Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques

18th International Workshop, APPROX 2015, and
19th International Workshop, RANDOM 2015
August 24–26, 2015, Princeton, USA

Edited by

Naveen Garg
Klaus Jansen
Anup Rao
José D. P. Rolim

LIPICS

Editors

Naveen Garg	Klaus Jansen
Indian Institute of Technology Delhi	University of Kiel
New Delhi	Kiel
`naveen@cse.iitd.ac.in`	`kj@informatik.uni-kiel.de`
Anup Rau	Jośe D. P. Rolim
University of Washington	University of Geneva
Washington	Geneva
`anuprao@cs.washington.edu`	`Jose.Rolim@unige.ch`

ACM Classification 1998
E.1 Data Structures, E.4 Coding and Information Theory, F. Theory of Computation, F.1.0 Computation by Abstract Devices, F.1.2 Modes of Computation, F.1.3 Complexity Measures and Classes, F.2.0 Analysis of Algorithms and Problem Complexity, F.2.1 Numerical Algorithms and Problems, F.2.2 Nonnumerical Algorithms and Problems G.1.2 Approximation, G.1.3 Numerical Linear Algebra, G.1.6 Optimization, G.2 Discrete Mathematics, G.2.1 Combinatorics, G.2.2 Graph Theory, G.2.3 Discrete Mathematics – Applications, G.3 Probability and Statistics, H.1.1 Coding and Information Theory, I.1.2 Algorithms, I.2.6 Learning, I.3.5 Computational Geometry and Object Modeling, J.4 Computer applications – Social and Behavioral Sciences

ISBN 978-3-939897-89-7

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-939897-89-7.

Publication date
August, 2015

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Digital Object Identifier: LIPIcs.APPROX-RANDOM.2015.i

ISBN 978-3-939897-89-7 **ISSN 1868-8969** **http://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8969

http://www.dagstuhl.de/lipics

Contents

18th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'15) / 19th Int'l Workshop on Randomization and Computation (RANDOM'15).
Editors: Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

This volume contains the papers presented at the 18th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2015) and the 19th International Workshop on Randomization and Computation (RANDOM 2015), which took place concurrently in Princeton University, USA during August 24–26, 2015.

APPROX focuses on algorithmic and complexity issues surrounding the development of efficient approximate solutions to computationally difficult problems, and was the 17th in the series after Aalborg (1998), Berkeley (1999), Saarbrücken (2000), Berkeley (2001), Rome (2002), Princeton (2003), Cambridge (2004), Berkeley (2005), Barcelona (2006), Princeton (2007), Boston (2008), Berkeley (2009), Barcelona (2010), and Princeton (2011), Boston (2012), Berkeley (2013), Barcelona (2014). RANDOM is concerned with applications of randomness to computational and combinatorial problems, and was the 18th workshop in the series following Bologna (1997), Barcelona (1998), Berkeley (1999), Geneva (2000), Berkeley (2001), Harvard (2002), Princeton (2003), Cambridge (2004), Berkeley (2005), Barcelona (2006), Princeton (2007), Boston (2008), Berkeley (2009), Barcelona (2010), Princeton (2011), Boston (2012), Berkeley (2013), Barcelona (2014).

Topics of interest for APPROX and RANDOM are: design and analysis of approximation algorithms, hardness of approximation, small space algorithms, sub-linear time algorithms, streaming algorithms, embeddings and metric geometry, mathematical programming methods, combinatorial problems in graphs and networks, algorithmic game theory and economic, computational geometric problems, approximate learning, online algorithms, approaches that go beyond worst case analysis, design and analysis of randomized algorithms, randomized complexity theory, pseudorandomness and derandomization, random combinatorial structures, random walks/Markov chains, expander graphs and randomness extractors, probabilistic proof systems, random projections and embeddings, error-correcting codes, average-case analysis, property testing, computational learning theory, and other applications of approximation and randomness.

The volume contains 26 contributed papers, selected by the APPROX Program Committee out of 61 submissions, and 30 contributed papers, selected by the RANDOM Program Committee out of 79 submissions.

We would like to thank all of the authors who submitted papers, the invited speakers, the members of the Program Committees, and the external reviewers. We gratefully acknowledge the Department of Computer Science and Engineering of the Indian Institute of Technology Delhi, the Institute of Computer Science of the Christian-Albrechts-Universität zu Kiel, the Department of Computer Science and Engineering of the University of Washington, and the Department of Computer Science of the University of Geneva.

August 2015

Naveen Garg
Klaus Jansen
Anup Rao
José D. P. Rolim

Organization

Program Committees

APPROX 2015

Parinya Chalermsook	Max-Planck Institute for Informatics, Germany
Michael Dinitz	Johns Hopkins University, USA
Naveen Garg (chair)	Indian Institute of Technology Delhi, India
Fabrizio Grandoni	IDSIA, Switzerland
Sungjin Im	University of California at Merced, USA
Amit Kumar	Indian Institute of Technology, India
Lap Chi Lau	The Chinese University of Hong Kong, Hong Kong
Nicole Megow	TU Berlin, Germany
Harald Raecke	Technische Universität München, Germany
Rishi Saket	IBM Research, USA
Piotr Sankowski	Warsaw University, Poland
Roy Schwarz	Princeton University, USA
René Sitters	University Amsterdam, The Netherlands
David Steurer	Cornell University, USA
Chaitanya Swamy	University of Waterloo, Canada
Andreas Wiese	Max-Planck Institute for Informatics, Germany

RANDOM 2015

Dimitris Achlioptas	University of California, USA
Alex Andoni	University of California, USA
Ken Clarkson	IBM Almaden Research Center, USA
Anindya De	University of California, USA
Hu Fu	Microsoft Research, USA
Nick Harvey	University of British Columbia, Canada
Xin Li	University of Illinois at Chicago, USA
Aleksander Madry	Massachusetts Institute of Technology, USA
Raghu Meka	Microsoft Research, USA
Eric Price	University of Texas at Austin, USA
Anup Rao (chair)	University of Washington, USA
Aaron Roth	University of Pennsylvania, USA
Mohit Singh	Microsoft Research, USA
Ali Sinop	University of California, USA
Madur Tulsiani	Toyota Technological Institute at Chicago, USA
David Woodruff	IBM Research, USA
Mary Wootters	Carnegie Mellon University, USA

External Reviewers

Scott Aaronson
Fidaa Abed
Marek Adamczyk
Sara Ahmadian
Elizabeth Allman
Hyung-Chan An
Elliot Anshelevich
Antonio Auffinger
Victor Bapst
Siddharth Barman
Alexander Barvinok
Frédérique Bassino
Jatin Batra
Mohammad Bavarian
Petra Berenbrink
Amey Bhangale
Aditya Bhaskara
Nayantara Bhatnagar
Abhishek Bhowmick
Sara Billey
Eric Blais
Antonio Blanca
Andrej Bogdanov
Gábor Braun
Michael Brautbar
Vladimir Braverman
Joshua Brody
Clément Canonne
Stefan Canzar
Arnaud Carayol
Amit Chakrabarti
Siu Man Chan
Siu On Chan
Timothy M. Chan
Chandra Chekuri
Lin Chen
Mahdi Cheraghchi
Stephen Chestnut
Flavio Chierichetti
Kai-Min Chung
Gil Cohen
Michael B. Cohen
Amin Coja-Oghlan
Ágnes Cseh
Marek Cygan
Artur Czumaj
Daniel Dadush
Samir Datta
Anindya De
Amit Deshpande
Jian Ding
Andrew Drucker
Zeev Dvir
Robert Elsässer
Alina Ene
Leah Epstein
Jittat Fakcharooenphol
Bill Fefferman
Vitaly Feldman
Matthias Feldotto
Michael A. Forbes
Nikolaos Fountoulakis
Alan Frieze
Zachary Friggstad
Benjamin Fuller
Andreas Galanis
Ankit Garg
Jugal Garg
Ran Gelles
Fulvio Gesmundo
Arijit Ghosh
George Giakkoupis
Mika Göös
Parikshit Gopalan
Michelangelo Grigni
Elena Grigorescu
Sudipto Guha
Abhradeep Guha Thakurta
Siyao Guo
Ankit Gupta
Varun Gupta
Moritz Hardt
Matan Harel
Pooya Hatami
Brett Hemenway
Sandy Heydrich
Zengfeng Huang
T-H. Hubert Chan
Satoshi Ikeda
Christian Ikenmeyer

Aukosh Jagannath
Hossein Jowhari
T.S. Jayram
Gautam Kamath
Graeme Kemkes
Sreyash Kenkre
Jeff Kinne
Pascal Koiran
Alexandra Kolla
Ilan Komargodski
Alexander Kononov
Swastik Kopparty
Pravesh Kothari
Robert Krauthgamer
Ravishankar Krishnaswamy
Mrinal Kumar
Tsz Chiu Kwok
Bundit Laekhanukit
Troy Lee
Marc Lelarge
Nikos Leonardos
Asaf Levin
Jian Li
Shi Li
Maarten Loffler
Po-Shen Loh
Anand Louis
Shachar Lovett
Chi-Jen Lu
Konstantin Makarychev
Yury Makarychev
Elitza Maneva
Alberto Marchetti-Spaccamela
Monaldo Mastrolilli
Julian Mestre
Daniele Micciancio
Sarah Miracle
Matthias Mnich
Tobias Mömke
Marco Molinaro
Ashley Montanaro
Cris Moore
Hiroki Morizumi
Luca Moscardelli
Benjamin Moseley
Dana Moshkovitz
Marcin Mucha
Viswanath Nagarajan
Jelani Nelson
Alantha Newman
Cyril Nicaud
Sotiris Nikoletseas
Aleksandar Nikolov
Igor Carboni Oliveira
Rafael Oliveira
Neil Olver
Krzysztof Onak
Alon Orlitsky
Denis Pankratov
Pablo Parrilo
Wesley Pegden
Chris Peikert
Britta Peis
Richard Peng
Eric Pricetes
Ilya Razenshteyn
Leo Reyzin
John Rhodes
Noga Ron-Zewi
Ben Rossman
Yogish Sabharwal
Sushant Sachdeva
Chandan Saha
Justin Salez
Steven Sam
Piotr Sankowski
Rahul Santhanam
Shubhangi Saraf
Cristiane Sato
Nitin Saurabh
Saket Saurabh
Kevin Schewior
Andreas Schmid
Or Sheffet
Bruce Shepherd
Alexander Sherstov
Igor Shinkar
Yaron Singer
Nitin Singh
Piotr Skowron
Allan Sly
Adam Smith
Christian Sohler
Zhao Song
José A. Soto
Frank Sottile

Andrea Sportiello
Piyush Srivastava
Daniel Stefankovic
David Steurer
Simon Straub
He Sun
Nike Sun
Ola Svensson
Mario Szegedy
Kenjiro Takazawa
Omer Tamuz
Li-Yang Tan
Augusto Teixeira
Prasad Tetali
Justin Thaler
Thomas Thierauf
Jacobo Toran
Madhur Tulsiani
Mario Ullrich
Sumedha Uniyal
Greg Valiant
Erik Jan van Leeuwen
Dieter van Melkebeek
Rob van Stee
Daniel Vaz
Juan Vera
José Verschae
Aravindan Vijayaraghavan
Fabian Wagner
Tandy Warnow
Thomas Watson
Udi Wieder
Ryan Williams
Pratik Worah
Nick Wormald
Bang Ye Wu
Yihong Wu
Ning Xie
Masafumi Yamashita
Grigory Yaroslavtsev
Yitong Yin
Rico Zenklusen
Noga Zewi
Qiang Zhang
Qin Zhang
Hang Zhou
Yuan Zhou
James Zou
Anna Zych

List of Authors

Fidaa Abed
Ittai Abraham
Jayadev Acharya
Anna Adamaszek
David Adjiashvili
Yogesh Anbalagan

Nikhil Bansal
Victor Bapst
Boaz Barak
Balthazar Bauer
Alok Baveja
Sebastian Berndt
Amey Bhangale
Vijay V.S.P. Bhattiprolu
Eric Blais
Antonio Blanca
Ralph C. Bottesch
Vladimir Braverman
Joshua Brody
Mark Bun

Clément L. Canonne
Marco L. Carmosino
Parinya Chalermsook
Amit Chavan
Shiri Chechik
Lin Chen
Stephen R. Chestnut
Julia Chuzhoy
Bouke Cloostermans
Edith Cohen
Gil Cohen
Amin Coja-Oghlan
Oliver Cooley
José Correa
Marek Cygan

Michael Dinitz
Zachary Drudi

Charilaos Efthymiou
Michael Elkin

David Felber
Hendrik Fichtenberger
Arnold Filtser
Michael A. Forbes
Zachary Friggstad

Andreas Galanis
Zhihan Gao
Dmitry Gavinsky
Rong Ge
Siyao Guo
Anupam Gupta
Venkatesan Guruswami

Bernhard Haeupler
David G. Harris
Nicholas J. A. Harvey
Johan Håstad
Chien-Chung Huang
Hao Huang
Sangxia Huang

Jennifer Iglesias
Russell Impagliazzo
Stephen Ingram
Kazuo Iwama

Nor Jaafari
Klaus Jansen

Valentine Kabanets
Gautam Kamath
Pritish Kamath
Mihyun Kang
Haim Kaplan
Andreas Karrenbauer
Shiva Prasad Kasiviswanathan
David H. K. Kim
Hartmut Klauck
Kim-Manuel Klein
Tomasz Kociumaka
Antonina Kolokolova
Ilan Komargodski
Robin Kothari
Robert Krauthgamer
Ravishankar Krishnaswamy

Euiwoong Lee
Shachar Lovett

Tengyu Ma
Rajsekar Manokaran
Pasin Manurangsi
Nicole Megow
Shuichi Miyazaki
Ankur Moitra
Shay Moran
Dana Moshkovitz
Elchanan Mossel

Viswanath Nagarajan
Ofer Neiman
Andrei Nikiforov
Sergey Norin

Ryan O'Donnell
Igor C. Oliveira
Rafail Ostrovsky

Pan Peng
Pablo Pérez-Lantero
Kirk Pruhs

David Racicot-Desloges
Prasad Raghavendra
Rajmohan Rajaraman
R. Ravi
Oded Regev
Roman Rischke
Sebastien Roch
Alan Roytman
Mark Rudelson

Mario Sanchez
Miklos Santha
Ramprasad Saptharishi
Kanthi K. Sarpatwar
Kevin Schewior
Baruch Schieber
Rocco A. Servedio
Hadas Shachnai
Alistair Sinclair
Kathrin Skubch
Christian Sohler
José Soto
Aravind Srinivasan
Daniel Štefankovič
Cliff Stein
Thomas Steinke
David Steurer
Leen Stougie
Francis Sullivan
Xiaoming Sun
Ravi Sundaram

Avishay Tal
Li-Yang Tan
Luca Trevisan

Girish Varma
Ameya Velingker
Rakesh Venkat
Adrian Vetta
Eric Vigoda
Aravindan Vijayaraghavan
Ilya Volkovich

Tal Wagner
Carol Wang
Andrew Warfield
Udi Wieder
Andreas Wiese
Jake Wires
David Witmer
Joel L. Wolf
David P. Woodruff
John Wright
Hehui Wu

Pan Xu

Hiroki Yanagisawa
Amir Yehudayoff

On Guillotine Cutting Sequences*

Fidaa Abed[1], Parinya Chalermsook[1], José Correa[2], Andreas Karrenbauer[1], Pablo Pérez-Lantero[3], José A. Soto[4], and Andreas Wiese[1]

1 **Max Planck Institute for Informatics**
Saarbrücken, Germany
{fabed,karrenba,parinya,awiese}@mpi-inf.mpg.de

2 **Department of Industrial Engineering, Universidad de Chile**
Santiago, Chile
correa@uchile.cl

3 **Escuela de Ingeniería Civil Informática, Universidad de Valparaíso**
Valparaiso, Chile
pablo.perez@uv.cl

4 **Department of Mathematical Engineering and CMM, Universidad de Chile**
Santiago, Chile
jsoto@dim.uchile.cl

Abstract

Imagine a wooden plate with a set of non-overlapping geometric objects painted on it. How many of them can a carpenter cut out using a panel saw making guillotine cuts, i.e., only moving forward through the material along a straight line until it is split into two pieces? Already fifteen years ago, Pach and Tardos investigated whether one can always cut out a constant fraction if all objects are axis-parallel rectangles. However, even for the case of axis-parallel squares this question is still open. In this paper, we answer the latter affirmatively. Our result is constructive and holds even in a more general setting where the squares have weights and the goal is to save as much weight as possible. We further show that when solving the more general question for rectangles affirmatively with only axis-parallel cuts, this would yield a combinatorial $O(1)$-approximation algorithm for the Maximum Independent Set of Rectangles problem, and would thus solve a long-standing open problem. In practical applications, like the mentioned carpentry and many other settings, we can usually place the items freely that we want to cut out, which gives rise to the two-dimensional guillotine knapsack problem: Given a collection of axis-parallel rectangles without presumed coordinates, our goal is to place as many of them as possible in a square-shaped knapsack respecting the constraint that the placed objects can be separated by a sequence of guillotine cuts. Our main result for this problem is a quasi-PTAS, assuming the input data to be quasi-polynomially bounded integers. This factor matches the best known (quasi-polynomial time) result for (non-guillotine) two-dimensional knapsack.

1998 ACM Subject Classification G.2.1 Combinatorics, I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Guillotine cuts, Rectangles, Squares, Independent Sets, Packing

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.1

* This work was partially supported by Iniciativa Científica Milenio through the Millennium Nucleus Information and Coordination in Networks RC130003.

1 Introduction

Two-dimensional cutting stock problems arise naturally in industrial manufacturing. The goal is to cut out a given set of geometric objects from a large piece of parent material such as wood, metal, or glass. Guillotine cutting sequences play an important role in such processes. Starting from the large piece, in each step such a sequence takes one of the available pieces and cuts it along a straight line into two smaller pieces. Eventually, each of the input objects corresponds to one of the resulting pieces. Using guillotine cuts is often required by the available technology since more complex cutting patterns are often not possible.

Guillotine cuts motivate interesting basic problems in combinatorics, computational geometry, and combinatorial optimization. For instance, Urrutia [16] asked the following simple and yet intricate question: Given a set of pairwise non-overlapping compact convex geometric objects in the plane, can we always separate a constant fraction of them using a guillotine cutting sequence? Since the answer would be trivially *no* if cutting through objects is forbidden, such cuts are allowed at the expense of losing partial objects completely. This is equivalent to asking how many of them we can cut out using only guillotine cuts, i.e., each piece must not contain more than one complete object at the end. Pach and Tardos [15] investigated this question and showed that already for straight line segments this cannot always be achieved. They give a family of instances with straight line segments yielding an upper bound of $O(n^{\log 2/\log 3})$ for the number of line segments that can be cut out using only guillotine cuts. On the other hand, they show that we can indeed cut out a constant fraction of the input objects if – loosely speaking – all input objects have roughly the same size.[1] In this paper, we investigate the natural related question when the objects to be cut are rectangles[2] as we describe in the sequel.

1.1 Guillotine cuts for squares and rectangles

Even though Urrutia's general conjecture about convex objects was refuted, Pach and Tardos [15] wrote that "it seems plausible" that the question can be answered in the affirmative if the input objects are axis-parallel rectangles. They provided a cut sequence that saves $\Omega(n/\log n)$ rectangles, and stated that they "were unable to verify [a bound of $\Omega(n)$] even for axis-parallel squares".

In our first result in this paper, we answer the latter open question by giving a guillotine cutting sequence that recovers an $1/81$-fraction of any set of axis-parallel squares. We first clean up the instance by placing a hierarchical grid with a random offset and delete some of the squares according to it (a standard step, see e.g., [7]). Then, we show that in each iteration of the cutting sequence we can find a cut that intersects at most $O(1)$ of the remaining squares while there is at least one surviving square on either side of the cut. By viewing this sequence as a binary tree, we can elegantly charge the number of the intersected squares to the number of surviving squares in the leaves.

Furthermore, we extend the previous result to the weighted case of the problem in which each square i has a weight w_i associated to it. As usual in combinatorial optimization, the weight of each object is a measure for its importance and we are looking for a cutting sequence that cuts out squares whose total weight is at least a constant fraction of the total weight of the input squares. As our above techniques do not carry over to this case, we give

[1] For a precise statement see Pach and Tardos [15].
[2] In this paper all rectangles considered are axis-parallel and open: a line going through a boundary side of such a rectangle does not destroy it.

a new algorithm that is based on a suitable conflict graph with one vertex for each input square. The graph has the important property that any independent set corresponds to a set of squares that can be cut out completely without any further loss. Even more, we show that we can color the vertices with at most 9 colors and thus there is an independent set corresponding to a 1/9-fraction of the entire weight of the input squares. Furthermore, our reasoning here directly extends to hypercubes in arbitrary dimensions. Thus, we can recover a 4/729-fraction of the weight of the squares in the two-dimensional plane, and a $1/2^{O(d)}$-fraction in d dimensions.

▶ **Theorem 1** (informal). *For axis-parallel squares there is always a guillotine cutting sequence that recovers a* $1/81$*-fraction in the unweighted case, a* $4/729$*-fraction in the weighted case, and a* $1/2^{O(d)}$*-fraction in the weighted case in* d *dimensions.*

An interesting aspect of our algorithms is that they only require axis-parallel cuts as opposed to the original question posed by Urrutia [16] (and investigated by Pach and Tardos [15]) where in principle also diagonal ones are allowed. Restricted to axis-parallel cuts we prove that for unit squares there are instances in which any cutting sequence can recover at most a 1/2-fraction of the squares (See Section 3 and Figure 3). Interestingly, although this is the strongest negative result we can obtain, we can easily show an algorithm finding a cutting sequence recovering 1/2-fraction of any set of unit squares, or more generally, for rectangles of equal height or width (see Section 2.3).

1.2 Connection to Independent Set of Rectangles

Inspired by the previous comment, we formulate a conjecture which is slightly stronger than the question investigated by Pach and Tardos [15].

▶ Conjecture 1.1. For any set of n non-overlapping axis-parallel rectangles there is a guillotine cutting sequence with only axis-parallel cuts separating $\Omega(n)$ of them.

If the conjecture was true this would have exciting consequences for the notoriously hard Maximum Independent Set of Rectangles (MISR) problem. Given a set of possibly overlapping axis-parallel rectangles, we want to compute a non-overlapping subset of maximum size. Finding a polynomial time $O(1)$-approximation algorithm for this problem is an important open problem (see e.g, [1, 7, 4, 5] and references therein.) We show that there is a simple dynamic program that computes the largest subset of the given rectangles that can be cut out completely using only guillotine cuts. Now, the conjecture implies that an $\Omega(1)$-fraction of the optimal solution of a MISR-instance can be cut out using guillotine cuts. Assuming this, we show that a simple dynamic programming (DP) algorithm yields an elegant combinatorial $O(1)$-approximation for MISR (see Section 4).

▶ **Theorem 2.** *If Conjecture 1.1 is true, then there is a* $O(1)$*-approximation algorithm for MISR with running time* $O(n^5)$*.*

1.3 Two-dimensional guillotine knapsack

The final contribution of this work concerns the two-dimensional guillotine knapsack problem. In this problem we are given a set of rectangles and a square-shaped knapsack[3], modeling a

[3] While with our techniques we can also handle the case of a rectangular knapsack, in this extended abstract for simplicity we assume the knapsack to be a square.

piece of parent material. We are interested in a placement of as many rectangles as possible in the knapsack so that there is a guillotine cutting sequence extracting them.

The two-dimensional geometric knapsack problem, without taking into account the guillotine cut constraint, is well-studied in the literature. For squares, a $(5/4+\epsilon)$-approximation is presented by Harren [10], which was subsequently improved to a PTAS by Jansen and Solis-Oba [12]. The best known polynomial time result for rectangles is a $(2+\epsilon)$-approximation result due to Jansen and Zhang [14]. The same authors presented a faster and simpler $(2+\epsilon)$-approximation for the unweighted case [13]. Recently, Adamaszek and Wiese presented a quasi-PTAS for rectangles that assumes the input data to be polynomially bounded integers [2]. For rectangles, there are $(1+\epsilon)$-approximation algorithms known if we are allowed to increase the size of the knapsack by a factor of $1+\epsilon$ in both dimensions [9], or even only in one [11]. Also, there is a PTAS if the profit of each item equals its area [3].

It is worth noting that many of the known results for this problem can be easily adjusted to take the guillotine cut constraint into account. However, this is not the case for the result giving the best known approximation factor for the problem: the recent $(1+\epsilon)$-approximation with quasi-polynomial running time (QPTAS) [2]. The algorithm is based on partitioning the placement area into $(\log n)^{O(1)}$ rectangular boxes such that there is a near-optimal solution in which – informally speaking – each box contains either only one big item, or high and narrow items, or wide and thin items. Then, the objects are assigned to the boxes via linear programming. This algorithm does not directly extend to the setting of guillotine cutting sequences. First, the mentioned near-optimal solution might not allow a guillotine cutting sequence (even if it is constructed based on such a solution) and second, the LP-rounding procedure does not necessarily produce such solutions either. We overcome these problems by showing that at additional (marginal) cost, we can construct a near-optimal solution in which essentially the mentioned boxes can be cut out using guillotine cuts. Then, we replace the LP-approach by a dynamic program. For this to work, we carefully round the items such that after rounding there are only $(\log n)^{O(1)}$ many different types of items and our DP guesses the correct guillotine cuts step by step, together with the distribution of the items on either side of the cut. In summary, our result is the following.

▶ **Theorem 3.** *There is a quasi-PTAS for the unweighted two-dimensional guillotine knapsack problem if all input data are quasi-polynomially bounded integers. This holds with and without the possibility to rotate items by 90 degrees.*

2 Guillotine cutting sequences for squares

In this section, we give guillotine cutting sequences recovering a constant fraction of a set of non-overlapping axis-parallel squares [4] of arbitrary sizes. Our results generalize to higher dimensions. First, we give some basic terminologies. In Section 2.1, we present a cutting sequence for unweighted squares, and then in Section 2.2, we prove the result for the weighted case.

Let P be a *piece*, i.e., a rectangle in the plane, and let H_1 and H_2 be the two open disjoint half-planes bounded by a straight line ℓ. Cutting P along ℓ gives us two sub-pieces $P_1 = P \cap H_1$ and $P_2 = P \cap H_2$. A *cutting strategy* is represented by a binary tree $\mathcal{T}$ where each non-leaf node $v \in V(\mathcal{T})$ is equipped with a piece P_v and a straight line ℓ_v such that cutting P_v along the straight line ℓ_v gives us P_{v_1} and P_{v_2}, where v_1, v_2 are the children of v.

[4] In this section, we henceforth implicitly assume all squares to be axis-parallel.

Let $\mathcal{O}$ be a set of objects. We say that the cutting strategy $\mathcal{T}$ *separates* $\mathcal{O}$ if the following statements hold

- The piece P_r associated with the root node r is a rectangle containing all objects in $\mathcal{O}$.
- For each non-leaf node v, the straight line ℓ_v intersects no object inside P_v.
- For each leaf node v, the piece P_v contains only one object in $\mathcal{O}$.

We also say that $\mathcal{O}$ is *guillotine separable* if there is a cutting strategy $\mathcal{T}$ separating $\mathcal{O}$. In the rest of this section, we focus on the case when our input objects are squares of arbitrary sizes. Let $\mathcal{R}$ denote the input set of squares.

Grid Lemma. One of the components in our proof is a collection of (multi-level) grid lines drawn on the plane in a "nice" way. These grid lines will be used to suggest our cut sequence, i.e., most of the straight lines in the strategy coincides with one of these grid lines. [5] We draw grid lines of various granularities and remove squares from $\mathcal{R}$ according to the grid lines, so that the remaining squares admit a guillotine cutting sequence. We say that a square $R \in \mathcal{R}$ is at level-i if its side length is in the interval $(N/2^{i+1}, N/2^i]$, where $N \in \mathbb{N}$ is used for normalization so that level-0 contains the largest squares.

In a first step, we independently pick two random numbers $x, y \in [0, N)$ defining a random shift to draw the grid. For each i (i.e. level), the vertical grid lines at level-i are drawn at $x, x+N/2^i, x+2\cdot N/2^i, \ldots$ (wrapping up appropriately); similarly, the horizontal grid lines at level-i are drawn at $y, y+N/2^i, y+2\cdot N/2^i \ldots$. Grid cells bounded by consecutive grid lines at level $i-1$ are said to be at level-i, so level-i grid cells are squares of size $(2N/2^i)$-by-$(2N/2^i)$. Note that the higher the level the more fine grained the grid is. A square $R \in \mathcal{R}$ is removed from this step if it intersects with grid lines at levels below it, i.e. if R is in level-i, then it is removed if it intersects a line at levels $i-1, \ldots, 0$. Let $\mathcal{R}_1$ be the set of squares that are not removed from this step.

▶ Claim 2.1. A level-i square $R \in \mathcal{R}$ of side length $\ell_R \in (N/2^{i+1}, N/2^i]$ remains in $\mathcal{R}_1$ with probability $(1-\mu_R)^2 \geq 1/4$, where $\mu_R = \ell_R 2^{i-1}/N$.

Proof. The probability that a horizontal grid line at level $i-1$ intersects R is $\mu_R = \ell_R 2^{i-1}/N$ (because of the random shift). Notice that μ_R is between $1/4$ and $1/2$. Now, since the shifts x, y are chosen independently, the probability that the square R survives in $\mathcal{R}_1$ is $(1-\mu_R)^2 \geq 1/4$. ◀

In a second step, we further sample $\mathcal{R}_1$ to obtain $\mathcal{R}_2$, where each square is sampled with relatively large probability. Now we consider each grid cell C at level-i that contains a subset of squares $\mathcal{R}_1^C$ at level-i. Cell C may contain up to 9 squares, so if we are not careful, we might end up paying an extra factor of 9 (giving 1/36 marginal only). So, we define a distribution on $\mathcal{R}_1^C$ where exactly one square R in $\mathcal{R}_1^C$ is kept, and R is kept with probability

$$\frac{1}{(1-\mu_R)^2 \cdot M_C}, \quad \text{for } M_C = \sum_{S \in \mathcal{R}_1^C} \frac{1}{(1-\mu_S)^2}.$$

Let $\mathcal{R}_2$ be the set of squares remaining after this process. We analyze the probability that a level-i square R remains in $\mathcal{R}_2$. This can be broken down into:

$$\begin{aligned} \mathbf{Pr}\left[R \in \mathcal{R}_1\right] \cdot \mathbf{Pr}\left[R \in \mathcal{R}_2 \mid R \in \mathcal{R}_1\right] &= (1-\mu_R)^2 \cdot \left(\frac{1}{(1-\mu_R)^2 M_C}\right) \\ &= 1/M_C. \end{aligned}$$

[5] We may deviate from this strategy when considering subsets with a constant number of squares.

▶ Claim 2.2. $M_C \leq 81/4$.

Proof. Each square $R \in \mathcal{R}_1^C$ satisfies $\mu_R \in (1/4, 1/2]$, and has side length ℓ_R strictly larger than $1/4$ of the side length of C. This implies $|\mathcal{R}_1^C| \leq 9$. Furthermore, R consumes a μ_R^2-fraction of the area of C, and $\sum_{R\in\mathcal{R}_1^C} \mu_R^2 \leq 1$ must be satisfied. We argue below that M_C is maximized when $|\mathcal{R}_1^C| = 9$ and $\mu_R = 1/3$ for each square $R \in \mathcal{R}_1^C$, i.e., when $\mathcal{R}_1^C$ is a set of 9 equal squares of area 1/9 of that of C. This gives $M_C \leq 9 \cdot (1-1/3)^{-2} = 9 \cdot (9/4) = 81/4$.

We ignore the geometry of the squares and focus only on the values of μ_R that satisfy the following:

$$\begin{aligned} \max \quad & \sum_{R\in\mathcal{R}_1^C} \frac{1}{(1-\mu_R)^2} \\ \text{s.t.} \quad & \sum_{R\in\mathcal{R}_1^C} \mu_R^2 \leq 1, \quad |\mathcal{R}_1^C| \leq 9, \quad \mu_R \in (1/4, 1/2]. \end{aligned}$$

Let q be an integer in $\{1, \dots, 9\}$. Notice that for a fixed choice of $|\mathcal{R}_1^C| = q$, the objective function is maximized when the values of μ_R are "balanced": One can check that if there are two values $\mu_R \neq \mu_{R'}$, then we could have averaged these values to $\mu'_R = \mu'_{R'} = \sqrt{(\mu_R^2 + \mu_{R'}^2)/2}$; the new choices of μ'_R and $\mu'_{R'}$ still satisfy all constraints while increasing the objective. Because $1/4 < \mu_R \leq 1/2$, the objective values cannot exceed $\frac{q}{(1-1/2)^2} \leq 20 < 81/4$ for $q \leq 5$. Moreover, for a fixed $q \in \{6, \dots, 9\}$, the objective is optimized by setting μ_R such that $q \cdot \mu_R^2 = 1$, i.e., $\mu_R = 1/\sqrt{q}$. An enumeration of the corresponding objective values reveals the maximum of $81/4$ for $q = 9$, which proves the claim. ◀

Notice that in the random subset $\mathcal{R}_2 \subseteq \mathcal{R}$, each square at level-$i$ is not intersected by grid lines at levels $i-1, \dots, 0$, and each level-i grid cell has at most one square in level-i. Then, using claims 2.1 and 2.2, we can state the main result of this section:

▶ **Lemma 4.** *There exists a distribution $\mathcal{D}\colon 2^{\mathcal{R}} \to [0,1]$ such that each subset $\mathcal{R}'$ in its support admits a grid drawing, with some shift, satisfying the following properties:*

- *Each square at level-i is not intersected by grid lines at levels $i-1, \dots, 0$.*
- *Each level-i grid cell has at most one square in level-i.*

Moreover, each square $R \in \mathcal{R}$ appears in a randomly drawn subset with probability at least $\epsilon_1 = 4/81$, i.e., $\mathbf{Pr}_{\mathcal{R}'\sim\mathcal{D}}[R \in \mathcal{R}'] \geq \epsilon_1$.

2.1 Unweighted case

Before we treat the more general weighted case, we first show how to save a linear number of squares using guillotine cuts. The approach of this section is not subsumed by the one from Section 2.2, as it yields a better constant than applying the latter with uniform weights. The high-level idea comes from the observation that the number of leaves in a proper binary tree is at least half the number of nodes. Thus, bounding the number of squares that are cut in each node yields a lower bound on the number of square that are saved.

▶ **Definition 5.** We call a line a *k-good separator*, if it intersects at most k squares and each side contains at least one square completely. We call a binary tree of a cutting strategy *k-good*, if each internal node has a k-good separator.

▶ **Lemma 6.** *If an instance admits a k-good tree, then there is a guillotine sequence such that $n/(k+1)$ squares survive.*

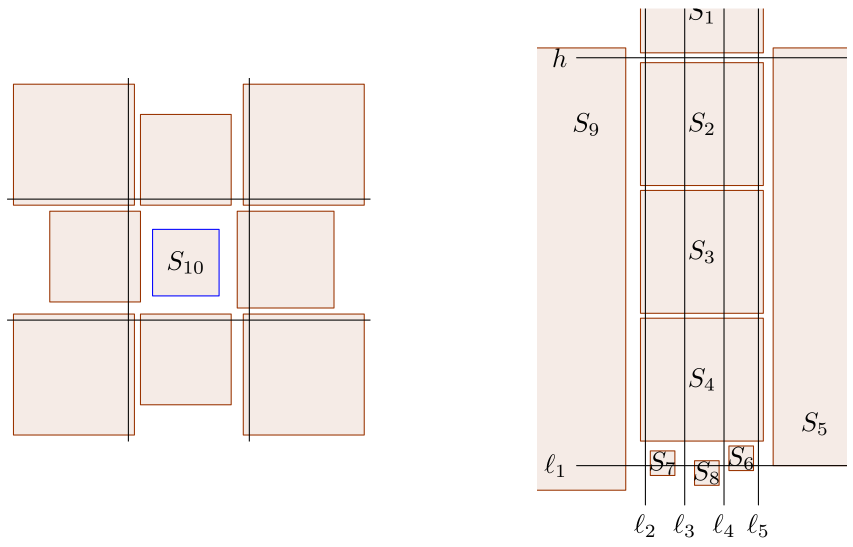


Figure 1 Arising cases in the proof of Lemma 7.

Proof. Observe that the number of surviving squares is equal to the number of leaves, say s, in the k-good tree. Let t be the number of nodes in this tree. The instance contains one square for each leaf and at most k squares for each internal node. Thus, we obtain $n \leq s + k \cdot (t - s) = s + k \cdot (s - 1) \leq (k + 1) \cdot s$, since $t = 2s - 1$ for a proper binary tree. ◀

▶ **Lemma 7.** *There is a guillotine sequence with a* 3*-good tree for any subset in the support of a distribution according to Lemma 4.*

Proof. We iteratively define the cut sequence that gives us a 3-good tree. Starting from the piece P_0 that contains every square, we show that, given a piece P', a 3-good separator for P' exists. Now we describe the existence proof of the separator. There are two cases to consider. First, if P' contains at least 10 squares, consider the squares in P' as a sequence $S_1, S_2, \ldots$ ordered by non-increasing side lengths. Let i be the level of S_{10}. We consider the grid cell of level i containing S_{10}. Observe (see left of Figure 1 for an illustration) that the edges of that cell can be covered by at most eight squares because these squares must be at lower levels than i and the distance between two adjacent corners of the cell, which is $2N/2^i$, is not wide enough to contain two squares of lower levels, which have side lengths of at least $N/2^i$, in its interior. Thus, one of the four grid lines defining the cell separates S_{10} from some other square while intersecting at most three. Since we choose a separator by the grid line at the level of S_{10}, this line cannot intersect any square in $\{S_{11}, S_{12}, \ldots\}$.

Let us now consider the other case when P' contains at most 9 squares. In this case, we would choose a separator that does not necessarily correspond to a grid line. We order the squares $S_1, \ldots, S_9$ by non-increasing y-coordinates of the bottom boundaries. Consider a horizontal line ℓ_1 that coincides with the bottom boundary of S_5, so ℓ_1 cannot "stab" any square in $\{S_1, \ldots, S_5\}$. If ℓ_1 stabs at most 3 squares in $\{S_6, S_7, S_8, S_9\}$, we would be done, ℓ_1 is our separator. Otherwise, ℓ_1 must stab all four squares and shares the border with S_5.

There are (at least) 4 combinatorially different vertical lines that separate squares in $\{S_5, \ldots, S_9\}$ without intersecting them. By "combinatorially different" we mean that they do not separate the squares in the exact same way. Denote by $\mathcal{L}$ a set of four such vertical lines. If there is a vertical line $\ell' \in \mathcal{L}$ that intersects at most 3 squares in $\{S_1, \ldots, S_4\}$, we would be done, as we can use ℓ' as our separator. Otherwise, each of these four lines stabs four squares in $\{S_1, \ldots, S_4\}$, and in this case, we can use the horizontal line h that coincides with the bottom boundary of S_1 as our separator; this line cannot overlap with any square in $\{S_1, \ldots, S_4\}$, and it can only overlaps with at most two of the squares in $\{S_5, \ldots, S_9\}$. ◀

▶ **Theorem 8.** *There is always a cutting strategy for a $\frac{1}{81}$-fraction of squares.*

Proof. We first apply the construction of Lemma 4. This guarantees the existence of a 3-good tree due to Lemma 7. This implies with Lemma 6 that a fraction of $\frac{4}{81} \cdot \frac{1}{4} = \frac{1}{81}$ of the given squares can be separated by guillotine cuts. ◀

2.2 Weighted case

In this setting, we are additionally given a weight function $w\colon \mathcal{O} \to \mathbb{R}_{\geq 0}$, and we want a cutting strategy separating a subset with the largest possible weight. We restate this question in an equivalent form with the following notion. An *ϵ-guillotine sampling* for objects $\mathcal{O}$ is a distribution $\mathcal{D}\colon 2^{\mathcal{O}} \to [0,1]$ such that any object $r \in \mathcal{O}$ is sampled by $\mathcal{D}$ with probability at least ϵ, i.e., $\mathbf{Pr}_{\mathcal{O}' \sim \mathcal{D}}[r \in \mathcal{O}'] \geq \epsilon$, and each subset $\mathcal{O}'$ in the support of $\mathcal{D}$ is guillotine separable.

▶ **Lemma 9.** *For any set of objects $\mathcal{O}$, the following are equivalent: (i) there is an ϵ-guillotine sampling for $\mathcal{O}$ and (ii) for any weight function $w : \mathcal{O} \to \mathbb{R}_{\geq 0}$, there is a subset $\mathcal{O}' \subseteq \mathcal{O}$ that is guillotine separable and $w(\mathcal{O}') \geq \epsilon \cdot w(\mathcal{O})$.*

Proof. The forward implication is easy to see. Suppose we have an ϵ-guillotine sampling $\mathcal{D}$ for $\mathcal{O}$. Let $w : \mathcal{O} \to \mathbb{R}_{\geq 0}$ be a weight function. We pick a random set $\mathcal{O}'$ according to the distribution $\mathcal{D}$. Then, we have

$$\mathbf{E}[w(\mathcal{O}')] \;=\; \sum_{r \in \mathcal{O}} w(r) \cdot \mathbf{Pr}[r \in \mathcal{O}'] \;\geq\; \epsilon \cdot w(\mathcal{O}).$$

This shows the existence of such a subset. For the backward implication, the proof is by LP duality. Let $\mathcal{F}_{\mathcal{O}}$ be the set of all guillotine separable subsets of $\mathcal{O}$. We write the following linear program that reflects the best ϵ guillotine sampling:

$$\begin{array}{lll}
\text{(LP)} & \max & \gamma \\
 & \text{s.t.} & \gamma \leq \displaystyle\sum_{\mathcal{O}' \in \mathcal{F}_{\mathcal{O}} : r \in \mathcal{O}'} p_{\mathcal{O}'} \text{ for all object } r \in \mathcal{O}, \\
 & & \displaystyle\sum_{\mathcal{O}' \in \mathcal{F}_{\mathcal{O}}} p_{\mathcal{O}'} = 1, \\
 & & p_{\mathcal{O}'} \geq 0 \text{ for all } \mathcal{O}'.
\end{array}$$

The dual of the above LP can be written as:

$$\begin{array}{lll}
\text{(LP')} & \min & \beta \\
 & \text{s.t.} & \beta \geq \displaystyle\sum_{r \in \mathcal{O}'} w_r \text{ for all } \mathcal{O}' \in \mathcal{F}_{\mathcal{O}}, \\
 & & \displaystyle\sum_{r \in \mathcal{O}} w_r = 1, \\
 & & w_r \geq 0 \text{ for all object } r \in \mathcal{O}.
\end{array}$$

Now, suppose that we can find a guillotine separable subset for any weight function w. This means that (LP') cannot be feasible for any (w, β) if $\beta < \epsilon$, so the optimal value of (LP') is at least ϵ. By duality, the optimal solution for (LP) gives us the distribution that is ϵ-guillotine sampling for $\mathcal{O}$. ◀

Lemma 9 allows us to focus on finding ϵ-guillotine samplings instead (this is an unweighted question). In what follows, we present such a sampling for any input set $\mathcal{R}$ of squares. We start by invoking Lemma 4 to find a distribution $\mathcal{D}'$ for $\mathcal{R}$. Consider each subset $\mathcal{R}'$ in the support of $\mathcal{D}'$ together with its grid drawing $\mathcal{G}'$. We will show that each such subset $\mathcal{R}'$ can be further partitioned into 9 guillotine separable subsets, which will imply the following theorem:

▶ **Theorem 10.** *Any set of squares $\mathcal{R}$ is ϵ-guillotine samplable for $\epsilon = 4/729$.*

For each square R at level-i, let $\mathsf{cell}(R)$ be the level-i cell containing R. We say that two squares R and S are *conflicting* if either R overlaps the boundary of $\mathsf{cell}(S)$ or S overlaps the boundary of $\mathsf{cell}(R)$. Observe that any pair of conflicting squares belong to different levels and if R overlaps the boundary of $\mathsf{cell}(\mathsf{S})$, then the level of R is smaller than that of S.

We define a conflict graph that encodes the conflict structures between squares. Let H be the graph such that the vertex set $V(H)$ corresponds to the squares in $\mathcal{R}'$, and there is an edge between squares R and S if and only if R and S are conflicting. The following two lemmas complete the proof.

▶ **Lemma 11.** *The graph H is 9-colorable.*

Proof. We prove this by induction on the number of vertices. The base case when $|V(H)| = 1$ is obvious. Now, consider any graph H with at least two vertices, and any square $R \in V(H)$ whose size is minimum among the squares in $V(H)$. Let ℓ be the side length of $\mathsf{cell}(R)$. Consider the set $N_H(R) \subset V(H)$ of the squares S defining an edge with R. It can only be that each square S in $N_H(R)$ is at the level below of that of R; so the lengths of these squares are strictly greater than $\ell/2$. We claim that $|N_H(R)| \leq 8$: There can be at most 4 squares in $N_H(R)$ that contain some corner of $\mathsf{cell}(R)$, and for each side of $\mathsf{cell}(R)$, there can be at most one square in $N_H(R)$ overlapping it and without containing a corner of $\mathsf{cell}(R)$. By the induction hypothesis, the graph obtained from H by removing the vertex R can be colored with 9 colors. Since the degree of R is at most 8, we can always assign a color to R, distinct from the colors of its neighbors $N_H(R)$. ◀

▶ **Lemma 12.** *Let $I \subseteq V(H)$ be an independent set. The squares $\{R\}_{R\in I}$ are guillotine separable.*

Proof. We prove this by iteratively defining the cutting strategy. Our cut sequence always cuts along grid lines, and any piece P produced in this process satisfies the following property: P contains at most one square, or, otherwise, let $\mathcal{R}_P$ be the set of squares inside P and ℓ the level of the *second* largest square in $\mathcal{R}_P$. Then, the sides of P are aligned with grid lines of levels at most ℓ.

Initially, let P_0 be the single piece that contains all squares, so the above properties are satisfied: One can assume that P_0 is bounded by four grid lines at level-0. Now, if every piece contains at most one square, then we are done. Otherwise, consider a piece P' with more than one square in $\mathcal{R}_{P'}$. Let R and R' be the largest and second largest squares in $\mathcal{R}_{P'}$, respectively.

There are two cases. See Figure 2 for reference. First, if R and R' belong to the same level ℓ, then we simply cut along any level-$(\ell-1)$ line L that separates the grid cells $\mathsf{cell}(R)$ and $\mathsf{cell}(R')$. Line L cannot intersect any square in $\mathcal{R}_{P'}$: By Lemma 4, these squares are

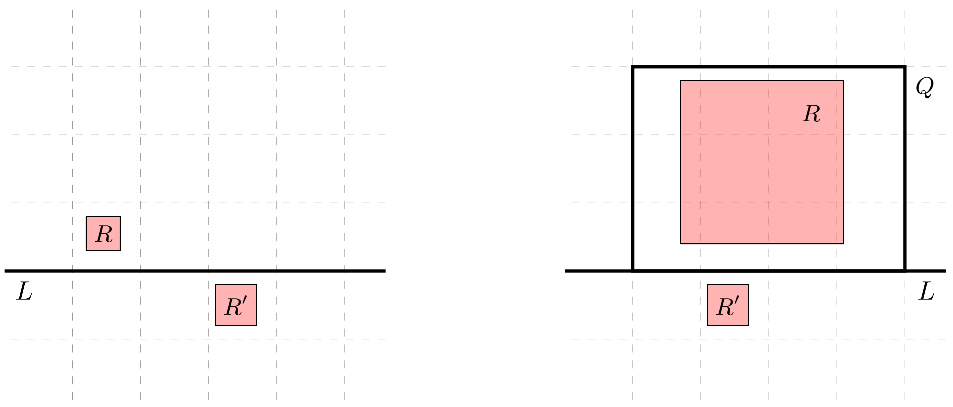


Figure 2 An illustration of the proof of Lemma 12. The figure on the left shows the case when the levels of R and R' are the same, and the right figure shows the other case when their levels are different.

at level at least ℓ, and cannot be intersected by level-$(\ell - 1)$ grid lines. Otherwise, suppose R and R' belong to different levels $\ell < \ell'$, respectively. Let Q be the union of level-ℓ' grid cells that overlap with square R. Notice that Q is a rectangle, and R' cannot be inside Q; otherwise, R would have intersected the boundaries of $\mathsf{cell}(R')$, contradicting the fact that they are not conflicting. This implies that Q and $\mathsf{cell}(R')$ are disjoint. Let L be any level-$(\ell' - 1)$ grid line that goes through a side of Q and separates Q from $\mathsf{cell}(R')$. Again, L cannot intersect any square in $\mathcal{R}_{P'}$ (since squares in $\mathcal{R}_{P'}$ are only at levels $\ell', \ell' + 1, \ldots$), and it separates R from R' in a way that maintains the properties. ◀

2.3 A cutting strategy for rectangles of the same width/height

Let $\mathcal{R}$ be a set of weighted rectangles of the same width or height . In this section, we prove the following lemma:

▶ **Lemma 13.** *There is a polynomial time algorithm to find a guillotine strategy that separates a set of rectangles of $\mathcal{R}$ with at least 1/2-fraction of its weight.*

Proof. Without loss of generality, we can assume that the rectangles in $\mathcal{R}$ have unit width. For every $x \in [0, 1]$, consider the set L_x of vertical lines at coordinates $\{x+2k : k \in \{0, \ldots, N\}$. Let $\mathcal{R}_x$ be the set of rectangles in $\mathcal{R}$ obtained by removing all rectangles intersecting a line in L_x, and all rectangles whose right side is contained in a line of L_x.

It is easy to see that $\mathcal{R}_x$ is guillotine separable. Indeed, we can first cut through all the lines in L_x to obtain a collection of vertical slabs of width 2 (In fact, we do not need to cut through all the $O(N)$ lines, since this could be non-polynomial in the number of rectangles. Precisely, we do not cut through the lines separating slabs without rectangles in $\mathcal{R}_x$). The rectangles in $\mathcal{R}_x$ of each vertical slab can then be separated using horizontal cuts. This is always possible, since the width of the slab does not allow two unit width rectangles side by side (recall that we remove the ones whose right side is aligned with a line in L_x).

Observe that if we choose x uniformly at random from $[0, 1]$, then every given rectangle $R \in \mathcal{R}$ belongs to $\mathcal{R}_x$ with probability 1/2. This means that there is a value x for which $\mathcal{R}_x$ contains at least 1/2 of the total weight of $\mathcal{R}$. In fact, this value can be found deterministically by standard techniques (the number of values of x with different sets $\mathcal{R}_x$ is linear in the number of rectangles). This concludes the proof of the lemma. ◀

2.4 Guillotine Cuts for d-Dimensional Boxes

By extending the ideas of this section, we can to get a $1/2^{O(d)}$-guillotine sampling for any set of d-dimensional cubes of arbitrary sizes. We first need to generalize the notion of guillotine cuts to higher dimensions. We say that a hyperplane is a *canonical hyperplane* if it is orthogonal to a vector in the standard basis, i.e. those hyperplanes illustrated by $x_i = a$ for $i \in \{1, \dots, d\}$ and $a \in \mathbb{R}$ would be canonical. When we say a piece $P \subseteq \mathbb{R}^d$, we mean the region that are bounded by $2d$ canonical hyperplanes, i.e. P can be represented by the intersection of $2d$ halfspaces and corresponds to $\bigcap_{i=1}^{d} \{x : (a_i \leq x_i \leq b_i)\}$.

Cutting a piece P along a canonical hyperplane h gives us two sub-pieces. A cutting strategy is represented by a binary tree $\mathcal{T}$ where each non-leaf node $v \in V(\mathcal{T})$ is equipped with a piece P_v and a hyperplane h_v such that cutting P_v along h_v gives us P_{v_1} and P_{v_2} where v_1 and v_2 are the children of v. We say that a set of objects $\mathcal{O}$ is guillotine separable if there is a cutting strategy $\mathcal{T}$ for $\mathcal{O}$ such that:

- The piece P_r associated with the root node contains all objects in $\mathcal{O}$.
- For each non-leaf node v, the canonical hyperplane h_v intersects no object inside P_v.
- For each leaf node v, the piece P_v contains only one object in $\mathcal{O}$.

To handle the weighted case, we can define a similar concept of guillotine sampling for objects. We prove the following theorem:

▶ **Theorem 14.** *There is an ϵ-guillotine sampling for any set of cubes of arbitrary sizes, for $\epsilon = 1/2^{O(d)}$.*

The rest of this section is spent on proving this theorem. The proof follows the same line of ideas used in the case of two dimensions. There are two steps. In the first step, we define the "high-dimensional grid" that will be used to suggest how the hyperplane will be selected. In the second step, we define the conflict graph that is shown to be $2^{O(d)}$ colorable.

Let $\mathcal{R}$ be a set of input cubes of arbitrary sizes. We say that a cube $r \in \mathcal{R}$ is at level-i if its edge length is in $(N/2^{i+1}, N/2^i]$. We define *special hyperplanes* of various granularities. For each integer i, for each axis x_j, the special hyperplanes of type-j are drawn so that consecutive hyperplanes are $N/2^i$ apart in distance (i.e. one can think of special hyperplanes of type-j as those corresponding to $x_j = kN/2^i$ for all integers k). The level-i grid cells are those regions that are bounded by consecutive special hyperplanes from level-$(i-1)$. The following lemma encapsulates the first step:

▶ **Lemma 15.** *There exists a distribution $\mathcal{D} : 2^{\mathcal{R}} \to [0, 1]$ such that each subset $\mathcal{R}'$ in its support admits a drawing of special hyperplanes with the following properties:*

- *Each cube at level-i is not intersected by hyperplanes at levels $i-1, \dots, 0$.*
- *Each level-i grid cell has at most one cube in level-i.*

Moreover, each cube $r \in \mathcal{R}$ appears in a randomly drawn subset with probability at least $\epsilon_1 = 1/2^{3d}$.

Proof. For each dimension j, we pick a random shift s_j and draw level-i special hyperplanes at $s_j + N/2^i, s_j + 2 \times N/2^i$, and so on. The probability that each level-i cube $r \in \mathcal{R}$ of edge length ℓ_r is intersected by special hyperplanes of type-$(i-1)$ is exactly $(1-\ell_r 2^{i-1}/N)^d \geq 1/2^d$ since $\ell_r \leq N/2^i$. This is the probability that each cube remains after placing the special hyperplanes.

Now, inside each cell, there can be more than one cube, and in such a case, we randomly select one of them to keep. This ensures that each cube that survives the first phase remains with probability at least $1/2^{2d}$ because there can be at most 2^{2d} cubes inside each cell (just because of the volume). This implies the lemma. ◀

Next, we consider a subset $\mathcal{R}'$ in the support of the distribution given by the above lemma. For each level-i cube $r \in \mathcal{R}$, we define $\mathsf{cell}(r)$ as the level-i grid cell that contains r. We say that two cubes r and r' are *conflicting* if either r overlaps the boundary of $\mathsf{cell}(r')$ or r' overlaps the boundary of $\mathsf{cell}(r)$.

We define the conflict graph H such that the vertex set $V(H)$ corresponds to the cubes in $\mathcal{R}'$, and there is an edge between r and r' if and only if r and r' are conflicting. The following two lemmas finish the proof.

▶ **Lemma 16.** *The graph H is $2^{O(d)}$ colorable.*

Proof. We prove this by induction on the number of vertices. The base case, when $|V(H)| = 1$, is obvious. Now consider any graph H with at least two vertices, and a cube $r \in V(H)$ whose size is minimum among the cubes in $V(H)$. Notice that $\mathsf{cell}(r)$ has $2d$ bounding hyperplanes.

Now, consider the neighborhood $N_H(r) \subset V(H)$ of r. There can be at most 2^d elements of $N_H(r)$ that contain some corner of $\mathsf{cell}(r)$. We then count those that do not contain any corner. Each cube $r' \in N_H(r)$ must be at the level below of that of r, so the overlapping volume of the intersection between r' and the bounding hyperplane of $\mathsf{cell}(r)$ must be at least $1/4^d$ fraction of the total surface volume of such bounding hyperplane. Then, each such hyperplane supports at most 4^d cubes in $N_H(r)$, and since there are $2d$ bounding hyperplanes in total, we have $2d4^d$ cubes that do not intersect any corner of $\mathsf{cell}(r)$.

This implies that $|N_H(r)| \leq 2d4^d + 2^d \leq 2d2^{3d} = 2^{O(d)}$. By induction hypothesis, we are done: We can inductively color the graph resulting from removing the vertex r from H, and since there are only $2^{O(d)}$ neighbors of r, we can assign a distinct color to r. ◀

Finally, the following lemma can be proved similarly to the 2-dimensional case.

▶ **Lemma 17.** *Let I be an independent set of H. Then I is guillotine separable.*

Proof. We argue that we can iteratively define a sequence of cuts that separate all cubes. Let P' be a piece that currently contains at least 2 cubes. Let R and R' be the largest and second largest cubes contained in P', respectively. If they are from the same level, we would be done: Pick any special hyperplane that separates the two cells $\mathsf{cell}(R)$ and $\mathsf{cell}(R')$. Otherwise, if R and R' are from levels $\ell < \ell'$, respectively, we define Q as the union of level-ℓ' cells that overlap with R. Notice that R' cannot be inside Q; otherwise, this would have contradicted the fact that they are independent in H. Now, we can pick any special hyperplane that separates Q from R'. ◀

3 Negative result for unit squares

In this section, we give a sequence of instances W_n for which no guillotine strategy separates more than a fraction of $\frac{1}{2} + o(1)$. All these instances are formed by unit squares.

Given positive integers a, and b, the *brick wall* $W(a,b) = \{S(i,j) \colon 1 \leq i \leq a, 1 \leq j \leq b\}$ is the set of unit squares such that the lower left corner of $S(i,j)$ is at coordinates $i \cdot (1,\delta) + j \cdot (-\delta, 1) = (i - j\delta, j + i\delta)$, where δ is an arbitrarily small positive constant (say, smaller than $1/(ab)$). For example, Figure 3 shows $W(5,4)$. Let $W_n = W(n,n)$.

▶ **Lemma 18.** *The number of squares of W_n that can be separated by a guillotine strategy is at most $\left\lceil \frac{n^2+2n-2}{2} \right\rceil \leq \frac{n^2}{2} + n$.*

In fact, we prove a more general version of Lemma 18. Define a *subwall* of a brick wall $W(a,b)$ as the subset of squares of $W(a,b)$ contained inside a given rectangle. For example,

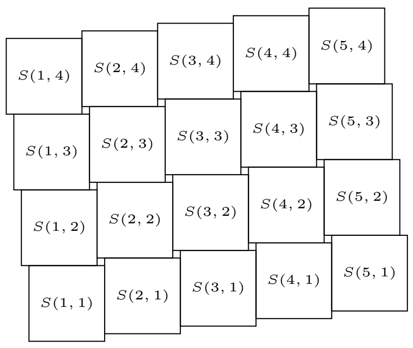


Figure 3 The grid wall $W(5,4)$.

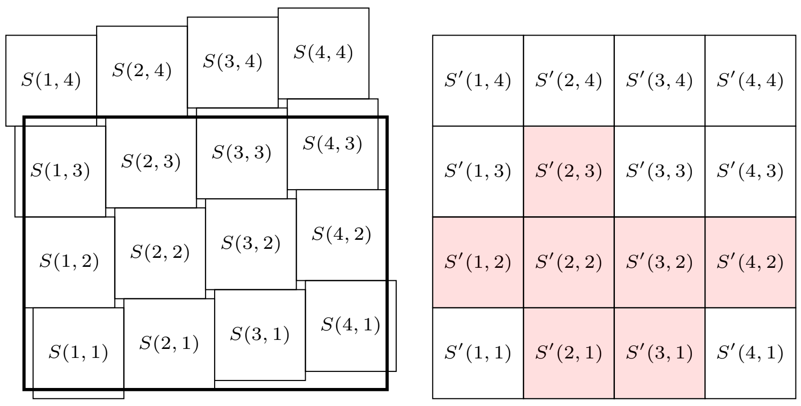


Figure 4 A subwall q of $W(4,4)$. On the right, the corresponding squares of q in the standard unit grid. The region has an area of 7 and a perimeter of 14.

in Figure 4 (left) the set $\{S(2,1), S(3,1), S(1,2), S(2,2), S(3,2), S(4,2), S(2,3)\}$ is a subwall of $W(4,4)$. It is easier to note the subwall in the "standard" square grid, as shown in Figure 4 (right). Define the *area* $A(q)$ of a subwall q as the number of squares it contains, and the *perimeter* $P(q)$ as the number of edges (i.e. sides of squares) such that only one of its incident squares is in q. The previous concepts coincide with the actual geometric area and perimeter of the union of the corresponding squares in a standard square grid. Let also $S(q)$ denote the maximum number of squares that can be separated using a guillotine cutting sequence.

▶ **Lemma 19.** *For any subwall q,*

$$S(q) \;\leq\; \left\lceil \frac{2A(q)+P(q)-4}{4} \right\rceil .$$

Proof. The proof is by induction on $A(q)$. When $A(q)=1$, $S(q)=1$ and $\lceil \frac{2A(q)+P(q)-4}{4} \rceil = \lceil \frac{2+4-4}{4} \rceil = 1$, so assume that $A(q) \geq 2$. Consider the first guillotine cut of an optimal cutting sequence for q (the one achieving $S(q)$). Without loss of generality, assume it is an horizontal cut. This cut divides q into two subwalls q_1 and q_2, both with at least one complete square inside. Let r be the number of squares cut in this first iteration. See Figure 5 for reference, where the cut squares are shown in black. It is easy to see that:

$$A(q) \;=\; A(q_1) + A(q_2) + r,$$

and

$$P(q_1) + P(q_2) - 2(r+1) \;\leq\; P(q) \;\leq\; P(q_1) + P(q_2) - 2r.$$

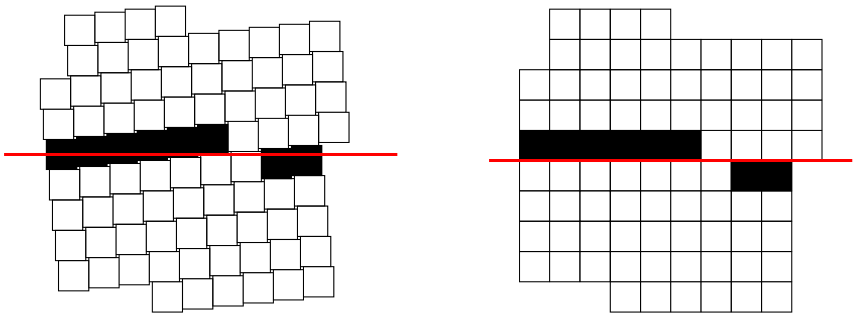


Figure 5 An horizontal guillotine cut.

Note that $P(q)$ attains the lower bound if the horizontal cut goes exactly through a square edge (like in Figure 5). Otherwise, an extra square is destroyed and $P(q)$ attains the upper bound. Using the previous observation we have that

$$\begin{aligned}
S(q) &= S(q_1) + S(q_2) && \text{(by optimality)}\\
&\le \left\lceil \frac{2A(q_1)+P(q_1)-4}{4} \right\rceil + \left\lceil \frac{2A(q_1)+P(q_1)-4}{4} \right\rceil && \text{(inductive step)}\\
&\le \left\lceil \frac{(2A(q_1)+2A(q_2))+(P(q_1)+P(q_2))-6}{4} \right\rceil && (\star)\\
&\le \left\lceil \frac{(2A(q)-2r)+(P(q)+2r+2)-6}{4} \right\rceil\\
&= \left\lceil \frac{2A(q)+P(q)-4}{4} \right\rceil .
\end{aligned}$$

To conclude the proof we need to check the validity of inequality $(\star)$. Note first that the perimeter of any subwall is always even (because it can be computed as the length of a closed path on the integer grid). Using this we only need to prove, for every pair of even integers x and y, that

$$\lceil x/4 \rceil + \lceil y/4 \rceil \ \le\ \lceil (x+y+2)/4 \rceil,$$

or equivalently, for every pair of integers a and b, that

$$\lceil a/2 \rceil + \lceil b/2 \rceil \ \le\ \lceil (a+b+1)/2 \rceil.$$

This is direct: if both a and b are even, then the left hand side (LHS) is $(a+b)/2$ and the right hand side (RHS) is $(a+b)/2+1$. If only one of them is even, then both the LHS and the RHS are equal to $(a+b+1)/2$. If both numbers are odd, then both the LHS and RHS are equal to $(a+b+2)/2$. ◀

Lemma 18 follows immediately from Lemma 19.

4 Proof of Theorem 2

We assume that Conjecture 1.1 is true, and present an $O(1)$-approximation algorithm for MISR. In our reasoning here, we assume that the given rectangles are open sets. Suppose we are given a set of axis-parallel rectangles $\mathcal{R}$. Since we consider the cardinality case in MISR, we can assume w.l.o.g. that the input does not contain any two rectangles R, R' such that

$R \subseteq R'$. The algorithm is essentially the algorithm GEO-DP from [1], when parametrized by $k = 4$. First, we can assume w.l.o.g. that all rectangles in $\mathcal{R}$ have integer coordinates in the range $\{0, ..., 2n\}$. This can be achieved easily by suitable stretching of the instance. Our algorithm is a dynamic program (DP) with a cell for every open rectangle $P \subseteq [0, 2n] \times [0, 2n]$ whose corners have integer coordinates. Note that there are $O(n^4)$ many of them. Each of these cells represents the problem of selecting the optimal Independent Set that consists only of rectangles that lie completely within P, i.e., rectangles R with $R \subseteq P$. Intuitively, the DP stores a near-optimal solution to this subproblem in the respective cell.

When computing the entry for such a cell, representing a rectangle P, the DP does the following procedure. In the case where the rectangle P does not (completely) contain any input rectangle, we define the empty set to be the solution corresponding to the cell. Also, in the case where P coincides with an input rectangle R, i.e. $P = R$, then from our assumption above we know that there is no rectangle R' with $R \neq R'$ such that $R' \subseteq P$ and therefore, we define $\{R\}$ to be the solution corresponding to the cell. Otherwise, it tries all possibilities of dividing P into two smaller pieces using a horizontal or vertical guillotine cut such that the horizontal/vertical coordinate of this cut is an integer (since the rectangles have integral coordinates we can safely restrict ourselves to those). Consider one such cut and let $P_1 \neq \emptyset \neq P_2$ denote the resulting pieces. The DP looks up the solutions for the cells representing P_1 and P_2 and combines them to a solution for P. It selects the cut yielding the optimal total profit from the resulting two subproblems. Since there are $O(n)$ possible cuts for each rectangle P, it takes $O(n)$ time to compute the solution for a given cell. Finally, we output the solution that the DP computes for the cell corresponding to the rectangle $[0, 2n] \times [0, 2n]$. Since there are $O(n^4)$ cells in total, we obtain a total running time of $O(n^5)$.

To analyze this algorithm, we first show that it finds the largest subset of the input rectangles that is guillotine separable. This can be easily shown by induction over the cells, i.e., for each cell the algorithm computes a solution that is at least as profitable as the best guillotine separable solution consisting only of rectangles completely contained in the rectangle that the cell represents.

▶ **Lemma 20.** *Let $\mathcal{R}' \subseteq \mathcal{R}$ be a largest subset of $\mathcal{R}$ that is guillotine separable. Then, the DP finds a set of size $|\mathcal{R}'|$.*

If now Conjecture 1.1 holds, then $|\mathcal{R}'| \geq \Omega(|OPT|)$, where OPT denotes the optimal solution to the given MISR instance. Hence, the DP finds a solution with at least $\Omega(|OPT|)$ rectangles and it is hence a $O(1)$-approximation algorithm for MISR. This completes the proof of Theorem 2.

We would like to note that the resulting algorithm is purely combinatorial and that it is faster than solving the natural LP-relaxation of the problem (which is a natural candidate for obtaining an $O(1)$-approximation algorithm for MISR).

5 QPTAS for two-dimensional geometric knapsack

In this section, we present a QPTAS for the unweighted two-dimensional guillotine knapsack problem. It builds on the recent QPTAS for the problem without the guillotine constraint [2] but requires substantial new ideas to handle this constraint. Like the latter QPTAS we require all input data to be quasi-polynomially bounded integers.

Let $\epsilon > 0$ and suppose that the knapsack has a capacity of $N \times N$ with $N \in \mathbb{N}$. For each item i denote by h_i and w_i its height and width, respectively. By standard shifting arguments we can assume that there are values $\mu, \delta > 0$ such that for each item i we have that $h_i \in [0, \mu \cdot N] \cup [\delta \cdot N, N]$ and $w_i \in [0, \mu \cdot N) \cup [\delta \cdot N, N]$, while losing only factor $1+\epsilon$ in the approximation ratio. The values μ, δ can be chosen such that $1 \geq \delta > \mu \geq (\frac{\epsilon}{\log n})^{O_\epsilon(1)}$ and $\mu = (\delta \cdot \frac{\epsilon}{\log n})^{O_\epsilon(1)}$ (see Lemma 1 in [2]). Moreover, since we study the unweighted case of the problem and allow quasi-polynomial running time, it is sufficient to compute a solution with a set of items I' such that $|I'| \geq (1-\epsilon)|OPT| - (\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ (see Proposition 2.1 in [2]). We will call such solutions *near-optimal* solutions in the sequel. Using this property, we assume from now on that there are no items i with $h_i \geq \delta \cdot N$ and $w_i \geq \delta \cdot N$ since there can be only $1/\delta^2 = O_\epsilon(\log n)^{O_\epsilon(1)}$ of them in the optimal solution OPT. We partition the remaining items into horizontal, vertical, and tiny items, given by sets H, V, and T, respectively. An item i is *horizontal* if $w_i \geq \delta \cdot N$ and $h_i \leq \mu \cdot N$, *vertical* if $w_i \leq \mu \cdot N$ and $h_i \geq \delta \cdot N$, and *tiny* if $w_i \leq \mu \cdot N$ and $h_i \leq \mu \cdot N$.

Box partition. The key ingredient in the QPTAS in [2] is a partition of the knapsack into boxes which intuitively describe the topology of the optimal solution. More precisely, it is shown that if $OPT \cap T = \emptyset$ then there exists a partition of the knapsack into a set $\mathcal{B}$ of at most $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ axis-parallel rectangular boxes and a near-optimal solution OPT' such that each box $B \in \mathcal{B}$ either contains only items from $OPT' \cap H$ or only items from $OPT' \cap V$ (the tiny items are added later into the remaining empty space). By being more careful in the construction, we can prove the following stronger statement which also takes our guillotine cutting constraint into account. Note that the following lemma is non-constructive, so our algorithm has to guess the partition given by it.

▶ **Lemma 21.** *There is a partition of the $N \times N$ knapsack into at most $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ rectangular boxes $\mathcal{B}$ with integral coordinates and a near-optimal solution $OPT' \subseteq OPT$ with the following properties:*

- *each item in OPT' is fully contained in some box $B \in \mathcal{B}$,*
- *each box $B \in \mathcal{B}$ is either a horizontal box which contains only items in $OPT' \cap (H \cup T)$ or a vertical box which contains only items in $OPT' \cap (V \cup T)$,*
- *OPT' is constructed by removing some items from OPT and moving the remaining items within the knapsack so that no horizontal item is moved to the left or right and no vertical items is moved up or down,*
- *for each box $B \in \mathcal{B}$ there exists a guillotine cutting sequence which cuts all items of OPT' that are contained in B.*

Proof. We start with the solution OPT that is the optimal solution satisfying the guillotine cut constraint. We use the same construction as in the proof of Lemma 2 in [2]. The proof of the lemma follows. Note that the last property is satisfied since OPT satisfies the guillotine cuts constraint by assumption, and after the modifications in the proof of Lemma 2 in [2] the property is still true for each box $B \in \mathcal{B}$. ◀

Guillotine cutting sequence. Using Lemma 21 we construct a guillotine cutting sequence which intersects the items of OPT' in a very controlled way, as given by the following lemma.

▶ **Lemma 22.** *There exists a guillotine cutting sequence with at most $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ cuts, cutting the knapsack into a set $\mathcal{B}'$ of rectangular pieces, such that the following properties hold:*

- *for each piece $B' \in \mathcal{B}'$ there is a box $B \in \mathcal{B}$ such that $B' \subseteq B$,*
- *at most $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ items from $OPT' \cap (H \cup V)$ are intersected,*
- *the intersected items from $OPT' \cap T$ have a total area of $\leq (\frac{\epsilon}{\log n})^{O_\epsilon(1)} \cdot N^2$.*

The intersected items from $OPT' \cap (H \cup V)$ will be lost which is justified since we are only interested in a near-optimal solution. For the intersected tiny items, we will argue that we can free up some space in the boxes in $\mathcal{B}'$ at only marginal cost to accommodate them.

We construct our guillotine cutting sequence now. Consider the first cut of the cutting sequence of OPT (which does not intersect any item of OPT!) and assume w.l.o.g. that it is a horizontal cut given by the line segment $[0, N] \times \{h\}$. By Lemma 21 we can show that if we used exactly the same cut in OPT' then we would not intersect any item in $OPT' \cap V$. Denote by h_1 the smallest integer such that $h_1 \geq h$ and either $[0, N] \times \{h_1 + 1\}$ split an item in $OPT' \cap V$, or there is a box $B \in \mathcal{B}$ such that $[0, N] \times \{h_1\}$ cuts along the top edge of B. Similarly, denote by h_2 the largest integer such that $h_2 \leq h$ and either $[0, N] \times \{h_2 - 1\}$ split an item in $OPT' \cap V$, or there is a box $B' \in \mathcal{B}$ such that $[0, N] \times \{h_2\}$ cuts along the bottom edge of B'. Our first two cuts are $[0, N] \times \{h_1\}$ and $[0, N] \times \{h_2\}$. We then cut the resulting piece between the two cuts by vertical cuts, such that we obtain smaller pieces having non-empty intersection with at most one box in $\mathcal{B}$. We continue iteratively on the other resulting pieces. Consider the piece $B_1 := [0, N] \times [0, h_1]$. Observe that it is contained in the piece $B_0 := [0, N] \times [0, h]$ which is the first piece that the optimal cutting sequence obtains. For cutting B_1 further, we consider the next cut on B_0 of the optimal cutting sequence which also cuts B_1 (i.e., we ignore cuts of the form $[0, N] \times \{\bar{h}\}$ with $h_1 < \bar{h} < h$). Suppose that it is a vertical cut $\{h'\} \times [0, h]$. Similarly as before, we denote by h'_1 the smallest integer such that $h'_1 \geq h'$ and either $\{h'_1 + 1\} \times [0, h_1]$ split an item in $OPT' \cap H$, or there is a box $B \in \mathcal{B}$ such that $\{h'_1\} \times [0, h_1]$ cuts along the right edge of B. We define h'_2 accordingly. Our next two cuts for B_1 are $\{h'_1\} \times [0, h_1]$ and $\{h'_2\} \times [0, h_1]$. We continue in the same manner till we end up with the set $\mathcal{B}'$.

For each box in $B \in \mathcal{B}$ we bound the number of cuts that go through B. There are two types of these cuts. Cuts of the first type contains the corner of some box in $\mathcal{B}$. Thus, the number of these cuts is at most $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$. For the other type of cuts we can give a charging scheme such that each box pays for at most $O(1/\delta)$ of them. Thus, the number of boxes in $\mathcal{B}'$ is bounded by $O(1/\delta) \cdot (\frac{\log n}{\epsilon})^{O_\epsilon(1)} \cdot |\mathcal{B}| \leq (\frac{\log n}{\epsilon})^{O_\epsilon(1)}$. Therefore, the total number of cuts is bounded by the same value. Each cut intersects at most $1/\delta = (\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ horizontal or vertical items and tiny items with a total area of at most $N \cdot \mu N \leq (\frac{\epsilon}{\log n})^{O_\epsilon(1)} \cdot N^2$.

Adding back tiny items. In the above process, we intersected some tiny items (with small total area). We do not want to lose them so we add them back now. To this end, we drop some of the remaining items and create an empty space in each box $B' \in \mathcal{B}'$. Suppose that B' contains only items from $OPT' \cap (H \cup T)$ and has height $h_{B'}$. Denote by $OPT'_{B'}$ the items of OPT' inside B'. We identify a horizontal slice of height $\epsilon \cdot h_{B'}$ inside B' such that there are at most $O(\epsilon) \cdot |OPT'_{B'}| + O(\frac{1}{\delta})$ items in $OPT'_{B'}$ that intersect this slice. We drop all these items. By doing this with each box $B' \in \mathcal{B}'$ we can show that this creates enough empty space to put back almost all tiny items that were intersected by our cutting sequence above. Even more, by assigning them into the empty space by the Next-Fit-Decreasing-Height (NFDH) algorithm [6] we can ensure that also a guillotine cutting sequence for them exists. Overall, we obtain the property that for the items in each box $B' \in \mathcal{B}'$ there exists a guillotine cutting sequence.

Details of this operation are the following: Observe that the number of guillotine cuts that separate the boxes in $\mathcal{B}'$ is exactly $|\mathcal{B}'| - 1$. Every cut intersects tiny items with total area at most μN^2. The total area of the tiny items intersected in all cuts is $\mu N^2 |\mathcal{B}'|$ which we can upper-bound by $\epsilon^3 N^2$ by choosing μ and δ such that $\mu |\mathcal{B}'| \leq \epsilon^3$. We call a box $B' \in \mathcal{B}'$ a *good box* if $h(B') \geq \frac{\mu N}{\epsilon^2}$ and $w(B') \geq \frac{\mu N}{\epsilon^2}$. Otherwise the box is called a *bad box*.

The total area of bad boxes is at most $(\frac{\mu N}{\epsilon^2})N|\mathcal{B}'|$. Recall that $\mu|\mathcal{B}'| \leq \epsilon^3$. This implies that the total area of bad boxes is at most ϵN^2. This means that the total area of good boxes is at least $(1-\epsilon)N^2$. In good boxes, we create empty space to accommodate the tiny items whose total area is at least $(\epsilon - \epsilon^2)N^2$. Since we use the NFDH algorithm to accommodate the tiny items, we only lose a constant fraction of the area [6]. Note that by definition any tiny item fits into the created empty space of a good box. This implies that we are able to accommodate all tiny items. We note that using the NFDH algorithm we can ensure that a guillotine cutting sequence for the tiny items exist.

Rounding of item sizes. As the last step of the non-constructive part, we round the sizes of the items such that at the end we have only $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ many items types. We say that two items i, i' are of the same type if $h_i = h_{i'}$ and $w_i = w_{i'}$. We round the item sizes in each box in $\mathcal{B}'$ separately. Like above, we create empty space of height $\epsilon \cdot h_{B'}$ inside each horizontal box $B' \in \mathcal{B}'$ while dropping only few items, at most $\epsilon|OPT'| + (\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ many (and we do a similar adjustment for the vertical boxes). We use this empty space to round up the height of each item to the next larger power of $1+\epsilon$, yielding $O_\epsilon(\log n)^{O(1)}$ many height classes (we use here that the input data are quasi-polynomially bounded integers). Using harmonic grouping like in De La Vega and Lueker [8] we round the widths of the items such that among each height class there are only $O_\epsilon(1)$ many widths arising, yielding $O_\epsilon(\log n)^{O(1)}$ different items types in total.

Algorithm. The algorithm first guesses the cutting sequence according to Lemma 22 for which there are only $n^{(\frac{\log n}{\epsilon})^{O_\epsilon(1)}}$ many options. Then, it guesses the item types that arise after the previous rounding. Note here that the height and width of each item type is either a power of $1+\epsilon$ or coincides with the height or width of an input item, and thus we have to choose $(\frac{\log n}{\epsilon})^{O_\epsilon(1)}$ types from $n^{O(1)}$ possible ones. So there are only $n^{(\frac{\log n}{\epsilon})^{O_\epsilon(1)}}$ many options for this in total. For each item type we guess how many items there are in the solution that we constructed above. Then we verify that our input items are consistent with this guess which can be done by finding a perfect bipartite matching. In this matching we have a node for every input item, and a set of nodes for each item type. For each item type the number of nodes equals the guessed number of items of this type in the searched-for solution. An edge between a node for an input item and a node for a guessed item of a type is added if the input item can be drawn inside the guessed item. In this case finding a perfect matching means that the guess is consistent with the input items, otherwise the guess is rejected. For each box $B' \in \mathcal{B}'$ we guess how many items of each type we have to assign in the box such that for them there is a guillotine cutting sequence. It remains to verify for each box $B' \in \mathcal{B}'$ that the items we guessed to be assigned to it actually fit into B'. To this end, we use a dynamic program. It guesses the first cut of the (existent) cutting sequence of the items and then guesses how we have to partition the items to the two sides of the cut. Then we recurse on both sides. Each arising subproblem is specified by a remaining rectangular piece and a set of items that is to be assigned to it. Since for both quantities together there are only $n^{(\frac{\log n}{\epsilon})^{O_\epsilon(1)}}$ many options, also this dynamic program runs in quasi-polynomial time. For the version of the problem where we are allowed to rotate items by 90 degrees the above methods can be adjusted easily. This completes the proof of Theorem 3.

Acknowledgements. J. Correa, P. Pérez-Lantero, and J. Soto are supported by Millennium Nucleus Information and Coordination in Networks ICM/FIC RC130003 (Chile). J. Soto is additionally supported by FONDECYT Grant 11130266. A. Karrenbauer is supported by the Max Planck Center for Visual Computing and Communication (`http://www.mpc-vcc.org`).

References

1 Anna Adamaszek and Andreas Wiese. Approximation schemes for maximum weight independent set of rectangles. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*, pages 400–409. IEEE, 2013.

2 Anna Adamaszek and Andreas Wiese. A quasi-PTAS for the two-dimensional geometric knapsack problem. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, pages 1491–1505, 2015.

3 Nikhil Bansal, Alberto Caprara, Klaus Jansen, Lars Prädel, and Maxim Sviridenko. A structural lemma in 2-dimensional packing, and its implications on approximability. In *Algorithms and Computation (ISAAC 2009)*, volume 5878 of *LNCS*, pages 77–86. Springer, 2009.

4 Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009)*, pages 892–901. SIAM, 2009.

5 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48:373–392, 2012.

6 Edward G Coffman, Jr, Michael R Garey, David S Johnson, and Robert Endre Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM J. Comput.*, 9(4):808–826, 1980.

7 Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM Journal on Computing*, 34(6):1302–1323, 2005.

8 Wenceslas Fernandez de la Vega and George S. Lueker. Bin packing can be solved within $1 + \varepsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

9 Aleksei V. Fishkin, Olga Gerber, Klaus Jansen, and Roberto Solis-Oba. Packing weighted rectangles into a square. In *Mathematical Foundations of Computer Science (MFCS 2005)*, volume 2337 of *LNCS*, pages 352–363. Springer, 2005.

10 Rolf Harren. Approximating the orthogonal knapsack problem for hypercubes. In *Automata, Languages and Programming (ICALP 2006)*, volume 4051 of *LNCS*, pages 238–249. Springer, 2006.

11 Klaus Jansen and Roberto Solis-Oba. New approximability results for 2-dimensional packing problems. In *Mathematical Foundations of Computer Science (MFCS 2007)*, volume 4708 of *LNCS*, pages 103–114. Springer, 2007.

12 Klaus Jansen and Roberto Solis-Oba. A polynomial time approximation scheme for the square packing problem. In *Proceedings of the 13th Integer Programming and Combinatorial Optimization Conference (IPCO 2008)*, pages 184–198. Springer, 2008.

13 Klaus Jansen and Guochuan Zhang. Maximizing the number of packed rectangles. In *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory (SWAT 2004)*, pages 362–371. Springer, 2004.

14 Klaus Jansen and Guochuan Zhang. Maximizing the total profit of rectangles packed into a rectangle. *Algorithmica*, 47(3):323–342, 2007.

15 János Pach and Gábor Tardos. Cutting glass. In *Proceedings of the 16th Annual Symposium on Computational Geometry (SOCG 2000)*, pages 360–369. ACM, 2000.

16 Jorge Urrutia. Problem presented at ACCOTA'96: Combinatorial and Computational Aspects of Optimization, Topology, and Algebra, Taxco, Mexico, 1996.

Approximate Nearest Neighbor Search in Metrics of Planar Graphs

Ittai Abraham[1], **Shiri Chechik**[2], **Robert Krauthgamer**[*3], **and Udi Wieder**[3]

1 **VMware Research, Palo Alto, CA, USA,** `{iabraham,uwieder}@vmware.com`
2 **Tel-Aviv University, Tel-Aviv, Israel,** `shiri.chechik@gmail.com`
3 **Weizmann Institute of Science, Israel,** `robert.krauthgamer@weizmann.ac.il`

Abstract

We investigate the problem of approximate Nearest-Neighbor Search (NNS) in graphical metrics: The task is to preprocess an edge-weighted graph $G = (V, E)$ on m vertices and a small "dataset" $D \subset V$ of size $n \ll m$, so that given a query point $q \in V$, one can quickly approximate $\mathbf{d_G}(q, D)$ (the distance from q to its closest vertex in D) and find a vertex $a \in D$ within this approximated distance. We assume the query algorithm has access to a distance oracle, that quickly evaluates the exact distance between any pair of vertices.

For planar graphs G with maximum degree Δ, we show how to efficiently construct a compact data structure – of size $\tilde{O}(n(\Delta + 1/\epsilon))$ – that answers $(1 + \epsilon)$-NNS queries in time $\tilde{O}(\Delta + 1/\epsilon)$. Thus, as far as NNS applications are concerned, metrics derived from bounded-degree planar graphs behave as low-dimensional metrics, even though planar metrics do not necessarily have a low doubling dimension, nor can they be embedded with low distortion into ℓ_2. We complement our algorithmic result by lower bounds showing that the access to an exact distance oracle (rather than an approximate one) and the dependency on Δ (in query time) are both essential.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Data Structures, Nearest Neighbor Search, Planar Graphs, Planar Metrics, Planar Separator

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.20

1 Introduction

In the Nearest Neighbor Search (NNS) problem, the input is a dataset $D = \{p_1, \dots, p_n\}$ containing n points that lie in some large host metric space (M, d). These points should be preprocessed into a data structure so that given a query point $q \in M$, the dataset point $p_i \in D$ closest to q can be reported quickly. The scheme's efficiency is typically measured by the space complexity of the data structure and the time complexity of the query algorithm. NNS is a fundamental problem with numerous applications, and has therefore attracted a lot of attention, including extensive experimental and theoretical analyses. Often, finding the exact closest neighbor is relaxed to finding an approximate solution, called $(1+\epsilon)$-NNS, where the goal is to report a dataset point $p_i \in D$ satisfying $d(q, p_i) \leq (1+\epsilon) \min\{d(q, p_j) \mid p_j \in D\}$.

Previous work on NNS has largely focused on the case where the host metric is ℓ_p-norm for some p, typically ℓ_1 or ℓ_2, over $M = \mathbb{R}^m$ for some dimension $m > 0$. In this common setting, exact NNS exhibits a "curse of dimensionality" – either the space (storage requirement) or

[*] Work supported in part by a US-Israel BSF grant #2010418 and an Israel Science Foundation grant #897/13.

the query time must be exponential in the dimension – which provides a strong motivation to study approximate solutions. When the dimension m is constant, known $(1+\epsilon)$-NNS algorithms achieve almost linear space and a polylogarithmic query time; but when the dimension is logarithmic (in n), all known algorithms, including approximate ones, require (in the worst-case) either a super-linear space or a polynomial query time.

While the ℓ_p-norm setting captures many applications, certain data types cannot be embedded (with low distortion) into $\mathbb{R}^m$, and it is therefore desirable to consider other metric spaces. However, the notion of dimensionality is not well-defined for general metrics, and it is unclear a-priori what type of internal structure suffices for efficient approximate NNS algorithms. A notable example of such an approach, initiated by [22, 17, 18], is the study of NNS in general metric spaces assuming the algorithm has access to a distance oracle, i.e., that the distance function can be evaluated in unit time; their motivation was drawn from the analysis of computer networks, where distances are derived from a huge graph (rather than by a norm or another simple function of the vertex names). It is known [18, 19, 8, 15, 12] that if the metric space restricted to the n dataset points has a bounded doubling dimension, then $(1+\epsilon)$-NNS can be solved with near-linear space and polylogarithmic query time.

1.1 Our Results

We look at another family of metric spaces, those derived by way of shortest paths from a graph with positive edge weights. Similarly to [22, 17, 18], we assume the NNS algorithm has access to a distance oracle.

As a motivating application consider the case where the graph represents a road network, say of the continental United States. Even though this graph has tens of millions of nodes, extremely efficient exact distance oracles have been built for it (see [14, 6, 2]). Now suppose we wish to find the nearest shop from a collection like the set of all Starbucks shops, which currently has roughly 12,000 locations in the US. This means we want to design a compact application (e.g., mobile app) that has access to a generic server (like google maps). While the server could use much larger space (but cannot be customized to support specialized operations like NNS), the application must be very efficient in terms of query time and space, and thus our goal is to build an NNS data structure whose efficiency depends on the significantly smaller number of shops ($n = |D|$ in our notation).

Our main result is that in planar graphs of bounded degree, $(1+\epsilon)$-NNS can be solved using near-linear space and polylogarithmic query time. Thus, bounded-degree planar metrics exhibit "low-dimensional" behavior, even though they do not necessarily have a low doubling dimension, nor can they be embedded with low distortion into ℓ_2. This phenomenon, namely, that the restricted topology of planar graphs maintains some of the geometric structure of the Euclidean plane, is known in other contexts like compact routing and TSP, but here we show it for the first time in the context of NNS.

▶ **Theorem 1.** *Let (M, d) be a metric derived from a plane graph of maximum degree $\Delta > 0$ with positive edge weights, and let $\epsilon > 0$. Then every dataset $D \subset M$ of size $n = |D|$ can be preprocessed into a data structure of size $O(\epsilon^{-1} n \log n \log |M| + n\Delta \log^2 n)$ words, which can answer $(1+\epsilon)$-NNS queries in time $O((\epsilon^{-1} \log\log n + t_{DO}) \log n \log |M| + \log n \cdot \Delta t_{DO})$, assuming the distance between any two points in M can be computed in time t_{DO}.*

The data structure's size is measured in words, where a single word can accommodate a point in M or a (numerical) distance value. The term plane graph refers to a planar graph accompanied with a specific drawing in the plane.

Theorem 1 makes two assumptions, that there is a bound on the maximum degree, and that the data structure has access to an exact distance oracle. We further show (in Section 4) that both of these assumptions are necessary. Roughly speaking, we prove that if the degree is large, or if the distance oracle is approximate, the graph could contain symmetries that only a large number of accesses to the distance oracle could break.

▶ Remark. We cannot dispose of the maximum degree assumption by "vertex splitting", where a high-degree vertex is replaced with a binary tree with zero edge weights, because we assume access to a distance oracle for G (but not necessarily for a modified graph G'), since the distance oracle models a generic server. For the same reason, we cannot make the usual assumptions that G is triangulated or perturb the pairwise distances to be all distinct.

1.2 Related Work

Our model of NNS for a dataset D embedded inside a (huge) graph G is related to vertex-sparsification of distances [20], where the goal is to construct a small graph G' that (i) contains all the dataset point D (called terminals here) and furthermore maintains all their pairwise distances; and (ii) is isomorphic to a minor of G.

Here is another interesting related problem that is open: Given only the distances between a dataset D, find in polynomial time a *planar* host graph G that contains D and realizes their given pairwise distances (perhaps even approximately).

A key difference of our NNS model from these two problems is that the vertices outside of D are actually used explicitly as query points. However, one may hope for some connections, at least at a technical level.

1.3 Techniques

At a very high level, our algorithm is reminiscent of the classical k-d tree algorithm for NNS due to Bentley [7], as it partitions the graph recursively using separators that split the current dataset in an approximately balanced manner. While k-d trees use hyperplanes as separators of the host space, for planar metrics we use shortest-paths as separators (see [23, 3, 1, 13]). This recursive partitioning process can be described as creating a "hierarchy" tree $\mathcal{T}$, whose nodes correspond to "regions" in the metric space, and every leaf node represents a region with at most one dataset point; in our graphical case, every region is an induced subgraph of G. Given a query point q, one often uses a top-down algorithm to identify the leaf node in the hierarchy tree $\mathcal{T}$ that "contains" q, by tracing the location of q along the recursive partitioning, a process that we call the "zoom-in" phase. While this phase is trivial in k-d trees, and simple in bounded treewidth graphs (see Section 1.4), it is quite non-trivial in planar graphs, as explained below.

The key observation that completes the k-d tree algorithm is that once the tree leaf node containing q has been located, the nearest neighbor of q must lie "near the boundary" of one of the regions along the tree path leading to this leaf. Thus, all we need to store is just the separators themselves and the dataset points near them.

In planar graphs, this master plan has two serious technical difficulties. First, the separators themselves are too large to be stored explicitly. Recalling that the separators consist of shortest paths, we can employ the known trick [23, 3, 1, 13] of using a carefully chosen "net" to store them within reasonable accuracy, but since we actually need a net of all nearby dataset points, our solution is more involved and roughly uses a net of nets. Second, tracing the path to q along the hierarchy tree is a major technical challenge because our storage is proportional to $n = |D|$, while the separator size could be much larger, even linear in $|M|$. Our solution is to store just enough auxiliary information to identify at each level of the tree a few (rather than one) potential nodes, which suffices to "zoom-in" towards a small set of leaves that one of them contains q. The construction is presented in Section 2.

As a warm-up to the main result, we demonstrate our approach on the much simpler case of graphs with a bounded treewidth, where our algorithm solves NNS *exactly* using a standard tool of vertex separators of bounded size. This is only an initial example how NNS algorithms can leverage topological information, and the case of planar graphs is considerably more difficult — our algorithm uses a path separator, whose size is not bounded, and consequently solves NNS *approximately* (within factor $1+\epsilon$).

1.4 Warmup: Bounded Treewidth Graphs

▶ **Theorem 2.** *Let D be a dataset of n points in a metric (V, d) derived from an edge-weighted graph of treewidth $w \geq 1$ and maximum degree $\Delta > 0$. Then D can be preprocessed into a data structure of size $O(\Delta wn)$ words, which can answer (exactly) nearest neighbor search queries in time $O(\Delta w \log n \cdot t_{DO})$, assuming the distance between any two points in V can be computed in t_{DO} steps.*

We assume for simplicity of exposition that an optimal tree decomposition is given to us; otherwise, it is possible to compute in polynomial time a tree decomposition of width $O(w \log w)$ [5] (or for fixed w, one of the algorithms of [9, 10]).

We use the following well-known property of bounded treewidth graphs: Given a set $X \subset V$, we can efficiently find a *separator* $S \subset V$ of size $|S| \leq w+1$ whose removal breaks G into connected components $V_1, V_2, \ldots$ such that $|V_i \cap X| \leq |X|/2$ for all i. (The separator can be found by picking a single suitable node in the tree decomposition, and the width bound implies the bound on the separator size, see e.g. [11, Lemma 6].) It follows that in G, every path from V_i to V_j for $i \neq j$, must intersect the separator S.

The preprocessing phase

Given a dataset $D = \{p_1, \ldots, p_n\}$, recursively compute a partition (using the above property) with respect to the dataset points in the current component (i.e., $D \cap V'$ where $V' \subseteq V$ denotes the current component), until no dataset points are left (they were all absorbed in the separators). It is easy to see that the depth of the recursion tree is $O(\log n)$, and the number of separators used (non-leaf nodes in the recursion tree) is at most $O(n)$, each with at most $w+1$ nodes. The data structure stores all the separators explicitly arranged in the form of their recursion tree. In addition, for each vertex u in any of these separators, it stores the following meta-data:

- All the neighbors (at most Δ) of u, along with the index of the part V_i to which they belong.
- A dataset point that is closest to u, i.e., $\text{argmin}_{p \in D} d(p,u)$, breaking ties arbitrarily.

The total number of vertices in all the separators is at most $O(nw)$ and the meta-data held for every separator vertex u is of size $O(\Delta)$, hence the total size is $O(\Delta wn)$ words.

The query phase

We first argue that given a query point $q \in V$ and a separator S, it is possible to check, using the meta-data stored in the preprocessing phase, which part V_i contains q. This would imply that we can trace the path along the recursion tree all the way down to the last component containing q, a process that was mentioned before as the "zoom-in" phase. Indeed, finding this V_i is done by finding a vertex $u \in S$ that is closest to q, this task takes at most $|S| = w+1$ distance queries. If $u = q$ we are done. Otherwise, compute u's neighbor that is closest to q, namely, $v = \operatorname{argmin}\{d(q,v) \mid (v,u) \in E\}$, which takes another Δ distance queries. Observe that q must lie in the same part V_i as v, because the shortest between these two vertices does not intersect S.

The next step after the zoom-in process is to find the nearest neighbor itself. Let S' be the union of all the vertices in all the separators encountered during the zoom-in on q. It is easy to verify the following two facts:

- $|S'| \leq O(w \log n)$.
- There exists $u \in S'$ that lies on the shortest-path between q and its nearest neighbor in the dataset D. So q and u have the same nearest neighbor.

Recall that the vertices in S' along with all their nearest neighbors are stored explicitly in the data structure, and they can be "compared" against q using a distance oracle. Hence, it is possible to find q's *exact* nearest neighbor in time $O(\Delta w \log n)$, assuming access to a distance oracle. This proves Theorem 2.

▶ Remark. Both the "zoom-in" phase and the calculation of the nearest neighbor itself were made easy by the fact that we stored all the separators explicitly in the data structure. Planar graphs on m nodes have separators of size $O(\sqrt{m})$, but in our model $m \gg n$, and thus storing the separators explicitly is prohibitively expensive.

2 Planar Graph Metrics

In this section we start proving Theorem 1. Let $G = (M, E)$ be a connected planar graph with positive edge weights $\omega : E \to \mathbb{R}_+$, and let $D \subseteq M$ be the dataset vertices. We denote $n = |D|$, $m = |M|$, and Δ is the maximum degree in G. Assume the minimal edge weight is 1, and let $Diam$ be the diameter of the graph. We use $\mathbf{d_G}(\cdot,\cdot)$ to denote the shortest path distance in G, and assume it can be computed in time t_{DO} e.g., by having access to a distance oracle.

We shall start with the case where shortest paths in G are unique, as it simplifies technical matters considerably. A common workaround to this uniqueness issue is to perturb the edge weights, but this solution is *not applicable* in our model of a black-box access to the distance function $\mathbf{d_G}(\cdot,\cdot)$, because its implementation could potentially exploit ties (e.g., by assuming all distances are small integers). The general case is sketched in Section 3.

It is worth pointing out that our algorithm relies on machinery developed in [1] to recursively partition a planar graph G, which relies in turn on a two-path planar separator. Unlike many algorithms for planar graphs, which use existence of *small* separators, this machinery, described in detail in Section 2.1, uses the fact that the separators are shortest paths, and therefore could be represented succinctly, even if they are large. While this idea had been used before, applying it in the context of NNS (and providing matching lower bounds) requires a considerable amount of technical novelty, as described in Section 2.2.

2.1 Building the Hierarchy tree $\mathcal{T}$

Preprocessing algorithm, step 1. The algorithm fixes some vertex $s \in M$. It then constructs a shortest-path tree T rooted at $s \in M$ by invoking Dijkstra's single-source shortest-path algorithm from s.

Two-path planar separator. We use a version of the well-known Planar Separator Theorem by Lipton and Tarjan [21], where the separator consists of two paths in the shortest-paths tree T of G. In the specific version stated below, we are only required to separate a subset $U \subset M$, and the balance constraint refers to a subset $W \subset U$. As usual, $G[U]$ denotes the subgraph of G induced on $U \subset M(G)$. We apply this theorem recursively using the same shortest-paths tree T. An additional concern for us is that this tree is too large to be stored entirely (it spans all nodes M), hence our algorithm will store partial information that suffices to efficiently perform the Zoom-In and Estimating the Distance operations.

Given a connected planar graph G, we assume throughout it is a plane graph, i.e., accompanied by a specific drawing in the plane, and that it is already triangulated. We let the new edges introduced by the triangulation have infinite weight (hence they do not participate in any shortest-path). While the triangulation operation may increase the maximum degree, it will not affect our runtime bounds (which depend on Δ), because our runtime bounds depend on the maximum degree in the tree T, which contains only edges from the original graph (and not from the triangulated one).[1]

For a rooted spanning tree $\tilde{T}$ of G, define the *root-path* of a vertex $v \in M$ in this tree, denoted $\tilde{T}_v$, to be the path in the tree $\tilde{T}$ connecting v to the root. (We use here $\tilde{T}$ for generality, but will soon instantiate it with the tree T constructed in step 1.) The next theorem has essentially the same proof as of [21, Lemma 2]. It is particularly convenient for a recursive application, where $U \subset M$ is the "current" subset to work on, yet G and the tree $\tilde{T}$ remain fixed through the recursive process. It shows that each set could be partitioned using a cycle which composed of two paths in the shortest path tree, plus an edge called the *separator edge*. We remark that $G[U]$ is not required to be connected.

▶ **Lemma 3** ([21]). *Given a triangulated plane graph $G = (M, E)$, a rooted spanning tree $\tilde{T}$ of G, a subset $U \subset M$, and a vertex subset $W \subseteq U$, one can find in linear time a non-tree edge $(u, v) \in E(G) \setminus E(\tilde{T})$ such that the cycle $\tilde{T}_u \cup \tilde{T}_v \cup \{(u, v)\}$ is a vertex-separator in the following sense: $U \setminus (\tilde{T}_u \cup \tilde{T}_v)$ can be partitioned into two subsets U_1 and U_2 such that (i) each of $U_1 \cap W$ and $U_2 \cap W$ is of size at most $2|W|/3$; and (ii) all paths from a vertex in U_1 to a vertex in U_2 intersect $\tilde{T}_u \cup \tilde{T}_v$ at a vertex.*

Preprocessing algorithm, step 2. We invoke the algorithm of Lemma 3 recursively to construct a "hierarchy" tree $\mathcal{T}$ as follows. Each node μ in $\mathcal{T}$ corresponds to a triple $\langle G(\mu), D(\mu), e(\mu)\rangle$ that records an invokation of Lemma 3 on G:

- $G(\mu)$ records the induced subgraph used as $G[U]$;
- $D(\mu) \subset D$ records the subset of data points used as W;
- the tree T constructed in step 1 is used as $\tilde{T}$ (the same tree for all μ); and
- $e(\mu)$ records the edge of G obtained by this invokation.

The root of $\mathcal{T}$ corresponds to $\langle G, D, e_0\rangle$ where e_0 is the edge obtained by invoking Lemma 3 with $U = M$, $W = D$. Consider now some node μ in $\mathcal{T}$, and let U_1 and U_2 be the two subsets of U obtained from the corresponding invokation of Lemma 3 on $G(\mu)$ and $D(\mu)$. The two children of μ in the tree, denoted μ_i for $i = 1, 2$, correspond, respectively, to the invokations of Lemma 3 on the induced subgraphs $G(\mu_i) = G[U_i]$ with the sets $D(\mu_i) = U_i \cap D$. The recursion stops at a node μ if the corresponding $G(\mu) \cap D = \emptyset$.

Structural properties of the hierarchy tree $\mathcal{T}$. We need several definitions and proofs from [1], which are repeated here for completeness. For a node $\mu \in \mathcal{T}$, let the *level* of

[1] One can also triangulate G while increasing its maximum degree by at most a constant factor [16, Theorem 4.4].

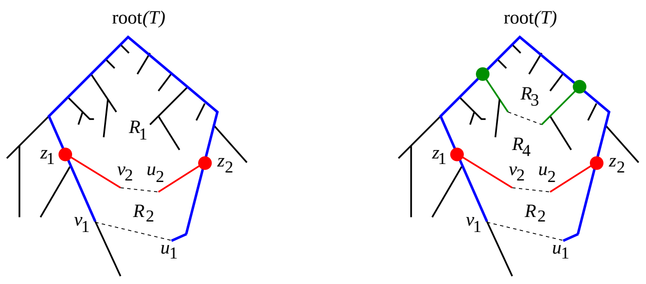


Figure 1 An illustration of apices and frames. The left figure shows the tree T using solid edges. The first separator-edge, at the hierarchy's root, is the dashed edge (v_1, u_1). Let R be the corresponding cycle's interior region. This region R has a separator-edge (v_2, u_2). The cycle-separator of this region R is the union of the two cycles defined by (v_1, u_1) and by (v_2, u_2). The apices of R are labeled z_1 and z_2.
The right figure then shows what happens in region R_1; the separator-edge is a green dashed line, and the new apices are green circles. The frame of R is the subgraph colored blue. The frame of R_2 contains all red and some blue edges. Note that all edges in the first separator $T_{v_1} \cup T_{u_1}$ belong to the frame of only one of R_1 and R_2.

μ, denoted $\textsc{Level}(\mu)$, be the number of edges in $\mathcal{T}$ from μ to the root of $\mathcal{T}$. Clearly, $0 \le \textsc{Level}(\mu) \le 1 + \log_{3/2} n \le 2 \log n$. We now associate with each node μ of $\mathcal{T}$ three subgraphs of G. First, define the *cluster* of μ to be

$$\textsc{Cluster}(\mu) := G(\mu).$$

Second, define the *cycle-separator* of μ recursively as follows. If μ is the root of $\mathcal{T}$, then $\textsc{Cycle-Sep}(\mu) := T_u \cup T_v \cup \{(u, v)\}$. Otherwise, let μ' be the parent of μ in $\mathcal{T}$, and let

$$\textsc{Cycle-Sep}(\mu) := \textsc{Cycle-Sep}(\mu') \cup T_u \cup T_v \cup \{(u, v)\}\,.$$

Third, define the *separator* of μ to be the subgraph of $\textsc{Cycle-Sep}(\mu)$ induced by the edges of T, formally,

$$\textsc{Sep}(\mu) := \textsc{Cycle-Sep}(\mu) \cap E(T).$$

▶ **Observation 4.** *For every μ, the subgraph $\textsc{Sep}(\mu)$ is a subtree of T containing its root s.*

▶ **Observation 5.** *For every μ (other than the root) and its parent μ', the vertices of $\textsc{Sep}(\mu')$ separate $\textsc{Cluster}(\mu)$ from the rest of G. This is immediate from Lemma 3.*

Define the *home* of a vertex $x \in M$, denoted $\textsc{Home}(x)$, as the node μ of $\mathcal{T}$ of smallest level such that x belongs to $\textsc{Sep}(\mu)$.

Define the *apices* of a node μ, denoted $\textsc{Apices}(\mu)$, as the set of vertices in $\textsc{Cycle-Sep}(\mu)$ that have degree ≥ 3; see Figure 1 for an illustration. The apices of μ turn out to be a key enabler of our solution. As we show below, there are very few apices per region and they concisely represent the topological connections between nearby regions (note that degree 2 vertices of $\textsc{Cycle-Sep}(\mu)$ simply form paths between pairs of apices and topologically each such path can be contracted into an edge)

The *new apices* of μ are defined as follows. If μ is the root of $\mathcal{T}$, then $\textsc{NewApices}(\mu) := \textsc{Apices}(\mu)$; otherwise, let $\textsc{NewApices}(\mu) := \textsc{Apices}(\mu) \setminus \textsc{Apices}(\mu')$, where μ' is the parent of μ in $\mathcal{T}$. Intuitively, the new apices of μ are the vertices where the separator of μ "disconnects" from its parent separator μ'.

▶ **Lemma 6** ([1]). *For every μ we have $|\textsc{NewApices}(\mu)| \leq 2$.*

We next define the *frame* of a node μ, which is, loosely speaking, a small subgraph of $\textsc{Sep}(\mu')$ that separates $\textsc{Cluster}(\mu)$ from the rest of G. Formally, if μ is the root of $\mathcal{T}$, then $\textsc{Frame}(\mu)$ is the empty graph. Otherwise, let μ' be the parent of μ, and let $\textsc{Frame}(\mu)$ be the subgraph of $\textsc{Sep}(\mu')$ induced by the vertices x in $\textsc{Sep}(\mu')$, which can be connected to (some vertex z in) $\textsc{Cluster}(\mu)$ by a path whose internal vertices (i.e., all but x, z) are all in $\textsc{Sep}(\mu') \setminus \textsc{Apices}(\mu')$. By construction, a path connecting a vertex of $\textsc{Cluster}(\mu)$ to a vertex outside the cluster has to intersect $\textsc{Frame}(\mu)$.

The *region* of μ, denoted $\textsc{Reg}(\mu)$, is the subgraph of G induced by all the vertices of $\textsc{Cluster}(\mu) \cup \textsc{Frame}(\mu)$.

▶ **Lemma 7** ([1]). *For every level $\ell \geq 0$ in $\mathcal{T}$, every edge $e \in E$ belongs to the frame of at most two nodes μ for which $\textsc{Level}(\mu) = \ell$.*

2.2 Finding the Query's Region

Our goal is to find for every level $\ell \leq \textsc{Level}(\textsc{Home}(q))$ the region that contains the query q. (There is exactly one such region, because q is not in the separator.) We provide a slightly weaker guarantee that is sufficient for our needs: we show how to compute for every $\ell = 0, 1, \ldots, 2\log n$ a set $\mathcal{A}_\ell(q)$ of at most two regions, such that whenever $\ell \leq \textsc{Level}(\textsc{Home}(q))$, the set $\mathcal{A}_\ell(q)$ contains the region containing q.

Query Algorithm for Region Finding (Zoom-In). For a node μ of $\mathcal{T}$, let its near-apices be the set of all edges incident to the apices of μ or to s (the root of the shortest path tree T); formally, $NA(\mu) := \{(y, x) \in E \mid x \in \textsc{Apices}(\mu) \cup \{s\}\}$, where we view each (y, x) as an ordered pair.

The query algorithm computes $\mathcal{A}_\ell(q)$ iteratively for level $\ell = 0, 1, \ldots, 2\log n$. After initializing $\mathcal{A}_0(q) = \{\text{root}(\mathcal{T})\}$, it computes the next set $\mathcal{A}_\ell(q)$ using $\mathcal{A}_{\ell-1}(q)$ as follows. Find the edge (y, x) for which y is furthest away from s, among all edges $(y, x) \in \cup_{\mu \in \mathcal{A}_{\ell-1}(q)} NA(\mu)$ such that (y, x) is on the shortest path from q to s. Here, we treat (y, x) as an ordered pair, and insist that y appears before x along the path from q to s, or equivalently, that $\mathbf{d_G}(q, y) + \omega(y, x) + \mathbf{d_G}(x, s) = \mathbf{d_G}(q, s)$, which can be checked using only a constant number of distance oracle queries. Notice such (y, x) always exists, because the root s is an apex and one of its incident edges is on the shortest paths from q to s. Next, let $\mathcal{A}_{\ell+1}(q)$ be the set of regions at level $\ell + 1$ that contain the edge (y, x), which we prepare in advance (during the preprocessing phase) as the set $\hat{\mathcal{A}}_{\ell'}((y, x))$. Finally, proceed to the next iteration.

```
algorithm FindRegions ($q$)
0.  let $\mathcal{A}_0(q) = \{\text{root}(\mathcal{T})\}$
1.  for $\ell = 1$ to $2\log n$ do
      a. pick $(y, x) \in E$ of maximal $\mathbf{d_G}(y, s)$ among all $(y, x) \in$
         $\cup_{\mu \in \mathcal{A}_{\ell-1}(q)} NA(\mu)$ that satisfy $\mathbf{d_G}(q, y) + \omega(y, x) + \mathbf{d_G}(x, s) = \mathbf{d_G}(q, s)$
      b. let $\mathcal{A}_\ell(q) = \hat{\mathcal{A}}_\ell((y, x))$
2.  return the sets $\mathcal{A}_\ell(q)$ for $\ell = 0, \ldots, 2\log n$
```

Figure 2 Zooming-in on the query's region.

▶ **Lemma 8.** *For every level $\ell \le \textsc{Level}(\textsc{Home}(q))$, the query vertex q belongs to a region in $\{\textsc{Reg}(\mu) \mid \mu \in \mathcal{A}_\ell(q)\}$.*

Proof. The proof is by induction on ℓ. For $\ell = 0$, we initialized $\mathcal{A}_0(q) = \{\text{root}(\mathcal{T})\}$ and the corresponding region $\textsc{Reg}(\text{root}(\mathcal{T}))$ is the entire graph G, hence $q \in \textsc{Reg}(\text{root}(\mathcal{T}))$. Assume now the claim holds for level ℓ and consider level $\ell+1$. Let (y,x) be the edge of maximal $\mathbf{d_G}(s,y)$ among all edges in $\cup_{\mu\in\mathcal{A}_\ell(q)} NA(\mu)$ that lie on a shortest path from q to s.

Suppose $q \in R$ for a region R at level $\ell+1$, and let us show that $R \in \mathcal{A}_\ell(q)$. Let μ be the corresponding node in $\mathcal{T}$, i.e., $\textsc{Reg}(\mu) = R$, and let μ' be the parent node of μ in $\mathcal{T}$. Observe that $q \in \textsc{Reg}(\mu')$, because q must be in $\textsc{Cluster}(\mu)$ (rather than $\textsc{Frame}(\mu)$), and all such vertices are inside the region of the parent μ'. Thus by the induction hypothesis $\mu' \in \mathcal{A}_\ell(q)$.

Let $P(q,s)$ be the unique shortest path from q to s, and let z be the first vertex along this path (furthest from the root s) which is in $\textsc{Apices}(\mu')$. Such z exists (because the root s is an apex) and is not the first vertex on the path (because q itself is not an apex), so let z_1 be vertex preceding z on this path.

We now claim that (z_1,z) is exactly the edge (y,x) chosen. Indeed, the edge (z_1,z) satisfies the two requirements (it is in $NA(\mu')$ and on the shortest path from q to s) by definition, and moreover, z was chosen to that it is closest to q and thus furthest from s.

The claim implies, by the construction of z as the first apex on $P(q,s)$, that the region R containing q also contains the edge $(z_1,z) = (y,x)$, which means that the algorithm will add $R = \textsc{Reg}(\mu)$ to $A_{\ell+1}(q)$. ◀

2.3 Estimating the Distance

Once we have located the region of q, we would like to complete the nearest neighbor search. Let $t^* \in D$ be the closest dataset point to q, and let $P(q,t^*)$ be the shortest path from q to t^*, then we would like to approximate the length of $P(q,t^*)$. Since we located q's region, we can follow the path from the root of the tree $\mathcal{T}$ to q's region. We observe that at some tree node μ we reach a situation where $P(q,t^*)$ intersects $\textsc{Frame}(\mu)$. Indeed, the region at the root of the tree contains both q and t^*; however, at the leaf μ of the tree, the region contains q and either (i) does not contain t^*, in which case the path $P(q,t^*)$ connects a vertex in $\textsc{Cluster}(\mu)$ to one outside $\textsc{Cluster}(\mu)$, and thus must intersect $\textsc{Frame}(\mu)$; or (ii) it does contain t^*, but only in its frame and not in its cluster (because $\textsc{Cluster}(\mu) \cap D = \emptyset$), in which case t^* is itself in the intersection.

If the query procedure could identify a vertex v on this intersection between $P(q,t^*)$ and $\textsc{Frame}(\mu)$, then it could solve finding the nearest neighbor problem for q by finding the nearest neighbor of v and reporting the exact same vertex (and this holds also for approximate nearest neighbor). This is exactly the approach taken in our warmup, the bounded treewidth case, where the preprocessing phase stores for every separator vertex its nearest neighbor in D, and the query procedure just considers all the separator vertices in all the regions encountered during the zoom-in process for q.

However, in the planar graph setting, the number of vertices on a single separator may be arbitrarily large (compared to n). So we must exploit the separator's structure as the union of a few shortest paths. At a very high level, our solution is to carefully choose net-points on the boundary of each region, and only for these net-point we store their nearest neighbor in D. The challenge is to choose the net-points in such a way that (i) for at least one "good" net-point in the sense that the distance from q through the net-point and then to D (i.e., to the nearest neighbor of this net-point) is guaranteed to approximate the optimal NNS answer; and (ii) the query procedure can examine very few net-points (compared to the total number of net-points stored, which is linear in n) until one of these good points is found.

Before describing the algorithm in more detail, let us introduce some useful notations. For a vertex c on a path P and a distance $\rho' > 0$, let $P(c, \rho')$ be all nodes in P at distance at most ρ' from c. Let $\mathcal{N}(P, c, \rho, \rho')$ be a set of nodes in $P(c, \rho')$ such that every node in $P(c, \rho')$ has a node in $\mathcal{N}(P, c, \rho, \rho')$ at distance ρ and every two nodes in $\mathcal{N}(P, c, \rho, \rho')$ are at distance at least ρ from one another (this set can be obtained by considering all nodes on the path $P(c, \rho')$ from endpoint of the path to another and adding to $\mathcal{N}(P, c, \rho, \rho')$ every node that does not have yet a node in $\mathcal{N}(P, c, \rho, \rho')$ at distance ρ from it). For a shortest path P and a node v, let

$$N_i(v, P) := \mathcal{N}(P, c_v, 2^i\epsilon/32, 2^{i+1}),$$

and

$$N(v, P) := \bigcup_{0 \le i \le \log(nDiam)} N_i(v, P),$$

where c_v is the vertex in P closest to v.

For a tree node $\mu \in \mathcal{T}$ such that $e(\mu) = (u, v)$, we let $N(T_u) := \cup_{w \in \textsc{Cluster}(\mu) \cap D} N(w, T_u)$, and similarly for $N(T_v)$. (Formally, it depends also on μ, but we suppress this.) For a shortest path P from s to some vertex f, let $N(P, d_1, d_2)$ to be a vertex in $N(P)$ as follows. If $d_2 > \mathbf{d_G}(s, f)$ then $N(P, d_1, d_2) := f$. If $d_1 < 0$ then $N(P, d_1, d_2) := s$. Otherwise let $N(P, d_1, d_2)$ be the vertex $x \in N(P)$ with minimal $\mathbf{d_G}(x, s)$ among all vertices x satisfying $d_1 \le \mathbf{d_G}(x, s) \le d_2$; if no such vertex x exists, set $N(P, d_1, d_2) := null$. For a tree node μ, let $\mathcal{P}(\mu) = \{T_u\} \cup \{T_v\}$ where $e(\mu) = (u, v)$.

Preprocessing. Let us describe the additional information stored by our data structure. For every node μ, where we denote $(u, v) = e(\mu)$, store the sets $N(T_z)$ for all $z \in \{u, v\}$. In addition, construct a range reporting data structure on $N(T_z)$ according to the distance from s. Namely, a data structure that given two distances d_1, d_2 returns in $O(\log\log n)$ time a vertex $x \in N(T_z)$ with $\mathbf{d_G}(x, s) \in [d_1, d_2]$ that has minimal $\mathbf{d_G}(x, s)$ among all such vertices, or returns null if no such vertex exists. Observe that the range reporting data structure on $N(P)$ makes it possible to find $N(P, d_1, d_2)$ in $O(\log\log n)$ time [4].

In addition, for every node μ, level $\ell \in \{1, \ldots, 2\log n\}$ and apex $x \in \textsc{NewApices}(\mu)$, store for every edge e incident to x the set $\hat{\mathcal{A}}_\ell(e) = \{\mu \in \mathcal{T} \mid e \in E(\textsc{Reg}(\mu)) \textit{ and } \textsc{Level}(\mu) = \ell\}$, namely, the set of level ℓ tree nodes μ for which e belongs to their region (recall there are at most two such nodes). The algorithm also stores the number $O_T(v)$ for every vertex v that is a neighbor of an apex of some node $\mu \in \mathcal{T}$ (for all apices).

For every tree node μ the algorithm stores an indicator $\mathrm{IL}(\mu)$ if μ is a leaf in $\mathcal{T}$. Note that if μ is a leaf in $\mathcal{T}$ then its cluster contains at most one dataset point, denoted by $D(\mu)$. The algorithm also stores the dataset point $D(\mu)$ in case μ is a leaf.

Distance Query. The distance query given a vertex q is performed as follows. (See Figure 3 for a pseudo-code description.) The algorithm starts by invoking Procedure **FindRegions** to obtain the sets $\{\mathcal{A}_i(q)\}$ for $1 \le i \le 2\log n$, where each set $\mathcal{A}_i(q)$ contains at most two nodes of level i in $\mathcal{T}$ such that q belongs to the region of at least one of them. The algorithm then iterates on all path separators P in $\mathcal{P}(q) = \cup_{1 \le i \le 2\log n} \cup_{\mu \in \mathcal{A}_i(q)} \mathcal{P}(\mu)$. For a path $P \in \mathcal{P}(q)$, let $\mu(P)$ be the node such that $P \in \mathcal{P}(\mu)$. For each such path separator P, the algorithm invokes Procedure **DistThroughPath** to estimate $\mathbf{d_G}(q, D \cap \textsc{Cluster}(\mu(P)), P)$, namely, the length of the shortest path from q to some vertex in $D \cap \textsc{Cluster}(\mu(P))$ among all such paths that go through some vertex in P. Let $\tilde{d}(P, q, D)$ be the estimated distance returned by this invokation of Procedure **DistThroughPath**. In addition, the algorithm iterates over

```
algorithm Dist$(q)$
1. $\{\mathcal{A}_i(q)\} \leftarrow$ FindRegions$(q)$.
2. Let $\mathcal{P}(q) = \cup_{\mu \in \mathcal{A}_i(q), 1 \le i \le 2\log n} \mathcal{P}(\mu)$.
3. For every $P \in \mathcal{P}(q)$ do the following:
   a. Set $\tilde{d}(q, D, P) \leftarrow$ DistThroughPath$(q, P)$.
4. Set $d_1 \leftarrow \infty$.
5. For every $\mu \in \cup_{\mu \in \mathcal{A}_i(q), 1 \le i \le 2\log n}$ do:
   a. If IL$(\mu)$ then set $d_1 \leftarrow \min\{d_1, \mathbf{d_G}(q, D(\mu))\}$.
6. Return $\min(\{\tilde{d}(q, D, P) \mid P \in \mathcal{P}(q)\} \cup \{d_1\})$.
```

Figure 3 Our main algorithm for estimating the distance between a given query vertex q and the closest data point to it in D.

```
algorithm DistThroughPath $(q, P)$
1. $found = false$.
2. Set $p \leftarrow s$.
3. While $(found = false)$
   a. Let $\tilde{d} = \mathbf{d_G}(p, q)$.
   b. Find an $\tilde{d}/8$-net $S'$ on $N(P) \cap P(p, 2\tilde{d})$.
   c. Set $p$ to be the vertex in $S' \cup \{p\}$ such that $\mathbf{d_G}(q, p)$ is minimal.
   d. If $\mathbf{d_G}(q, p) > \tilde{d}/2$ then set $found = true$.
4. For $i$ from 1 to $\log nM$ do the following.
   a. Find an $2^i\epsilon/8$-net $S_i$ on $N(P) \cap P(p, 2^{i+3})$.
   b. Set $\tilde{d}(q, D, P)_i$ to be the minimal distance $\mathbf{d_G}(q, x) + \mathbf{d_G}(q, D)$ for $x \in S_i$.
5. Set $\tilde{d}(q, D, P)$ to be the minimal distance $\tilde{d}(q, D, P)_i$.
6. Return $\tilde{d}(q, D, P)$.
```

Figure 4 A procedure for estimating $\mathbf{d_G}(q, D, P)$, which is the minimum length of a path from a given query vertex q to some vertex in D among all such paths that go through some vertex in P.

all nodes $\mu \in \cup_{1 \le i \le 2\log n} \mathcal{A}_i(q)$ to check if μ is a leaf in $\mathcal{T}$, and among all such leaf nodes μ, the algorithm finds the node $\tilde{\mu}$ such that $\mathbf{d_G}(q, D(\tilde{\mu}))$ is minimal, denoting it d_1. (This computation is straightforward, since $|D(\tilde{\mu})| \le 1$ for leaf nodes.) The algorithm then returns $\min(\{d_1\} \cup \{\tilde{d}(q, D, P) \mid P \in \mathcal{P}(q)\})$.

Procedure **DistThroughPath** is given (q, P) and works in two stages. (See Figure 4 for a pseudo-code description.) The first stage finds a vertex $p \in P$ that is "close" to q, and the second one uses this p to compute the estimated distance $\tilde{d}(q, D, P)$. The first stage is done as follows. Initialize $p = s$ and $found = false$, and now while $found = false$, do the following: first, let $\tilde{d} = \mathbf{d_G}(p, q)$; second, find a $\tilde{d}/8$-net S' on $N(P) \cap P(p, 2\tilde{d})$; third, set p to be a vertex in $S' \cup \{p\}$ that minimizes $\mathbf{d_G}(q, p)$; finally, if $\mathbf{d_G}(q, p) > \tilde{d}/2$ then set $found = true$ (namely, the first part is finished).

The second stage is then done as follows. For i from 1 to $\log(nM)$ do the following. First, find a $2^i\epsilon/8$-net S_i on $N(P) \cap P(p, 2^{i+3})$. Second, set $\tilde{d}(q, D, P)_i$ to be the minimal distance $\mathbf{d_G}(q, x) + \mathbf{d_G}(q, D)$ for $x \in S_i$. Now return the minimal distance $\tilde{d}(q, D, P)_i$ as the final answer $\tilde{d}(q, D, P)$.

2.4 Analysis

Recall that $t^* \in D$ is the closest data point to q.

▶ **Lemma 9.** *Consider a node $\mu \in \mathcal{T}$, let $C = \text{Cluster}(\mu)$ and $e(\mu) = (u, v)$. Let $P \in \{P_u, P_v\}$. Consider a vertex $x \in P$, there is a vertex $z \in N(P)$ at distance at most $\epsilon\, \mathbf{d_G}(x, D \cap C)/16$ from x.*

Proof. Let $t \in D \cap C$ be the vertex of minimal $\mathbf{d_G}(x, t)$, namely, $\mathbf{d_G}(x, t) = \mathbf{d_G}(x, D \cap C)$. Let i be the index such that $2^{i-1} \le \mathbf{d_G}(x, t) \le 2^i$. Note that $\mathbf{d_G}(c_t, x) \le \mathbf{d_G}(c_t, t) + \mathbf{d_G}(t, x) \le 2\, \mathbf{d_G}(t, x)$. Hence $x \in P(c_t, 2^{i+1})$.

Recall that $N(P)$ contains $\mathcal{N}(P, c_t, 2^i\epsilon/32, 2^{i+1})$, namely, for every vertex $y \in P(c_t, 2^{i+1})$ there is a vertex $z' \in N(P)$ such that $\mathbf{d_G}(y, z') \le 2^i\epsilon/32 \le \epsilon\, \mathbf{d_G}(x, D \cap C)/16$. Hence in particular there is a vertex $z \in N(P)$ at distance at most $\epsilon\, \mathbf{d_G}(x, D \cap C)/16$ from x. ◀

Consider a node $\hat{\mu} \in \mathcal{T}$ such that $t^* \in \text{Cluster}(\hat{\mu})$ and let $e(\hat{\mu}) = (\hat{u}, \hat{v})$. Let $P \in \{T_{\hat{u}}, T_{\hat{v}}\}$. Let $\mathbf{d_G}(q, t^*, P)$ be the distance of the shortest path from q to t^* among all q to t^* paths that contain at least one vertex in P. Consider Procedure **DistThroughPath** when invoking on (q, P). Let $p_{final}(P)$ be the vertex p when the algorithm reaches step 4 of Procedure **DistThroughPath** invoked on (q, P).

▶ **Lemma 10.** $\mathbf{d_G}(q, p_{final}(P)) \le 4\, \mathbf{d_G}(q, t^*, P)$.

Proof. Let c_q be the closest vertex to q in P. Let p_i be the vertex p in the beginning of the i'th iteration of the while loop in step 3 of Procedure **DistThroughPath**.

Note that the algorithm continues to the next iteration as long $\mathbf{d_G}(q, p_{i+1}) \le \mathbf{d_G}(q, p_i)/2$. Let $p_r = p_{final}(P)$. Note also that $\mathbf{d_G}(q, p_r) > \mathbf{d_G}(q, p_{r-1})/2$. From triangle inequality it follows that $c_q \in P(p_{r-1}, 2\, \mathbf{d_G}(q, p_{r-1}))$.

Let S' be the $\mathbf{d_G}(q, p_{r-1})/8$-net on $N(P) \cap P(p, 2\, \mathbf{d_G}(q, p_{r-1}))$ from step 3*b* of the while loop. If $\mathbf{d_G}(q, p_{r-1}) \le 4\, \mathbf{d_G}(q, t^*, P)$ then we are done as $\mathbf{d_G}(q, p_r) \le \mathbf{d_G}(q, p_{r-1})$. Seeking a contradiction assume $\mathbf{d_G}(q, p_{r-1}) > 4\, \mathbf{d_G}(q, t^*, P)$. By Lemma 9, $N(P)$ contains a vertex z_1 at distance $\epsilon\, \mathbf{d_G}(c_q, t^*)/16$ from c_q. Recall that S' is an $\mathbf{d_G}(q, p_{r-1})/8$-net on $N(P) \cap P(p_{r-1}, 2\, \mathbf{d_G}(q, p_{r-1}))$. Hence there is a vertex $z_2 \in S'$ at distance at most $\mathbf{d_G}(q, p_{r-1})/8$ from z_1.

We get that,

$$\begin{aligned}
\mathbf{d_G}(q, p_r) &\le \mathbf{d_G}(q, z_2)\\
&\le \mathbf{d_G}(q, c_q) + \mathbf{d_G}(c_q, z_2)\\
&\le \mathbf{d_G}(q, t^*, P) + \epsilon\, \mathbf{d_G}(c_q, t^*)/16 + \mathbf{d_G}(q, p_{r-1})/8\\
&\le \mathbf{d_G}(q, t^*, P) + \epsilon(\mathbf{d_G}(c_q, q) + \mathbf{d_G}(q, t^*))/16 + \mathbf{d_G}(q, p_{r-1})/8\\
&\le \mathbf{d_G}(q, t^*, P) + \epsilon(\mathbf{d_G}(q, t^*, P) + \mathbf{d_G}(q, t^*, P))/16 + \mathbf{d_G}(q, p_{r-1})/8\\
&= \mathbf{d_G}(q, t^*, P)(1 + \epsilon/8) + \mathbf{d_G}(q, p_{r-1})/8\\
&\le \mathbf{d_G}(q, p_{r-1})(1 + \epsilon/8)/4 + \mathbf{d_G}(q, p_{r-1})/8\\
&\le \mathbf{d_G}(q, p_{r-1})/2,
\end{aligned}$$

contradiction. ◀

Let i be the index such that $2^{i-1} \le \mathbf{d_G}(q, t^*, P) \le 2^i$.

▶ **Lemma 11.** *The distance $\tilde{\mathbf{d_G}}(q, D, P)$ returned by the Procedure* **DistThroughPath** *satisfies $\mathbf{d_G}(q, D, P) \le \tilde{\mathbf{d_G}}(q, D, P) \le (1 + \epsilon)\, \mathbf{d_G}(q, t^*, P)$.*

Proof. It is not hard to verify that $\mathbf{d_G}(q, D, P) \le \tilde{\mathbf{d_G}}(q, D, P)$, we therefore only need to show the second direction where $\tilde{\mathbf{d_G}}(q, D, P) \le (1 + \epsilon)\, \mathbf{d_G}(q, t^*, P)$.

Let $w \in P \cap P(q, t^*, P)$. Note that $\mathbf{d_G}(w, t^*) \leq \mathbf{d_G}(q, t^*, P)$. By Lemma 9 there is a vertex $x \in N(P)$ at distance at most $\epsilon\, \mathbf{d_G}(w, t^*)/16 \leq \epsilon\, \mathbf{d_G}(q, t^*, P)/16$ from w.

We have

$$\begin{aligned}
\mathbf{d_G}(x, p_{final}(P)) &\leq \mathbf{d_G}(x, w) + \mathbf{d_G}(w, q) + \mathbf{d_G}(q, p_{final}(P)) \\
&\leq \mathbf{d_G}(x, w) + \mathbf{d_G}(w, q) + 4\,\mathbf{d_G}(q, t^*, P) \\
&\leq \epsilon\, \mathbf{d_G}(q, t^*, P)/16 + \mathbf{d_G}(q, t^*, P) + 4\,\mathbf{d_G}(q, t^*, P) \\
&< 6\,\mathbf{d_G}(q, t^*, P) \\
&\leq 6 \cdot 2^i \\
&< 2^{i+3},
\end{aligned}$$

where the second inequality follows by Lemma 10.

We get that $x \in P(p_{final}(P), 2^{i+3})$. Recall that S_i is an $2^i\epsilon/8$-net on $N(P) \cap P(p_{final}(P), 2^{i+3})$. Hence there is a vertex $x_2 \in S_i$ at distance at most $2^i\epsilon/8$ from x. Note that $\mathbf{d_G}(x_2, w) \leq \mathbf{d_G}(w, x) + \mathbf{d_G}(x, x_2) \leq \epsilon\, \mathbf{d_G}(w, t^*)/16 + 2^i\epsilon/8 \leq \epsilon\, \mathbf{d_G}(q, t^*, P)/2$. We get that

$$\begin{aligned}
\tilde{\mathbf{d_G}}(q, D, P) &\leq \mathbf{d_G}(q, x_2) + \mathbf{d_G}(x_2, t^*) \\
&\leq \mathbf{d_G}(q, w) + \mathbf{d_G}(w, x_2) + \mathbf{d_G}(x_2, w) + \mathbf{d_G}(w, t^*) \\
&\leq \mathbf{d_G}(q, t^*, P) + 2\,\mathbf{d_G}(x_2, w) \\
&\leq (1 + \epsilon)\,\mathbf{d_G}(q, t^*, P).
\end{aligned}$$

◀

The following lemma shows that the estimated distance returned by the algorithm satisfies the desired stretch.

▶ **Lemma 12.** *The distance $\tilde{\mathbf{d_G}}(q, D)$ returned by the algorithm satisfies $\mathbf{d_G}(q, D) \leq \tilde{\mathbf{d_G}}(q, D) \leq (1 + \epsilon)\,\mathbf{d_G}(q, D)$.*

Proof. it is not hard to verify that $\mathbf{d_G}(q, D) \leq \tilde{\mathbf{d_G}}(q, D)$, we therefore only need to show the other direction, namely, $\tilde{\mathbf{d_G}}(q, D) \leq (1 + \epsilon)\,\mathbf{d_G}(q, D)$. Let μ be the leaf node in $\mathcal{T}$ that contains t^*.

If $q \in \textsc{Reg}(\mu)$, then note that by Lemma 8 $\mu \in \{\mu \in \mathcal{A}_i(q) \mid 1 \leq i \leq 2 \log n\}$, therefore the algorithm examines the distance $\mathbf{d_G}(q, D(\mu))$ and returns it if this is the minimal distance examined by the algorithm. We get that $\tilde{\mathbf{d_G}}(q, D) \leq \mathbf{d_G}(q, D(\mu)) = \mathbf{d_G}(q, D)$. So assume $q \notin \textsc{Reg}(\mu)$. Notice that there must be an ancestor node μ' such that $P(q, t^*) \cap P \neq \emptyset$ for some $P \in \{T_u, T_v\}$ where $e(\mu') = (u, v)$. Notice that $P \in \mathcal{P}(q)$ and thus by the algorithm and Lemma 11 we have $\tilde{\mathbf{d_G}}(q, D) \leq (1 + \epsilon)\,\mathbf{d_G}(q, D, P) = (1 + \epsilon)\,\mathbf{d_G}(q, D)$. ◀

▶ **Lemma 13.** *The query algorithm runs in time $O(\frac{1}{\epsilon} \cdot \log\log n + t_{DO}) \log n \log Diam + \log n \cdot \Delta t_{DO})$.*

Proof. Let us start with bounding the time to find the sets $\mathcal{A}_\ell(q)$ for $1 \leq \ell \leq 2 \log n$ in Procedure **FindRegions**. Recall that in order to find the sets $\mathcal{A}_{\ell+1}(q)$ the algorithm examines all $(y', x') \in \cup_{\mu \in \mathcal{A}_\ell(q)} NA(\mu)$ and check which ones satisfy $\mathbf{d_G}(q, y') + \omega(y', x') + \mathbf{d_G}(x', s) = \mathbf{d_G}(q, s)$, and among the ones that satisfy the equality, the algorithm picks the edge $e = (y, x)$ of minimal $\hat{O}_T(e)$. Checking if an edge (y', x') satisfy $\mathbf{d_G}(q, y') + \omega(y', x') + \mathbf{d_G}(x', s) = \mathbf{d_G}(q, s)$ can be done by constant queries to the distance oracle and thus takes $O(t_{DO})$ time.

Let $\tilde{\mu}$ be a node in $\mathcal{A}_{\ell+1}(q)$ and let $\tilde{\mu}'$ be its parent in $\mathcal{T}$. Recall that $\tilde{\mu}' \in \mathcal{A}_\ell(q)$. Recall also that $NA(\tilde{\mu})$ is the set of all edges incident to the apices of $\tilde{\mu}$ or to s. Since $\tilde{\mu}$ is a child of $\tilde{\mu}'$ we have $\textsc{Apices}(\tilde{\mu}) \subseteq \textsc{Apices}(\tilde{\mu}') \cup \textsc{NewApices}(\tilde{\mu})$.

For level j let $e = (x_j, y_j) \in \cup_{\mu \in \mathcal{A}_j(q)} NA(\mu)$ be the edge of minimal $O_T(y_j)$ among all edges $(x', y') \in \cup_{\mu \in \mathcal{A}_j(q)} NA(\mu)$ that satisfy $\mathbf{d_G}(q, y') + \omega(y', x') + \mathbf{d_G}(x', s) = \mathbf{d_G}(q, s)$.

Let $NNA(\mu)$ be the set of all edges incident to the new apices of μ. It is not hard to verify that in order to find $e = (x_{j+1}, y_{j+1})$ given the edge $e = (x_j, y_j)$, it is enough to find the edge (x, y) of minimal $O_T(y)$ among all edges $(x', y') \in \cup_{\mu \in \mathcal{A}_j(q)} NNA(\mu)$ that satisfy $\mathbf{d_G}(q, y') + \omega(y', x') + \mathbf{d_G}(x', s) = \mathbf{d_G}(q, s)$ and compare it with (x_j, y_j). Recall that $\mathcal{A}_{j+1}(q)$ contains at most two nodes. Hence the time spend for level $j + 1$ is $O(\Delta \cdot t_{DO})$. Hence the total time to find all sets $\mathcal{A}_\ell(q)$ is $O(\log n \Delta \cdot t_{DO})$.

We now turn to bound the running time of Procedure **DistThroughPath** invoked on (q, P). Recall that Procedure **DistThroughPath** has two main parts. The first part finds a vertex p_{final} such that $\mathbf{d_G}(q, p_{final}(P)) \le 4\, \mathbf{d_G}(q, t^*, P)$ and the second part uses $p_{final}(P)$ to find an estimation on $\mathbf{d_G}(q, D, P)$.

Let p_i be the vertex p in the beginning of the i'th iteration of the while loop in step 3 of Procedure **DistThroughPath**. The first part is done in iterations, where the algorithm continues to the next iteration i as long as $\mathbf{d_G}(q, p_i) \le \mathbf{d_G}(q, p_{i-1})$. Therefore the number of iteration is $O(\log Diam)$. It is not hard to see that the time of iteration i is dominated by the maximum of the time for finding a $\tilde{d}/8$-net S' on $N(P) \cap P(p_i, 2\tilde{d})$ and the time for invoking the distance oracle a constant number of times. Finding a $\tilde{d}/8$-net S' on $N(P) \cap P(p_i, 2\tilde{d})$ can be done by $O(\log \log n)$ using the range reporting data structure on $N(P)$ as follows.

For j from -16 to 15, find $N(P, \mathbf{d_G}(s, p_i) + j\tilde{d}/8, \mathbf{d_G}(s, p_i) + (j+1)\tilde{d}/8)$ and add it to S' (initially set to be empty). It is not hard to verify that S' is indeed $\tilde{d}/8$-net on $N(P) \cap P(p_i, 2\tilde{d})$.

The time for a single invokation of the range reporting data structure takes $O(\log \log n)$. Note that the range reporting data structure is invoked a constant number of times. We get that each iteration of the first part takes $O(\log \log n + t_{DO})$ time.

Hence the first part takes $O((\log \log n + t_{DO}) \log Diam)$ time.

Let us now turn to the second part of Procedure **DistThroughPath**. The second part consists of $\log nM = O(\log Diam)$ iterations. It is not hard to see that the time of each iteration is dominated by the maximum of the time for finding a $2^i\epsilon/8$-net S_i on $N(P) \cap P(p, 2^{i+3})$ and the time for invoking the distance oracle $O(1/\epsilon)$ times.

Similarly as explained in the first part finding a $2^i\epsilon/8$-net S_i on $N(P) \cap P(p, 2^{i+3})$ can be done in $O(1/\epsilon \log \log n)$ time. Thus the total time for the second part is $O((1/\epsilon \log \log n + t_{DO}) \log Diam)$ time. We get that the total time for Procedure **DistThroughPath** is $O((1/\epsilon \log \log n + t_{DO}) \log Diam)$.

Finally, we turn to bound the running time of Procedure **Dist**. Procedure **Dist** starts by invoking Procedure **FindRegions** to obtain the sets $\{\mathcal{A}_i(q)\}$ for $1 \le i \le 2 \log n$. This takes $O(\log n \Delta \cdot t_{DO})$ as explained above.

The algorithm then iterates on all path separators P in $\mathcal{P}(q) = \cup_{\mu \in \mathcal{A}_i(q), 1 \le i \le 2 \log n} \mathcal{P}(\mu)$. Recall that there are at most $O(\log n)$ such paths. For each such path P the algorithm invokes Procedure **DistThroughPath** which takes $O((1/\epsilon \log \log n + t_{DO}) \log Diam)$ time. In addition, the algorithm iterates over all nodes $\mu \in \cup_{\mu \in \mathcal{A}_i(q), 1 \le i \le 2 \log n}$ and invokes the distance oracle a constant number of items for each iteration.

We get that the total running time of Procedure **Dist** is $O((1/\epsilon \log \log n + t_{DO}) \log n \log Diam + \log n \Delta \cdot t_{DO})$. ◀

▶ **Lemma 14.** *The space requirement of the data structure is* $O(\frac{1}{\epsilon} n \log n \log Diam + n\Delta \log^2 n)$.

Proof. The number of nodes in $\mathcal{T}$ is $O(n \log n)$. It is not hard to verify that the depth of $\mathcal{T}$ is $O(\log n)$ as for every node μ with parent node μ' we have $\textsc{Cluster}(\mu) \cap D \le$

$2/3 \cdot \textsc{Cluster}(\mu') \cap D$. In addition, the number of nodes in each level is at most n as the clusters of the nodes are disjoint and the cluster of each node contains a vertex in D.

For every node μ, every level ℓ and every apex $x \in \textsc{NewApices}(\mu)$, the algorithm stores for every edge e that is incident to x the set of at most two nodes $\hat{\mathcal{A}}_\ell(e) = \{\mu \in \mathcal{T} \mid e \in E(\textsc{Reg}(\mu))$ *and* $\textsc{Level}(\mu) = \ell\}$. The algorithm also stores the number $O_T(v)$ for every vertex v that is a neighbor of an apex of some node $\mu \in \mathcal{T}$.

There are at most two apices in $\textsc{NewApices}(\mu)$. For each such apex x there are most Δ incident edges. For each such edge e and for each level ℓ the size of $\hat{\mathcal{A}}_\ell(e)$ is two. We get that the size stored for each node μ for this part is $O(\Delta \log n)$. There are at most $O(n \log n)$ nodes. Thus the total size for this part is $O(n\Delta \log^2 n)$.

For every node μ such that $e(\mu) = (u, v)$, the algorithm stores the sets $N(T_z)$ in an increasing distance from s for $z \in \{u, v\}$. In addition, construct a range reporting data structure on $N(T_z)$ according to the distance from s. The size of the range reporting data structure is $|N(T_z)|$. We thus need to bound the size of all $N(P)$ for all path separators P.

Every vertex $w \in D$ belongs to the clusters of at most $2 \log n$ nodes μ. For each such node μ such that $e(\mu) = (u, v)$, w contributes at most $O(\log Diam/\epsilon)$ vertices to $N(T_z)$. We get that the sum of the sizes of all $N(P)$ for all path separators P is $O(n \log n \log Diam/\epsilon)$. Thus the total size for this part is $O(n \log n \log Diam/\epsilon)$.

Finally, for every node μ the algorithm stores an indicator $\text{IL}(\mu)$ if μ is a leaf in $\mathcal{T}$. The algorithm also stores the data-point $D(\mu)$ in case μ is a leaf. Thus the total size for this part is $O(n \log n)$. Overall, we get that the total size of the data structure is $O(n \log n \log Diam/\epsilon + n\Delta \log^2 n)$. ◀

3 The General Case: Non-Unique Shortest Paths

In this section we show how to handle the general and seemingly much more involved case of non-unique shortest paths. As mentioned above, the common workaround of perturbing the edge weights is not applicable here because we assume only a black-box access to a distance oracle. The main challenge is to efficiently perform the zoom-in operation. In the unique shortest paths case, if we found a node x that is on the shortest path from q to s, then we knew that x is an ancestor of q in the tree T. This provided us with a better idea on where the query q is and and thus we could zoom in to the right regions. The main idea in handling the non-unique case is to have a consistent way of breaking ties in the preprocessing phase while constructing the shortest path tree T. This also considerably complicates the analysis of the zoom-in operation, and we need to use planarity to show that our consistent way of breaking the ties together with planarity is enough to be able to zoom-in correctly (it is easy to create examples where the graph is not planar and then our way of breaking the ties does not give us more information on where the query q is).

Let us start with the modifications needed in the preprocessing phase. We will later show the modifications needed in the zoom-in operation and in the analysis.

3.1 Preprocessing: The General Case

The main difference in the preprocessing phase is in the way the algorithm chooses the shortest path tree T. The definitions below provide a consistent way of breaking such ties, and will be used later extensively.

Identifiers. Fix some vertex $s \in M$, and assign each vertex $v \in M$ a unique *identifier* $\mathbf{id}(v) \in [1..m]$, such that for all $v_1, v_2 \in M$ with $\mathbf{d_G}(s, v_1) < \mathbf{d_G}(s, v_2)$, we have $\mathbf{id}(v_1) >$

$\mathbf{id}(v_2)$. Such identifiers can be computed easily by ordering the vertices according to their distance from s, breaking ties arbitrarily.

Partial-order on shortest paths. The unique identifiers and $s \in M$ induce the following partial order $\lessdot$ on shortest paths in the graph. Let $P = (s = z_1, z_2, \ldots, z_r)$ and $P' = (s = z'_1, z'_2, \ldots, z'_{r'})$ be two shortest paths originating from the same vertex s. (Our definition below actually extends to every two shortest paths, but we will only need the case $z_1 = z'_1 = s$.) We say that P *is smaller than* P' *with respect to* $\lessdot$, denoted $P \lessdot P'$, if the smallest index $j \geq 1$ for which $z_j \neq z'_j$, satisfies $\mathbf{id}(z_j) < \mathbf{id}(z'_j)$. If no such index j exists, which happens if P is a subpath of P' or the other way around, then the two paths are incomparable under $\lessdot$. A shortest path P from $s \in M$ to $v \in M$ is called *minimal with respect to* $\lessdot$ if it is smaller with respect to $\lessdot$ than every other shortest path from s to v. Observe that for every $v \in M$, every two non-identical shortest paths from s to v are comparable, and thus exactly one of all these shortest paths is minimal. We remark that in the above description, and also in the foregoing discussion, it is convenient to implicitly consider paths as "directed" from one endpoint to the other one (usually going further away from s).

Tree with ordered shortest paths. Let T be a shortest-path tree rooted at the fixed vertex $s \in M$, and let $P(s, v, T)$ denote the path in the tree from s to vertex $v \in M$. We say that the tree T is *minimal with respect to* $\lessdot$ if for every vertex $v \in M$ the path $P(s, v, T)$, which is obviously a shortest path, is minimal with respect to $\lessdot$.

Preprocessing algorithm, step 1'. The algorithm fixes some vertex $s \in M$, and gives the vertices unique identifiers as described above. It then constructs a shortest-path tree T rooted at $s \in M$ that is minimal with respect to $\lessdot$, by invoking Dijkstra's single-source shortest-path algorithm from s, with the following slight modification. When there is a tie, namely, the algorithm has to choose an edge (x, y) that minimizes $\mathbf{d_G}(s, x, T) + \omega(x, y)$, then among all the edges achieving the minimum, the algorithm selects the (unique) one for which the path $P(s, x_i, T)$ is minimal with respect to $\lessdot$.

▶ **Claim 3.1.** *A tree T constructed as above is indeed minimal with respect to $\lessdot$.*

We now define a total order on the vertices induced by $\lessdot$ and T as follows. We say that $v \lessdot_T u$ if either (i) v and u are not related and $P(s, v, T) \lessdot P(s, u, T)$; or (ii) v and u are related and v is a descendant of u. The algorithm assigns every vertex v a number $O_T(v)$ from $[1..m]$ such that $O_T(u) < O_T(v)$ iff $u \lessdot_T v$.

In addition, the algorithm assigns every ordered edge $e = (y, x)$ a number $\hat{O}_T(e) \in [1..3m]$ such that for two edges $e = (y, x)$ and $e' = (y', x')$, we have $\hat{O}_T(e) < \hat{O}_T(e')$ iff $O_T(x) < O_T(x')$ or $O_T(x) = O_T(x')$ and $O_T(y) < O_T(y')$.

The rest of the preprocessing phase is similar to the unique-distances case, with the slight modification that every vertex v (resp., edge e) the algorithm stores, it also stores $O_T(v)$ (resp., $\hat{O}_T(e)$).

3.2 Finding the Query's Region: The General Case

In this section we describe the modifications needed in the zoom-in operation for the general non-unique case. The main difference is in the analysis of the zoom-in operation.

The only modification to the zoom-in operation is as follows. Instead of picking the edge $(y, x) \in \cup_{\mu \in \mathcal{A}_\ell(q)} NA(\mu)$ such that (y, x) is on any shortest path from q to s of maximum

$\mathbf{d_G}(s,x)$ and zooming in to regions of (y,x) on the next level, the algorithm picks the edge (y,x) with minimal $O_T(y)$ among all edges $(y,x) \in \cup_{\mu\in\mathcal{A}_\ell(q)} NA(\mu)$ such that (y,x) is on any shortest path from q to s (not necessarily the path in T).

The following main lemma proves the correctness of the zoom-in operation for the non-unique case (the proof is quite technical and is omitted from this version).

▶ **Lemma 15.** *For every level $\ell \leq \textsc{Level}(\textsc{Home}(q))$, the query vertex q belongs to a region in $\{\textsc{Reg}(\mu) \mid \mu \in \mathcal{A}_\ell(q)\}$.*

4 Lower Bounds

Our approximate NNS scheme, presented in Theorem 1, requires access to an *exact* (rather than approximate) distance oracle, and its space and time complexity bounds depend linearly on the graph's maximum degree Δ. In this section we prove that these two requirements are necessary. The graphs used in our lower bounds are in fact trees (and thus certainly planar). Let $\mathbf{DO}(u,v)$ denote (the answer for) a distance-oracle probe for the distance between points u, v.

4.1 Linear Dependence on the Degree Δ

We first assume access to an exact distance oracle, and prove a lower bound on the NNS worst-case query time, assuming that the space requirement is not prohibitively large. We actually prove a stronger assertion, and bound the NNS query time only by the number of distance-oracle probes, regardless of any other operations; in particular, we allow the NNS query procedure to read the entire data structure!

Consider a c-approximate NNS (randomized) scheme with the following guarantee: When given a planar graph with N vertices and maximum degree $\log N \leq \Delta \leq n$, together with a dataset of n vertices, it produces a data structure of size s. Using this data structure, for every query vertex q, with probability at least $1/2$ it finds q's $c-$approximate nearest neighbor using at most t distance-oracle probes. We are interested in the setting where $N \gg n$, say $N \geq n^2$. The following theorem shows that unless s is huge, the query time t must grow linearly with the maximum degree Δ. Let us justify the above requirements on Δ; the assumption $\Delta \leq n$ is necessary because $t \leq n$ is always achievable, by answering NNS queries using exhaustive search (with no preprocessing); the assumption $\Delta \geq \log N$ is for ease of exposition, and can probably be removed with some extra technical work.

▶ **Theorem 16.** *If $s \leq O(N/(\Delta \log_\Delta n))$ bits, then $t \geq \Omega(\Delta \log_\Delta n)$.*

Outline. We prove the theorem by presenting a single distribution over inputs, which is "hard" for all *deterministic* algorithms. That is, every deterministic algorithm is unlikely to succeed in producing a correct answer, under certain space/time constraints (Lemma 20). A bound for randomized algorithms is achieved by fixing the best possible coins (the easy direction of Yao's minimax principle).

The bound for deterministic algorithms is obtained in three steps. First we assume there is no data structure, i.e., memory size $s = 0$, and show that no deterministic algorithm can succeed with more than a constant probability (Lemma 18). We then amplify the bound by considering a series of query points (Lemma 19), at which point the success probability is so tiny that a small data structure cannot help.

4.1.1 The hard distribution

We specify a distribution over NNS instances, namely, a distribution over tree graphs T of size roughly N and degree roughly Δ, data sets D of size n and a query points q. All but the last level of the tree would be fixed, and the randomization occurs only in the way the leaves are connected.

The fixed part of T: Start with a complete tree of arity Δ and exactly N leaves, which means the tree's depth is $H := \log_\Delta N$. (We shall assume for simplicity that all values are integral, to avoid the standard yet tedious rounding issues.) The dataset D is formed by the n vertices at depth (also called level) $h := \log_\Delta n$, and they are labeled $p_1, \dots, p_n$. Let all edges have unit length, except for the edges at level h, which have length cH. (To extend our results to unweighted trees, replace these edges with paths of corresponding length.) In particular, the distance between every two distinct dataset points is at least $2cH$. Since this part of the tree is fixed, we assume the algorithm "knows it", i.e., it can compute distances without any distance-oracle probes.

The random part of T: The last level $H+1$ of the tree is random; it is constructed by hanging N leaves labeled $\ell_1, \dots, \ell_N$ independently at random. In other words, for each vertex ℓ_i we sample uniformly at random one of the N/Δ nodes at level $H-1$ and connect to it. By standard tail bounds, with probability greater than $1 - 1/n$, at most 2Δ leaves are attached to the same node, so the maximum degree of the graph is $\leq 2\Delta$. We note that this is the only place where we use that $\Delta \geq \log N$. We denote this input distribution by $\mathcal{T}$.

Finally, we need to specify the distribution of query points. Throughout our analysis the query point is chosen uniformly at random from the leaves, namely, a vertex ℓ_q for uniformly random $q \in [N]$. Observe that the nearest neighbor of ℓ_q is the dataset point p_i which is the unique ancestor of ℓ_q at level h. In fact, this p_i is the unique c-approximate nearest neighbor, because $d(\ell_q, p_i) = H - h$ while for $i' \neq i$ we have $d(\ell_q, p_{i'}) \geq 2cH$. Thus, in all these instances, exact NNS is equivalent to c-approximate NNS.

4.1.2 The no-preprocessing case

Let A be a deterministic algorithm that solves c-approximate NNS without any preprocessing, in other words, A has zero space requirements and consists of only a query algorithm. Define T_A as the number of distance-oracle probes that A makes given a query. Under the above input distribution, T_A is a random variable, and our goal is to show that it is likely to be $\Omega(\Delta \log_\Delta n)$. Towards this end, we shall make a few adaptations to T_A and to the algorithm A.

Let T'_A be the number of distance-oracle probes of the form $\mathbf{DO}(\ell_q, \cdot)$. Clearly $T'_A \leq T_A$ so it suffices to bound T'_A. We next show that in effect, we may restrict attention to algorithms that do not probe the distance from ℓ_q to vertices at level bigger than h (i.e., strict descendants of the dataset D).

▶ **Lemma 17.** *There is an algorithm A_1 that probes $\mathbf{DO}(\ell_q, w)$ only for vertices w at level at most h, and with probability* 1 *(i.e., on every instance in the support), $T'_{A_1} \leq T'_A$.*

Proof. Algorithm A_1 simulates A probe by probe, except that when A probes $\mathbf{DO}(\ell_q, w)$ for some w at level bigger than h, algorithm A_1 probes $\mathbf{DO}(\ell_q, p_i)$ where p_i is the dataset point which is the ancestor of w. Now, if p_i is also the ancestor of ℓ_q then p_i is the nearest neighbor and A_1 can output p_i. Otherwise observe that $d(\ell_q, w) = d(\ell_q, p_i) + d(p_i, w)$, so A_1

can compute $d(\ell_q, w)$ and continue the simulation of A. Since A_1 uses one query to simulate a query of A, clearly $T'_{A_1} \le T'_A$.

The remaining thing to specify is how does A_1 find the ancestor of w. Now, if w is not a leaf, then it is part of the fixed graph, and the ancestor is hard-wired into A_1. If w is a leaf, we will assume that the parent p_i is just given to A_1 for free. Formally we allow A_1 to query $DO(w, p_j)$ for various j's until w's ancestor is found, and note that these queries are not counted in T'_{A_1}. ◀

For illustration, consider the case where $n = \Delta$ and $N = n^2$, which means that the tree has depth $H = 2$, and the dataset D lies at level $h = 1$. The algorithm A_1 is given a label for a leaf ℓ_q and has to find its parent. We only consider distance-oracle probes of the form $\mathbf{DO}(\ell_q, p_i)$, which have value 1 if p_i is the nearest neighbor of ℓ_q, and value 3 otherwise. Observe that queries that don't involve ℓ_q carry no information on the parent of ℓ_q, and in queries of the form $\mathbf{DO}(\ell_q, \ell_{q'})$, we effectively replace $\ell_{q'}$ with its parent. The situation is thus identical to searching in an unsorted array of size n. The algorithm "scans" the vertices p_i in some order, which is not deterministic (as it might depend on parents of other leaves) but is independent of the correct answer (the parent of ℓ_q). Therefore, T'_{A_1} is distributed uniformly in $[\Delta]$.

We now return to the general hard distribution $\mathcal{T}$ which follows the same intuition but requires additional technical maneuvers. Our goal is to show that without loss of generality, A_1 could be thought as finding the ancestors of q level by level, starting from the root at level 0 and proceeding down to level $h = \log_\Delta n$. Each level requires a search over Δ items, hence we will obtain a lower bound of $\Omega(\Delta h)$.

Given algorithm A_1, define a new algorithm A_2 as follows. Simulate A_1, but whenever A_1 probes $\mathbf{DO}(\ell_q, w)$, probe instead $\mathbf{DO}(\ell_q, w')$, where w' is the minimum-level ancestor of w for which $\mathbf{DO}(\ell_q, w')$ wasn't probed yet. Now, based on the answer, detect whether ℓ_q is a descendant of w' and proceed according to the case at hand:

- If ℓ_q is a not descendant of w', proceed in the simulation of A_1 by calculating the distance $d(\ell_q, w) = d(\ell_q, w') + d(w', w)$ without probing $\mathbf{DO}(\ell_q, w)$ directly.
- If ℓ_q is a descendant of w', probe the entire path from w' to w (namely, the distance between ℓ_q and each vertex along this path) until you can compute $d(\ell_q, w)$, which could happen by reaching either w itself or a vertex which is not an ancestor of ℓ_q.

We point out two crucial observations. First, A_2 recovers the ancestors of ℓ_q one by one starting from level 0 (the root) down to level h (some dataset point). Second, the extra probes are along the path from the root to ℓ_q, with at most one probe outside that path at each level. This is done only up to level h and without repeating the same probe. Thus in total, A_2 always makes at most $2h = 2\log_\Delta n$ more probes of the form $\mathbf{DO}(\ell_q, \cdot)$ than A_1 does, i.e., $T'_{A_2} \le T'_{A_1} + 2h$. The next lemma analyzes this "well-behaved" algorithm A_2.

▶ **Lemma 18.** $\Pr_{\mathcal{T},q}[T'_{A_2} \le h\Delta/3] \le 1/16$.

Proof. Let $w_0, w_1, \dots, w_h$ denote the ancestors of ℓ_q from level 0 to level h (e.g., w_0 is the root and $w_h \in D$). We say A_2 *recovers* w_i, the first time it queries $DO(\ell_q, w_i)$ and we recall that a key feature of A_2 is that it recovers these vertices one by one.

Denote by X_j, for $j \in [h]$, the number of distance-oracle probes of the form $\mathbf{DO}(\ell_q, \cdot)$ that A_2 makes after recovering w_{j-1} and until recovering w_j. The main observation is that X_j dominates a random value chosen uniformly from $[\Delta]$, even when conditioned on the sequence of probes made prior to recovering w_{j-1}. Indeed, when sampling the location of ℓ_q, the decision which child of w_{j-1} is the ancestor of ℓ_q could be deferred to the moment the children of w_{j-1} are being probed. Hence algorithm A_2 is essentially performing an exhaustive search, akin to searching in an unsorted array. It follows that $\mathbb{E}[X_j] \geq \Delta/2$ and $\mathbb{E}[T'_{A_2}] \geq h\Delta/2$. Moreover, by applying Azuma's inequality (assuming n is large enough), $\Pr[T'_{A_2} \geq h\Delta/3] \leq 1/16$. ◀

Recalling that $T_A \geq T'_A \geq T'_{A_1} \geq T'_{A_2} - 2h$ (always) and assuming $\Delta \geq 24$, we obtain using Lemma 18 that

$$\Pr_{\mathcal{T},q}\left[T_A \leq h\Delta/4\right] \leq \Pr_{\mathcal{T},q}\left[T'_{A_2} - 2h \leq h\Delta/4\right] \leq 1/16. \tag{1}$$

A sequence of queries (with no preprocessing). We say algorithm A (with no preprocessing) *succeeds* on a query ℓ_q if it outputs a correct answer and makes at most $h\Delta/4$ distance-oracle probes. Eq. (1) states that $\Pr_{\mathcal{T},q}[A \text{ succeeds on query } \ell_q] \leq 1/16$. In order to extend the argument to the case with preprocessing we need to decrease that probability to be exponentially small, which we achieve by looking at a sequence of several queries. Let $q_1, \dots, q_m \in [N]$ be chosen uniformly at random and independently.

▶ **Lemma 19.** *For every $m \leq N/(4h\Delta)$,*

$$\Pr_{\mathcal{T},q_1,\dots,q_m}[A \textit{ succeeds on all queries } \ell_{q_1}, \dots, \ell_{q_m}] \leq (\tfrac{1}{8})^m.$$

Proof. The main difficulty here is that there might be dependencies between different query points, e.g., if the algorithm's first probe is $\mathbf{DO}(\ell_6, \ell_7)$, then there is a chance that $q_1 = 6$ and $q_2 = 7$, and we cannot argue the success of A on q_1 and on q_2 are independent. In particular, we cannot assume that A never probes other leaves (other than the query point).

The way we handle it is by sampling the tree using deferred decisions, meaning that we attach every leaf of the tree only when it is needed. Trace the executions of A on query q_i for $i = 1, \dots, m$ one by one, where each execution is restricted to at most $h\Delta/4$ distance-oracle probes. Every time a leaf is probed for the first time (more precisely, the distance from/to that leaf), determine its location by attaching it to a random vertex at level $H - 1$.

Now, assume algorithm A succeeded on queries $q_1, \dots, q_{i-1}$ and consider its execution on q_i. The number of leaves attached prior to this execution is at most $(i-1)h\Delta/4 \leq mh\Delta/4 \leq N/16$. Thus, the probability that a random q_i is one of these leaves is at most $1/16$; if this is not the case, then q_i is completely random leaf, and Eq. (1) applies to it. Thus, assuming A already succeeded on queries $q_1, \dots, q_{i-1}$ the probability it succeeds on query q_i is at most $1/8$. The theorem follows. ◀

4.1.3 Algorithm with preprocessing

We turn to the case where an algorithm can prepare a data structure of size s, and prove a lower bound under the same input distribution as before. Specifically, an input tree is first drawn from the distribution $\mathcal{T}$, and then the tree is processed to create a data structure of s bits. Next, a random leaf ℓ_q is chosen as a query point, and algorithm B, which can read the entire data structure (as "advice"), answers the query. As before, we say that algorithm B *succeeds* on a query ℓ_q if it outputs a correct answer and makes at most $h\Delta/4$ distance-oracle probes.

▶ **Lemma 20.** *If $s \leq N/(4h\Delta)$ then $\Pr_{\mathcal{T},q}[B$ succeeds on a query $q] < 1/2$.*

Proof. Assume for contradiction that B succeeds with probability at least $1/2$. Define a new algorithm A that guesses the s bits of advice at random and then simulates algorithm

B. We will show that this algorithm A (which obviously uses no preprocessing) contradicts Lemma 19.

Now consider sampling a tree from $\mathcal{T}$. In expectation, algorithm B would succeed on at least half the leaves of this tree (as query points), hence by Markov's inequality, with probability at least 1/4 (over the choice of the tree) algorithm B succeeds on at least $1 - \frac{4}{3} \cdot \frac{1}{2} = \frac{1}{3}$ of the leaves. In such a tree, the probability that B succeeds on m random leaves is at least $(\frac{1}{3})^m$.

Recall that Algorithm A guesses the advice strings independently at random. Thus, the probability that A succeeds on all the m leaves of a random tree is at least $(\frac{1}{2})^s \cdot \frac{1}{4} \cdot (\frac{1}{3})^m$. Taking $m = N/(4h\Delta)$ and observing $s \le m$, we have

$$\Pr_{\mathcal{T}, q_1, \ldots, q_m}[A \text{ succeeds on all queries } \ell_{q_1}, \ldots, \ell_{q_m}] \ge \tfrac{1}{4} \cdot (\tfrac{1}{6})^m.$$

Now observe that for N (and hence m) large enough, we have contradicted Lemma 19. ◀

We can now complete the proof of Theorem 16.

Proof of Theorem 16. Assume towards contradiction there is a randomized algorithm C for the c-approximate NNS problem, such that on tree instances with maximum degree at most $2\Delta + 2$, we have (the theorem would then follow by substituting $\Delta' = 2\Delta + 2$): (a) for each query point, with probability at least 3/4, the algorithm answers correctly (a c-approximate nearest neighbor); (b) the space requirement is $s \le N/(4h\Delta) = N/(4\Delta \log_\Delta n)$; and (c) for each query point, the algorithm makes at most $t \le h\Delta/4$ distance-oracle probes.

By fixing the coins of algorithm C optimally, it immediately follows there exists some *deterministic* algorithm B that achieves $\Pr_{\mathcal{T},q}[B \text{ succeeds on a query } q] \ge \frac{3}{4}$, in addition to satisfying the space requirement (b) and query time bound (c), which contradicts Lemma 20. The theorem follows. ◀

4.2 Approximate Distance Oracles

We now sketch the argument claiming it is essential to have an exact distance oracle (rather than an approximate one). Suppose the distance oracle provides a $(1 + \delta)$ multiplicative approximation of the distance for some $\delta > 0$. We show an instance with the following properties:

- The total number of nodes is $O(n^2)$ and the maximum degree is $O(\log n)$.
- The aspect ratio across edge weights is $\max\{(2 \log n)/\delta, (1 + \epsilon) \log n\}$.
- If the space of the data structure is $\le n^2$ bits then the number of distance-oracle probes needed is $\Omega(n)$.

The construction is as follows. Build a binary tree of height $\log n$ from root s, so the binary tree has n leaves, and call this part of the graph the *top tree*. Now, from each leaf of the top tree hang an edge of length $\max\{2 \log n/\delta, (1 + \epsilon) \log n\}$. To simplify the exposition, assume this maximum is $2 \log n/\delta$. The bottom nodes of these edges are labeled $p_1, \ldots, p_n$ and these are the dataset points. Finally, from each dataset point hang a binary tree of depth $\log n$, so we have a total of n^2 leaves to which we hang random n^2 nodes labeled $l_1, \ldots, l_{n^2}$, thus creating nodes of maximum degree $O(\log n)$ with high probability. We call these trees the *bottom trees*.

Observe that for each leaf l_i there is one p_j for which $d(l_i, p_j) = \log n$ while for $k \ne j$ $d(l_i, p_k) \ge (1 + \epsilon) \log n$. Thus, for a query l_i the algorithm must output p_j, which is the unique $(1+\epsilon)$-approximate nearest neighbor. When probed for $\mathbf{DO}(u, v)$, the distance oracle answer is as follows:

- If both u, v belong to the top tree or to the *same* bottom tree, the oracle answers with the exact distance between them.
- If u belongs to a bottom tree and v belongs to a top tree, the oracle answers $2\log n/\delta$. Note that the exact distance is in the range $[2\log n/\delta, (1+\delta)2\log n/\delta]$.
- If u and v belong to different bottom trees, the oracle outputs $4\log n/\delta$. Again, observe that the correct distance is in the range $[4\log n/\delta, (1+\delta)4\log n/\delta]$.

The way the distance oracle is set up, the NNS algorithm faces a situation which is similar to the case where the root has degree n, and is connected to the all the dataset points by distinct edges. In this case the total size of the graph is n^2. The proof of the previous section essentially shows that unless the data structure is of size roughly n^2, the query time is $\Omega(\Delta) = \Omega(n)$.

References

1 Ittai Abraham, Shiri Chechik, and Cyril Gavoille. Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels. In *Proceedings of the 44th symposium on Theory of Computing*, pages 1199–1218. ACM, 2012.

2 Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. Hierarchical hub labelings for shortest paths. In *Proceedings of the 20th Annual European conference on Algorithms*, ESA'12, pages 24–35. Springer-Verlag, 2012.

3 Ittai Abraham and Cyril Gavoille. Object location using path separators. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC'06, pages 188–197. ACM, 2006.

4 Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. New data structures for orthogonal range searching. In *41st Annual Symposium on Foundations of Computer Science*, pages 198–207, 2000.

5 Eyal Amir. Approximation algorithms for treewidth. *Algorithmica*, 56(4):448–479, January 2010.

6 Holger Bast, Stefan Funke, Peter Sanders, and Dominik Schultes. Fast routing in road networks with transit nodes. *Science*, 316(5824):566, 2007.

7 Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

8 A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *23rd international conference on Machine learning*, pages 97–104. ACM, 2006.

9 Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, December 1996.

10 Hans L. Bodlaender, Pal Gronas Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. An $O(c^k n)$ 5-approximation algorithm for treewidth. In *54th Annual Symposium on Foundations of Computer Science*, pages 499–508, 2013.

11 Hans L. Bodlaender, John R. Gilbert, Hjálmtýr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *J. Algorithms*, 18(2):238–255, March 1995.

12 R. Cole and L.-A. Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *38th Annual ACM Symposium on Theory of Computing*, pages 574–583. ACM, 2006.

13 David Eisenstat, Philip N. Klein, and Claire Mathieu. Approximating k-center in planar graphs. In *25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 617–627. SIAM, 2014.

14 Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *Proceedings of the 7th International Conference on Experimental Algorithms*, WEA'08, pages 319–333. Springer-Verlag, 2008.

15 S. Har-Peled and M. Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006.

16 Goos Kant and Hans L. Bodlaender. Triangulating planar graphs while minimizing the maximum degree. *Inf. Comput.*, 135(1):1–14, May 1997.

17 D. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *34th Annual ACM Symposium on the Theory of Computing*, pages 63–66, 2002.

18 R. Krauthgamer and J. R. Lee. Navigating nets: Simple algorithms for proximity search. In *15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 791–801, January 2004.

19 R. Krauthgamer and J. R. Lee. The black-box complexity of nearest-neighbor search. *Theoret. Comput. Sci.*, 348(2-3):262–276, 2005.

20 R. Krauthgamer, H. Nguyen, and T. Zondiner. Preserving terminal distances using minors. *SIAM Journal on Discrete Mathematics*, 28(1):127–141, 2014.

21 R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, 1979.

22 C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory Comput. Syst.*, 32(3):241–280, 1999.

23 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, November 2004.

How to Tame Rectangles: Solving Independent Set and Coloring of Rectangles via Shrinking

Anna Adamaszek[1], **Parinya Chalermsook**[2], and **Andreas Wiese**[2]

1 **Department of Computer Science (DIKU), University of Copenhagen, Denmark,** `anad@di.ku.dk`

2 **Max-Planck-Institut für Informatik, Saarbrücken, Germany,** `{parinya,awiese}@mpi-inf.mpg.de`

Abstract

In the Maximum Weight Independent Set of Rectangles (MWISR) problem, we are given a collection of weighted axis-parallel rectangles in the plane. Our goal is to compute a maximum weight subset of pairwise non-overlapping rectangles. Due to its various applications, as well as connections to many other problems in computer science, MWISR has received a lot of attention from the computational geometry and the approximation algorithms community. However, despite being extensively studied, MWISR remains not very well understood in terms of polynomial time approximation algorithms, as there is a large gap between the upper and lower bounds, i.e., $O(\log n/\log\log n)$ v.s. NP-hardness. Another important, poorly understood question is whether one can color rectangles with at most $O(\omega(\mathcal{R}))$ colors where $\omega(\mathcal{R})$ is the size of a maximum clique in the intersection graph of a set of input rectangles $\mathcal{R}$. Asplund and Grünbaum obtained an upper bound of $O(\omega(\mathcal{R})^2)$ about 50 years ago, and the result has remained asymptotically best. This question is strongly related to the integrality gap of the canonical LP for MWISR.

In this paper, we settle above three open problems in a relaxed model where we are allowed to shrink the rectangles by a tiny bit (rescaling them by a factor of $(1-\delta)$ for an arbitrarily small constant $\delta > 0$.) Namely, in this model, we show (i) a PTAS for MWISR and (ii) a coloring with $O(\omega(\mathcal{R}))$ colors which implies a constant upper bound on the integrality gap of the canonical LP.

For some applications of MWISR the possibility to shrink the rectangles has a natural, well-motivated meaning. Our results can be seen as an evidence that the shrinking model is a promising way to relax a geometric problem for the purpose of better algorithmic results.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Approximation algorithms, independent set, resource augmentation, rectangle intersection graphs, PTAS

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.43

1 Introduction

The main motivation of this paper is to study barriers in designing approximation algorithms for the Maximum Weight Independent Set of Rectangles (MWISR) problem and propose a way to break them. In this problem, we are given a collection of weighted axis-parallel rectangles in the plane, and our goal is to select a maximum weight subset of pairwise non-overlapping rectangles. Besides being a special case of Maximum Independent Set, which has been one of the most extensively studied problems in combinatorial optimization, MWISR is a fundamental geometric problem in itself. The problem arises in multiple applications and has connections to other problems in various areas of computer science, such as map labeling [4], data mining [19], networking [26], and pricing [14]. Therefore, it is not surprising that MWISR has received a significant amount of attention from researchers in both computational geometry and approximation algorithms communities.

While for Maximum Independent Set in general it is NP-hard to obtain approximation ratios of $n^{1-\epsilon}$ for any $\epsilon > 0$ [22, 28], much better approximation ratios are possible for MWISR. Agarwal, van Kreveld and Suri first proposed the problem with tentative applications in map labeling, where they showed the first $O(\log n)$-approximation algorithm [4]. Since then, a significant amount of research has been done in various directions: (i) Proposing $O(\log n)$ approximation algorithms with faster running times [7, 12, 27] or (ii) Showing approximation schemes or constant factor approximation algorithms for special cases, when the input rectangles are squares [16, 11], unit-height rectangles [24], or have restricted positions [26, 15]. Currently the best result for the general case is a $O(\log n/\log\log n)$-approximation by Chan and Har-Peled [13]. When all rectangles have unit weights, Chalermsook and Chuzhoy [10] present an $O(\log\log n)$-approximation algorithm. A much better approximation is possible for super-polynomial time algorithms. Recently, Adamaszek and Wiese [1] showed a quasi-polynomial time approximation scheme for MWISR, thus showing that the problem cannot be APX-hard unless $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{\mathrm{poly}(\log n)})$.

Despite extensive effort of various groups of researchers, the approximability status of MWISR has so far remained elusive. On one hand, the existence of the recent QPTAS suggests that a PTAS is possible, but on the other hand, even a sub-logarithmic approximation has not been obtained for two decades. No substantial progress in the lower bound has been made, and even for the integrality gap of the natural LP relaxation we only have a lower bound of 2!

Closely related to MWISR (and notoriously hard) is the question of rectangle coloring. In this problem, we are given a collection of axis-parallel rectangles in the plane, and the goal is to color the rectangles so that intersecting rectangles have different colors, while minimizing the number of colors used. In 1948 [8] Bielecki asked whether one can bound the number of colors in such a coloring by the clique size of the intersection graph of the input rectangles. Denote the clique size of the intersection graph of $\mathcal{R}$ by $\omega(\mathcal{R})$. In 1960, Asplund and Grunbaum [6] showed that at most $O(\omega(\mathcal{R})^2)$ colors are needed. This status has not changed for half a century. The upper bound of $O(\omega(\mathcal{R})^2)$ is still asymptotically the best known result, while the best known lower bound is $3\omega(\mathcal{R})$ [6]. Closing this gap is seen as a challenging open problem in discrete mathematics (see, e.g., a survey by Kostochka [25]).

The state of the art of these two problems gives convincing evidence that rectangle problems are hard to deal with, and clearly new insights are needed.

1.1 A Relaxed Model: Shrinkable Rectangles

Motivated by the barriers of designing approximation algorithms for MWISR, we study a slight relaxation of the problem. Instead of computing a set of pairwise non-overlapping rectangles, we allow our algorithm to output a subset of rectangles that is *almost feasible* in the following sense. The subset of the rectangles must be pairwise non-overlapping *after* we shrink each rectangle by a multiplicative factor of $1-\delta$ for some small constant $\delta > 0$. Formally, this means that a rectangle $(a, a+x) \times (b, b+y)$ will become $(a+\frac{\delta}{2}x, a+(1-\frac{\delta}{2})x) \times (b+\frac{\delta}{2}y, b+(1-\frac{\delta}{2})y)$. We compare the value of our (almost feasible) solution to the value of an optimal feasible solution. We call this problem *δ-MWISR*. Observe that δ-MWISR remains NP-hard (see Appendix A for a proof). We remark that similar models have been studied before. In particular, Har-Peled and Lee showed approximation algorithms for geometric set cover problems for fat objects when the input objects are allowed to expand slightly [21]. In fact, this relaxed model still serves the purposes of many applications such as map labeling where it is tolerable to slightly shrink the rectangles without losing much benefit.

1.2 Our Contributions

We solve three long-standing open problems in the domain of rectangle intersection graphs in our new model. First, we give a polynomial time approximation scheme for δ-MWISR while, as mentioned above, the best known polynomial time algorithm in the ordinary setting has a superconstant approximation ratio of $O(\log n/\log\log n)$.

▶ **Theorem 1.** *Let $\epsilon, \delta > 0$ be any constants. There is a $(1+\epsilon)$ approximation algorithm for δ-MWISR that runs in time $n^{(\frac{1}{\epsilon\delta})^{O(1/\epsilon)}}$.*

The core of this result is a plane cutting procedure that follows the framework of [1]. The high-level idea is that we recursively partition the input plane into a collection of axis-parallel polygons. Rectangles overlapping the boundaries of the partition are lost. In [1], it has been shown that for any set of pairwise non-overlapping rectangles there exists such a cutting sequence where only an ϵ-fraction of all rectangles (or rectangles of small total weight) is cut and the maximum complexity of a polygon arising in this sequence is bounded by $(\log n/\epsilon)^{O(1)}$. When guessing this cut sequence recursively, we obtain an $(1+\epsilon)$-approximation algorithm with a running time of $n^{(\log n/\epsilon)^{O(1)}}$, i.e., quasi-polynomial. For our relaxed model, we construct a totally different cut sequence,

where any polygon arising in this sequence has *constant complexity*, and still only an ϵ-fraction of the overall weight is lost. Therefore, when embedding the search for this cut sequence into a dynamic program, we obtain a *polynomial* time approximation scheme.

Next, we study the rectangle coloring problem. Let us first give a formal statement of the problem. For any collection $\mathcal{R}$ of axis-parallel rectangles in the plane, one can define an intersection graph $G = (V, E)$ by introducing one vertex in V for each rectangle in $\mathcal{R}$ and connecting two vertices if and only if their corresponding rectangles overlap. We denote by $\omega(\mathcal{R})$ the clique number of the resulting intersection graph of $\mathcal{R}$ and by $\chi(\mathcal{R})$ its chromatic number. For rectangles, the clique number is identical to the minimum number q such that any point in the plane is contained in at most q rectangles. Clearly, $\chi(\mathcal{R}) \geq \omega(\mathcal{R})$. The main open question is whether $\chi(\mathcal{R}) = O(\omega(\mathcal{R}))$ for any collection of rectangles $\mathcal{R}$.

The relation between $\chi(\mathcal{R})$ and $\omega(\mathcal{R})$ is also interesting in our model. We now want to compute a minimum number of colors c for which there exists a c-coloring of the rectangles such that after the shrinking operation rectangles with the same color are pairwise non-overlapping. We prove the following result.

▶ **Theorem 2.** *For any $\delta > 0$, any collection of axis-parallel rectangles $\mathcal{R}$ in the plane can be colored with $O((\frac{1}{\delta})^2\log^2(\frac{1}{\delta}))\omega(\mathcal{R})$ colors, such that after shrinking each rectangle by a multiplicative factor of $(1-\delta)$ the resulting rectangles with the same color are pairwise non-overlapping. Moreover, we can compute such a coloring in polynomial time.*

We prove this theorem by showing a rather general partitioning lemma that splits any collection of rectangles into $O((\frac{1}{\delta})^2\log^2(\frac{1}{\delta}))$ sub-collections. Each of the resulting collections has the property that its rectangles can be shrunk by a factor of at most $(1-\delta)$ such that any two overlapping rectangles are either contained in one another or they do not overlap on a corner, i.e., they cross each other. It has been shown in [9] (building on the previous work [6, 13, 26]) that such collections of rectangles $\mathcal{R}'$ admit a coloring algorithm with at most $\omega(\mathcal{R}')$ colors. This gives us the desired result.

Due to a connection between coloring and the integrality gap of the natural LP-relaxation of MWISR (see, e.g, [9]), we obtain the following corollary (in fact, our partitioning lemma also yields this directly). We will define this relaxation formally in Section 3.

▶ **Corollary 3.** *The integrality gap for the natural LP relaxation for δ-MWISR is at most $O((\frac{1}{\delta})^2 \log^2(\frac{1}{\delta}))$ and there is a polynomial time $O((\frac{1}{\delta})^2 \log^2(\frac{1}{\delta}))$-approximation algorithm for δ-MWISR that rounds this LP.*

1.3 Other Related Work

The framework of Adamaszek and Wiese has been further extended in [2, 20] to give a QPTAS for the maximum independent set of polygons in general. In polynomial time, the best result is a n^{ϵ}-approximation by Fox and Pach [18] for independent set of arbitrary curves in the plane. For the rectangle coloring problem better bounds are known for some special cases of rectangles [26, 3]. Also, a small improvement over Asplund and Grünbaum was discussed in [23]. We refer the readers to a nice survey by Kostochka for a more complete literature on the coloring problem for other objects [25].

Finally, we remark that special cases of both MWISR and rectangle coloring when intersection patterns are restricted are much simpler than the general problem. When one rectangle is not allowed to contain any corner of another, the intersection graph is a perfect graph; therefore both problems are polynomial time solvable (see, e.g., [13, 26]).

1.4 Problem Definition and Notation

We are given a set of n axis-parallel rectangles $\mathcal{R} = \{R_1, \ldots, R_n\}$ in the plane. Each input rectangle R_i is specified by an open set $R_i := \{(x,y) | x_i^{(1)} < x < x_i^{(2)} \wedge y_i^{(1)} < y < y_i^{(2)}\}$ together with its weight w_i[1]. For each rectangle R_i we denote its *width* and *height* by $g_i := |x_i^{(1)} - x_i^{(2)}|$ and $h_i = |y_i^{(1)} - y_i^{(2)}|$, respectively. We say that a subset of rectangles $\mathcal{S} \subseteq \mathcal{R}$ is an *independent set* if every pair of rectangles $R_i, R_j \in \mathcal{S}$ satisfies $R_i \cap R_j = \emptyset$.

Our model uses the following relaxed notion of an independent set. For $R_i \in \mathcal{R}$, a δ-shrunk rectangle $R_i^{-\delta}$ is defined by the x-coordinates $x_i^{(1)} + \frac{1}{2}\delta g_i$ and $x_i^{(2)} - \frac{1}{2}\delta g_i$, and the y-coordinates $y_i^{(1)} + \frac{1}{2}\delta h_i$ and $y_i^{(2)} - \frac{1}{2}\delta h_i$ respectively. Then, for any subset $\mathcal{S} \subseteq \mathcal{R}$ of rectangles, we denote by $\mathcal{S}^{-\delta}$ the collection of δ-shrunk rectangles of $\mathcal{S}$, i.e., $\mathcal{S}^{-\delta} = \{R_i^{-\delta} : R_i \in \mathcal{S}\}$. We say that a subset $\mathcal{S} \subseteq \mathcal{R}$ is a *δ-independent set* if $\mathcal{S}^{-\delta}$ is an independent set.

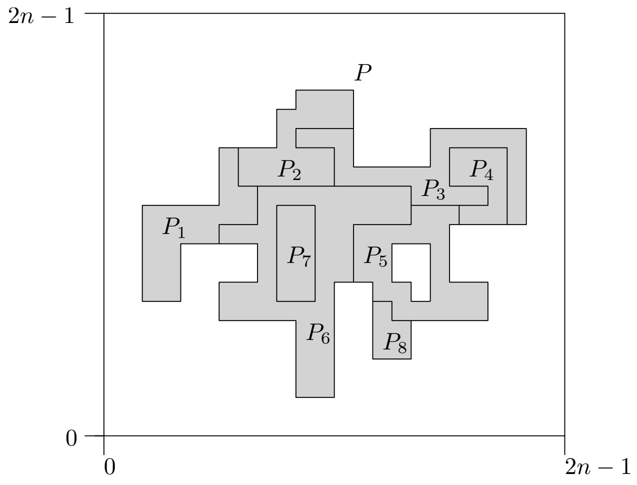
Now we define our problems formally. In δ-MWISR our goal is to find a maximum weight subset $\mathcal{S} \subseteq \mathcal{R}$ that is δ-independent. For the coloring problem, we define a δ-chromatic number, denoted by $\chi^{-\delta}(\mathcal{S})$, of a collection $\mathcal{S} \subseteq \mathcal{R}$ as the minimum integer c such that rectangles in $\mathcal{S}$ can be colored using c colors so that rectangles with the same color form a δ-independent set. Our goal is to bound $\chi^{-\delta}(\mathcal{R})$ in terms of $\omega(\mathcal{R})$[2].

2 Approximation Scheme for Independent Set

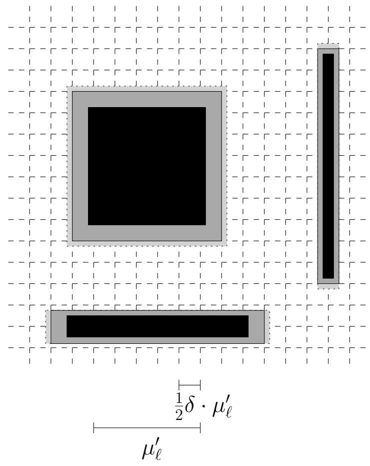
In this section, we present a polynomial time approximation scheme for δ-MWISR for any constant $\delta > 0$. More precisely, for any constants $\epsilon > 0$ and $\delta > 0$, we present a $(1+\epsilon)$-approximation algorithm for δ-MWISR with a running time of $n^{(\frac{1}{\epsilon\delta})^{O(1/\epsilon)}}$. Denote by $N := \max_i\{x_i^{(1)}, y_i^{(1)}, x_i^{(2)}, y_i^{(2)}\}$. Suppose for now that N is bounded by a polynomial in n. We will show later how to remove this assumption.

[1] By linear scaling we can assume the rectangle coordinates to be integers, even if in the actual input we are given rationals.

[2] Notice that there is a collection $\mathcal{R}$ for which the lower bound of $\omega(\mathcal{R})$ still holds, e.g., consider a collection of identical rectangles.



(a) Partitioning of polygons



(b) Shrinking of rectangles

Figure 1 Illustration of our algorithm. Figure 1a denotes the partition of polygon P (grey area) into at most k smaller polygons, each with at most k edges. In Figure 1b depicted in light grays are original rectangles R_i from the input, in dark gray the slightly shrunk rectangles R'_i, and in black the shrunk rectangles.

2.1 The Algorithm

Our algorithm, called GEO-DP, is exactly the same dynamic program which was used in [1] for obtaining a QPTAS for MWISR (without the shrinking assumption). GEO-DP is parametrized by a value $k \in \mathbb{N}$. For our purposes, we will later choose $k := (\frac{1}{\epsilon\delta})^{O(1/\epsilon)}$.

Fix a parameter $k \in \mathbb{N}$. Let $\mathcal{P}$ denote the set of all simple polygons within the $[0,N]\times[0,N]$ input square whose corners have only integer coordinates, and which have at most k axis-parallel edges each. We introduce a DP-cell for each polygon $P \in \mathcal{P}$, where a cell corresponding to P stores a near-optimal solution $sol(P) \subseteq \mathcal{R}_P^{-\delta}$, where $\mathcal{R}_P^{-\delta}$ denotes the set of all rectangles from $\mathcal{R}^{-\delta}$ contained in P. Here, near-optimal means with respect to the optimal solution using (original) rectangles from $\mathcal{R}$ contained in P.

▶ **Proposition 4.** *The number of DP-cells is at most $N^{O(k)}$.*

To compute the solution $sol(P)$ for some polygon $P \in \mathcal{P}$ we use the following procedure. If $\mathcal{R}_P^{-\delta} = \emptyset$ or $|\mathcal{R}_P^{-\delta}| = 1$ then we set $sol(P) := \mathcal{R}_P^{-\delta}$ and terminate. Otherwise, we enumerate all possibilities to partition P into k' polygons $P_1, \ldots, P_{k'} \in \mathcal{P}$ such that $k' \leq k$. See Figure 1a for an illustration. Since by Proposition 4 we have $|\mathcal{P}| \leq N^{O(k)}$, the number of potential partitions we need to consider is upper bounded by $\binom{N^{O(k)}}{k} = N^{O(k^2)}$. Let $P_1, \ldots, P_{k'}$, where $k' \leq k$, be a feasible partition, i.e., each P_j has at most k edges and they form a partition of $\mathcal{P}$. For any enumerated set $\{P_1, \ldots, P_{k'}\} \subseteq \mathcal{P}$, one can efficiently verify whether this is a feasible partition since all polygons have axis-parallel edges with integer coordinates in $\{0, \ldots, N\}$. For each polygon $P_i \in \{P_1, \ldots, P_{k'}\}$ we look up the DP-table value $sol(P_i)$ and compute $\sum_{i=1}^{k'} w(sol(P_i))$. We set $sol'(P) := \bigcup_{i=1}^{k'} sol(P_i)$ for the partition $\{P_1, \ldots, P_{k'}\}$ which yields the maximum profit. Now we define $sol(P) := sol'(P)$ if $w(sol'(P)) > \max_{R\in\mathcal{R}_P^{-\delta}} w(R)$, and otherwise $sol(P) := \{R_{\max}\}$ where $R_{\max} \in \mathcal{R}_P^{-\delta}$ is the rectangle with maximum weight in $\mathcal{R}_P^{-\delta}$. At the end, the algorithm outputs the value in the DP-cell which corresponds to the polygon containing the entire input region $[0,N] \times [0,N]$.

Thus, the entry in the DP-table for each polygon $\mathcal{P}$ can be computed in time $N^{O(k^2)}$, assuming that all entries for all polygons $\mathcal{P}' \subsetneq \mathcal{P}$ have been computed already. Since we have $|\mathcal{P}| \leq N^{O(k)}$, we get the following upper bound on the running time of GEO-DP.

▶ **Proposition 5.** *When parametrized by k the running time of GEO-DP is upper bounded by $N^{O(k^2)}$.*

For bounding the approximation ratio of GEO-DP for any parameter k, it is sufficient to consider only the special case that the input set $\mathcal{R}$ is already an optimal feasible solution. This can be proven formally by induction on the DP-cells. For $\mathcal{R}^* \subseteq \mathcal{R}$ being the optimal solution, we can prove that when GEO-DP is given $\mathcal{R}$ as input, the value for each DP-cell is at least as high as when given $\mathcal{R}^*$ as input. Therefore, we will assume from now on in our whole argumentation about GEO-DP that $\mathcal{R}$ is already the (optimal) independent set.

2.2 A Suitable Shrunk Solution

Consider $\epsilon, \delta > 0$ such that $\epsilon\delta < 1$. We define $k := (\frac{1}{\delta\epsilon})^{O(1/\epsilon)}$ and show that for this choice of the parameter, GEO-DP yields a $(1+\epsilon)$-approximate solution for δ-MWISR. Starting with an optimal solution $\mathcal{R}^* \subseteq \mathcal{R}$ for the (non-shrunk) input set $\mathcal{R}$, we first define a $(1+\epsilon)$-approximative set $\mathcal{R}'$ consisting of one rectangle R_i' for each rectangle $R_i \in \mathcal{R}^*$ such that $R_i^{-\delta} \subseteq R_i' \subseteq R_i$. Then, in the second step, we show that if the input consisted only of $\mathcal{R}'$, then GEO-DP would compute the whole set $\mathcal{R}'$ as a feasible solution. This implies that GEO-DP finds a $(1+\epsilon)$-approximate solution for δ-MWISR.

Now we start with the description of the first step. Let $\mathcal{R}^* \subseteq \mathcal{R}$ be the maximum weight set of pairwise non-overlapping rectangles, i.e., where $w(\mathcal{R}^*) = \mathsf{OPT}$. Assume for simplicity that $1/\epsilon$ and $1/\delta$ are integers. We partition the rectangles of $\mathcal{R}$ into $O_{\delta,\epsilon}(\log N)$ groups $\mathcal{R}_\ell$, according to the lengths of their respective longer edge (where $O_{\delta,\epsilon}$ hides constants that depend only on ϵ and δ). Using standard shifting techniques (see, e.g., Hochbaum and Maas [24]), by losing only a factor of $1+\epsilon$ in our objective function, we can assume that for any two rectangles in different groups, the lengths of their respective longer edge differ at least by a factor of $\frac{1}{\epsilon\delta}$, and for any two rectangles in the same group they differ at most by a factor of $(\frac{1}{\epsilon\delta})^{1/\epsilon}$.

▶ **Lemma 6.** *By losing a factor of $1+\epsilon$ in the value of the optimal solution, we can assume that there is a partition of the rectangles $\mathcal{R}$ into $O(\log N)$ groups $\mathcal{R}_\ell$ and values $\mu_\ell', \mu_\ell \in \mathbb{N}$ for each group $\mathcal{R}_\ell$ such that*

- *$\mu_\ell' \leq \max\{g_i, h_i\} < \mu_\ell$ for each $R_i \in \mathcal{R}_\ell$ (recall that g_i and h_i are width and height of rectangle R_i respectively), and*
- *$\delta\epsilon \cdot \mu_\ell' = \mu_{\ell+1}$ and $\mu_\ell/\mu_\ell' = (1/\delta\epsilon)^{1/\epsilon}$ for each ℓ.*

Proof. We first group rectangles in $\mathcal{R}$ into $\mathcal{R}_1, \ldots, \mathcal{R}_m$ for $m = O(\log N)$ based on their values $v_i = \max\{h_i, g_i\}$, where $\mathcal{R}_j = \{R_i : v_i \in [(1/\delta\epsilon)^{j-1}, (1/\delta\epsilon)^j)\}$. Then, we again group every $1/\epsilon$ consecutive groups $\mathcal{R}_j$ together to obtain *supergroups*. We define supergroups with respect to different values of "shifts" as follows. For each shift $s \in \{1, \ldots, 1/\epsilon\}$, the supergroup $\mathcal{T}_{s,0} = \bigcup_{j=1}^{s-1} \mathcal{R}_j$ and for each $\alpha \geq 1$, we have $\mathcal{T}_{s,\alpha} = \bigcup_{j=s+(\alpha-1)/\epsilon+1}^{s+\alpha/\epsilon-1} \mathcal{R}_j$. Notice that for each fixed s, if we take the union of supergroups $\mathcal{T}_{s,\alpha}$, we would get $\mathcal{T}_s = \bigcup_\alpha \mathcal{T}_{s,\alpha} = \bigcup_{j:j \not\equiv s \pmod{1/\epsilon}} \mathcal{R}_j$.

▶ **Observation 7.** $\sum_{s=1}^{1/\epsilon} \mathsf{OPT}(\mathcal{T}_s) \geq (1-\epsilon)\mathsf{OPT}/\epsilon$.

Proof. Let $\mathcal{R}^*$ be an optimal solution. We argue that

$$\sum_{s=1}^{1/\epsilon} w(\mathcal{T}_s \cap \mathcal{R}^*) \geq (1-\epsilon)w(\mathcal{R}^*)/\epsilon$$

Notice that each rectangle $R_i \in \mathcal{R}^*$ appears in $(1/\epsilon) - 1$ terms on the left-hand-side (more precisely, if $R_i \in \mathcal{R}_j$ where $j = s \pmod{1/\epsilon}$, then the contribution from rectangle R_i does not appear). The claim then follows. ◀

Then there must be a shift $s \in \{0, \ldots, 1/\epsilon - 1\}$ such that $w(\mathcal{T}_s \cap \mathcal{R}^*) \geq (1-\epsilon)w(\mathcal{R}^*)$. We complete the proof of this lemma by observing that for each s, the collection $\mathcal{T}_s$ has the following properties:

- For any α, for any two rectangles $R_i, R_{i'} \in \mathcal{T}_{s,\alpha}$, we have $v_i/v_{i'} \leq (1/\delta\epsilon)^{(1/\epsilon)}$.
- For two integers $\alpha < \alpha'$, for rectangles $R_i \in \mathcal{T}_{s,\alpha}, R_{i'} \in \mathcal{T}_{s,\alpha'}$, we have $h_{i'}/h_i \geq \frac{1}{\delta\epsilon}$.

◀

The readers may think of the values $\mu_0, \mu'_0, \mu_1, \ldots, \mu'_q$ as being the values $N, N(\delta\epsilon)^{1/\epsilon}$, $N(\delta\epsilon)^{1+1/\epsilon}, N(\delta\epsilon)^{1+2/\epsilon}, \ldots$. Next, we place a grid with a random offset in the plane. Let $a \in \{0, \ldots, \mu_0 - 1\}$ be a random offset. We draw the grid cells of various granularities, and we use the notion of *levels* to indicate the granularities of the cells. Denote by G_ℓ the grid of level ℓ. Each grid cell of G_ℓ has a width and height of $w_\ell = 2\delta \cdot \mu'_\ell$ and there is one grid cell whose top left corner has the coordinates (a, a). More formally, the horizontal (resp. vertical) grid lines at level ℓ are those with y-coordinates (resp. x-coordinates) $a, a + w_\ell, a + 2w_\ell, \ldots$. Observe that each grid line in G_ℓ is a also a grid line in $G_{\ell'}$ whenever $\ell' > \ell$.

For each set $\mathcal{R}_\ell$ we remove all rectangles which are intersected by a grid $G_{\ell'}$ with $\ell' < \ell$. The next lemma shows that this comes at a negligible cost, by exploiting the fact that the grid granularity $w_{\ell'}$ of each grid $G_{\ell'}$ is at least by a factor of $1/\epsilon$ larger than $\max\{g_i, h_i\}$ for any rectangle R_i in a set $\mathcal{R}_\ell$ with $\ell' < \ell$, and the fact that a was a random offset.

▶ **Lemma 8.** *Let $\epsilon > 0$ be any constant. There is a randomized algorithm that, given a collection $\mathcal{R}$ of rectangles, produces a new collection $\mathcal{R}' \subseteq \mathcal{R}$ together with grid lines $\{G_\ell\}$ such that no rectangle in group $\mathcal{R}_\ell \cap \mathcal{R}'$ is intersected by grid lines $G_{\ell'}$ for $\ell' < \ell$. Moreover,* $\mathsf{OPT}(\mathcal{R}') \geq (1-\epsilon)\mathsf{OPT}(\mathcal{R})$ *in expectation.*

Proof. We first argue that, for any $\ell' < \ell$, the probability that a rectangle $R_i \in \mathcal{R}_\ell$ is intersected by a grid line of $G_{\ell'}$ is at most $\epsilon^{\ell-\ell'}$: Consider a rectangle $R_i \in \mathcal{R}_\ell$. Two consecutive parallel grid lines of the grid $G_{\ell'}$ have a distance of $w_{\ell'} = 2\delta\mu'_{\ell'} > \frac{2}{\epsilon^{\ell-\ell'}}\mu_\ell > \frac{2}{\epsilon^{\ell-\ell'}}\max\{g_i, h_i\}$. Therefore, the probability that R_i is intersected by a horizontal grid line of $G_{\ell'}$ is at most $\epsilon^{\ell-\ell'}/2$; similarly, the probability that R_i is intersected by a vertical grid line of $G_{\ell'}$ is at most $\epsilon^{\ell-\ell'}/2$. By the union bound the probability that R_i is intersected by some grid line of $G_{\ell'}$ is bounded by $\epsilon^{\ell-\ell'}$.

Now let $\mathcal{R}^*$ be an optimal solution. Observe that any rectangle $R \in \mathcal{R}_\ell \cap \mathcal{R}^*$ is removed if it intersects some a grid line of $G_{\ell'}$ with $\ell' < \ell$. So the probability that R is removed from the instance is, by the union bound, at most $\sum_{\ell':\ell'>\ell} \epsilon^{\ell'-\ell} \leq 2\epsilon$. Therefore, in expectation, the total weight of the remaining rectangles in $\mathcal{R}^*$ is at least $(1-2\epsilon)w(\mathcal{R}^*) \geq \frac{1}{(1+3\epsilon)}w(\mathcal{R}^*)$. ◀

We remark that if N is polynomially bounded in the number of input rectangles, our algorithm does not need to execute this lemma; only the existential statement is sufficient for the DP to find a good solution. The lemma is only needed when N is superpolynomial.

Denote by $\tilde{\mathcal{R}}$ the set of rectangles from the optimal solution in the set obtained by Lemma 8. We will now shrink these rectangles for the purpose of proving that GEO-DP finds a good solution. We remark that our algorithm does not need to compute this shrinking. For each rectangle $R_i \in \tilde{\mathcal{R}}$ we define a new rectangle R'_i such that $R_i^{-\delta} \subseteq R'_i \subseteq R_i$. Consider a rectangle $R_i \in \tilde{\mathcal{R}} \cap \mathcal{R}_\ell$. If $\mu'_\ell \leq h_i < \mu_\ell$ then we move the top and bottom boundaries of R_i towards each other so that they align with the closest horizontal grid lines of G_ℓ. If $\mu'_\ell \leq g_i < \mu_\ell$, then we move the left and right boundaries of R_i towards each other so that they align with the closest vertical grid lines of G_ℓ. Note that $R_i^{-\delta} \subseteq R'_i \subseteq R_i$ since we apply the above procedure only to the edges that are at least $\mu'_\ell = w_\ell/2\delta$ units long and in their corresponding dimension R_i crosses at least $1/2\delta$ grid lines of G_ℓ. See Figure 1b for an illustration. Note, the actual shrinking for R_i is always $R_i^{-\delta}$ (R'_i is defined only for analysis.) Denote by $\mathcal{R}'$ the solution consisting of all rectangles R'_i for $R_i \in \tilde{\mathcal{R}}$.

2.3 Analysis of the Dynamic Program

In this section we show that, when given the set $\mathcal{R}$ as an input, GEO-DP will find the solution $\mathcal{R}'$ when parametrized by $k := (\frac{1}{\epsilon\delta})^{10/\epsilon}$. Using the fact that $w(\mathcal{R}') \geq (1-O(\epsilon))w(\mathcal{R}^*)$ (from Lemmas 6 and 8), this implies that GEO-DP is a $(1+\epsilon)$-approximation algorithm for δ-MWISR.

In its recursion, GEO-DP tries all possibilities to partition the input square $[0, N] \times [0, N]$ into at most k smaller polygons and then selects the most profitable partition. For each polygon in the latter partition, it again computes an optimal partition into at most k smaller polygons and so on. The sequence of cuts produced by GEO-DP can be described by a tree T where each node v is associated with a region P_v in the plane. We say that a tree T is a *good* $(k, \mathcal{R}')$*-region decomposition* if the following holds:

- For each node v in T and each rectangle $R \in \mathcal{R}'$, we have that if R does not coincide with P_v, i.e., $R \neq P_v$, then either R is contained in P_v, or R is disjoint from P_v.
- For tree nodes u and v such that u is a parent of v, we have $P_v \subseteq P_u$. Each node $v \in T$ has at most $k' \leq k$ children $u_1, \ldots, u_{k'}$ in T, and $\bigcup_{i=1}^{k'} P_{u_i} = P_v$.
- For each leaf node v of T, the polygon P_v coincides with a rectangle in $\mathcal{R}'$ or P_v has empty intersection with every rectangle in $\mathcal{R}'$.

▶ **Lemma 9.** *If a good* $(k, \mathcal{R}')$*-region decomposition exists, then the algorithm GEO-DP parametrized by* k *is a* $(1+\epsilon)$*-approximation algorithm for* δ*-MWISR.*

Proof. We assume that there is a non-overlapping set of rectangles $\mathcal{R}'$ with $w(\mathcal{R}') \geq (1-O(\epsilon))\mathsf{OPT}$ for which a $(k, \mathcal{R}')$-region decomposition exists. For each $R'_i \in \mathcal{R}'$, we denote by R_i the original, non-shrunk counterpart of R'_i. Let T be the tree that represents the region decomposition for $\mathcal{R}'$. We now prove the following statement by induction on the structure of T from its leaves to the root:

For any node $u \in T$, when GEO-DP processes the instance given by the input rectangles that are contained in P_u, it outputs a set of rectangles $\bar{\mathcal{R}}_u$ whose weight $w(\bar{\mathcal{R}}_u)$ is at least the total weight of the rectangles in $\mathcal{R}'$ that are contained in P_u.

In particular, this statement implies that for the root node r with $P_r = [0, N] \times [0, N]$ GEO-DP computes a set of rectangles $\bar{\mathcal{R}}_r$ with weight $w(\bar{\mathcal{R}}_r) \geq w(\mathcal{R}') \geq (1 - O(\epsilon))\mathsf{OPT}$ as desired.

The base case is obvious: For each leaf node v its polygon P_v coincides with a rectangle $R'_i \in \mathcal{R}'$ and thus $R_i^{-\delta}$ is in P_v; so GEO-DP returns a solution whose weight is at least $w(R'_i)$. Now for the inductive step, consider a node v for which the induction hypothesis holds for all children of v. Let $\mathcal{R}'_v$ denote all rectangles from $\mathcal{R}'$ that are contained in

P_v. Denote the children of v by $v_1, \dots, v_{k'}$ for some $k' \leq k$. We have that $P_v = \bigcup_{j=1}^{k'} P_{v_j}$ and that the polygons $P_{v_1}, \dots, P_{v_{k'}}$ are pairwise disjoint. For each $j \in \{1, \dots, k'\}$ let $\mathcal{R}'_{v_j}$ denote the rectangles from $\mathcal{R}'$ that are contained in P_{v_j}. Since each rectangle in $\mathcal{R}'_v$ is contained in some polygon P_{v_j} the sets $\mathcal{R}'_{v_j}$ form a partition. In particular, this implies that $w(\mathcal{R}'_v) = \sum_{j=1}^{k'} w(\mathcal{R}'_{v_j})$. Moreover, GEO-DP considers the cut which partitions P_v into $P_{v_1}, \dots, P_{v_{k'}}$ and returns, by the induction hypothesis, a solution $\bar{\mathcal{R}}_v$ consisting of one solution $\bar{\mathcal{R}}_{v_j}$ for each polygon P_{v_j} such that $w(\bar{\mathcal{R}}_v) = \sum_{j=1}^{k'} w(\bar{\mathcal{R}}_{v_j}) \geq \sum_{j=1}^{k'} w(\mathcal{R}'_{v_j}) = w(\mathcal{R}'_v)$. This completes the proof. ◀

We prove the existence of a $(k, \mathcal{R}')$-region decomposition by iteratively cutting the polygons. Initially, before the first iteration, we have the tree T which contains only the root r with corresponding region $P_r = [0, N] \times [0, N]$ (the whole input square). Denote the grid lines we have by $\{G_\ell\}_{\ell=0}^{q}$. In each iteration ℓ, we use grid G_ℓ as a template to further cut the polygons into sub-polygons (updating the tree T accordingly). We will ensure that the following invariant holds at the beginning of iteration ℓ: For each leaf node $v \in T$, the polygon P_v has only four edges (i.e., it is a rectangular region[3]), and P_v is either contained in a grid cell of $G_{\ell-1}$ or P_v coincides with some rectangle in $\mathcal{R}'$; each region P_v has empty intersection with every rectangle in $\mathcal{R}' \cap (\bigcup_{\ell' < \ell} \mathcal{R}_{\ell'})$. Finally, every internal node has degree at most k. It is not hard to see that if we have maintained the invariant until the last iteration q, the tree T would satisfy all properties of good $(k, \mathcal{R}')$-region decomposition.

Partition into groups of cells

Now assume that we have so far maintained the invariant up to iteration ℓ, and we will provide a sequence of cuts extending the so far constructed tree such that the invariant holds for $\ell + 1$. Consider a leaf node v of T. If P_v coincides with a rectangle in $\mathcal{R}'$, no further partition is necessary (it satisfies the invariant until the end). Otherwise, we consider the grid G_ℓ restricted to P_v. Denote by $\mathcal{R}^{cor}_{v,\ell} \subseteq \mathcal{R}' \cap \mathcal{R}_\ell$ all rectangles of $\mathcal{R}' \cap \mathcal{R}_\ell$ that overlap corners of G_ℓ inside P_v. We add each such rectangle as a child node of v. Notice that these nodes satisfy the invariant for level $\ell + 1$. Let $M = (\mu'_\ell / \mu'_{\ell+1})^2$ (i.e., M equals the maximum number of grid cells of G_ℓ within P_v). Since $|\mathcal{R}^{cor}_{v,\ell}| \leq M$, the polygon P_v after removing such rectangles has at most $4M + 4$ edges. We then focus on the other rectangles. The way we shrunk rectangles guarantees the following.

▶ **Observation 10.** *Consider a grid cell C in G_ℓ. Either the cell C is not touched by any rectangle $R'_i \in \mathcal{R}_\ell \cap \mathcal{R}'$, i.e., $C \cap R'_i = \emptyset$ for all $R'_i \in \mathcal{R}_\ell \cap \mathcal{R}'$, or C is crossed by a rectangle $R'_i \in \mathcal{R}_\ell \cap \mathcal{R}'$, i.e., C without the relative interior of R'_i has two connected components.*

Since their longer edges start and end at grid coordinates, the rectangles in $\mathcal{R}^{cor}_{v,\ell}$ partition the grid cells into three disjoint groups: cells which are not crossed by any rectangle in $\mathcal{R}_\ell \cap \mathcal{R}'$, cells which are horizontally crossed, and cells which are vertically crossed (see Figure 2). The cells of the first group already satisfy the invariant for $\ell + 1$ because no rectangle in $\mathcal{R}_\ell \cap \mathcal{R}'$ intersects it (but we remark that there may be rectangles in $\mathcal{R}_{\ell+1}, \dots,$ that may still be in such cells). For each of them we create a child node v' of v. We partition the remaining grid cells into at most M *groups* $\mathcal{C}_1, \mathcal{C}_2, \dots$ such that two adjacent grid cells are in the same group if and only if there is a rectangle $R'_i \in \mathcal{R}_\ell$ crossing both of them. For each

[3] A rectangular region refers to a region in the plane which may not coincide with any input rectangle.

group $\mathcal{C}_j$ we add a child node v_j to v and we define the region $Q_j = (\bigcup_{C \in \mathcal{C}_j} C) \setminus \bigcup_{R_i \in \mathcal{R}^{cor}_{v,\ell}} R_i$ corresponding to node v_j.

▶ **Lemma 11.** *All cells in each group $\mathcal{C}_j$ are contained in either a grid row or a grid column of G_ℓ. Moreover, the region Q_j has at most $9M$ edges, and no rectangle in $\mathcal{R}' \cap (\bigcup_{\ell' \geq \ell} \mathcal{R}_{\ell'})$ touches its boundary.*

Proof. Assume for contradiction that there is a group $\mathcal{C}_j$ that is not horizontally or vertically contained in a grid row or column. Then $\mathcal{C}_j$ contains more than one cell and thus each cell in $\mathcal{C}_j$ is crossed horizontally or vertically but no both. If there is no cell in $\mathcal{C}_j$ that is crossed vertically then no two cells from $\mathcal{C}_j$ in different rows can be in the same group which is a contradiction since we assumed $\mathcal{C}_j$ not to be contained in one grid row. The same reasoning applies if no cell in $\mathcal{C}_j$ is crossed horizontally. Thus, there must be a grid cell C in $\mathcal{C}_j$ that is crossed horizontally and another grid cell C' that is crossed vertically. However, then the cells in $\mathcal{C}_j$ that are crossed horizontally and the ones that are crossed vertically must be in different groups.

Moreover, the edges of Q_j consist of the grid cell boundaries of G_ℓ (at most $4M$ edges as there are M such cells with 4 edges each), the boundaries of rectangles in $\mathcal{R}^{cor}_\ell$ (at most $4M$ edges as there are at most M such rectangles), and the boundaries of the polygon P_v (at most 4 edges by the induction hypothesis). So Q_j has at most $8M + 4 \leq 9M$ edges. Also, no rectangle in a set $\mathcal{R}_{\ell'}$ with $\ell' \geq \ell$ touches the boundary of Q_j because no rectangle in $\mathcal{R}' \cap \mathcal{R}_{\ell'}$ can cross a grid line of G_ℓ (by Lemma 8), the boundary of other rectangles in $\mathcal{R}'$, or the boundary of the polygon P_v (by the induction hypothesis). ◀

So the "correct" partition of P_v has one polygon for each cell that is not crossed by a rectangle in $\mathcal{R}_\ell$, one polygon for each group $\mathcal{C}_j$, and one polygon for each rectangle in $\mathcal{R}^{cor}_{v,\ell}$. Note that in total those are at most $5M$ many. Notice that these tree nodes for a group $\mathcal{C}_j$ do not necessarily satisfy the invariant since Q_j might not be contained in a grid cell of G_ℓ.

While this partition has similarities to quad-tree approaches like in Arora's algorithm for Euclidean TSP [5] we note that in such classical approaches the pieces arising in the recursive partition (typically squares) do not depend on the instance and are predetermined. In constrast, in our case this partition depends on the structure of the optimal solution $\mathcal{R}'$ and the algorithm has to guess the correct one. Furthermore, before proceeding to the next level we must further refine the partitions that correspond to groups $\mathcal{C}_j$ step-by-step as we explain in the sequel.

Further partitioning of each group

Next, we show that there is a sequence of cuts that further partition each group $\mathcal{C}_j$ into a family of smaller polygons such that at each intermediate step each polygon has at most k edges. Consider group $\mathcal{C}_j$ that is horizontally crossed (the other case is symmetric). We construct a (planar) graph $H_j = (V_j, E_j)$ within Q_j, see Figure 3 for a sketch. The set V_j has a node for each vertex of the polygon Q_j and for each intersection of the top or bottom edge of a rectangle $R'_i \in \mathcal{R}' \cap \mathcal{R}_\ell$ with a vertical grid line in G_ℓ (including the corners of R'_i). Denote by $V_j^{(0)}, V_j^{(1)}, V_j^{(2)}, \ldots$ the vertices in V_j ordered by the vertical grid lines they appear on, i.e. $V_j^{(p)}$ contains the vertices in V_j on the p^{th} vertical grid line in G_ℓ inside Q_j. For each p, we introduce a horizontal edge in E_j between two vertices $v \in V_j^{(p)}$, $v' \in V_j^{(p+1)}$ if and only if v and v' lie on the same edge of a rectangle in $\mathcal{R}' \cap \mathcal{R}_\ell$; also we add a vertical edge in E_j between two vertices $v \in V_j^{(p)}$, $v' \in V_j^{(p)}$ if the line segment L between v and v'

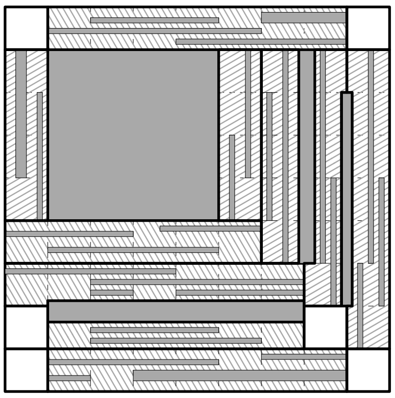


Figure 2 The pieces P'_j for the groups $\mathcal{C}_j$ and the grid cells that are not touched by any rectangle. The shading indicates whether the group is a horizontal or a vertical group.

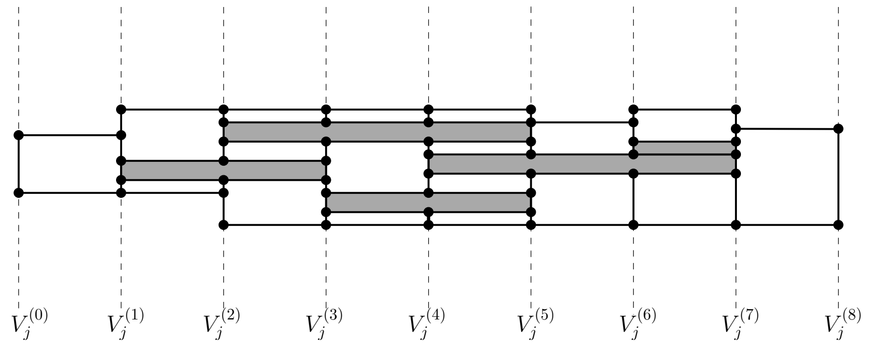


Figure 3 The graph H_j for one piece P'_j. The thick lines represent the edges E_j of H_j.

does not cross any rectangle $\mathcal{R}' \cap \mathcal{R}_\ell$ and also no other vertex $v'' \in V_j^{(p)}$ with $v'' \notin \{v, v'\}$. By construction, no edge in E_j crosses through any rectangle in $\mathcal{R}'$.

Now we cut the region Q_j step-by-step along simple paths in H_j which go from left to right, visiting a vertex in $V_j^{(p)}$ after having visited a vertex in $V_j^{(p-1)}$, for each p. We call such paths *cutting paths*. Each polygon arising in this partition sequence can be described as the polygon $P(\sigma, \sigma')$ between two cutting paths σ and σ' that start at some common point s and end at t; also they are disjoint except at the two endpoints. Observe that such polygons have at most $O(M)$ edges each and that Q_j itself equals $P(\sigma_T, \sigma_B)$ where σ_T and σ_B denote the paths describing the top and bottom boundary of Q_j, respectively. Now the idea is that if a polygon $P(\sigma, \sigma')$ for two cutting paths σ, σ' does not satisfy the invariant, then it can be further partitioned along another cutting path σ'', as the following lemma shows (we will prove it later in Section 2.4).

▶ **Lemma 12.** *Let σ, σ' be two cutting paths in Q_j. Then either*
- *$P(\sigma, \sigma')$ has rectangular shape, is contained in a grid cell of G_ℓ, and has empty intersection with each rectangle in $\mathcal{R}_\ell \cap \mathcal{R}'$, or*
- *$P(\sigma, \sigma')$ has rectangular shape and it coincides with a rectangle in $\mathcal{R}' \cap \mathcal{R}_\ell$, or*
- *there is a cutting path σ'' with $\sigma \neq \sigma'' \neq \sigma'$ such that $P(\sigma, \sigma') = P(\sigma, \sigma'') \dot\cup P(\sigma'', \sigma')$.*

We invoke Lemma 12 on each region Q_j until the invariant is satisfied: If invoking the lemma on Q_j holds with the first or second cases, then we are done; otherwise, Q_j can be further partitioned into Q' and Q'' based on the cutting path. In such case, we add two nodes corresponding to regions Q' and Q'' into the tree T as children of Q_j, and then invoke the lemma on Q' and Q''. Since these polygons are always defined by two cutting paths, their complexities are bounded by $O(M)$. Now each leaf node that does not coincide with a rectangle in $\mathcal{R}' \cap \mathcal{R}_\ell$ satisfied the invariant. The above shows that there is a $(k, \mathcal{R}')$-region decomposition for $k = O(M) = (1/\delta\epsilon)^{1/\epsilon}$.

Note that already in the last part—the partitioning of each group—one single group might be partitioned into up to $\Omega(n)$ pieces. Thus, we cannot use an approach which guesses this partition in a single step only. In particular, to ensure polynomial running time we crucially need our DP and cannot replace it by a brute-force recursive algorithm since the depth of T can be up to $\Omega(\log n)$. This is a key difference to the QPTAS in [1] where instead of the DP one could alternatively use such a recursion and obtain the same result.

Superpolynomial input data

To remove the assumption that N is bounded by a polynomial, observe that there are only $O(\log N)$ recursion levels, which is polynomial in the length of the input encoding. Each coordinate used in our cut sequence coincides with a coordinate of a rectangle in $\mathcal{R}'$ or with a horizontal or vertical grid line (these coordinates can be computed efficiently in a randomized fashion by Lemma 8). While the last recursion level can give rise to up to $\Omega(N)$ of those, it suffices to consider only grid lines belonging to grid cells C such that there exists an input rectangle R with $R \subseteq C$. In each of the $O(\log N)$ levels, there can be only n such grid cells which bounds the total number of needed coordinates by $O(n \log N)$. This completes the proof of Theorem 1.

2.4 Proof of Lemma 12

In this section we prove Lemma 12. Assume w.l.o.g. that the polygon Q_j is completely contained in a grid row. Consider the polygon $P(\sigma, \sigma')$ defined by two cutting paths where σ is above σ', i.e. paths σ and σ' contain the upper and lower boundaries of polygon $P(\sigma, \sigma')$ respectively. Let H'_j be the subgraph of H_j induced by all vertices that are used by σ or σ' or which lie in the relative interior of $P(\sigma, \sigma')$. We assume w.l.o.g. that paths σ and σ' do not intersect except at the endpoints, i.e. they both start at some node $s \in V(H'_j)$ and end at some node $t \in V(H'_j)$. We will argue that one of the three cases of Lemma 12 applies.

We say that a path τ in H'_j is *monotone* if τ is empty or can be written as $\tau = (v_0, v_1, \ldots, v_z)$ such that for each i, vertex v_i is either on the left of v_{i+1} or on the top (i.e. the monotone path only goes right or down.) First, we need the following lemma.

▶ **Lemma 13.** *Let $u \in V(H'_j)$ be a vertex that corresponds to the bottom-right corner of a rectangle R in $\mathcal{R}_\ell$. Then there is a monotone path τ from vertex u to some vertex v' on path σ'; symmetrically, any top-left corner of a rectangle is reachable from a vertex in σ by a monotone path.*

Proof. We only prove this statement when $u \notin \sigma'$; otherwise, it is trivial (notice that u cannot be on σ.) To prove this statement, it is sufficient to show that there is a monotone path τ' that either connects vertex u to the bottom-right corner of another rectangle R' or to some vertex on σ': Applying this claim iteratively gives us the lemma.

Now notice that vertex u is on the right boundary of rectangle R, so $u \in V_j^{(p)}$ for some p. From the way we construct graph H_j, there must be a downward edge from u to either a vertex on the top boundary of some other rectangle R' or on the path σ'. In the latter case, we are immediately done. In the former case, let u' be a vertex on the top boundary of R'

that is connected to u via an edge (u, u'). We define path τ' that first takes an edge (u, u') and then from u' there is always a monotone path to the bottom-right corner of R' using edges on the boundary of R'. ◀

Using this lemma, we now prove Lemma 12. We have the following cases:

- First, if there is a vertex $u \in V(H_j)$ that is a corner of some rectangle $R \in \mathcal{R}_\ell \cap \mathcal{R}'$, we show that we can find a cutting path σ'' implying the third case of the lemma. Define σ''_t to be the monotone path that connects the top-left corner u_{top} of R to u (this path could be empty). Also σ''_b is the monotone path that connects u to the bottom-right corner u_{bot} of R. Observe that σ''_t is disjoint from σ''_b and that at least one edge in $\sigma''_b \cup \sigma''_t$ is in the interior of $P(\sigma, \sigma')$.
  We now apply Lemma 13 to find a path τ_t that connects a vertex v_{top} on σ to u_{top}, and similarly we can find a path τ_b that connects vertex u_{bot} to some vertex v_{bot} on σ'. It is easy to see that all paths $\sigma''_b, \sigma''_t, \tau_b, \tau_t$ are disjoint. Now the cutting path σ'' is easily defined: Start from s, follow path σ until it reaches v_{top}, then follow the paths $\tau_t, \sigma''_t, \sigma''_b, \tau_b$ in this order until v_{bot} is reached, and finally from v_{bot} we use the path σ' towards vertex t. This is a cutting path because we always go from left to right and the path cuts through the interior.
- Now assume that there is no such corner in the interior. There are two possibilities. First if there is no rectangle in $\mathcal{R}_\ell \cap \mathcal{R}'$ that lies in polygon $P(\sigma, \sigma')$, then either $P(\sigma, \sigma')$ is contained in one cell (in which case we are done with the first case of Lemma 12 applied), or there is a vertical edge that connects two vertices in $V_j^{(p)}$ for some p where we can cut. Otherwise, there is a rectangle $R \in \mathcal{R}_\ell \cap \mathcal{R}'$ that lies in $P(\sigma, \sigma')$ where all four corners lie on the border of polygon $P(\sigma, \sigma')$, i.e. on $\sigma \cup \sigma'$. If the upper boundary of R does not lie on σ, we could cut the polygon $P(\sigma, \sigma')$ using this upper boundary as our σ'' (in which case, the third case of Lemma 12 applies.) Similar arguments hold for the bottom boundary of R. Hence, the only case left to analyze is when the top and bottom boundaries of R lie on σ and σ' respectively. In such case, polygon $P(\sigma, \sigma')$ coincides with rectangle R, and the second case of Lemma 12 applies.

3 Coloring and Integrality Gap

In this section, we consider the rectangle coloring problem and bound the integrality gap of the LP for MWISR in our model. Both results rely on a partitioning lemma that divides rectangles into sub-collections with "nice" properties. We will first define these properties precisely and state the partitioning lemma. Then we will describe how it can be used to prove Theorem 2 and Corollary 3.

For pairs of intersecting rectangles we distinguish three types of intersections: crossing, containment, and corner intersections. We say that two rectangles R, R' have a *crossing* intersection if no rectangle contains a corner of the other, a *containment* intersection if one rectangle completely contains the other, and otherwise they have *corner* intersection. We call a collection of rectangles *nice* if no two rectangles in $\mathcal{R}$ have corner intersections (but may still have containment).

It is known that if a collection of rectangles $\mathcal{R}$ is nice then we have $\chi(\mathcal{R}) = \omega(\mathcal{R})$, see e.g., [9, Theorem 4] (which implies that then the intersection graph is perfect). Note that this statement is slightly more general than the classical result in [6] that the latter equality holds if the rectangles in $\mathcal{R}$ have only crossing intersections (and thus no containment intersections). Our partitioning scheme is formally summarized in the following lemma that we will prove later in Section 3.1.

▶ **Lemma 14** (Partitioning lemma). *Let $\mathcal{R}$ be a set of rectangles. For any $\delta > 0$, there is a value $M = O((\frac{1}{\delta})^2 \log^2(1/\delta))$ and a polynomial time algorithm computing a partition of $\mathcal{R}$ into groups $\mathcal{R}_1, \dots, \mathcal{R}_M$ and a rectangle S_i for each rectangle $R_i \in \mathcal{R}$ such that $R_i^{-\delta} \subseteq S_i \subseteq R_i$. The computed partition and the rectangles S_i have the property that each collection $\mathcal{S}_j = \{S_i : R_i \in \mathcal{R}_j\}$ is nice.*

We explain now how to use Lemma 14 in order to prove Theorem 2 and Corollary 3.

Rectangle Coloring

It is straightforward to see that Lemma 14 implies the coloring algorithm. Partition the input collection $\mathcal{R}$ into $M = O((\frac{1}{\delta})^2 \log^2(1/\delta))$ collections $\mathcal{R}_1, \dots, \mathcal{R}_M$. Now we know that each set $\mathcal{S}_j$ is nice, so we can color its rectangles with $\omega(\mathcal{S}_j) \leq \omega(\mathcal{R})$ colors while using a different set of colors for each set $\mathcal{S}_j$. In total, the number of used colors is at most $M \cdot \omega(\mathcal{R})$. This proves that $\chi^{-\delta}(\mathcal{R}) \leq O((\frac{1}{\delta})^2 \log^2(1/\delta))\omega(\mathcal{R})$ and thus Theorem 2.

Integrality Gap

We use Lemma 14 in order to bound the integrality gap of the natural LP-formulation of MWISR in our shrinking model. To this end, we first define this LP and the meaning of an integrality gap in our model and subsequently prove Corollary 3.

First recall the following standard LP relaxation for MWISR. For each rectangle R_i, we have a variable x_i which indicates whether rectangle R_i is included in the solution.

$$\begin{aligned} \text{(LP-IS)} \quad & \max \quad \sum_{R_i \in \mathcal{R}} w_i x_i \\ & \text{s.t.} \quad \sum_{R_i : p \in R_i} x_i \leq 1 \text{ for all } p \in \mathcal{P} \\ & \qquad\quad x_i \geq 0 \text{ for all } R_i \in \mathcal{R} \end{aligned}$$

Here $\mathcal{P}$ denotes the set of "interesting points" defined as follows: define a non-uniform grid by drawing a horizontal and a vertical line through each corner of an input rectangle. Note that each point in the interior of a grid cell is overlapped by exactly the same set of rectangles. For each grid cell add an arbitrary point from its interior to $\mathcal{P}$. Note that $|\mathcal{P}| \leq O(|\mathcal{R}|^2)$. In the MWISR problem, the integrality gap is the maximum possible ratio $\sup_{\mathcal{R}} \frac{\mathsf{LP}(\mathcal{R})}{\mathsf{OPT}(\mathcal{R})}$ where $\mathsf{LP}(\mathcal{R})$ denotes the optimal value of (LP-IS) on the instance $\mathcal{R}$. For the model of shrinking the rectangles, we use the following natural modification of the integrality gap definition. For each collection $\mathcal{R}$, let $\mathsf{OPT}_\delta(\mathcal{R})$ be the weight of a maximum-weight δ-feasible independent set $\mathcal{R}' \subseteq \mathcal{R}$. Notice that for any $\delta > 0$ we have that $\mathsf{OPT}_\delta(\mathcal{R}) \geq \mathsf{OPT}(\mathcal{R})$. Then the *$\delta$-shrunk integrality gap* is defined as $\sup_{\mathcal{R}} \frac{\mathsf{LP}(\mathcal{R})}{\mathsf{OPT}_\delta(\mathcal{R})}$. We need the following lemma.

▶ **Lemma 15** (Implied by Theorem 4 in [9]). *Let $\mathcal{R}$ be a nice collection of rectangles and let x be a solution to (LP-IS) for $\mathcal{R}$. Then there is a set of independent rectangles $\mathcal{R}' \subseteq \mathcal{R}$ with $w(\mathcal{R}') \geq \sum_{R_i \in \mathcal{R}} w_i x_i$.*

Now we prove Corollary 3. Let x^* be an optimal LP solution to an input collection $\mathcal{R}$ of rectangles, so we have $\sum_{R_i \in \mathcal{R}} w_i x_i^* = \mathsf{LP}(\mathcal{R})$. Use Lemma 14 to partition $\mathcal{R}$ into $\mathcal{R}_1, \dots, \mathcal{R}_M$. By the pigeon hole principle there must be a group $\mathcal{R}_j$ with $\sum_{R_i \in \mathcal{R}_j} w_i x_i^* \geq \mathsf{LP}(\mathcal{R})/M$. Together with Lemma 15, applied on a nice set $\mathcal{R}_j$, this yields the proof of Corollary 3.

3.1 Proof of the Partitioning Lemma

We prove Lemma 14 now. Our algorithm deals with the x and y coordinates of the input rectangles separately in the following way: We compute two collections of intervals $\mathcal{I}^x, \mathcal{I}^y$ obtained by projecting the rectangles in $\mathcal{R}$ onto the x and y-axes, respectively. Then for each such collection we invoke the following lemma where for any interval $I = (a, a+x)$ we define $I^{-\delta} := (a + \frac{\delta}{2}x, a + (1 - \frac{\delta}{2})x)$. For simplicity, we prove the following lemma only for open intervals, as also our rectangles are defined as open sets. However, it holds also for general intervals.

▶ **Lemma 16.** *Let $\mathcal{I} = \{I_1, \dots, I_n\}$ be a set of open intervals with integral start and end points. There is a value $M = O((1/\delta)\log(1/\delta))$ and a polynomial time algorithm computing a partition of $\mathcal{I}$ into groups $\mathcal{I}_1, \dots, \mathcal{I}_M$ and an open interval I'_i with $I_i^{-\delta} \subseteq I'_i \subseteq I_i$ for each interval $I_i \in \mathcal{I}$ such that each collection $\mathcal{I}'_j = \{I'_i : I_i \in \mathcal{I}_j\}$ is* nested *(i.e. any two intervals in it are either disjoint or one is contained in another.)*

It follows straightforwardly that invoking this lemma for $\mathcal{I}^x$ and $\mathcal{I}^y$ gives the desired result: Let $\{\mathcal{I}^x_j\}_{j=1}^M$ and $\{\mathcal{I}^y_j\}_{j=1}^M$ be the partitioning obtained by the lemma. We can define a partition $\{\mathcal{R}_{j,k}\}_{j,k=1}^M$ where $\mathcal{R}_{j,k} = \{R_i : I^x_i \in \mathcal{I}^x_j \text{ and } I^y_i \in \mathcal{I}^y_k\}$. Notice that any two overlapping rectangles in the same set $\mathcal{R}_{j,k}$ must be nested in both x and y coordinates, so either they are crossing or one is contained in the other.

The proof of the above lemma has two main steps. In the first step, we group intervals into many groups by their lengths, where intervals in the same groups have roughly the same length, and the ratio of lengths of two intervals in different groups is sufficiently large. We pay a factor of $O(\log(1/\delta))$ in this step. In the second step, we partition the intervals into at most $O(1/\delta)$ groups and shrink intervals in each group to obtain the claimed properties.

Step 1: Preprocessing

We first group the intervals geometrically by their lengths into $\mathcal{I} = \bigcup_j \mathcal{I}_j$ such that each set $\mathcal{I}_j$ contains all intervals whose lengths are within $[2^j, 2^{j+1})$. Let $L := \lceil \log \frac{8}{\delta} \rceil$. For each $r \in \{0, \dots, L-1\}$ we define a collection $\Gamma_r = \{\mathcal{I}_j : j \equiv r \mod L\}$. Notice that, for any collection Γ_r, if we take two intervals from different sets $\mathcal{I}_j$, their lengths differ by at least a factor of $4/\delta$. This property will be crucial in our algorithm. In the next step, we further partitioning each collection Γ_r into $O(1/\delta)$ sub-collections.

Step 2: Shrinking

Recall that our intervals have integral start and end points and assume w.l.o.g. that they are all contained in $[0, N]$ for some large integer N. Consider a collection Γ_r. By the first step, we know that $\Gamma_r = \{\mathcal{I}_r, \mathcal{I}_{L+r}, \mathcal{I}_{2L+r}, \dots, \mathcal{I}_{\ell_{\max}L+r}\}$ with $\ell_{\max}$ being the largest integer such that $\mathcal{I}_{\ell_{\max}L+r} \neq \emptyset$. We say that an interval is at level-ℓ if it belongs to $\mathcal{I}_{\ell L+r}$, i.e., its length is in the interval $[2^{\ell L+r}, 2 \cdot 2^{\ell L+r})$. For later convenience, we define $\mu'_\ell = 2^{\ell L+r}$ and $\mu_\ell = 2^{\ell L+r+1}$. Note that $\mu'_{\ell+1}/\mu'_\ell = 2^L \geq 8/\delta$ for each ℓ. Moreover, for each ℓ we define a collection of level-ℓ points $\mathcal{P}_\ell = \{k \cdot \delta\mu'_\ell | k \in \mathbb{Z}\}$.

▶ **Observation 17.** *Each level-ℓ interval contains at least $1/\delta$ points in $\mathcal{P}_\ell$. Moreover, for any two consecutive points $p, p' \in \mathcal{P}_{\ell+1}$ there are $\mu'_{\ell+1}/\mu'_\ell - 1$ points in $\mathcal{P}_\ell \cap (p, p')$.*

We now describe our shrinking process. For each interval $I_i = (x_i, y_i)$ at level ℓ, we shrink the left-endpoint of I_i towards its centroid to the closest point in $\mathcal{P}_\ell$; similarly for the right endpoint. Formally, we define $I_i' := \left(\left\lceil \frac{x_i}{\delta\mu_\ell'} \right\rceil \cdot \delta\mu_\ell', \left\lfloor \frac{y_i}{\delta\mu_\ell'} \right\rfloor \cdot \delta\mu_\ell'\right)$ to be the shrunk interval corresponding to I_i. From the above observation, each interval gets shrunk by a factor of at most $(1-2\delta)$. Let $\mathcal{I}^{good}$ be the set of intervals that do not contain points of levels higher than the interval itself, i.e., each interval I_i is contained in $\mathcal{I}^{good}$ if and only if I_i does not contain a point in $\mathcal{P}_{\ell+1}$. Note that the latter condition implies that I_i does not contain a point in $\mathcal{P}_{\ell+2}, \mathcal{P}_{\ell+3}, \ldots$ since the values $\mu_\ell', \mu_{\ell+1}', \ldots$ pairwise divide each other.

▶ **Lemma 18.** *The collection of intervals $\mathcal{I}^{good}$ can be partitioned into $M' = O(1/\delta)$ sub-collections such that each shrunk sub-collection is nested.*

Proof. We define $M' := 2^L = O(1/\delta)$. Note that $\mu_{\ell+1}'/\mu_\ell' = M'$ for each ℓ. We partition $\mathcal{I}^{good}$ into $\{\mathcal{J}_a\}_{a=0}^{M'-1}$ as follows. Since each good level-ℓ interval I_i does not contain a point in $\mathcal{P}_{\ell+1}$, its shrunk counterpart I_i' is of the form $(K_i\delta\mu_{\ell+1}' + a_i(\delta\mu_\ell'), K_i\delta\mu_{\ell+1}' + b_i(\delta\mu_\ell'))$ for some integers K_i, a_i, b_i, where $a_i, b_i \in \{0, \ldots, M'-1\}$; that is for an interval $I_i = (x_i, y_i)$ with $I_i' = (x_i', y_i')$, we have that

$$K_i = \left\lfloor \frac{x_i'}{\delta\mu_{\ell+1}'} \right\rfloor, a_i = \frac{x_i' - K_i\delta\mu_{\ell+1}'}{\delta\mu_\ell'} \text{ and } b_i = \frac{y_i' - K_i\delta\mu_{\ell+1}'}{\delta\mu_\ell'}$$

We include each such interval I_i in the set $\mathcal{J}_{a_i}$. There can be at most $O(1/\delta)$ such sets.

Now we argue that each set $\mathcal{J}_a$ is nested. Consider a set $\mathcal{J}_a$ for some a and two intervals $I_i, I_j \in \mathcal{J}_a$ that are in levels ℓ_i and ℓ_j respectively. If I_i and I_j are disjoint, we are done, so assume that they are overlapping. If $\ell_i \neq \ell_j$ then one interval must contain the other. Here we use that for each ℓ no level-ℓ interval contains a point in $\mathcal{P}_{\ell+1}, \mathcal{P}_{\ell+2}, \ldots$. If $\ell_i = \ell_j$ we have $K_i = K_j$, and therefore one interval must contain the other. ◀

Finally, we need to deal with intervals in $\mathcal{I}^{bad} = \mathcal{I} \setminus \mathcal{I}^{good}$. The intuition is that, if we define point sets similar to $\mathcal{P}_\ell$ but with respect to some shift s, then the bad intervals are behaving like the good intervals above. Formally, we define $s = \delta \sum_\ell \frac{\mu_\ell'}{4}$ be the shift and for each ℓ we define $\mathcal{P}_\ell' = \{s + k \cdot \delta\mu_\ell' | k \in \mathbb{Z}\}$. The intervals in $\mathcal{I}^{bad}$ are shrunk with respect to these new points in a way similar to intervals in $\mathcal{I}^{good}$ but instead we use the points in $\{\mathcal{P}_\ell'\}$ rather than $\{\mathcal{P}_\ell\}$. Formally, for each interval $I_i = (x_i, y_i) \in \mathcal{I}^{bad}$ we define a new shrunk counterpart $I_i'' := \left(\left\lceil \frac{x_i - s}{\delta\mu_\ell'} \right\rceil \cdot \delta\mu_\ell' + s, \left\lfloor \frac{y_i - s}{\delta\mu_\ell'} \right\rfloor \cdot \delta\mu_\ell' + s\right)$.

▶ **Lemma 19.** *Any level-ℓ interval $I_i \in \mathcal{I}^{bad}$ does not contain any point in $\mathcal{P}_{\ell+1}'$.*

Proof. Assume otherwise that some level-ℓ interval $I_i \in \mathcal{I}^{bad}$ intersects some new point $q' \in \mathcal{P}_{\ell+1}'$. Since $I_i \in \mathcal{I}^{bad}$, the interval intersects some old point q in $\mathcal{P}_{\ell+1}$ as well. Recall that the length of the interval I_i is strictly smaller than $2\mu_\ell' \leq \frac{\delta}{4}\mu_{\ell+1}'$.

It must be the case that the coordinate of q is a multiple of $\delta\mu_{\ell+1}'$, while the coordinate of q' is equal to $s + k'\delta\mu_{\ell+1}'$ for some $k' \in \mathbb{Z}$. The shift s can also be written as $k''\delta\mu_{\ell+1}' + \delta\sum_{\ell' \leq \ell+1} \frac{\mu_{\ell'}'}{4}$ for some $k'' \in \mathbb{Z}$ (because the terms $\delta\mu_{\ell+2}', \delta\mu_{\ell+3}', \ldots$, are multiples of $\delta\mu_{\ell+1}'$.) Observe that the term $\delta\sum_{\ell' \leq \ell+1} \frac{\mu_{\ell'}'}{4}$ is at least $\delta\mu_{\ell+1}'/4$ and at most $3\delta\mu_{\ell+1}'/4$ as the values of μ_ℓ' are geometrically increasing in ℓ. This implies that the distance between q and q' is at least $\delta\mu_{\ell+1}'/4$, and since the interval I_i contains both points, its length must be at least that much. This is a contradiction. ◀

With similar arguments as in Lemma 18 we can partition $\mathcal{I}^{bad}$ into $O(1/\delta)$ sub-collections whose respective shrunk counterparts I_i'' are nested.

References

1 Anna Adamaszek and Andreas Wiese. Approximation schemes for maximum weight independent set of rectangles. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*, pages 400–409. IEEE, 2013.

2 Anna Adamaszek and Andreas Wiese. A QPTAS for maximum weight independent set of polygons with polylogarithmically many vertices. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*, pages 645–656. SIAM, 2014.

3 Pankaj K. Agarwal and Nabil H. Mustafa. Independent set of intersection graphs of convex objects in 2D. *Computational Geometry: Theory and Applications*, 34(2):83–95, 2006.

4 Pankaj K. Agarwal, Marc J. van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications*, 11(3-4):209–218, 1998.

5 Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM (JACM)*, 45:753–782, 1998.

6 E. Asplund and Branko Grünbaum. On a coloring problem. *Mathematica Scandinavica*, 8:181–188, 1960.

7 Piotr Berman, Bhaskar DasGupta, S. Muthukrishnan, and Suneeta Ramaswami. Improved approximation algorithms for rectangle tiling and packing. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001)*, pages 427–436. SIAM, 2001.

8 A. Bielecki. Problem 56. *Colloquium Mathematicum*, 1:333, 1948.

9 Parinya Chalermsook. Coloring and maximum independent set of rectangles. In *Proceedings of the 14th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX 2011)*, pages 123–134. Springer, 2011.

10 Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009)*, pages 892–901. SIAM, 2009.

11 Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *Journal of Algorithms*, 46(2):178–189, 2003.

12 Timothy M. Chan. A note on maximum independent sets in rectangle intersection graphs. *Information Processing Letters*, 89(1):19–23, 2004.

13 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. In *Proceedings of the 25th Annual Symposium on Computational Geometry (SoCG 2009)*, pages 333–340. ACM, 2009.

14 George Christodoulou, Khaled M. Elbassioni, and Mahmoud Fouz. Truthful mechanisms for exhibitions. In *Proceedings of the 6th International Workshop on Internet and Network Economics (WINE 2010)*, pages 170–181. Springer, 2010.

15 José R. Correa, Laurent Feuilloley, and José A. Soto. Independent and hitting sets of rectangles intersecting a diagonal line. In *Proceedings of the 11th Latin American Symposium on Theoretical Informatics (LATIN 2014)*, pages 35–46. Springer, 2014.

16 Thomas Erlebach and Klaus Jansen. The complexity of path coloring and call scheduling. *Theoretical Computer Science*, 255(1-2):33–50, 2001.

17 Robert J Fowler, Michael S Paterson, and Steven L Tanimoto. Optimal packing and covering in the plane are np-complete. *Information processing letters*, 12(3):133–137, 1981.

18 Jacob Fox and János Pach. Computing the independence number of intersection graphs. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pages 1161–1165. SIAM, 2011.

19 Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Data mining with optimized two-dimensional association rules. *ACM Transactions on Database Systems (TODS)*, 26(2):179–213, 2001.

20 Sariel Har-Peled. Quasi-polynomial time approximation scheme for sparse subsets of polygons. In *Proceedings of the 30th Annual Symposium on Computational Geometry (SoCG 2014)*, page 120. ACM, 2014.
21 Sariel Har-Peled and Mira Lee. Weighted geometric set cover problems revisited. *JoCG*, 3(1):65–85, 2012.
22 Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Electronic Colloquium on Computational Complexity*, 4(38), 1997.
23 C. Hendler. Schranken für Färbungs-und Cliquenüberdeckungszahl geometrisch repräsentierbarer Graphen. Master's thesis, FU Berlin, Germany, 1998.
24 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32(1):130–136, 1985.
25 Alexandr V. Kostochka. Coloring intersection graphs of geometric figures with a given clique number. *Contemporary Mathematics*, 342, 2004.
26 Liane Lewin-Eytan, Joseph Naor, and Ariel Orda. Routing and admission control in networks with advance reservations. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX 2002)*, pages 215–228. Springer, 2002.
27 Frank Nielsen. Fast stabbing of boxes in high dimensions. *Theoretical Computer Science*, 246:53–72, 2000.
28 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3:103–128, 2007.

A NP-Hardness Proof

We give a sketch for the proof that δ-MWISR is NP-hard. We note that (ordinary) MWISR is NP-hard even for unit squares [17, Theorem 2]. Let $\mathcal{R}$ be an instance produced by this reduction. By analysing the proof in [17] one can easily show that the intersection area between any pair of intersecting squares in $\mathcal{R}$ is at least a constant ϵ. Notice that shrinking a (unit) square $R \in \mathcal{R}$ by a factor of $(1-\delta)$ reduces its area by at most $1-(1-\delta)(1-\delta) = 2\delta - \delta^2 \leq 2\delta$. This implies that if δ is chosen such that $4\delta < \epsilon$ then any collection of rectangles $\mathcal{S} \subseteq \mathcal{R}$ is non-overlapping if and only if $\mathcal{S}^{-\delta}$ is non-overlapping. Thus, for the instance $\mathcal{R}$, any subset $\mathcal{S} \subseteq \mathcal{R}$ is δ-independent if and only if $\mathcal{S}$ is independent. So if one were able to compute a δ-independent set of value $\mathsf{OPT}(\mathcal{R})$ in polynomial time, it would also imply that such an algorithm can compute an optimal independent set of $\mathcal{R}$.

Non-Uniform Robust Network Design in Planar Graphs

David Adjiashvili

ETH Zürich
Rämistrasse 101, 8092 Zürich, Switzerland
addavid@ethz.ch

Abstract

Robust optimization is concerned with constructing solutions that remain feasible also when a limited number of resources is removed from the solution. Most studies of robust combinatorial optimization to date made the assumption that every resource is equally vulnerable, and that the set of scenarios is implicitly given by a single budget constraint. This paper studies a robustness model of a different kind. We focus on **bulk-robustness**, a model recently introduced [3] for addressing the need to model non-uniform failure patterns in systems.

We significantly extend the techniques used in [3] to design approximation algorithm for bulk-robust network design problems in planar graphs. Our techniques use an augmentation framework, combined with linear programming (LP) rounding that depends on a planar embedding of the input graph. A connection to cut covering problems and the dominating set problem in circle graphs is established. Our methods use few of the specifics of bulk-robust optimization, hence it is conceivable that they can be adapted to solve other robust network design problems.

1998 ACM Subject Classification G.2.2 Network problems

Keywords and phrases Robust optimization, Network design, Planar graph, Approximation algorithm, LP rounding

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.61

1 Introduction

Robust optimization is concerned with finding solutions that perform well in any one of a given set of scenarios. Many paradigms were proposed for robust optimization in the last decades. Some models assume uncertainty in the cost structure of the optimization problem. Such robust models typically have as scenarios different cost structures for the resources in the system, and ask to find a solution whose worst-case cost is as small as possible. Another kind of robustness postulates uncertainty in the feasible set of the optimization problem. Typically, in such models scenarios correspond to different realizations of the feasible set. A minimum-cost solution is then sought that is feasible in any possible realization of the feasible set.

This paper deals with the latter class of robust models, i.e. ones that incorporate uncertainty in the feasible set. Concretely, we are interested in **robust network design problems**, that are generally defined as follows. The input specifies a graph $G = (V, E)$, a set of **failure scenarios** Ω, consisting of subsets of the nodes and edges of G, and some connectivity requirement. The goal is to find a minimum-cost subgraph of G satisfying the connectivity requirement, even when the elements in any one single scenario are removed from the solution. Different problems are obtained for different types of connectivity requirement and when different representations of the scenario set are assumed.

Most existing models of robust network design assume **uniform scenario sets**. Given interdiction costs for the resources, and a bound $B \in \mathbb{Z}_{\geq 0}$, such models assume that the adversary can remove **any** subset of resources of interdiction cost at most B. In fact, unit interdiction costs are almost always assumed. While such uniform robust network design problems often enjoy good algorithms, they also often do not reflect realistically the uncertainty in the modeled system, which feature highly non-uniform failure patterns.

In a recent paper, Adjiashvili, Stiller and Zenklusen [3] introduced a new model for robust network design called **bulk-robustness**, specifically designed to model such highly non-uniform failure patterns. In bulk-robust optimization failure scenarios are given explicitly, as a list of subsets of the resources. These subsets may be arbitrary, and in particular, they are allowed to vary in size. The goal, as in robust network design problems, is to find a minimum-cost set of resources that contains a feasible solution, even when the resources in any one of the scenarios are removed.

The authors justify the model by bringing many example from health care optimization, computer systems, digitally controlled systems, military applications, financial systems and more. For example, in computer systems, different components of the network rely on the different resources, such a databases, power sources etc. At down-times of such resources, the components that can not operate properly are exactly those that depend of the downed resource. While no uniform failure model can capture such failure patterns, bulk-robutness seems to be a suitable choice.

In [3] the authors study a number of problems in the bulk-robust model, including the s-t connection problem, and the spanning tree problem. In particular, the approximability of these problems is studied in general graphs. The goal of this paper is to extend the existing tool set available for designing approximation algorithm for robust network design in this model. In this paper we focus on the important special case of **planar graphs**. We show a widely-applicable method for computing approximate solutions to bulk-robust network design problem in planar graphs.

1.1 Results and methods

For an integer $r \in \mathbb{Z}_{\geq 0}$ we let $[r] = \{1, \cdots, r\}$ and $[r]_0 = \{0, 1, \cdots, r\}$. The **bulk-robust network design problem** is defined as follows. Given an undirected graph $G = (V, E)$, a weight function $w : E \to \mathbb{Z}_{\geq 0}$, a connectivity requirement $\mathcal{C}$ and a set of m *scenarios* $F_1, \cdots, F_m$, each comprising a set of edges $F_i \subseteq E$, find a minimum-cost set of edges $S \subseteq E$, such that $(V, S \setminus F_i)$ satisfies $\mathcal{C}$ for every $i \in [m]$. When $\mathcal{C}$ is the requirement that two specific nodes $s, t \in V$ are to be connected we obtain the **bulk-robust s-t connection problem**. When $\mathcal{C}$ is the requirement that all nodes are pair-wise connected, we obtain the **bulk-robust spanning tree problem**. Other bulk-robust problems such as **bulk-robust Steiner tree** and **bulk-robust survivable network design** are obtained analogously, by choosing the appropriate $\mathcal{C}$. We let $n = |V|$, and $k = \max_{i \in [m]} |F_i|$ denote the maximum size of a scenario. The parameter k is called **the diameter** of the instance. Adjiashvili et. al. [3] proved the following theorem.

▶ **Theorem 1** (Adjiashvili et. al. [3]). *The bulk-robust s-t connection problem admits a polynomial 13-approximation algorithm in the case $k = 2$. The bulk-robust spanning tree problem admits an $(\log n + \log m)$-approximation algorithm.*

On the complexity side, the authors prove set cover hardness for all considered bulk-robust counterparts, implying a conditional $\log m$ lower bound on the approximation in general graphs. In terms of the parameter k, the authors show that in general graphs a sub-exponential approximation factor is likely not achievable for certain variants of the bulk-robust s-t connection problem.

1.1.1 Contribution

Our goal is to prove a significant strengthening of Theorem 1 for the special case where the input graph is planar. Concretely, we prove the following theorem.

▶ **Theorem 2.** *The bulk-robust s-t connection and the bulk-robust spanning tree problems admit polynomial $O(k^2)$-approximation algorithms on planar graphs.*

The latter result implies constant-factor approximation algorithms for the case of fixed k. In light of the results in [3] this is qualitatively best possible. To complement our algorithmic result we also prove the following stronger inapproximability result.

▶ **Theorem 3.** *For some fixed constant $c > 0$ it is NP-hard to approximate the bulk-robust s-t connection and bulk-robust spanning tree problems within a factor of ck, even when the input graphs are restricted to series-parallel graphs.*

The expression ck in the latter theorem can be replaced with the concrete expression $\frac{1}{2}k - 1 + \frac{1}{2k} - \epsilon$. Theorem 3 suggests that the dependence of the approximation factor on k is necessary.

For concreteness and clarity of the exposition we prove Theorem 2 for the s-t connection problem. We discuss the necessary minor adaptation needed for the spanning tree problem later. Furthermore, the methods we employ use very little of the particularities of bulk-robust optimization, and are thus likely to be adaptable to other robust problems on planar graphs.

1.1.2 Our methods

Our algorithm is a combination of combinatorial and LP-based techniques. On the top level, our algorithm employs an augmentation framework, which constructs a feasible solution by solving a sequence of relaxations of the problem. The lowest level corresponds to a simple polynomial problem, while the last level corresponds to the original instance. The idea of augmentation is well known in the literature of network design (see e.g. [26, 14]). We use here the variant of the augmentation framework defined for bulk-robust optimization in [3].

We solve each stage of the augmentation problem by considering a suitable set cover problem, the analysis of which comprises the core technical contribution of the paper. Using a combinatorial transformation that amounts to finding certain shortest paths in the graph, we obtain a simpler covering problem, which we call **the link covering problem**. The remainder of the algorithm relies on the analysis of the standard LP relaxation of the latter problem. Using properties of planar graphs, we show that the obtained LP has an integrality gap of $O(k)$, and that a solution of this quality can be obtained in polynomial time. Our rounding procedure relies on a decomposition according to the planar embedding of the graph, and a connection to the dominating set problem in circle graphs, for which we develop an LP-respecting constant-factor approximation algorithm. The line of our proof follows that of the proof in [3]. Our main technical contribution can hence be seen in the additional techniques developed to deal with planar graphs. As we mentioned before, these new techniques seem more general than the bulk-robust model, and are likely to be applicable to other network design problems in planar graphs.

The proof of Theorem 3 relies on a reduction form the minimum vertex cover problem in m-uniform, m-partite hypergraphs.

1.1.3 Organization

In the remainder of this section we review related work. In Section 2 we present the algorithm for the bulk-robust s-t connection problem, and prove Theorem 2 for this case. The required modification for the bulk-robust spanning tree problem and possible extensions of our results are discussed in Section 3. The proof of Theorem 3 is brought in Appendix B.

1.2 Related work

For a comprehensive survey on general models for robust optimization we refer the reader to the paper of Bertsimas, Brown and Caramanis [7].

Robustness discrete optimization with cost uncertainty was initially studied by Kouvelis and Yu [22] and Yu and Yang [27]. These works mainly consider the min-max model, where the goal is to find a solution that minimizes the worst-case cost according to the given set of cost functions. See the paper of Aissi, Bazgan and Vanderpooten [5] for a survey. A closely related class of multi-budgeted problem has received considerable attention recently (see e.g. [25, 24, 8, 16] and references therein).

An interesting class of problems with uncertainty in the feasible set was introduced by Dhamdhere, Goyal, Ravi and Singh [11]. In this two-stage models the feasibility condition is only fully revealed in the second stage. While resources can be bought in both stages, they are cheaper in the first stage, in which only partial information about the feasible set is available. This model was subsequently studied by several other authors (see [15, 12, 21]). Different two-stage model was proposed in [4, 2] for the shortest path problem. Several other important network design problems are motivated by robust optimization. Such problems include the minimum k-edge connected spanning subgraph problem [9, 13] and the survivable network design problem [19, 20]. Various other robust variants of classical combinatorial optimization problems were proposed. For a survey of these results we refer the reader to the theses of Adjiashvili [1] and Olver [23].

2 Bulk-robust s-t connection in planar graphs

In this section we are concerned with the **bulk-robust s-t connection problem**, which given an undirected graph $G = (V, E)$, a weight function $w : E \to \mathbb{Z}_{\geq 0}$, two terminals $s, t \in V$ and a set of m scenarios $F_1, \cdots, F_m \subseteq E$, asks to find minimum-cost set of edges S, such that $S \setminus F_i$ contains an s-t path for every $i \in [m]$.

The remainder of the section is organized as follows. First we explain the augmentation framework in general. Then we define the set cover problem for the i-th augmentation step and analyze its properties. Finally, we propose a LP-based approximation algorithm for the set cover problem.

2.1 The augmentation framework

Consider the following sequence of relaxations of the given instance of the bulk-robust s-t connection problem. For an integer $i \in [k]_0$ define Ω_i to be the collection of subsets of cardinality at most i of the failure scenarios $F_1, \cdots, F_m$, i.e.

$$\Omega_i = \{F \subseteq E \mid \exists j \in [m]\ F \subseteq F_j \ \wedge \ |F| \leq i\}.$$

Now, define the ***i*-th level relaxation** P_i of our instance to be the instance where Ω is replaced by Ω_i. Clearly P_0 is simply the shortest path problem, as $\Omega_0 = \{\emptyset\}$, and P_k is the original instance. Furthermore, we indeed obtained a sequence of relaxations, as any feasible solution for P_i is feasible for P_j if $i \geq j$.

The augmentation framework constructs the solution for the given instance by iteratively adding additional edges to the solution. The solution X_{i-1} obtained until the beginning of the i-th augmentation step is feasible for P_{i-1}. The **i-th augmentation problem** is to augment X_{i-1} with additional edges A_i of minimum cost so that $X_{i-1} \cup A_i$ is feasible for P_i. We denote by AUG_i the optimal value the i-th augmentation problem.

2.2 The i-th augmentation problem

In the first iteration, the problem P_0 becomes the shortest s-t path problem, and is solved in polynomial time by any shortest path algorithm. We denote by $X_{i-1} \subseteq E$ the set of edges presented to the i-augmentation problem, and let $G_{i-1} = (V, X_{i-1})$.

Consider the i-th augmentation problem for some $i \geq 1$. Since X_{i-1} is a feasible solution of P_{i-1}, we know that any scenario Ω_j for $j < i$ does not disconnect s from t in G_{i-1}. The same may hold true for some scenarios in Ω_i. If this holds for all scenarios in Ω_i, then X_{i-1} is already feasible for P_i, and we can set $X_i = X_{i-1}$. In the other case, some scenarios in Ω_i are still relevant, i.e. they disconnect s from t in G_{i-1}. We abuse notation and let Ω_i denote this set of relevant scenarios.

Let us formulate the i-th augmentation problem as a set cover problem. To this end we let $E_i = E \setminus X_{i-1}$ denote the set of edges not yet chosen to be included in the solution. Let $\bar{V} = V[X_{i-1}]$ be the set of nodes incident to X_{i-1}. Let us define the following useful notion of links.

▶ **Definition 4.** Let $u, v \in \bar{V}$ be distinct nodes. Define the u-v **link** $L_{u,v}$ to be any shortest u-v path in (V, E_i). Let $\ell_{u,v} = w(L_{u,v})$ denote the length of this path.

Consider any optimal solution A^* to the i-th augmentation problem. It is easy to see that A^* is acyclic, i.e. it forms a forest in (V, E_i). Instead of looking for forests, however, we would like to restrict our search to collections of links.

The advantage of using links is twofold. On the one hand, it is possible to compute all links using a shortest path algorithm in polynomial time. On the other hand, using links will allows us to decompose the augmentation problem in a later stage. Let us define the notion of covering with links next.

▶ **Definition 5.** A link $L_{u,v}$ is said to **cover** $F \in \Omega_i$ if its endpoints u and v lie on different sides of the cut formed by F.

It is easy to see that a union of links forms a feasible solution to the augmentation problem if and only if for every set $F \in \Omega_i$, at least one of the links in the union covers F. This formulation naturally gives rise to our desired set cover problem, defined next.

▶ **Definition 6.** The **i-th link covering problem** asks to find a collection of links of minimum total cost, covering every scenario $F \in \Omega_i$.

We also know that feasible solutions to the i-th links covering problem correspond to feasible solutions of the i-th augmentation problem with the same objective function value, or better. The following lemma from [3] states that any feasible solution to the i-th augmentation problem corresponds to a feasible solution to the i-th links covering problem of at most twice the cost, thus by solving the link covering problem we lose at most a factor of 2.

▶ **Lemma 7** (Adjiashvili et. al. [3]). *There exists a collection $Q_1, \cdots, Q_r \subseteq E_i$ of paths, such that for each $F \in \Omega_i$, the collection contains at least one path covering F and $\sum_{j=1}^{r} w(Q_j) \leq 2\mathrm{AUG}_i$.*

Before proposing an approximation algorithm for the link covering problem let us make the following additional assumption. We assume that **every** edge $e \in X_{i-1}$ appears in at least one scenario from Ω_i. This assumption does not compromise generality, as any edge not satisfying the latter condition can be safely contracted for the solution of the i-th augmentation problem.

2.3 Approximating the link covering problem

We focus next on approximating the i-th link covering problem. For simplicity we drop the index i from our notation in this section and use X, Ω and P for X_{i-1}, Ω_i and P_i, respectively. The case $i = 1$ is particularly simple and is treated as follows. In the case $i = 1$ the set X simply corresponds to an s-t path. This path can be seen as a line and links can be seen as intervals on this line. Scenarios $F \in \Omega$ are singletons corresponding to edges on this path, and are interpreted as points on the line. The link covering problem now becomes an interval covering problem that can be solved exactly in polynomial time using various algorithms.

In the case $i \geq 2$, which we henceforth assume, the situation is much more complex. Consider next the following standard linear programming relaxation of the link covering problem. We include a variable $x_{u,v} \in [0, 1]$ for each link $L_{u,v}$, where $x_{u,v} = 1$ is interpreted as including the link $L_{u,v}$. Furthermore, we denote by $cover(F)$ all pairs $\{u, v\} \subseteq \bar{V} \times \bar{V}$ such that the link $L_{u,v}$ covers F.

$$\min \left\{ \ell(x) : x_{u,v} \geq 0 \ \forall \{u, v\} \in \bar{V} \times \bar{V}, \quad \sum_{\{u,v\} \in cover(F)} x_{u,v} \geq 1 \quad \forall F \in \Omega \right\}$$

It is well-known that in general, the latter LP has an integrality gap as large as $\log N$, where N is the size of the ground set of the set cover problem. Our goal here is to show that in the case of the link covering problem and when the input graph is required to be **planar**, a stronger bound can be proved. Concretely, we will show that a fractional solution x^* to the LP can be rounded in polynomial time to an integral solution with cost at most $8i\ell(x^*)$, thus also proving a bound of $8i$ on the integrality gap.

2.3.1 Solving the LP

Before we turn to our rounding algorithm, let us discuss the problem of solving the latter LP. Clearly, if k is a fixed constant, the size of the LP is polynomial, and any polynomial time LP algorithm can be used. In the other case, when the diameter k is not bounded by a constant, the sets Ω might have exponential size, as they potentially contain all subsets of cardinality $i \leq k$ of sets of cardinality k. It is however not difficult to design a polynomial-time separation procedure for the latter LP as follows. Given a fractional vector x, we can check if it is feasible for the LP by checking for every one of the polynomially many failure scenarios $F_1, \cdots, F_m$, if it contains a subset F of size i that is both an s-t cut in $\bar{G} = (V, X)$, and

$$\sum_{\{u,v\} \in \bar{V} \times \bar{V} \text{ covers } F} x_{u,v} < 1.$$

Let us call a set F of the latter type **violating**. This can be achieved as follows. Let F_j be the scenario from the family of input scenarios that we would like to test. Let $H = (\bar{V}, Y)$ be the graph obtained from $\bar{G}$ by adding the direct edge $\{u,v\}$ for every pair of distinct nodes $u, v \in \bar{V}$. The new edge $\{u,v\}$ represents the link $L_{u,v}$. Define an edge capacity vector $c : Y \to \mathbb{R}_{\geq 0}$ on the new edge set Y setting $c_j(e) = 1$ if $e \in F_j$, $c_j(e) = \infty$ if $e \in X \setminus F_j$ and $c_j(e) = x_e$ if $e \in Y \setminus X$. It is now easy to verify that a violating set $F \subseteq F_j$ exists if and only if the capacity of the minimum s-t cut in H with capacity vector c is strictly bellow $i+1$. Furthermore, if such a cut exists, the set F can be chosen to be all edges of F_j crossing the minimum cut. Polynomiality of the latter transformation and the minimum s-t cut problem now imply that the Ellipsoid algorithm can be used to solve the LP in polynomial time.

2.3.2 Rounding the LP

Let x^* denote an optimal solution to our LP. We describe our rounding procedure next.

Our rounding technique heavily exploits the planarity of the input graph G. Let us henceforth assume that G is presented with a planar embedding Γ. Such an embedding can be computed in polynomial time. We let $\psi^1, \cdots, \psi^{\bar{q}}$ denote the faces of the embedding of $\bar{G}$, induced by the embedding of G.

▶ **Definition 8.** We say that link $L_{u,v}$ is of **type** j if it connects two nodes u, v on ψ^j, and if $L_{u,v}$ is completely contained in the face ψ^j. We call a link **typed** if it is of type j for some $j \in [\bar{q}]$. Links that are not typed are called **untyped**.

For what follows it will be convenient to assume that x^* is **clean**, i.e. that $x^*_{u,v} = 0$ holds for every untyped link $L_{u,v}$. This assumption does not compromise generality, as we state and prove in the following lemma.

▶ **Lemma 9.** *Restricting the solutions of the LP to satisfy $x_{u,v} = 0$ for every untyped link $L_{u,v}$ does not change the optimal value of the LP.*

Proof. Assume that x^* is an optimal solution to the LP with minimum possible weight assigned to untyped links

$$\sum_{\{u,v\} \in \bar{V} \times \bar{V} \text{ untyped}} x^*_{u,v}.$$

Assume towards contradiction that $x^*_{u,v} > 0$ holds for some untyped link $L_{u,v}$. Since $L_{u,v}$ is untyped, it forms a shortest path between the nodes u and v, composed of edges contained in several faces of $\bar{G}$. Let $u = v_1, \cdots, v_p = v$ be nodes on $L_{u,v}$ with the following properties.

- The nodes appear in this order on $L_{u,v}$, when it is traversed from u to v.
- For every $i \in [p-1]$, it holds that $L_{v_i, v_{i+1}}$ is a typed link, i.e. it holds that $v_i, v_{i+1} \in \bar{V}$ and the sub-path of $L_{u,v}$ between v_i and v_{i+1} is completely contained in some face ψ^{j_i}.

Now, consider the LP solution y where

- $y_{u,v} = 0$,
- $y_{v_i, v_{i+1}} = \min\{1, x^*_{v_i, v_{i+1}} + x^*_{u,v}\}$ for every $i \in [p-1]$, and
- $y_{w,z} = x^*_{w,z}$ everywhere else.

Since all links are shortest paths we have $w(L_{u,v}) = \sum_{i \in [p-1]} w(L_{v_i, v_{i+1}})$, and thus $\ell_{u,v} = \sum_{i \in [p-1]} \ell_{v_i, v_{i+1}}$. This implies that $\ell(y) \leq \ell(x^*)$.

The new solution is also a feasible LP solution. To see this we only need to verify that for every $F \in \Omega$, the constraint

$$\sum_{\{z,w\} \in cover(F)} y_{z,w} \geq 1$$

holds. If $L_{u,v}$ does not cover F, this is obvious from feasibility of x^*, since $y_{z,w} \geq x^*_{z,w}$ for all links except $L_{u,v}$.

In the remaining case $L_{u,v}$ covers F. Now, since the union of links $\cup_{i \in [p-1]} L_{v_i,v_{i+1}}$ contains a u-v path, clearly at least one of these links, say $L_{v_{i^*},v_{i^*+1}}$, also covers F. If $y_{v_{i^*},v_{i^*+1}} = 1$ we are clearly done. In the other case

$$y_{v_{i^*},v_{i^*+1}} = x^*_{v_{i^*},v_{i^*+1}} + x^*_{u,v},$$

and thus what is lost by reducing $x^*_{u,v}$ is compensated by increasing $x^*_{v_{i^*},v_{i^*+1}}$, and the constraint is also satisfied.

Finally, we obtained a new optimal solution y with a lower weight assigned to untyped links, as all the links of the form $L_{v_i,v_{i+1}}$ are typed links, and the link $L_{u,v}$ is untyped. This contradicts the choice of x^*. ◀

A set of links S is **clean** if it only contains typed links. The following lemma proves certain useful connections between the planar embedding of G and the link covering problem, which we later use to round the LP solution. We say that an edge is **on the boundary of a face** if both of its endpoint lie on the face.

▶ **Lemma 10.** *Let $F \in \Omega$ be some failure scenario and let $\psi \in \{\psi^1, \cdots, \psi^{\bar{q}}\}$ be some face. Then, if $i \geq 2$, the number of edges of F that lie on the boundary of ψ is either zero or two. Furthermore, the number of faces $\{\psi^1, \cdots, \psi^{\bar{q}}\}$ that contain two edges of F on their boundary is exactly i.*

Proof. Since $F \in \Omega$ we know that F is an s-t cut in $\bar{G}$. Observe that $(\bar{V}, X \setminus F)$ contains exactly two connected components, one $C^s(F)$ containing s and one $C^t(F)$ containing t. This holds since, by definition of Ω and the augmentation problem, the set X is feasible for P_{i-1}, and thus any subset of F is **not** an s-t cut in $\bar{G}$.

This implies that all edges in F can be directed unambiguously from the node in $C^s(F)$ to the node in $C^t(F)$. Now consider any edge $e \in F$ and any face $\psi \in \{\psi^1, \cdots, \psi^{\bar{q}}\}$ which contains e on its boundary. Since ψ corresponds to a cycle in $\bar{G}$, the number of edges of F on its boundary cannot be odd, as an odd number of such edges would imply the existence of path in $\bar{G}$ connecting $C^s(F)$ to $C^t(F)$, and containing no edge of F.

Next we prove that this number must be two, i.e. that ψ contains exactly one more edge of F. Assume towards contradiction that there are at least four such edges. By traversing the cycle in $\bar{G}$, forming the face ψ, the cut defined by F is crossed every time an edge of F is crossed. In particular, there are some four nodes u_1, v_1, u_2, v_2 appearing in this order on the face, and such that u_1, u_2 belong to $C^s(F)$ and v_1, v_2 belong to $C^t(F)$. Let Q and R be a u_1-u_2 path in $C^s(F)$ and a v_1-v_2 path in $C^t(F)$, respectively. Since ψ is a face, the embedding of both Q and R is disjoint from the interior of ψ. Now, since Q and R form continuous curves in the plane, and are connected to alternating nodes on the boundary of a face, they must intersect at some point, contradicting the fact that $C^s(F)$ and $C^t(F)$ are different connected components in $(\bar{V}, X \setminus F)$. Figure 1 illustrates this argument.

Finally, since every edge $e \in F$ belongs to the boundary of exactly two faces from $\psi^1, \cdots, \psi^{\bar{q}}$, and since every face containing some edge of F on the boundary contains exactly two such edges, we conclude that there are exactly $|F| = i$ faces containing some edge of F on the boundary. In the first assertion we assumed there are no cut edges in $\bar{G}$. For $i \geq 2$ this can be assume without loss of generality, as cut edges are either contracted in the pre-processing stage before the augmentation step, or, they are redundant, and can be removed from X. ◀

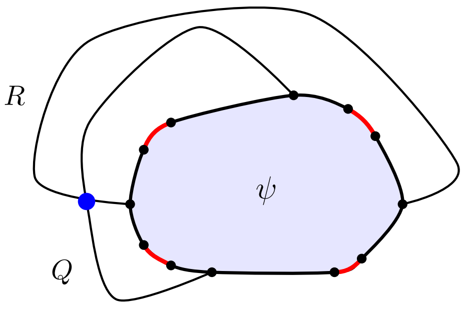


Figure 1 The situation in the proof of Lemma 10.

For simplicity we say that a **scenario** $F \in \Omega$ **is contained in a face** $\psi \in \{\psi^1, \cdots, \psi^{\bar{q}}\}$, if two edges of F lie on the boundary of ψ. Lemma 10 implies that a clean set S of links is feasible if and only if for every $F \in \Omega$, there exists a face ψ^j containing F, and a link $L_{u,v} \in S$ of type j with u and v on different sides of the cut defined by F. With this criterion we are ready to prove the main lemma of this section.

▶ **Lemma 11.** *Let x be a clean feasible solution to the LP. Then, there exists a feasible set S of links with total cost at most $8i\ell(x)$.*

Proof. We construct the desired set of links in two steps. First, we partition the set of scenarios Ω into $\bar{q}$ parts, one for each face of $\bar{G}$. In the second stage, we process the faces of $\bar{G}$ one by one, and for each face we use the part of the LP solution x corresponding to the face to construct a set of links that cover the scenarios assigned to that face.

Consider any scenario $F \in \Omega$. Since x is feasible we have

$$\sum_{\{u,v\} \in cover(F)} x_{u,v} \geq 1.$$

Let $\psi^{p_1}, \cdots, \psi^{p_i} \in \{\psi^1, \cdots, \psi^{\bar{q}}\}$ be the set of faces that contain F. According to Lemma 10, there are exactly i such faces. Now, since x is clean, the latter sum can be decomposed as follows.

$$\sum_{\{u,v\} \in cover(F)} x_{u,v} = \sum_{j=1}^{i} \sum_{\substack{\{u,v\} \in cover(F) \\ L_{u,v} \text{ type } p_j}} x_{u,v}$$

Let us denote the second sum on the right hand side by $\sigma^j[F]$, i.e. let

$$\sigma^j[F] = \sum_{\substack{\{u,v\} \in cover(F) \\ L_{u,v} \text{ type } p_j}} x_{u,v}.$$

Now since $\sum_{j=1}^{i} \sigma^j[F] \geq 1$, there exists at least one index $j \in [i]$ such that $\sigma^j[F] \geq \frac{1}{i}$. We let $j[F] \in [i]$ be one such index. If several indices $j \in [i]$ satisfy the latter condition, one is chosen arbitrarily. Note that the index $j[F]$ is chosen is such a way that in the LP solution x, the total weight of links of type $p^{j[F]}$ that cover F is at least $\frac{1}{i}$.

We are now ready to define the partition of Ω into $\bar{q}$ parts, corresponding to the $\bar{q}$ faces of $\bar{G}$. For $j \in [\bar{q}]$ we let

$$\Omega^{(j)} = \left\{ F \in \Omega \mid p^{j[F]} = j \right\}.$$

Clearly, $\Omega = \cup_{j\in[\bar{q}]}\Omega^{(j)}$ is a partition of Ω. To conclude the first stage of the our procedure, it remains to define a corresponding decomposition $x = \sum_{j\in[\bar{q}]} x^{(j)}$ of the LP solution x. The vector $x^{(j)}$ is defined by setting $x^{(j)}_{u,v} = x_{u,v}$ if $L_{u,v}$ is of type j, and $x^{(j)}_{u,v} = 0$ otherwise. This concludes the first step of the rounding procedure.

In the second step we construct for every $j \in [\bar{q}]$, a set of links $S^{(j)}$ of total cost $8i\ell(x^{(j)})$ that covers all scenarios in $\Omega^{(j)}$. By doing so we clearly conclude the proof of the lemma, since by taking $\cup_{j\in[\bar{q}]}S^{(j)}$ we obtain a feasible solution with total cost of at most $\sum_{j\in[\bar{q}]} 8i\ell(x^{(j)}) = 8i\ell(x)$, as desired.

It remains to show how a single set $S^{(j)}$ can be constructed. Our plan is the following. First, we observe that, by construction, $ix^{(j)}$ is an LP solution that fractionally covers all scenarios in $\Omega^{(j)}$. Then, we observe that the link covering problem restricted to links of type j, and to scenarios in $\Omega^{(j)}$ essentially becomes a variant of the **dominating set problem on circle graphs.** We explain the required transformation next, and conclude by proving that the integrality gap of the standard LP relaxation for the latter problem is constant, and that the corresponding integral solution can be found in polynomial time.

Recall that a **circle graph** is an intersection graph of the set of chords in a circle. The dominating set problem in circle graphs hence corresponds to finding a minimum-cost collection of chords that intersect every chord of the graph. We are interested in a variant of this problem, where chords are partitioned into two groups called **demand chords** and **covering chords**, and the goal is to find a minimum-cost set of covering chords that dominates all the demand chords. We call this problem the **restricted dominating set problem in circle graphs**.

The link covering problem restricted to a the face ψ can now be seen as a dominating set problem on circle graphs as follows. Let $v_0, v_1, \cdots, v_d = v_0$ be the nodes on the boundary of ψ. We subdivide each edge $\{v_j, v_{j+1}\}$ for $j \in [d]$ by adding the node w_j. This new cycle corresponds to the circle of the circle graph we construct. Let us define the chords of the graph, and their corresponding weights, next. For every scenario F contained in ψ we add the demand chord α_F connecting w_{j_1} to w_{j_2}, where $\{u_{j_1}, u_{j_1+1}\} \in F$ and $\{u_{j_2}, u_{j_2+1}\} \in F$ (recall from Lemma 10 that there are exactly two such edges). Next, for every link of the form L_{v_l,v_r}, we add the covering chord β_{v_l,v_r} connecting v_l with v_r. The cost of this chord is set to ℓ_{v_l,v_r}, i.e. we set $c(\beta_{v_l,v_r}) = \ell_{v_l,v_r}$. This concludes the transformation.

To see that the latter problem indeed models the desired link covering problem it suffices to make the following simple observation. Sets of chords corresponding to links that form a restricted dominating set in the circle graph are in one-to-one correspondence with sets of links that cover all scenarios, with identical costs. This is true, since a link L_{v_l,v_r} covers a scenario F if and only if the chords α_F and β_{v_l,v_r} intersect.

We can now naturally interpret the solution $y^{(j)} = ix^{(j)}$ as a feasible fractional solution to the standard LP relaxation of the restricted dominating set problem on the obtained circle graph. In the following claim we show that the integrality gap of the latter LP is constant. The proof of the claim uses a connection to a special case of the **axes-parallel rectangle covering problem**, for which Bansal and Pruhs [6] provided an LP-respecting 2-approximation with the natural LP.

▶ **Lemma 12.** *The integrality gap of the standard LP relaxation of the restricted dominating set problem on circle graphs is bounded by* 8.

Proof. Let $H = (V^d \cup V^c, E)$ be the given circle graph with V^d and V^c corresponding to the demand chords and the covering chords, respectively. Let $g : V^c \to \mathbb{Z}_{\geq 0}$ denote the cost function for the covering chords. Let $p_0, \cdots, p_m = p_0$ be all the points on the circle to which

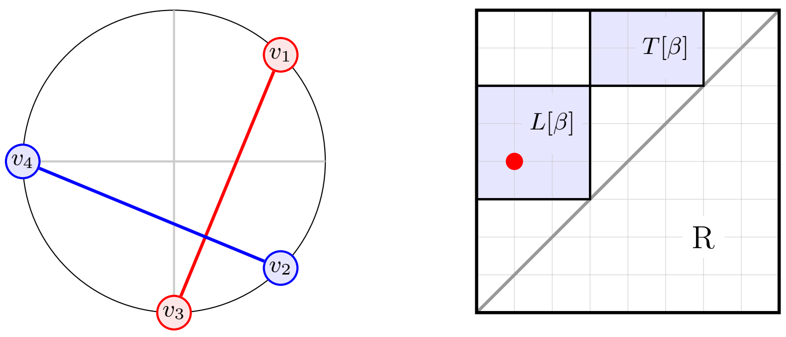


Figure 2 An illustration of the transformation. The chords $\alpha = (v_1, v_3)$ and $\beta = (v_2, v_4)$ are demand and covering chords, respectively. The ordering of the points on the circle is clockwise starting from the highest point.

chords are connected, in the order that they appear when the circle is traversed in some arbitrary direction. For a chord $\alpha \in V^d \cup V^c$ we write $\alpha = (p_l, p_r)$ with $l \le r$ to indicate the endpoints of the chord in the circle.

We interpret the restricted dominating set problem as a kind of **point covering problem by axis-aligned rectangles** as follows. Construct a large square R with side length m. The points in R are indexed by pairs of points on the circle, with (p_0, p_0) and (p_{m-1}, p_{m-1}) being, respectively, the lower-left corner of R and the upper-right corner of R. For four points $p_{l_1}, p_{l_2}, p_{r_1}, p_{r_2}$ with $l_1 \le l_2$ and $r_1 \le r_2$ we denote by $[p_{l_1}, p_{r_1}] \times [p_{l_2}, p_{r_2}] \subseteq \mathrm{R}$ the rectangle contained in R with lower-left point and upper-right point (p_{l_1}, p_{r_1}) and (p_{l_2}, p_{r_2}), respectively.

Demand chords are interpreted as **points in** R. The chord $\alpha = (p_l, p_r) \in V^d$ is interpreted as the point $Q[\alpha] = (p_l, p_r)$ inside R. Observe that $Q[\alpha]$ is contained above the main diagonal in R, that is the line connecting (p_0, p_0) and (p_{m-1}, p_{m-1}), as $p_l \le p_r$.

Covering chords are interpreted as **pairs of rectangles** contained in R. The chord $\beta = (p_l, p_r) \in V^d$ is interpreted as the pair of rectangles

$$L[\beta] = [p_0, p_l] \times [p_l, p_r] \text{ and } T[\beta] = [p_l, p_r] \times [p_r, p_{m-1}].$$

Observe the following property. $L[\beta]$ intersects the left side of R and $T[\beta]$ intersects the top side of R.

It is now straightforward to verify that a covering chord β dominates a demand chord α if and only if

$$Q[\alpha] \in L[\beta] \cup T[\beta].$$

Finally, we arrived at the desired covering problem, namely the problem of selecting a minimum cost set of rectangles pairs $L[\beta] \cup T[\beta]$ in R, corresponding to covering chords, so as to cover every point $Q[\alpha]$, corresponding to demand chords. The cost of a rectangle pair is simply the cost of the corresponding covering chord. Figure 2 illustrates the transformation.

The standard LP relaxation for this covering problem reads

$$\min \left\{ g(z) \mid z_\beta \ \ge 0 \ \ \forall \beta \in V^c, \qquad \sum_{\beta \,:\, Q[\alpha] \in L[\beta] \cup T[\beta]} z_\beta \ge 1 \quad \forall \alpha \in V^d \right\}.$$

Let z be a fractional feasible solution to the latter LP. We construct an integral solution as follows. First, observe that for every demand chord $\alpha \in V^d$, at least one of the following holds due to feasibility of z:

- $\sum_{\beta\,:\,Q[\alpha]\in L[\beta]} z_\beta \geq \frac{1}{2}$
- $\sum_{\beta\,:\,Q[\alpha]\in T[\beta]} z_\beta \geq \frac{1}{2}$

Let $V_L^d \subseteq V^d$ be the set of all $\alpha \in V^d$ for which the first condition holds. Let $V_T^d = V^d \setminus V_L^d$ be all other demand chords. We show how to construct an integral solution of cost at most $4g(z)$ that dominates all chords in V_L^d. From symmetry, this implies that another integral solution can be constructed for V_T^d with cost at most $4g(z)$. This will then prove the claim, as the union of both solutions is an integral feasible solution of cost at most $8g(z)$.

To this end observe that $2z$ is a fractional feasible solution to the LP

$$\min\left\{ g(z) \;\middle|\; z_\beta \;\geq 0 \;\; \forall \beta \in V^c, \qquad \sum_{\beta\,:\,Q[\alpha]\in L[\beta]} z_\beta \geq 1 \quad \forall \alpha \in V_L^d \right\}.$$

Now, it remains to observe that the latter LP is the natural LP relaxation of an ordinary rectangle covering problem. The rectangles $\{L[\beta] \mid \beta \in V^c\}$ also have the additional property that their left side lies on the left side of R. This restricted variant of the rectangle covering problem was studied by Bansal and Pruhs [6], who proved that the standard LP relaxation of the problem has integrality gap of 2. This implies that there exists an integral solution covering V_L^d with cost $4g(z)$. This solution can also be constructed in polynomial time. This concludes the proof of the claim. ◀

Lemma 12 concludes the proof. We note that Lemma 12 provides an 8-approximation algorithms for the dominating set problem in circle graphs based on the natural LP relaxation. While constant factor approximations were known for this problem (see e.g. [10]), to best of our knowledge, none of the existing algorithms use LP rounding. ◀

2.3.3 Putting it all together

We are ready to prove Theorem 2.

Proof. The feasibility of the solution obtained after the final augmentation step is obvious. It remains to compute the approximation guarantee. Let ALG denote the cost of the solution returned by the algorithm. Clearly, $\text{AUG}_i \leq 2\text{OPT}$ holds for every $i \in [k]$, as any optimal solution is feasible for any augmentation problem, and Lemma 7 asserts that by using unions of paths we lose a factor of at most 2. According to Lemma 11, an $8i$-approximation can be obtained for the i-th augmentation problem in polynomial time. Also, the shortest path comprising the solution of P_0 has cost of at most OPT. In total, we obtain the bound $\text{ALG} \leq \text{OPT} + \sum_{i=0}^{k} 8i \cdot 2\text{OPT} = O(k^2)\text{OPT}$.

◀

3 Bulk-robust spanning trees and further extensions

3.1 Bulk-robust spanning trees

Let us discuss first the minor changes needed to prove Theorem 2 for the bulk-robust spanning tree problem. More details are given in Appendix A.

The augmentation procedure remains unchanged, except that at every iteration i, the sets in Ω_i are those sets of cardinality i that disconnect the graph in any way. The link covering problem and the obtained LP are essentially the same. The separation procedure of the LP now requires a polynomial minimum cut algorithm, instead of a minimum s-t cut algorithm. The rounding procedure only used the property that every set $F \in \Omega_i$ creates exactly two connected components when removed from X_{i-1}. Finally, the augmentation algorithm for $i = 1$ is no longer an interval covering problem, but can be treated as in Lemma 11, with the outer face being the only face of $\bar{G}$.

3.2 Further extensions

Let us conclude by discussing some further extensions and implications of our techniques. First, our techniques can clearly be applied to other bulk-robust network design problems. A treatment of the bulk-robust survivable network design problem is deferred to the full version of the paper.

Also, as we mentioned in the introduction, our methods seem to be suitable for solving other robust problems in planar graphs. Consider, for example, the uniform model with varying interdiction costs, where each edge has an interdiction cost $c(e) \in \mathbb{Z}_{\geq 0}$, and the set of scenarios is exactly the set of all edge subsets with total interdiction cost at most $B \in \mathbb{Z}_{\geq 0}$. Our methods can be used to approximate this problem provided that a suitable (approximate) separation oracle is provided for the resulting LP. In general, however, this separation problem coincides with difficult interdiction problems (see e.g. [18, 28] and references therein).

References

1 D. Adjiashvili. *Structural Robustness in Combinatorial Optimization.* PhD thesis, ETH Zürich, Zürich, Switzerland, 2012.

2 D. Adjiashvili, G. Oriolo, and M. Senatore. The online replacement path problem. In *Proceedings of 18th Annual European Symposium on Algorithms (ESA)*, pages 1–12. Springer, 2013.

3 D. Adjiashvili, S. Stiller, and R. Zenklusen. Bulk-robust combinatorial optimization. *Mathematical Programming*, 149(1-2):361–390, 2014.

4 D. Adjiashvili and R. Zenklusen. An s-t connection problem with adaptability. *Discrete Applied Mathematics*, 159:695–705, 2011.

5 H. Aissi, C. Bazgan, and D. Vanderpooten. Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, 2009.

6 N. Bansal and K. Pruhs. The geometry of scheduling. In *51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 407–414, 2010.

7 D. Bertsimas, D. B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM review*, 53:464–501, 2011.

8 C. Chekuri, J. Vondrák, and R. Zenklusen. Multi-budgeted matchings and matroid intersection via dependent rounding. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1080–1097, 2011.

9 J. Cheriyan and R. Thurimella. Approximating Minimum-Size k-Connected Spanning Subgraphs via Matching. *SIAM J. Comput*, 30:292–301, 2000.

10 M. Damian-Iordache and S. V. Pemmaraju. A (2+ ε)-approximation scheme for minimum domination on circle graphs. *Journal of Algorithms*, 42(2):255–276, 2002.

11 K. Dhamdhere, V. Goyal, R. Ravi, and M. Singh. How to pay, come what may: approximation algorithms for demand-robust covering problems. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 367–376, October 2005.

12 U. Feige, K. Jain, M. Mahdian, and V. Mirrokni. Robust combinatorial optimization with exponential scenarios. In Matteo Fischetti and David Williamson, editors, *IPCO 2007*, volume 4513 of *Lecture Notes in Computer Science*, pages 439–453. Springer Berlin / Heidelberg, 2007.

13 H. N. Gabow, M. X. Goemans, É. Tardos, and D. P. Williamson. Approximating the smallest k-edge connected spanning subgraph by lp-rounding. *Networks*, 53(4):345–357, 2009.

14 H. N. Gabow, M.X. Goemans, and D.P. Williamson. An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming, Series B*, 82(1-2):13–40, 1998.

15 D. Golovin, V. Goyal, and R. Ravi. Pay today for a rainy day: Improved approximation algorithms for demand-robust min-cut and shortest path problems. In Bruno Durand and Wolfgang Thomas, editors, *STACS 2006*, volume 3884 of *Lecture Notes in Computer Science*, pages 206–217. Springer Berlin / Heidelberg, 2006.

16 F. Grandoni, R. Ravi, M. Singh, and R. Zenklusen. New approaches to multi-objective optimization. *Mathematical Programming*, 146(1-2):525–554, 2014.

17 V. Guruswami, S. Sachdeva, and R. Saket. Inapproximability of minimum vertex cover on k-uniform k-partite hypergraphs. *SIAM Journal on Discrete Mathematics*, 29(1):36–58, 2015.

18 E. Israeli and R. K. Wood. Shortest-path network interdiction. *Networks*, 40:97–111, 2002.

19 K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21:39–60, 2001.

20 H. Kerivin and A.R. Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005.

21 R. Khandekar, G. Kortsarz, V. Mirrokni, and M. Salavatipour. Two-stage robust network design with exponential scenarios. In Dan Halperin and Kurt Mehlhorn, editors, *ESA 2008*, volume 5193 of *Lecture Notes in Computer Science*, pages 589–600. Springer Berlin / Heidelberg, 2008.

22 P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications.* Kluwer Academic Publishers, Boston., 1997.

23 N.-K. Olver. *Robust network design.* PhD thesis, McGill University, Montreal, Quebec, Canada, 2010.

24 C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 86–92, 2000.

25 R. Ravi, M. V. Marathe, Ravi S. S., Rosenkrantz D. J., and Hunt H. B. Many birds with one stone: Multi-objective approximation algorithms. In *25th annual ACM Symposium on the Theory of Computing (STOC)*, pages 438–447, 1993.

26 D.P. Williamson, M.X. Goemans, M. Mihail, and V.V. Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. *Combinatorica*, 15(3):435–454, 1995.

27 G. Yu and J. Yang. On the robust shortest path problem. *Computers & Operations Research*, 25(6):457–468, 1998.

28 R. Zenklusen. Matching interdiction. *Discrete Applied Mathematics*, 158(15):1676–1690, 2010.

A Bulk-Robust Spanning Trees

In this section we describe the proof of Theorem 2 for the bulk-robust spanning tree problem. As the changes are minor, we choose to follow the outline of the proof given in the main text, and describe the required modifications.

A.1 The augmentation framework

We use the same sets Ω_i, $i \in [k]_0$ to define the relaxations of the problem. The i-th augmentation problem P_i is to augment the set X_{i-1} of edges chosen so far to a set X_i with the property that $(V, X_i \setminus F)$ is a connected graph for all $F \in \Omega_i$.

As for the bulk-robust s-t connection problem, the optimal solution to any augmentation problem is a forest. As a consequence, Lemma 7 still applies, so we can again use unions of paths to approximate the augmentation problem, at the loss of a factor 2.

The notion of links and covering by links is defined as before, except that now cuts formed by sets $F \in \Omega_i$ are arbitrary cuts in the graph, and not just s-t cuts.

A.2 Solving the link covering problem

The approximate solution to the link covering problems are obtained in essentially the same way for the bulk-robust spanning tree problem, as for the bulk-robust s-t connection problem. The differences are minor and are explained next.

The solution for P_0 is computed by computing a minimum spanning tree in the input graph in polynomial time. The cost of this tree is clearly at most OPT.

A.2.1 The case $i = 1$

The link covering problem corresponding to P_1 is no longer equivalent to an interval covering problem, but it can be approximated as follows. Recall that the solution obtained before the first augmentation problem is a spanning tree of the graph. Each edge in this tree either belong to some failure scenario F_i, in which case it comprises a failure scenario in Ω_1, or it is not contained in any failure scenario. In the latter case the edge can be simply contracted, so we henceforth assume that all edges of the tree form a scenario in Ω_1.

The augmentation problem now becomes a standard **connectivity augmentation problem**, where, given a spanning tree T of a graph G, the task is to compute a minimum-cost set of edges, not in the tree, whose addition to the tree will increase the size of the minimum cut in the resulting graph to two. Indeed, on the one hand any set A of edges satisfying that the graph $(V, T \cup A)$ has no cut of size one is feasible, as the removal of any edge of T cannot disconnect this graph. On the other hand, if a set A is such that $(V, T \cup A)$ does contain a cut edge, this edge must belong to T (since T is a spanning tree of G). Since all edges of T are assumed to comprise failure scenarios in Ω_1, this means that A is infeasible for the augmentation problem.

It remains to note that the latter connectivity augmentation problem can be efficiently approximated within a constant factor. One way to achieve this is to use the algorithm for survivable network design in [19].

A.2.2 The case $i \geq 2$

As before, we omit the index i from our notation, as we now discuss the i-th link covering problem for some arbitrary $i \geq 2$.

The set cover LP appropriate for modeling the link covering problem for the bulk-robust spanning tree problem remain exactly the same as before. There is, however, a slight difference in the design of the separation oracle for the LP. Concretely, the construction of the capacitated graph H remains the same, but now violating sets correspond to sets of edges in minimum cuts (instead of minimum s-t cuts), if the value of the minimum cut is bellow

$i+1$. Since minimum cuts can be found in polynomial time, this separation procedure is also polynomial.

Finally, the rounding procedure and its analysis remain unchanged. While it may seem that the proof of Lemma 10 used the fact that $(\bar{V}, X \setminus F)$ contains exactly two connected components $C^s(F)$ and $C^t(F)$, one containing s and the other containing t, the fact that two specific nodes were separated by the cut was never used. The only property that is used is that $(\bar{V}, X \setminus F)$ contains exactly two connected components. Here we can simply use instead the fact that $(V, X \setminus F)$ contains exactly two connected components.

This concludes the description of the required modifications.

B Proof of Theorem 3

Recall that a **hypergraph** is a pair $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a finite set of **nodes**, and $\mathcal{E} \subseteq 2^{\mathcal{V}}$ is a set of subsets of $\mathcal{V}$ called **edges**. Let $m \in \mathbb{Z}_{\geq 0}$. We say that $\mathcal{H}$ is m**-uniform** if $|e| = m$ for every $e \in \mathcal{E}$. Observe that 2-uniform hypergraphs are graphs. $\mathcal{H}$ is m**-partite** if $\mathcal{V}$ can be partitioned into m parts $\mathcal{V} = \mathcal{V}_1 \cup \cdots \cup \mathcal{V}_m$ such that for every $e \in \mathcal{E}$ and for every $j \in [m]$ it holds that

$$|e \cap \mathcal{V}_j| \leq 1.$$

A **vertex cover** of $\mathcal{H}$ is a set $S \subseteq \mathcal{V}$ of nodes that touches every edge, i.e. such that $|S \cap e| \geq 1$ holds for every $e \in \mathcal{E}$. The **hypergraph minimum vertex cover** problem is to find a vertex cover of $\mathcal{H}$ of minimum cardinality.

Our reduction relies on the following hardness-of-approximation result of Guruswami, Sachdeva and Saket [17].

▶ **Theorem 13** (Guruswami et. al. [17]). *For any $\epsilon > 0$ and any $m \geq 4$ it is NP-hard to approximate the minimum hypergraph vertex cover problem within a factor $\frac{m}{2} - 1 + \frac{1}{2m} - \epsilon$, even when the hypergraph is restricted to be m-uniform and m-partite, and the m-partition is given as input.*

We show that the minimum hypergraph vertex cover problem on k-uniform and k-partite hypergraphs can be transformed to an equivalent instance of bulk-robust s-t connection with diameter k, provided that the k-partition is given as input. We then extend the argument to the bulk-robust spanning tree problem.

To this end let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a k-uniform, k-partite hypergraph, and let $\mathcal{V} = \mathcal{V}_1 \cup \cdots \cup \mathcal{V}_k$ be the given k-partition of $\mathcal{V}$. We construct the series-parallel graph to be the input of the bulk-robust s-t connection problem as follows.

Let $p = |\mathcal{E}|$ and $n_j = |\mathcal{V}_j|$ for $j \in [k]$. For every $j \in [k]$ we construct an ordering $e_1^j, \cdots, e_p^j$ of $\mathcal{E}$ corresponding to $\mathcal{V}_j$. This ordering is constructed as follows. First, order the nodes in $\mathcal{V}_j$ in an arbitrary way, say $v_1^j, \cdots, v_{n_j}^j$. Now, construct the ordering of edges by first including all edges incident v_1^j in any order, then all edges incident to v_2^j in any order and so on, until all vertices are traversed. Since $\mathcal{V}_j$ is a part in a k-partition, the latter procedure succeeds in producing an ordering of $\mathcal{E}$, as every edge is incident to exactly one node in $\mathcal{V}_j$. By design, the latter construction satisfies the following useful property that we will use later. For every node $v \in \mathcal{V}$ there exists an index $j \in [k]$ such that the set of edges $\mathcal{E}_v = \{e \in \mathcal{E} \mid v \in e\}$ incident to v appear as a sub-sequence in the j-th ordering. This index j can be chosen such that $v \in \mathcal{V}_j$.

Next, start constructing the series-parallel graph $G = (V, E)$. First, include the nodes s, t and connect them by k node-disjoint paths $P^1, \cdots, P^k$ (the nodes s and t are common to all paths). Each path P^j contains exactly p edges $f_1^j, \cdots, f_p^j$, appearing in this order when P^j

is traversed from s to t. The edge f_l^j is associated with the edge $e \in \mathcal{E}$ of the hypergraph in the l-th position of the j-th ordering, i.e., the edge e_l^j. Clearly, every edge $e \in \mathcal{E}$ is associated with exactly one edge of G on every path P^j and thus, in total, it is associated with k edges of G.

Next, for every $j \in [k]$ and $v \in \mathcal{V}_j$ add the edge α_v to G, connecting two nodes w_v^j and z_v^j on P^j. These nodes are selected so that the set of edges between these nodes on the path P^j are exactly those that are associated with edges in $\mathcal{E}_v$. Since the hypergraph edge in $\mathcal{E}_v$ appear as a sub-sequence in the order used to construct P^j, such two nodes w_v^j and z_v^j exist. It is straightforward to verify that G is series-parallel.

To complete the construction of the graph we set the weights of all edges on the paths P^j for $j \in [k]$ to zero, while the weight of edges of the type α_v for $v \in \mathcal{V}$ is set to one.

To conclude the reduction it remains to specify the scenario set Ω of the bulk-robust s-t connection problem. For $e \in \mathcal{E}$ we include in Ω a single failure scenario $F_e \subseteq E$. The set F_e contains the k edges, one from every path P^j, $j \in [k]$, that are associated with e. Since every edge $e \in \mathcal{E}$ is associated with exactly one edge on every such path, we have $|F_e| = k$, as required.

We conclude the proof by showing that the resulting instance of the bulk-robust s-t connection problem is equivalent to the hypergraph vertex cover instance. Formally, we show that a solution to the hypergraph vertex cover instance can be transformed to a solution of the bulk-robust s-t connection instance with the same cost, and vice versa.

Assume first that $S \subseteq \mathcal{V}$ is solution to the hypergraph vertex cover problem. Construct a solution $X \subseteq E$ to the the bulk-robust s-t connection problem as follows. Include in X all paths P^j for $j \in [k]$ (at zero cost), as well as all edges α_v such that $v \in S$. This solution has cost $|S|$, as required. To see that this solution is feasible, consider any $F = F_e \in \Omega$. Since S is a vertex cover, there exists some $v \in S$ such that $v \in e$. Let $j \in [k]$ be such that $v \in \mathcal{V}_j$. Observe that the path starting at s, following P^j until w_v^j, then crossing α_v and then continuing to t on P^j is contained in $X \setminus F$.

Assume next that $X \subseteq E$ is a feasible solution to the bulk-robust s-t connection instance. Let $S = \{v \in \mathcal{V} \mid \alpha_v \in X\}$. By the cost structure of the reduction we know that the cost of X is exactly $|S|$. It remains to prove that S is a vertex cover. Consider any $e \in \mathcal{E}$. Since X is feasible, there exists an s-t path $Q \subseteq X \setminus F_e$. This s-t path must use some edge of the form α_v for $v \in S$ as, by construction, every path P^j intersects every failure scenario, and these are node-disjoint s-t paths. Furthermore, we can assume that α_v is the only such edge, as from every path P^j, only one edge is contained in F_e. Now, this edge $\alpha_v \in Q$ connects some nodes w_v^j and z_v^j on some path P^j. Consequently, the unique edge in $F_e \cup P^j$ is contained on the sub-path connecting w_v^j and z_v^j and hence, by construction, $v \in e$. We conclude that S is a vertex cover, as required.

The proof of the theorem for the bulk-robust s-t connection problem now directly follows from the latter reduction and Theorem 13.

The reduction for the bulk-robust spanning tree problem is identical, and so is the correspondence between feasible solutions to the hypergraph vertex cover and the bulk-robust spanning tree problems. The paths $P^1, \cdots, P^k$ together already span the entire graphs. Since all cuts formed by failure scenarios are s-t cuts, the result readily follows from our previous arguments.

◀

Large Supports are Required for Well-Supported Nash Equilibria

Yogesh Anbalagan*[1], Hao Huang[2], Shachar Lovett†[3], Sergey Norin‡[4], Adrian Vetta§[1,4], and Hehui Wu[5]

1 School of Computer Science, McGill University
Montreal, Canada
yogesh.anbalagan@mail.mcgill.ca

2 Institute for Mathematics and its Applications, University of Minnesota
Minneapolis, USA
huanghao@ima.umn.edu

3 Computer Science and Engineering Department, University of California
San Diego, USA
slovett@cs.ucsd.edu

4 Department of Mathematics and Statistics, McGill University
Montreal, Canada
{snorin, vetta}@math.mcgill.ca

5 Department of Mathematics, University of Mississippi
University, USA
hhwu@olemiss.edu

Abstract

We prove that for any constant k and any $\epsilon < 1$, there exist bimatrix win-lose games for which every ϵ-WSNE requires supports of cardinality greater than k. To do this, we provide a graph-theoretic characterization of win-lose games that possess ϵ-WSNE with constant cardinality supports. We then apply a result in additive number theory of Haight [8] to construct win-lose games that do not satisfy the requirements of the characterization. These constructions disprove graph theoretic conjectures of Daskalakis, Mehta and Papadimitriou [7] and Myers [10].

1998 ACM Subject Classification G.2.3 Discrete Mathematics – Applications, J.4 Computer Applications – Social and Behavioral Sciences

Keywords and phrases bimatrix games, well-supported Nash equilibria

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.78

1 Introduction

A Nash equilibrium of a bimatrix game (A, B) is a pair of strategies that are mutual best-responses. Nash equilibria always exist in a finite game [11], but finding one is hard, unless $PPAD \subseteq P$ [5]. This has lead to the study of relaxations of the equilbrium concept. A notable example is an *ϵ-approximate Nash equilibrium* (ϵ-NE). Here, every player must receive an expected payoff within ϵ of their best response payoff. Thus ϵ-NE are numerical relaxations of Nash Equilibria. Counterintuitively, however, given that Nash's existence result is via a

* Supported by a MITACS fellowship and NSERC grant 429598.
† Partially supported by NSF CAREER award 1350481.
‡ Supported by NSERC grant 418520.
§ Supported by NSERC grants 288334 and 429598.

fixed point theorem, Nash equilibria are intrinsically combinatorial objects. In particular, the crux of the equilibrium problem is to find the supports of the equilibrium. In particular, at an equilibrium, the supports of both strategies consist only of best responses. This induces a combinatorial relaxation called an *ϵ-well supported approximate Nash equilibrium* (ϵ-WSNE). Now the content of the supports are restricted, but less stringently than in an exact Nash equilibrium. Specifically, both players can only place positive probability on strategies that have payoff within ϵ of a pure best response.

Observe that in an ϵ-NE, no restriction is placed on the supports of the strategies. Consequently, a player might place probability on a strategy that is arbitrarily far from being a best response! This practical deficiency is forbidden under ϵ-WSNE. Moreover, the inherent combinatorial structure of ϵ-WSNE has been extremely useful in examining the hardness of finding Nash equilibria. Indeed, Daskalakis, Goldberg and Papadimitriou [6] introduced ϵ-WSNE in proving the PPAD-completeness of finding a Nash equilibrium in multiplayer games. They were subsequently used as the notion of approximate equilibrium by Chen, Deng and Teng [5] when examining the hardness of bimatrix games.

This paper studies the (non)-existence of ϵ-WSNE with small supports. Without loss of generality, we may assume that all payoffs in (A, B) are in $[0, 1]$. Interestingly, for ϵ-NE, there is then a simple $\frac{1}{2}$-NE with supports of cardinality at most two [7]. Take a row r. Let column c be a best response to r, and let r' be a best response to c. Suppose the row player places probability $\frac{1}{2}$ on r and r', and the column player plays column c as a pure strategy. It is easy to verify that this is a $\frac{1}{2}$-NE. On the other hand, Althöfer [1] showed the existence of zero-sum games for which every ϵ-NE, with $\epsilon < \frac{1}{4}$, requires supports of cardinality at least $\log n$. This result is almost tight; a probabilistic argument shows the existence of ϵ-NE with supports of cardinality $O(\frac{log\, n}{\epsilon^2})$, for any $\epsilon > 0$; see Lipton et al. [9].

For the case of well-supported equilibria, Anbalagan et al. [2] recently showed the existence of win-lose games for which every ϵ-WSNE, with $\epsilon < \frac{2}{3}$, require supports of cardinality at least $\sqrt[3]{\log n}$. They also proved, in contrast to ϵ-NE, that with supports of cardinality at most two, it is not possible to guarantee the existence of an ϵ-WSNE, for any $\epsilon < 1$.

The outstanding open problem in the area is whether there is a constant k and an $\epsilon < 1$ such that, for any bimatrix game, there is a ϵ-WSNE with supports of cardinality at most k. In the paper we prove this is not the case. This result illustrates a fundamental structural distinction between ϵ-WSNE and ϵ-NE. This structural distinction also has practical implications with regards to behavioural models and popular equilibria search algorithms that focus upon small supports. The key to our result is the disproof of graph theoretic conjectures of Daskalakis, Mehta and Papadimitriou [7] and Myers [10] via an old result in additive number theory of Haight [8].

2 WSNE and a Graph Theoretic Conjecture

A bimatrix game is a 2-player game with $m \times n$ payoff matrices A and B. We consider normal form games with entries in the payoff matrices in $[0, 1]$. A pair of mixed strategies $\{p, q\}$ forms an *ϵ-well supported Nash equilibrium (ϵ-WSNE)* if every pure strategy in the support of p (resp. q) is an ϵ-approximate best response to q (resp p). Thus $\{p, q\}$ forms an ϵ-WSNE if and only if:

$$\forall i : p_i > 0 \;\Rightarrow\; \quad {e_i}^T Aq \geq {e_j}^T Aq - \epsilon \quad \forall j = 1, .., m$$

and

$$\forall i : q_i > 0 \;\Rightarrow\; \quad p^T Be_i \geq p^T Be_j - \epsilon \quad \forall j = 1, .., n$$

To analyse well-supported equilibria in a win-lose game (A, B), Daskalakis et al. [7] applied a *decorrelation transformation* to obtain a pair of decorrelated matrices (A^*, B^*). The exact details of this decorrelation transformation are not important here. What is pertinent, however, is that the $n \times n$ square $0-1$ matrix A^* induces a directed, possibly non-bipartite, graph $H = (V, E)$. There are n vertices in V, and there is an arc $ij \in E$ if an only if $A^*_{ij} = 1$. Moreover, Daskalakis et al. proved that the original win-lose game has a $(1 - \frac{1}{k})$-WSNE with supports of cardinality at most k if H contains either a directed cycle of length k or a set of k undominated[1] vertices. Furthermore, they conjectured that every directed graph contains either a small cycle or a small undominated set.

▶ **Conjecture 1** ([7]). *There are integers k and l such that every digraph either has a cycle of length at most k or an undominated set of l vertices.*

Indeed, they believed the conjecture was true for $k = l = 3$ and, consequently, that every bimatrix win-lose game has a $\frac{2}{3}$-WSNE with supports of cardinality at most three. Interestingly, motivated by the classical Caccetta-Haggkvist conjecture [3] in extremal graph theory, a similar conjecture was made previously by Myers [10].

▶ **Conjecture 2** ([10]). *There is an integer k such that every digraph either has a cycle of length at most k or an undominated set of two vertices.*

Myers conjectured that this was true even for $k = 3$, but Charbit [4] proved this special case to be false.

We say that D is a (k, l)*-digraph* if every directed cycle in D has length at least k, and every $S \subseteq V(D)$ of cardinality at most l is dominated. In Section 4, we will prove that there exists a finite (k, l)-digraph for every pair of positive integers k and l. This will imply that Conjectures 1 and 2 are false.

3 A Characterization for Games with Small Support ϵ-WSNE.

In this section, we show Daskalakis et al.'s sufficiency condition extends to a characterization of when a win-lose game has ϵ-WSNE with constant supports. To do this, rather than non-bipartite graphs, it is more natural for bimatrix games to work with bipartite graphs. In particular, any win-lose game (A, B) has a simple representation as a bipartite directed graph $G = (R \cup C, E)$. To see this, let G contain a vertex for each row and a vertex for each column. There exists an arc $(r_i, c_j) \in E$ if and only if $A_{ij} = 1$. So r_i is the best response for the row player against the strategy c_j of the column player. Similarly, there exists an arc $(c_j, r_i) \in E$ if and only if $B_{ij} = 1$. So, c_j is a best response for the column player against the strategy r_i of the row player.

We will now show that a win-lose game has a $(1 - \frac{1}{k})$-WSNE with supports of cardinality at most k *if and only if* the corresponding directed bipartite graph has either a small cycle or a small set of undominated vertices. Thus we obtain a characterization of win-lose games that have ϵ-WSNE with small cardinality supports.

It what follows, we will only consider undominated sets that are contained either in R or in C

▶ **Lemma 3.** *Let G be a win-lose game with minimum out-degree at least one. If G contains an undominated set of cardinality k then there is a $(1 - \frac{1}{k})$-WSNE with supports of cardinality at most k.*

[1] A set S is *undominated* if there is no vertex v that has an arc to every vertex in S.

Proof. Without loss of generality, let $U = \{r_1, ..., r_k\}$ be the undominated set. Let the row player play a uniform strategy p on these k rows. Since U is undominated, any column has expected payoff at most $1 - \frac{1}{k}$ against p. Therefore every column c_j is a $(1 - \frac{1}{k})$-approximate best response against p.

By assumption, each row vertex r_i has out-degree at least one. Let $c_{f(i)}$ be an out-neighbour of r_i (possibly $f(i) = f(j)$ for $j \neq i$). Now let the column player play a uniform strategy q on $\{c_{f(i)}\}_{i=1}^{k}$. Because q has support cardinality at most k, each pure strategy $r_i \in U$ has an expected payoff at least $\frac{1}{k}$ against q. Thus, these r_i's are all $(1-\frac{1}{k})$-approximate best responses for the row player against q. So $\{p, q\}$ is a $(1 - \frac{1}{k})$-WSNE with supports of cardinality at most k. ◀

▶ **Lemma 4.** *If G contains a cycle of length $2k$ then there is a $(1 - \frac{1}{k})$-WSNE with supports of cardinality k.*

Proof. Let W be a cycle of length $2k$ in G. Since G is bipartite, k of the vertices in the cycle are row vertices and k are column vertices. Let p be the uniform strategy on the rows in W and let q be the uniform strategy on the columns in W. We claim that p and q form a $(1 - \frac{1}{k})$-WSNE. To prove this, consider the subgraph F induced by the vertices of W. Every vertex in F has out-degree (and in-degree) at least one since $W \subseteq F$. So, every pure strategy in p, gives the row player an expected payoff of at least $\frac{1}{k}$ against q. Thus, every pure strategy in p is a $(1 - \frac{1}{k})$-best response for the row player against q. Similarly, every pure strategy in q is a $(1 - \frac{1}{k})$-best response for the column player against p. ◀

Lemma 3 and Lemma 4 immediately give the following corollary.

▶ **Corollary 5.** *Let G be a win-lose game with minimum out-degree at least one. If G contains a cycle of length $2k$ or an undominated set of cardinality k then then the win-lose game has $(1 - \frac{1}{k})$-WSNE with supports of cardinality at most k.*

Importantly, the converse also holds.

▶ **Lemma 6.** *Let G be a win-lose game with minimum out-degree at least one. If there is an ϵ-WSNE (for any $\epsilon < 1$) with supports of cardinality at most k then G either contains an undominated set of cardinality k or contains a cycle of length at most $2k$.*

Proof. Take a win-lose game $G = (R \cup C, E)$ and let p and q be an ϵ-WSNE. Suppose the supports of p and q, namely $P \subseteq R$ and $Q \subseteq C$, have cardinality at most k.

We may assume that every set of cardinality every set of k (on the same side of the bipartition) is dominated; otherwise we are already done. In particular, both P and Q are dominated. Consequently, the row player has a best response with expected payoff 1 against q. Similarly, the column player has a best response with expected payoff 1 against p. Thus, for the ϵ-WSNE $\{p, q\}$, we have:

$$\begin{aligned} \forall i: \quad p_i > 0 \quad &\Rightarrow \quad {e_i}^T R q \geq 1 - \epsilon > 0 \\ \forall j: \quad q_j > 0 \quad &\Rightarrow \quad p^T C e_j \geq 1 - \epsilon > 0 \end{aligned}$$

Here the strict inequalities follow because $\epsilon < 1$. Therefore, in the subgraph F induced by $P \cup Q$, every vertex has an out-degree at least one. But then F contains a cycle W. Since H contains at most $2k$ vertices, the cycle W has length at most $2k$. ◀

Corollary 5 and Lemma 6 then give the following characterization for win-lose games with ϵ-WSNE with small cardinality supports

▶ **Theorem 7.** *Let G be a win-lose game with minimum out-degree at least one. Take any constant k and any ϵ such that $1 - \frac{1}{k} \leq \epsilon < 1$. The game contains an ϵ-WSNE with supports of cardinality at most k if and only if G contains an undominated set of cardinality k or a cycle of length at most $2k$.*

4 Digraphs of Large Girth with every Small Subset Dominated

In this section, we will first prove that there exists a finite (k, l)-digraph for every pair of positive integers k and l and, hence, disprove Conjecture 1. We then adapt the resulting counterexamples in order to apply Theorem 7 and deduce that, for any constant k and any $\epsilon < 1$, there exist bimatrix win-lose games for which every ϵ-WSNE require supports of cardinality greater than k.

The main tool we require is a result of Haight [8] from additive number theory. We will require the following notation. Let Γ be an additive group. Then, for $X \subseteq \Gamma$, denote

$$\begin{aligned} X - X &= \{x_1 - x_2 \mid x_1, x_2 \in X\}, \text{ and} \\ (k)X &= \{x_1 + x_2 + \ldots + x_k \mid x_i \in X \text{ for } 1 \leq i \leq k\}. \end{aligned}$$

Finally, let $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$ denote the additive group of $\mathbb{Z}/q\mathbb{Z}$, the integers modulo q. Haight [8] proved:

▶ **Theorem 8** ([8]). *For all positive integers k and l, there exists a positive integer $\hat{q} = \hat{q}(k, l)$ and a set $X \subseteq \mathbb{Z}_{\hat{q}}$, such that $X - X = \mathbb{Z}_{\hat{q}}$, but $(k)X$ omits l consecutive residues.*

To construct the finite (k, l)-digraph we will use the following corollary.

▶ **Corollary 9.** *For every positive integer k, there exists a positive integer $q = q(k)$ and a set $Y \subseteq \mathbb{Z}_q$, such that $Y - Y = \mathbb{Z}_q$, but $0 \notin (k)Y$.*

Proof. Let $l = k$ and apply Theorem 8 with $q = q(k) = \hat{q}(k, k)$. Thus, we obtain a set $X \subseteq \mathbb{Z}_q$ with the properties that: (i) $X - X = \mathbb{Z}_q$, and (ii) $(k)X$ omits k consecutive residues. But these k consecutive residues must contain ky for some $y \in \mathbb{Z}_q$. Thus, there exists $y \in \mathbb{Z}_q$ such that $ky \notin (k)X$.

Now, define $Y := X - y$. Then $Y - Y = X - X = \mathbb{Z}_q$. Furthermore, $ky \notin (k)(Y + y)$. This implies that $0 \notin (k)Y$, as desired. ◀

We now construct a counter-example to Conjecture 2 of Myers. We will then show how the construction can be extend to disprove Conjecture 1.

▶ **Theorem 10.** *For any positive integer κ, there exists a $(\kappa, 2)$-digraph D.*

Proof. Set $k = (\kappa - 1)!$ and apply Corollary 9. Thus we find $Y \subseteq \mathbb{Z}_q$, with $q = q(k)$ where $Y - Y = \mathbb{Z}_q$, and $0 \notin (k)Y$. From Y, we create a directed graph D as follows. Let the vertex set be $V(D) = \mathbb{Z}_q$. Let the arc set be $E(D) = \{z_1 z_2 \mid z_1 - z_2 \in Y\}$.

Now take any pair $z_1, z_2 \in \mathbb{Z}_q$. Because $Y - Y = \mathbb{Z}_q$, there exist $y_1, y_2 \in Y$ such that $z_1 - z_2 = y_1 - y_2$. We now claim that the vertex pair $z_1, z_2 \in V(D)$ is dominated. To see this consider the vertex $x \in V(D)$ where $x = z_1 + y_2 = z_2 + y_1$. Then xz_1 is an arc in $E(D)$ because $x - z_1 = (z_1 + y_2) - z_1 = y_2 \in Y$. On the other hand $x = z_2 + y_1$ and so $x - z_2 = y_1 \in Y$. Consequently, xz_2 is also in $E(D)$. Hence, every subset of $V(D)$ of cardinality at most 2 is dominated.

It remains to prove that D contains no directed cycle of length less than κ. So, assume there is a cycle C with ordered vertices $z_1, z_2, \ldots, z_s$, where $s < \kappa$. As $z_i z_{i+1}$ is an arc we have that $z_i - z_{i+1} = y_i$ where $y_i \in Y$, for $1 \leq i \leq s$ (here we assume $z_{s+1} = z_1$). Summing around the cycle we have that $y_1 + y_2 + \cdots + y_s = 0$ modulo q. This implies that $0 \in (s)Y$ as $y_1, y_2, \ldots, y_s \in Y$. Consequently, $0 \in (ts)Y$ for any positive integer t. In particular, $0 \in (k)Y = ((\kappa - 1)!)Y$, as $s \leq \kappa - 1$. This contradicts the choice of Y and, so, D is a $(\kappa, 2)$-digraph, as desired. ◀

▶ **Theorem 11.** *For every pair of positive integers k and l, there exists a finite (k, l)-digraph.*

Proof. Without loss of generality, assume $l \geq 2$. By Theorem 10, there exists a $((k-1)(l-1)+1, 2)$-digraph D'. We claim that the $(l-1)$-st power of D' is a (k, l)-digraph. More precisely, let the digraph D be defined by $V(D) = V(D')$ and $vw \in E(D)$ if and only if there exists a directed walk from v to w in D' using at most $(l-1)$ edges.

Suppose D has a cycle of length at most $k-1$. This corresponds to a closed directed walk of length a most $(k-1)(l-1)$ in D'. This is a contradiction as D' has no cycles of length shorter than $(k-1)(l-1)+1$. Therefore, the shortest directed cycle in D has length at least k.

It remains to prove that every $S \subseteq V(D)$ with $|S| = l$ is dominated. So take $S = \{v_1, v_2, \ldots, v_l\}$. Recall that every pair of vertices in $V(D') = V(D)$ is dominated in D'. So there is a vertex z_1 dominating v_1 and v_2 in D'. Now let z_{i+1} be a vertex dominating z_i and v_{i+2} for $1 \leq i \leq l-2$. By construction, there is a directed walk in D' from z_{l-1} to v_i of length at most $l-1$, for every $1 \leq i \leq l$. Thus $z_{l-1}v_i \in E(D)$, and S is dominated in D, as desired. ◀

Observe that these constructions are non-bipartite. To exploit the characterization of Theorem 7 (and therefore conclude that there are games with no ϵ-WSNE with small supports), we desire bipartite constructions. These we can create using a simple mapping from non-bipartite to bipartite graphs. Given a non-bipartite graph $G = (V, E)$, we build a win-lose game, that is, a bipartite directed graph $G' = (R \cup C, E')$ as follows. We set $R = C = V$. Thus, for each $v_i \in V$ we have a row vertex $r_i \in R$ and a column vertex $c_i \in C$. Next, for each arc $a = (v_i, v_j)$ in G, we create two arcs (r_i, c_j) and (c_i, r_j) in G'. Finally, for each $v_i \in V$ we add an arc (r_i, c_i).

Now let's understand what this mapping does to cycles and undominated sets. First, suppose G contains a cycle of length k. Then observe that G' contains a cycle of length k if k is even and of length $k+1$ if k is odd. On the other hand, suppose the minimum length cycle in G' is $k+1$. This cycle will contain at most one pair of vertices type $\{r_i, c_i\}$, and if it contains such a pair then these vertices are consecutive on the cycle. (Otherwise we can find a shorter cycle in G'.) Thus, G contains a cycle of length k or $k+1$.

Second, consider an undominated set $S \subseteq V$ of size ℓ in G. Then $S \subseteq R$ is undominated in G'. (Note $S \subseteq C$ may be dominated because we added arcs of the form (r_i, c_i) to G'). On the other hand if S is undominated in G' (either in R or C) then S is also undominated in G.

Applying this mapping to a non-bipartite $(2k+1, k)$-digraph produces a bipartite digraph for which every set of k vertices (on the same side of the bipartition) is dominated but that has no cycle of length at most $2k$. Thus, by Theorem 7, the corresponding game has no ϵ-WSNE, for any $\epsilon < 1$, with supports of cardinality at most k.

▶ **Theorem 12.** *For any constant k and any $\epsilon < 1$, there exist bimatrix win-lose games for which every ϵ-WSNE requires supports of cardinality greater than k.*

References

1 I. Althöfer. On sparse approximations to randomized strategies and convex combinations. *Linear Algebra and its Applications*, 199:339–355, 1994.

2 Y. Anbalagan, S. Norin, R. Savani, and A. Vetta. Polylogarithmic supports are required for approximate well-supported Nash equilibria below 2/3. In *Proceedings of Ninth Conference on Web and Internet Economics (WINE)*, pages 15–23, 2013.

3 L. Caccetta and R. Häggkvist. On minimal digraphs with given girth. *Linear Algebra and its Applications*, 21:181–187, 1978.

4 P. Charbit. *Circuits in Graphs and Digraphs via Embeddings.* Doctoral Thesis, University of Lyon, 2005.

5 X. Chen, X. Deng, and S. Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM*, 56(3):1–57, 2009.

6 C. Daskalakis, P. Goldberg, and C. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

7 C. Daskalakis, A. Mehta, and C. Papadimitriou. A note on approximate Nash equilibria. *Theoretical Computer Science*, 410(17):1581–1588, 2009.

8 J. Haight. Difference covers which have small k-sums for any k. *Mathematika*, 20:109–118, 1973.

9 R. Lipton, E. Markakis, and Mehta A. Playing large games using simple startegies. In *Proceedings of Fourth Conference on Electronic Commerce (EC)*, pages 36–41, 2003.

10 J. Myers. *Extremal Theory of Graph Minors and Directed Graphs.* Doctoral Thesis, University of Cambridge, 2003.

11 J. Nash. Non-cooperative games. *Annals of Mathematics*, 54:289–295, 1951.

Minimizing Maximum Flow-time on Related Machines*

Nikhil Bansal and Bouke Cloostermans

Eindhoven University of Technology
P.O. Box 513, 5600 Eindhoven, The Netherlands
`{b.cloostermans,n.bansal}@tue.nl`

Abstract

We consider the online problem of minimizing the maximum flow-time on related machines. This is a natural generalization of the extensively studied makespan minimization problem to the setting where jobs arrive over time. Interestingly, natural algorithms such as Greedy or Slow-fit that work for the simpler identical machines case or for makespan minimization on related machines, are not $O(1)$-competitive. Our main result is a new $O(1)$-competitive algorithm for the problem. Previously, $O(1)$-competitive algorithms were known only with resource augmentation, and in fact no $O(1)$ approximation was known even in the offline case.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Related machines scheduling, Maximum flow-time minimization, On-line algorithm, Approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.85

1 Introduction

Scheduling a set of jobs on machines to optimize some quality of service measure is one of the most well studied problems in computer science. A very natural measure of service received by a job is the flow-time, defined as the amount of time the job spends in the system. In particular, if a job j arriving at time r_j completes its processing at time C_j, then its flow-time F_j is defined as $C_j - r_j$; i.e., its completion time minus its arrival time. Over the last few years, several variants of flow-time related problems have received a lot of attention: on single and multiple machines, in online or offline setting, for different objectives such as total flow-time, ℓ_p norms of flow-time, stretch etc., with or without resource augmentation, in weighted or unweighted setting and so on. We refer the reader to [15, 13, 12, 4] for a survey of some of these results.

In this paper we focus on the objective of minimizing the maximum flow-time. This is desirable when we want to guarantee that *each* job has a small delay. Maximum flow-time is also a very natural generalization of the minimum makespan or the load-balancing problem, that has been studied extensively (see e.g. [5, 9, 1] for a survey). In particular, if all jobs have identical release times, then the maximum flow-time value is precisely equal to the makespan. Minimizing the maximum flow-time is also related to deadline scheduling problems. In particular, the maximum flow-time is at most D if and only if each job j completes by $r_j + D$. Moreover, note that arbitrary deadlines d_j can be modeled by considering the weighted version of maximum flow-time[1].

* Supported by NWO grant 639.022.211 and ERC consolidator grant 617951.

[1] In deadline scheduling however, the deadlines are typically considered fixed and the focus is on maximizing the throughput.

Known results for maximum flow-time

For a single machine, it is easy to see that First In First Out (FIFO) is an optimal (online) algorithm for minimizing the maximum flow-time. For identical multiple machines, Bender et al. [8] showed that the Greedy algorithm, that schedules the incoming job on the least loaded machine is $3 - 2/m$ competitive, where m is the number of machines. They also showed that this bound is tight for the Greedy algorithm. If jobs can be preempted and migrated (moved from one machine to another), [2] gave a 2-competitive algorithm.

A systematic investigation of the problem for various machine models was initiated recently by Anand et al. [3]. Recall that in the related machines model each machine i has speed s_i, and processing job j on machine i takes $p_{ij} = p_j/s_i$ units of time. In the more general unrelated machines model, p_{ij} can be completely arbitrary.

Among other results, Anand et al. [3] gave a $(1+\epsilon)$-speed $O(1/\epsilon)$-competitive algorithm for the unrelated machine case, for any given $\epsilon > 0$. Here the online algorithm can process $1+\epsilon$ units of work per time step, but is compared to an offline optimum that does not have this extra *resource augmentation* [14, 15]. They also showed that in the unrelated setting any algorithm without resource augmentation must be $\Omega(m)$ competitive. For the weighted maximum flow-time objective, they gave a $(1+\epsilon)$-speed, $O(1/\epsilon^3)$-competitive algorithm for the related machines setting, and showed that no $O(1)$-speed, $O(1)$-competitive algorithm exists in the unrelated setting.

A natural question that remains is the complexity of the problem for related machines: Is there an $O(1)$-competitive algorithm for the related machines setting, without using resource augmentation?

This question is particularly intriguing as it is not at all clear what the right algorithm should be [2]. In fact, no $O(1)$-approximation is known even in the offline case. One issue is that the natural Slow-fit algorithm, that is $O(1)$-competitive for makespan minimization (even when the jobs are temporary and have unknown durations [6]), is not $O(1)$-competitive for maximum flow-time (Lemma 2 below). The algorithm of [3] for weighted maximum flow-time with resource augmentation is also a variant of Slow-fit. Recently, [7] obtained an $O(\log n)$ approximation for minimizing maximum flow-time on unrelated machines, where n is the number of machines. However their techniques do not seem to give anything better for the related machines setting either.

Our main result is the following.

▶ **Theorem 1.** *There is a* 13.5 *competitive algorithm,* Double-fit*, for minimizing maximum flow-time on related machines.*

This also gives the first $O(1)$ approximation for the offline problem. We also show that no such result is possible in the weighted case (without resource augmentation), and give an $\Omega(W)$ lower bound on the competitive ratio where W is the maximum to minimum weight ratio.

High-level approach

There are two competing trade-offs while scheduling on related machines. On one hand the algorithm should keep as many machines busy as possible, otherwise load might accumulate and delay future jobs. This accumulated load could be impossible to get rid of if there is no resource augmentation. On the other hand, the algorithm should keep fast machines empty for processing large jobs that might arrive later. In particular, fast machines are a scarce resource that should not be wasted on processing small jobs unnecessarily. It is instructive to consider the lower bounds in Section 2, where both Slow-fit and Greedy are shown to perform badly due to these opposite reasons.

To get around this, we design an algorithm that combines the good properties of both Slow-fit and Greedy. In particular, the algorithm uses a two phase strategy while assigning jobs to machines at each step. First, the jobs are spread out to ensure that machines are busy as much as possible. Once machines are *saturated*, the algorithm shifts into a *Slow-fit* mode, which ensures that small jobs do not unnecessarily go on fast machines.

The key difficulty in the analysis is to control how the two phases interact with each other. To do this, we maintain two invariants that capture the dynamics of the algorithm, and control how much the online algorithm's load on a subset of machines deviates from the offline algorithm's load on those machines. The main part of the argument is to show inductively that these invariants are maintained over time.

Notation and formal problem description

There are m machines indexed by non-decreasing order of speeds $s_1 \leq s_2 \leq \ldots \leq s_m$. The processing requirement of job j is p_j, and it requires time p_j/s_i on machine i. We will call p_j the work of j, and p_j/s_i its load on machine i. Jobs arrive online over time and p_j is known immediately upon its release time r_j. The goal is to find a schedule that minimizes the maximum flow-time, and we assume that a job cannot be migrated from one machine to another. We use Opt to denote some fixed optimum offline schedule, and also to denote the value of this solution.

2 Lower bounds on Slow-fit and Greedy

Slow-fit

Algorithm Slow-fit takes as input a threshold F_{opt} (the current guess on optimum), and schedules every incoming job on the slowest possible machine while keeping the load below F_{opt}. If the jobs cannot be feasibly scheduled on any machine, the algorithm fails and the threshold is doubled.

▶ **Lemma 2.** Slow-fit *has a competitive ratio of* $\Omega(m)$.

Proof. We describe an instance where the threshold F_{opt} keeps doubling until it reaches m even though Opt $= 2$.

There are m identical machines (but we arbitrarily order them from slow to fast). Next, we assume that $F_{\text{opt}} \geq 2$, which can be achieved by giving $2m$ unit-size jobs initially at $t = 0$.

At each time step $t \geq 2$, m unit-length jobs arrive. As Slow-fit will not use all m machines initially, there will be some time t_0 at which all the machines $1, \ldots, m-1$ have load F_{opt}. At time $t_0 + 1$, when these initial $m-1$ machines have $F_{\text{opt}} - 1$ pending jobs, we release $2m$ unit-size jobs. As there is at most $m - 1 + F_{\text{opt}}$ total capacity available, these jobs cannot be scheduled feasibly if $F_{\text{opt}} \leq m$. On the other hand, at each time step Opt distributes the incoming jobs over all machines and achieves value 2. ◀

Intuitively, Slow-fit unnecessarily builds up load on slow machines while keeping the fast machines empty, and cannot recover if there is small burst of jobs.

Greedy

When a job j arrives, Greedy schedules j on the machine that minimizes the flow-time of j (assuming FIFO order). Ties are broken arbitrarily. The following bound is well-known [11], but we sketch it here for completeness. The idea is that Greedy puts too many slow jobs on fast machines, which causes problems when large jobs arrive.

▶ **Lemma 3.** Greedy *has a competitive ratio of* $\Omega(\log m)$.

Proof. Consider an instance where we have k groups of machines where group G_i contains 2^{2k-2i} machines of speed 2^i. Note that the total processing power in group G_i is equal to $S_i = 2^{2k-i}$. The processing power of groups $i, \dots, k$ combined is thus equal to $P_i = \sum_{i'=i}^{k} 2^{2k-i'} \le 2S_i$.

We receive k sets of jobs, all at time 0, but in order. For all $i = 1, \dots, k$, set J_i contains 2^{2k-2i} jobs of size 2^i. Again, note that the total size of jobs in set J_i is equal to 2^{2k-i}. Greedy will spread jobs from set i over groups $i, \dots, k$. Group k (containing only a single machine of speed 2^k) will receive a $S_k/P_i \ge \frac{1}{2} S_k/S_i = 2^{-k+i-1}$ fraction of these jobs. This means group k receives $\sum_{i=1}^{k} 2^{2k-i} 2^{-k+i-1} = k2^{k-1}$ work. Since group k has a single machine of speed 2^k, finishing these jobs takes $\Omega(k)$ time.

However, optimum can schedule the i-th batch of jobs on group i machines, incurring a maximum load of 1 (i.e., it does Slow-fit with threshold 1). ◀

3 The Algorithm Double-fit

We describe our algorithm, denoted by Double-fit hereafter. Double-fit takes an input a parameter F_{opt}, which is supposed to be our estimate of Opt. By a slight variation on the doubling trick that loses an additional factor of 1.5 (see Section 3.4), we will assume henceforth that $F_{\text{opt}} \in [\text{Opt}, 1.5\text{Opt})$.

We divide time into intervals I_k of size $3F_{\text{opt}}$ as $I_k = [3(k-1)F_{\text{opt}}, 3kF_{\text{opt}})$. We refer to time $3kF_{\text{opt}}$ as the k-th epoch. For each $k = 1, 2, \dots,$ Double-fit batches the jobs that arrive during I_k and schedules them at epoch k using the algorithm in Figure 1. We use $[i : m]$ to denote the machines $i, \dots, m$. If the total remaining work on jobs on machine i is $w(i)$ at time t, we say that it has load $w(i)/s_i$.

```
1. Let $J$ denote the set of jobs arriving during $I_k$.
2. Partition jobs in $J$ into classes $J_1, \dots, J_m$, where each job $j$ is in class $J_i$ with the
   smallest index $i$ such that $p_j \le s_i \cdot F_{\text{opt}}$.
3. For $i = m, m-1, \dots, 1$
4.     Consider the jobs $j$ in $J_i$ in arbitrary order and assign them as follows:
5.         (Saturation Phase:) If some machine in $[i : m]$ is loaded below $3F_{\text{opt}}$
6.             schedule $j$ on the slowest such machine.
7.         (Slow-fit Phase:) Else schedule $j$ on the slowest machine in $[i : m]$
8.             such that its load stays below $6F_{\text{opt}}$.
9.         If no such machine exists return FAIL.
```

Figure 1 Algorithm Double-fit for the epoch k.

Description

First, Double-fit classifies the jobs arriving during I_k depending on the smallest machine on which they have size no larger than F_{opt}. Note that as $F_{\text{opt}} \geq \text{Opt}$, if job j is put in class J_i, then Opt cannot schedule job j onto a machine smaller than i either.

Double-fit considers jobs from classes J_m down to J_1 (this ordering will be used crucially). Each class is scheduled in two phases. In the saturation phase, when scheduling a job j, it checks if there is some machine in $[i:m]$ with load less than $3F_{\text{opt}}$. If so, j is scheduled on the slowest such machine. If no such machine exists, the algorithm enters the Slow-fit phase (for class J_i), and performs Slow-fit for class J_i on machines $[i:m]$ with threshold $6F_{\text{opt}}$.

3.1 Analysis

Our goal in this section is to show the following result.

▶ **Theorem 4.** *If $F_{opt} \geq Opt$, then the algorithm never fails.*

This directly implies Theorem 1 as follows. Each job spends at most $3F_{\text{opt}}$ time waiting to be assigned, and at most $6F_{\text{opt}}$ on its designated machine, thus the flow-time of any job is at most $9F_{\text{opt}}$. As $F_{\text{opt}} \leq 1.5\text{Opt}$ by the doubling trick, this implies a competitive ratio of 13.5

For the purpose of analysis, it will be convenient to consider a *restricted* Opt that also batches jobs and schedules the jobs arriving in I_k at epoch k. Note that such a restricted algorithm has objective at most $3F_{\text{opt}} + \text{Opt} \leq 4F_{\text{opt}}$ (as we can take the original schedule and delay every job by $3F_{\text{opt}}$). To prove theorem 4, we will in fact prove the following stronger result: Double-fit never fails for any instance where the restricted Opt has value at most $4F_{\text{opt}}$.

The Invariants

Fix an epoch k. Let $A_i(k)$ and $B_i(k)$ denote the total work on machines $[i:m]$ in Double-fit's schedule just before and just after all the jobs from interval I_k are scheduled respectively. Similarly, let $A_i^{\text{opt}}(k)$ and $B_i^{\text{opt}}(k)$ be the total work remaining on machines $[i:m]$ in Opt's schedule.

We will show that the following two invariants hold at each epoch k.

$$A_i(k) \leq A_i^{\text{opt}}(k) + F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}. \tag{1}$$

$$B_i(k) \leq \max\left\{3F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}, B_i^{\text{opt}}(k)\right\} + F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}. \tag{2}$$

Roughly speaking, invariants (1) and (2) show that the load on any suffix of Double-fit's machines stays close to Opt's load on those machines, both before and after the jobs are scheduled in epoch k. We will prove that (1) and (2) hold by a careful induction over i and k.

Before we prove these invariants, let us first see why they imply Theorem 4.

Proof of Theorem 4. Consider a fixed epoch k. As the (restricted) Opt has maximum flow-time at most $4F_{\text{opt}}$, for each i it must hold that $B_i^{\text{opt}}(k) \leq 4F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}$. Thus by (2) it follows that $B_i^{\text{opt}}(k) \leq 5F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}$ for each i. Choosing $i = m$, this implies that

Double-fit never loads machine m above $5F_{\text{opt}}$ and thus never fails (as machine m always has room for an additional job). ◀

Proving the Invariants

The strategy for proving that (1) and (2) hold at all epochs k will be to show the following two lemmas.

▶ **Lemma 5.** *If at epoch k,* (1) *holds for all machines, then* (2) *also holds for all machines.*

The next step will be to relate the conditions at epochs k and $k+1$.

▶ **Lemma 6.** *If at any epoch k,* (2) *holds for all machines, then* (1) *also holds for all machines at epoch $k+1$.*

As (1) trivially holds for $k = 0$ (as $A_i(0) = A_i^{\text{opt}}(0) = 0$ for all i), applying Lemma 5 and Lemma 6 alternately implies that (1) and (2) hold for all k.

3.2 Proof of Lemma 5

We first show that Double-fit is conservative in scheduling small jobs on fast machines.

▶ **Lemma 7.** *Let $i_1 < i_2$. If some job j of class i_1 is scheduled by* Double-fit *onto machine i_2 during the saturation phase (i.e. using threshold $3F_{opt}$), then* all *jobs of class i for $i_1 < i \leq i_2$ are also scheduled during the saturation phase.*

Proof. Consider the state of Double-fit's machines just before j was scheduled. As j is scheduled on machine i_2 during the saturation phase, the load on i_2 must be below $3F_{\text{opt}}$ at that point. As jobs of class i for $i_1 < i \leq i_2$ were considered before class i_1-jobs, the load on i_2 was also below $3F_{\text{opt}}$ after scheduling class i jobs, and thus Double-fit must have never switched to the Slow-fit phase while considering class i. ◀

Next we define the notion of *separated* machines, which will play a crucial role in the analysis.

▶ **Definition 8.** Machines i_1 and i_2 ($i_1 < i_2$) are *separated* at epoch k if Double-fit scheduled no jobs from classes $[1 : i_1]$ onto machines $[i_2 : m]$ at epoch k.

The following lemma shows that if two consecutive machines are separated, it is easy to relate epochs k and $k+1$.

▶ **Lemma 9.** *If machines $i-1$ and i are separated at epoch k, then* (1) *implies* (2) *for machine i. Moreover this trivially holds for machine $i = 1$.*

Proof. As machines $i-1$ and i are separated at epoch k, no jobs from class $[1 : i-1]$ were scheduled onto machines $[i : m]$ at epoch k. Thus

$$B_i(k) = A_i(k) + \sum_{i'=i}^{m} |J_{i'}|, \tag{3}$$

where $|J_i|$ represents the total work of all jobs in J_i.

As jobs from J_i cannot be scheduled onto machines $[1 : i-1]$ in an optimal schedule, we also obtain

$$B_i^{\text{opt}}(k) \geq A_i^{\text{opt}}(k) + \sum_{i'=i}^{m} |J_i|. \tag{4}$$

This implies that

$$B_i(k) = A_i(k) + \sum_{i'=i}^{m} |J_{i'}| \leq A_i(k) + B_i^{\text{opt}}(k) - A_i^{\text{opt}}(k) \leq B_i^{\text{opt}}(k) + F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}, \tag{5}$$

where the last step follows by our assumption that (1) holds for (i, k).

Finally for $i = 1$, we observe that both (3) and (4) hold with equality, and hence the result holds trivially. ◀

We now have all the tools we need to prove Lemma 5.

Proof of Lemma 5. We use induction over i in the order of larger to smaller i. In particular, to prove that (2) holds for some pair (i, k), we assume that (1) holds for all (i', k) and that (2) holds for all (i', k) with $i' > i$. As the base case note that this is vacuously true for $i = m + 1$ (as all relevant quantities are 0).

We consider three cases depending on how DOUBLE-FIT assigns jobs from classes $[1 : i-1]$ to machines $[i : m]$.

1. *No jobs from class* $[1 : i-1]$ *were scheduled onto machines* $[i : m]$*:* In this case, machines $i - 1$ and i are separated and (2) follows from Lemma 9.

2. *Jobs from classes* $[1 : i-1]$ *are only scheduled onto machines* $[i : m]$ *during the saturation phase:* Let $i_{\max} \geq i$ denote the smallest index such that machines $i - 1$ and $i_{\max} + 1$ are separated (if no such machine exists, set $i_{\max} = m$). By the inductive hypothesis, we can assume that (2) holds for $i_{\max} + 1$. In the case where $i_{\max} = m$, this holds vacuously. As jobs from classes $[1 : i - 1]$ are assigned to $[i : m]$ (and hence to $i_{\max}$) during the saturation phase, Lemma 7 implies that all jobs in classes $[i : i_{\max}]$ were also scheduled during the saturation phase, which implies that all machines $[i : i_{\max}]$ are loaded below $4F_{\text{opt}}$. This gives us the following:
$$\begin{aligned} B_i(k) &\leq 4F_{\text{opt}} \sum_{i'=i}^{i_{\max}} s_{i'} + B_{i_{\max}+1}(k) \\ &\leq 4F_{\text{opt}} \sum_{i'=i}^{i_{\max}} s_{i'} + \max\left\{3F_{\text{opt}} \sum_{i'=i_{\max}+1}^{m} s_{i'}, B_{i_{\max}+1}^{\text{opt}}(k)\right\} + F_{\text{opt}} \sum_{i'=i_{max}+1}^{m} s_{i'} \\ &= 3F_{\text{opt}} \sum_{i'=i}^{i_{\max}} s_{i'} + \max\left\{3F_{\text{opt}} \sum_{i'=i_{\max}+1}^{m} s_{i'}, B_{i_{\max}+1}^{\text{opt}}(k)\right\} + F_{\text{opt}} \sum_{i'=i}^{m} s_{i'} \\ &\leq \max\left\{3F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}, B_i^{\text{opt}}(k)\right\} + F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}, \end{aligned}$$
where the second inequality follows from the inductive hypothesis for machine $i_{\max} + 1$.

3. *Some job j from class* $[1 : i-1]$ *was scheduled onto machines* $[i : m]$ *during Slow-fit phase (using threshold $6F_{opt}$):* We assume that $i > 1$, otherwise the result follows from case 1. Let $i_{min} < i$ denote the largest index such that machines $[i_{min} : i - 1]$ have load more than $5F_{\text{opt}}$ and machine $i_{\min} - 1$ has load at most $5F_{\text{opt}}$. If no such machine exists, set $i_{\min} = 1$. $i_{\min}$ is well-defined as $i > 1$ and machine $i - 1$ must have load more than $5F_{\text{opt}}$ as job j from class $[1 : i - 1]$ was assigned to a machine in $[i : m]$ during the Slow-fit phase.

▶ **Claim 1.** *Machines $i_{\min} - 1$ and $i_{\min}$ are separated or $i_{\min} = 1$.*

Proof. This is trivially true if $i_{\min} = 1$.
If $i_{\min} > 1$, suppose that some job j' from class $[1 : i_{\min} - 1]$ was scheduled onto machines $[i_{\min} : m]$. Now j' cannot be scheduled during the Slow-fit phase as this would imply that the load on $i_{\min} - 1$ was more than $5F_{\text{opt}}$, which contradicts the choice of $i_{\min}$.
So all jobs in $[1 : i_{\min} - 1]$ that were assigned to $[i_{\min} : m]$ must have been assigned during the saturation phase. Let $i' \geq i_{\min}$ denote the largest index where such a job is assigned. By Lemma 7, it must be that all machines $[i_{\min} : i']$ were assigned load during the saturation phase and must have load at most $4F_{\text{opt}}$. This contradicts that $i_{\min}$ has load more than $5F_{\text{opt}}$. ◀

By Lemma 9 applied to $i_{\min}$, we get that (2) holds for machine $i_{\min}$ and thus

$$B_{i_{\min}}(k) \leq \max\left\{3F_{\text{opt}} \sum_{i'=i_{\min}}^{m} s_{i'}, B_{i_{\min}}^{\text{opt}}(k)\right\} + F_{\text{opt}} \sum_{i'=i_{\min}}^{m} s_{i'}. \tag{6}$$

Furthermore, by choice of $i_{\min}$ all the machines in $[i_{\min} : i - 1]$ are loaded above $5F_{\text{opt}}$. This implies that

$$B_i(k) \leq B_{i_{\min}}(k) - 5F_{\text{opt}} \sum_{i'=i_{\min}}^{i-1} s_{i'}. \tag{7}$$

As every machine is loaded below $4F_{\text{opt}}$ in an optimal schedule, we also have

$$B_{i_{\min}}^{\text{opt}}(k) \leq B_i^{\text{opt}}(k) + 4F_{\text{opt}} \sum_{i'=i_{\min}}^{i-1} s_{i'}. \tag{8}$$

Adding (6) and (7) we obtain that

$$\begin{aligned}
B_i(k) &\leq \max\left\{3F_{\text{opt}} \sum_{i'=i_{\min}}^{m} s_{i'}, B_{i_{\min}}^{\text{opt}}(k)\right\} + F_{\text{opt}} \sum_{i'=i_{\min}}^{m} s_{i'} - 5F_{\text{opt}} \sum_{i'=i_{\min}}^{i-1} s_{i'} \\
&\leq \max\left\{4F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}, B_{i_{\min}}^{\text{opt}}(k) + F_{\text{opt}} \sum_{i'=i}^{m} s_{i'} - 4F_{\text{opt}} \sum_{i'=i_{\min}}^{i-1} s_{i'}\right\} \\
&\leq \max\left\{4F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}, B_i^{\text{opt}}(k) + F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}\right\} \qquad \text{By (8)} \\
&= \max\left\{3F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}, B_i^{\text{opt}}(k)\right\} + F_{\text{opt}} \sum_{i'=i}^{m} s_{i'},
\end{aligned}$$

which implies that (2) holds for i.

◀

3.3 Proof of Lemma 6

We now prove Lemma 6, which is relatively easier.

Proof of Lemma 6. We will apply induction over i (in decreasing order of machines). Consider epoch k. We assume that (2) holds for all i' at epoch k, and that (1) holds for all $i' > i$ at epoch $k + 1$. For the base case of $i = m + 1$ the lemma follows trivially since all the relevant quantities are 0.

Consider some machine i. We consider two cases depending on the load of machine i after the jobs were scheduled at epoch k.

1. *Machine i has load at most $4F_{opt}$ after epoch k, i.e., $B_i(k) - B_{i+1}(k) \leq 4F_{opt} \cdot s_i$:* At epoch $k+1$ before the jobs arriving during interval I_{k+1} are scheduled, the load of machine i will be at most F_{opt}. Thus we have that
$$\begin{aligned} A_i(k+1) &\leq A_{i+1}(k+1) + F_{\text{opt}} \cdot s_i \\ &\leq A^{\text{opt}}_{i+1}(k+1) + F_{\text{opt}} \sum_{i'=i+1}^{m} s_{i'} + F_{\text{opt}} \cdot s_i \\ &\leq A^{\text{opt}}_{i}(k+1) + F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}. \end{aligned}$$
Here the second inequality follows by the inductive hypothesis for machine $i+1$, and the third inequality follows as $A^{\text{opt}}_i(k+1)$ is non-decreasing as i decreases.
2. *Machine i is loaded above $4F_{opt}$ after epoch k, i.e., $B_i(k) - B_{i+1}(k) > 4F_{opt} \cdot s_i$:* In this case, some job j must have been scheduled onto machine i during the Slow-fit phase. This only happens if j could not be scheduled during the saturation phase. In particular, this implies that all the machines $[i : m]$ (which is surely a subset of machines where j could have been scheduled) were loaded above $3F_{\text{opt}}$. So the total work on all machines $[i : m]$ decreases by exactly $3F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}$ during interval I_{k+1}.
Thus we have that
$$A_i(k+1) = B_i(k) - 3F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}. \tag{9}$$
Similarly, as Opt can complete at most $3F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}$ on machines $[i : m]$ during this interval, we have
$$A^{\text{opt}}_i(k+1) \geq B^{\text{opt}}_i(k) - 3F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}. \tag{10}$$
As (2) holds for each i at epoch k, we obtain that
$$\begin{aligned} A_i(k+1) &\leq B_i(k) - 3F_{\text{opt}} \sum_{i'=i}^{m} s_{i'} \\ &\leq B^{\text{opt}}_i(k) + F_{\text{opt}} \sum_{i'=i}^{m} s_{i'} - 3F_{\text{opt}} \sum_{i'=i}^{m} s_{i'} \qquad \text{By (2)} \\ &\leq A^{\text{opt}}_i(k+1) + F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}, \end{aligned}$$
and hence (1) holds for i at epoch $k+1$, which completes the proof. ◀

3.4 Removing the assumption of knowledge of Opt

We describe a variant of the standard doubling trick where we increase the online estimate of Opt by only 1.5 times at each step.

Consider some epoch k where the algorithm first fails with the current guess of F_{opt}. It must be that (2) does not hold. In particular, (1) holds at epoch k as (2) holds at $k-1$. Now, Lemma 5 implies that $F_{\text{opt}} < \text{Opt}$. We then abort epoch k, and do not schedule any jobs. Instead, we set $F'_{\text{opt}} = 1.5F_{\text{opt}}$ and redefine the new epoch to be the time $(k-1)F_{\text{opt}} + 3F'_{\text{opt}}$. Note that between these epochs $4.5F_{\text{opt}}$ time passes, so at the next epoch the load on all machines in the schedule of DOUBLE-FIT will be at most $6F_{\text{opt}} - 3F'_{\text{opt}} = 1.5F_{\text{opt}} = F'_{\text{opt}}$.

This implies that for all i

$$A_i(k) \leq F_{\text{opt}} \sum_{i'=i}^{m} s_{i'} \leq A_i^{\text{opt}}(k) + F_{\text{opt}} \sum_{i'=i}^{m} s_{i'}.$$

The crucial point is that (1) holds for all machines i at this new epoch irrespective of the workload of the new restricted Opt (with parameter F'_{opt}). Thus, (2) holds if $F_{\text{opt}} \geq \text{Opt}$ and DOUBLE-FIT proceeds as normal.

4 Other Lower bounds

We also show simple (but strong) lower bounds for weighted maximum flow-time and maximum stretch.

▶ **Lemma 10.** *Any algorithm for minimizing maximum weighted flow-time on identical machines must have a competitive ratio of $\Omega(W)$, where W is the ratio between the largest and smallest weight.*

Proof. Consider the following instance on 2 machines. At time $t = 0$ we receive 2 jobs of size w with weight 1. Now, any algorithm has three options: (i) it schedules both jobs on the same machine, (ii) it schedules both jobs on different machines, or (iii) it waits before assigning jobs. In all three cases, we show that it will end up trailing by at least w work behind an optimal schedule.

In option (i), w work remains at time $t = w$, while optimum is empty. In option (ii) we instantly receive another $2w$-sized job with weight 1, so that one of our machines has load $3w$. At time $t = 2w$ we have w load remaining while an optimal schedule is empty. In option (iii) we receive no jobs until algorithm decide to choose (i) or (ii). If we have not chosen by time $t = w$ we are trailing $2w$ work behind an optimal schedule.

Once we trail w work behind optimum, at every unit time step we receive 2 unit-size jobs of weight w. If the trailing jobs are ever to be finished, at least $w/2$ delay is incurred on the weight w jobs, implying an objective value of $\Omega(w^2)$. Opt on other hand has value $O(w)$. ◀

A lower bound of $\Omega(W^{0.4})$ for maximum weighted flow-time follows from [10], using the analogy between delay factor and weighted maximum flow-time described in [3]. By replacing the unit size jobs by unit weight jobs in the above lower bound instance, this also directly implies an $\Omega(S)$ lower bound on the competitive ratio for maximum stretch [3] where S is the ratio between the size of the largest and the smallest job.

5 Concluding Remarks

Note that our algorithm DOUBLE-FIT is not immediate dispatch, i.e., it does not dispatch a job to a machine immediately upon arrival. We are unable to extend the ideas here to obtain an $O(1)$-competitive immediate dispatch algorithm, and it is not clear to us whether such an algorithm exists. Given that in the unrelated setting, there can be no $O(1)$-speed, $O(1)$-competitive immediate dispatch algorithm [3] (while there is a $(1+\epsilon)$-speed, $O(1/\epsilon)$-competitive algorithm), it would be quite interesting to resolve this question.

References

1 Susanne Albers. *Introduction to scheduling*, chapter Online scheduling, pages 51–73. Chapman and Hall/CRC, 2010.
2 Christoph Ambühl and Monaldo Mastrolilli. On-line scheduling to minimize max flow time: an optimal preemptive algorithm. *Oper. Res. Lett.*, 33(6):597–602, 2005.
3 S. Anand, Karl Bringmann, Tobias Friedrich, Naveen Garg, and Amit Kumar. Minimizing maximum (weighted) flow-time on related and unrelated machines. In *ICALP (1)*, pages 13–24, 2013.
4 S. Anand, Naveen Garg, and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1228–1241, 2012.
5 Yossi Azar. On-line load balancing. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms*, volume 1442 of *Lecture Notes in Computer Science*, pages 178–195. Springer, 1998.
6 Yossi Azar, Bala Kalyanasundaram, Serge A. Plotkin, Kirk Pruhs, and Orli Waarts. On-line load balancing of temporary tasks. *J. Algorithms*, 22(1):93–110, 1997.
7 Nikhil Bansal and Janardhan Kulkarni. Minimizing flow-time on unrelated machines. In *Symposium on Theory of Computing, STOC*, 2015, to appear.
8 Michael A. Bender, Soumen Chakrabarti, and S. Muthukrishnan. Flow and stretch metrics for scheduling continuous job streams. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 270–279, 1998.
9 Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3):93–263, 2009.
10 Chandra Chekuri and Benjamin Moseley. Online scheduling to minimize the maximum delay factor. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1116–1125. Society for Industrial and Applied Mathematics, 2009.
11 Yookun Cho and Sartaj Sahni. Bounds for list schedules on uniform processors. *SIAM Journal on Computing*, 9(1):91–103, 1980.
12 Naveen Garg. Minimizing average flow-time. In *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, pages 187–198, 2009.
13 Sungjin Im, Benjamin Moseley, and Kirk Pruhs. A tutorial on amortized local competitiveness in online scheduling. *SIGACT News*, 42(2):83–97, 2011.
14 Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.
15 Kirk Pruhs, Jiri Sgall, and Eric Torng. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter Online Scheduling. CRC Press, 2004.

A 2-Competitive Algorithm For Online Convex Optimization With Switching Costs

Nikhil Bansal[*1], **Anupam Gupta**[2], **Ravishankar Krishnaswamy**[†3], **Kirk Pruhs**[‡4], **Kevin Schewior**[§5], and **Cliff Stein**[¶6]

1 **Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands**
2 **Computer Science Department, Carnegie Mellon University, USA**
3 **Microsoft Research India, India**
4 **Computer Science Department, University of Pittsburgh, USA**
5 **Department of Mathematics, TU Berlin, Germany**
6 **Departmemt of IEOR, Columbia University, USA**

Abstract

We consider a natural online optimization problem set on the real line. The state of the online algorithm at each integer time t is a location x_t on the real line. At each integer time t, a convex function $f_t(x)$ arrives online. In response, the online algorithm picks a new location x_t. The cost paid by the online algorithm for this response is the distance moved, namely $|x_t - x_{t-1}|$, plus the value of the function at the final destination, namely $f_t(x_t)$. The objective is then to minimize the aggregate cost over all time, namely $\sum_t (|x_t - x_{t-1}| + f_t(x_t))$. The motivating application is rightsizing power-proportional data centers. We give a 2-competitive algorithm for this problem. We also give a 3-competitive memoryless algorithm, and show that this is the best competitive ratio achievable by a deterministic memoryless algorithm. Finally we show that this online problem is strictly harder than the standard ski rental problem.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems: Sequencing and Scheduling

Keywords and phrases Stochastic, Scheduling

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.96

1 Introduction

We consider a natural online optimization problem on the real line. The state of the online algorithm after each integer time $t \in \mathbb{Z}_{\geq 0}$ is a location on the line. At each integer time t, a convex function $f_t(x)$ arrives online. In response, the online algorithm from its previous location $x_{t-1} \in \mathbb{R}$ to a new location $x_t \in \mathbb{R}$. The cost paid by the online algorithm for this response is the distance moved, namely $|x_t - x_{t-1}|$, plus the value of the function at the final destination, namely $f_t(x_t)$. The objective is to minimize the aggregate

* Supported by NWO grant 639.022.211 and an ERC consolidator grant 617951.
† Part of this work was done when the author was at Columbia University and supported by NSF grant CCF-1349602.
‡ Supported in part by NSF grants CCF-1115575, CNS-1253218, CCF-1421508, and an IBM Faculty Award.
§ Supported by the Deutsche Forschungsgemeinschaft within the research training group "Methods for Discrete Structures" (GRK 1408).
¶ Supported in part by NSF grants CCF-1349602 and CCF-1421161.

cost $\sum_t (|x_t - x_{t-1}| + f_t(x_t))$ over all time. We refer to this problem as *Online Convex Optimization with Switching Costs* (OCO). This problem is also referred to as *Smoothed Online Convex Optimization* in the literature.

1.1 Motivation and Related Results

The OCO problem has been extensively studied recently, partly due to its application within the context of rightsizing power-proportional data centers, see for example [1, 15, 12, 14, 10, 11, 13]. In these applications, the data center consists of a homogeneous collection of servers/processors that are speed scalable and that may be powered down. The load on the data center varies with time, and at each time the data center operator has to determine the number of servers that will be operational. The standard assumption is that there is some fixed cost for powering a server on, or powering the server off. Most naturally this cost incorporates the energy used for powering up or down, but this cost may incorporate ancillary terms such as the cost of the additional wear and tear on the servers. As for the processor speeds, it natural to assume that the speed of a processor is scaled linearly with its load (as would be required to maintain a constant quality of service), and that there is a convex function $P(s)$ that specifies the power consumed as a function of speed. The most commonly used model for $P(s)$ is $s^\alpha + \beta$ for constants $\alpha > 1$ and β. Here the first term s^α is the dynamic power and the second term β is the static or leakage power. At each time, the state of the online algorithm represents the number of servers that are powered on. In a data center, there are typically sufficiently many servers so that this discrete variable can be reasonably be modeled a continuous one. Then, in response to a load L_t at time t, the data center operator decides on a number of servers x_t to use to handle this load. The algorithm pays a cost of $|x_{t-1} - x_t|$ for either powering-up or powering-down servers, and a cost of $x_t((L_t/x_t)^\alpha + \beta)$ for handling the load, which is the most energy efficient way to service the load L_t using x_t processors. Note that the function $x_t((L_t/x_t)^\alpha + \beta)$ is convex in x_t, and hence this application can be directly cast in our general online model where $f_t(x) = x((L_t/x)^\alpha + \beta)$.

Lin et al. [12] observed that the offline problem can be modeled as a convex program, and thus is solvable in polynomial time, and that if the line/states are discretized, then the offline problem can be solved by a straight-forward dynamic program. They also give a 3-competitive deterministic algorithm. The algorithm computes (say via solving a convex program) the optimal solution to date if moving to the left on the line was free, and the optimal solution to date if moving to the right on the line was free, and then moves the least distance possible so that it ends up between the final states of these two solutions. Note that this algorithm solves a (progressively larger) convex program at each time. Andrew et al. [1] show that there is an algorithm with sublinear regret, but that $O(1)$-competitiveness and sublinear regret cannot be simultaneously achieved. They also claim that a particular randomized online algorithm, RBG, is 2-competitive, but this claim has been withdrawn [16].

The OCO problem is also related to several classic online optimization problems. It is a special case of the *metrical task system* problem in which the metric is restricted to be a line and the costs are restricted to be convex functions on the real line. The optimal deterministic competitive ratio for a general metrical task system is $2n-1$, where n is the number of points in the metric [5], and the optimal randomized competitive ratio is $\Omega(\log n/\log\log n)$ [4, 3], and $O(\log^2 n \log\log n)$ [8]. The OCO problem is closely related to the *allocation problem* defined in [2], that arises when developing a randomized algorithm for the classic k-server problem using tree embeddings of the underlying metric space [7, 2]. In fact, the algorithm RBG in [1] is derived from a similar algorithm in [7] for this k-server "subproblem". The classic *ski rental* problem, where randomized algorithms are allowed, is a special case of the OCO problem. The optimal competitive ratio for randomized algorithms for the ski rental problem is $e/(e-1)$ [9] and this translates to a matching lower bound for any online algorithm for the OCO problem. The ski rental problem where only deterministic algorithms are allowed is a special case of the deterministic version of the OCO problem, and the optimal deterministic competitive ratio for the ski rental problem is exactly 2.

1.2 Our Results

2-Competitive Algorithm. Our main result, presented in Section 3, is a 2-competitive algorithm (thus we improve the upper bound on the optimal competitive ratio from 3 to 2). It will be convenient to first present a "fractional algorithm" $\mathcal{A}$ that maintains a probability distribution p over locations. In Section 2 we show how to convert a fractional algorithm into a randomized algorithm, and how to convert any c-competitive randomized algorithm into a c-competitive deterministic algorithm. Although the observation that randomization is not helpful is straight-forward, as best as we can tell, it has not previously appeared in the literature on this problem. The deterministic algorithm that results from these two conversations maintains the invariant that the current location is the expected location given the probability distribution over the states that $\mathcal{A}$ maintains.

We now describe the fractional algorithm $\mathcal{A}$. In response to the arrival of a new function $f_t(x)$, the algorithm $\mathcal{A}$ computes a point x_r to the right of the minimizer x_m of $f_t(x)$ such that the derivative of $f_t(x_r)$ is equal to twice the total probability mass to the right of x_r. Similarly the algorithm $\mathcal{A}$ computes a point x_l to the left of the minimizer x_m such that the (negative) derivative of $f_t(x_l)$ is equal to twice the total probability mass to the left of x_l. Then, the probability mass at each state $x \in [x_l, x_r]$ is increased by half the second derivative of $f_t(x)$ at that point, while the probability mass for each state $x \notin [x_l, x_r]$ is set to 0. A simple calculation shows that this operation, along with our choices of x_l and x_r, preserves the property that p is a valid probability distribution. One can convert such a probability distribution into a deterministic algorithm by initially picking a random number $\gamma \in [0, 1]$, and at any time t, moving to the state x_t such that the probability mass to the left of x_t in the current distribution is exactly γ.

The analysis of $\mathcal{A}$ uses an amortized local competitiveness argument, using the potential function

$$\Phi(p, x^*) = 2\int_{y=-\infty}^{\infty} |x^* - y| p(y)\, dy - \int_{x=-\infty}^{\infty}\int_{y=-\infty}^{x} p(x)p(y)(x-y)\, dx\, dy.$$

where x^* is the position of the adversary. The first term is depends on the expected distance between $\mathcal{A}$'s state and the adversary's state, and the second term is proportional to the expected distance of two randomly drawn states from $\mathcal{A}$'s probability distribution on states. This potential function can be viewed as a fractional generalization of the potential function used to show that that the Double Cover algorithm is k-competitive for the k-server problem on a line metric [6].

3-Competitive Memoryless Algorithm. Our algorithm $\mathcal{A}$ requires time and memory roughly proportional to the number of states and/or the number of time steps to date. Similarly, the 3-competitive algorithm from [12], requires solving a convex program (with the entire history) at each time step. However, as pointed out in [12], this may well be undesirable in settings where the data center operator wants to adapt quickly to changes in load. Previously it was not known if $O(1)$-competitiveness can be achieved by a "memoryless" algorithm. Intuitively in a memoryless algorithm the next state x_t only depends upon the previous state x_{t-1} and the current function $f_t(x)$. In Section 4 we show that $O(1)$-competitiveness is achievable by a memoryless algorithm – we give a simple memoryless algorithm $\mathcal{M}$, and show that it is 3-competitive. Given function $f_t(x)$ at time t, this algorithm $\mathcal{M}$ moves in the direction of the minimizer of $f_t(x)$ until either it reaches the minimizer, or it reaches a state where its movement cost equals twice the function cost of this state. The analysis is via an amortized local-competitiveness argument using the distance between the online algorithm's state and the adversary's state (times three) as the potential function.

Lower Bounds. In Section 5 we show a matching lower bound of 3 on the competitiveness of any deterministic memoryless online algorithm. We also give a general lower bound of 1.86 on the competitiveness of any algorithm, which shows that in some sense this problem is strictly harder than ski rental, which has an $e/(e-1)$-competitive randomized algorithm.

2 Reduction From Randomized to Deterministic

In this section, we explain how to convert a probability distribution over locations into randomized algorithm, and present a simple derandomization of any randomized algorithm.

Converting a Fractional Algorithm into a Randomized Algorithm: The randomized algorithm initially picks a number $\gamma \in [0,1]$ uniformly at random. Then the randomized algorithm maintains the invariant that at each time t the location x_t has the property that the probability mass to the left of x_t in the distribution for the fractional algorithm is exactly γ.

▶ **Theorem 2.1.** *For the OCO problem, if there is a c-competitive randomized algorithm $\mathcal{R}$ then there is a c-competitive deterministic algorithm $\mathcal{D}$.*

Proof. Let $\mathcal{R}$ denote the randomized algorithm, and let x_t denote the random variable for its position at time t. Then, our deterministic algorithm $\mathcal{D}$ sets its location to be the expected location of $\mathcal{R}$, i.e., its location at time t is $\mu_t := \mathbb{E}[x_t]$. It is then a simple application of Jensen's inequality to observe that $\mathcal{D}$'s cost is at most $\mathcal{R}$'s expected cost for each time t. Indeed, first observe that $\mathcal{D}$'s cost at time t is $|\mu_t - \mu_{t-1}| + f_t(\mu_t)$, and $\mathcal{R}$'s expected cost is $\mathbb{E}[|x_t - x_{t-1}|] + \mathbb{E}[f_t(x_t)]$. Now, notice that both the absolute value function and the function $f_t(\cdot)$ are convex functions, and therefore $\mathcal{R}$'s cost is at least $|\mathbb{E}[x_t - x_{t-1}]| + f_t(\mathbb{E}[x_t])$, which is precisely the cost incurred by the algorithm $\mathcal{D}$. Summing over all t completes the proof. ◀

3 The Algorithm A and its Analysis

In this section, we describe the online algorithm $\mathcal{A}$ and prove that it is 2-competitive. For simplicity, we will assume that the functions $f_t(x)$ are all continuous and smooth. That is, we assume that the first derivative $f'_t(x)$ and second derivative $f''_t(x)$ of $f_t(x)$ are well defined functions. We also assume that $f_t(x)$ has a unique bounded minimizer x_m, and $f'_t(x_m) = 0$. The assumptions are merely to simplify our presentation; we discharge these assumptions in Section 3.1.

The algorithm $\mathcal{A}$ was informally described in the introduction, and is more formally described in Figure 1. At any time t, the state of algorithm $\mathcal{A}$ is described by a probability distribution $p_t(x)$ over the possible states x. So $\int_a^b p_t(x)dx$ is the probability that $x_t \in [a,b]$.

When a new function $f_t(\cdot)$ arrives:

(i) Let $x_m = \text{argmin} f_t(x)$ denote the minimizer of f_t, $x_r \geq x_m$ denote the point to the right of x_m where $\frac{1}{2}\int_{x_m}^{x_r} f''(y)\,dy = \int_{x_r}^{\infty} p_{t-1}(y)\,dy$.

(ii) Let $x_l \leq x_m$ denote the point to the left of x_m where $\frac{1}{2}\int_{x_l}^{x_m} f''(y)\,dy = \int_{-\infty}^{x_l} p_{t-1}(y)\,dy$.

(iii) We update the probability density function of our online algorithm as $p_t(x) = p_{t-1}(x) + \frac{1}{2}f''(x)$ for all $x \in [x_l, x_r]$ and $p_t(x) = 0$ for all other x.

Figure 1 The 2-competitive Online Algorithm $\mathcal{A}$.

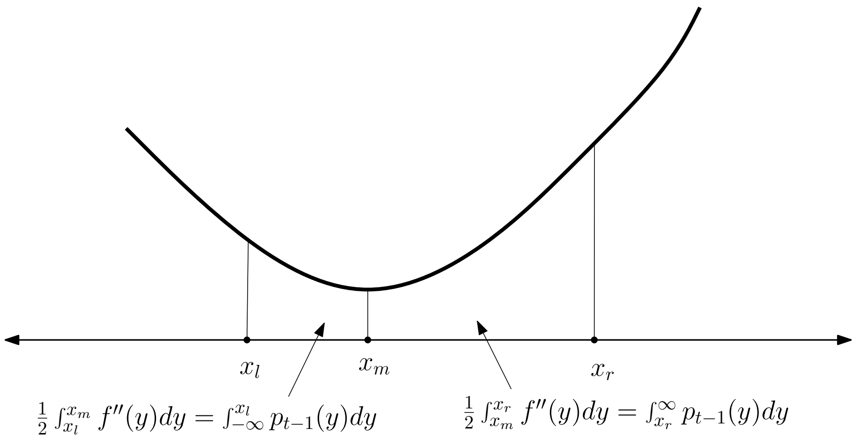

x_l
x_m
x_r
$\frac{1}{2}\int_{x_l}^{x_m} f''(y)dy = \int_{-\infty}^{x_l} p_{t-1}(y)dy$
$\frac{1}{2}\int_{x_m}^{x_r} f''(y)dy = \int_{x_r}^{\infty} p_{t-1}(y)dy$

Figure 2 Illustration of x_m, x_l and x_r.

Before beginning our analysis of $\mathcal{A}$, let us introduce some notation. Let $H_t = \mathbb{E}\left[f_t(x_t)\right] = \int_{y=-\infty}^{\infty} f_t(y)p_t(y)\,dy$ denote the *expected hit cost* for algorithm $\mathcal{A}$ at time t. Let $M_t = \mathbb{E}\left[|x_t - x_{t-1}|\right]$, which is equal to the earthmover distance between the two probability distributions[1], denote the *expected move cost* for algorithm $\mathcal{A}$ at time t. Similarly, let x_t^* be the adversary's state after time t. Let $H_t^* = f_t(x_t^*)$ be the hit cost for the adversary at time t, and $M_t^* = |x_t^* - x_{t-1}^*|$ be the movement cost for the adversary at time t. The analysis will use the potential function:

$$\Phi(p, x_t^*) = \Phi_1(p, x_t^*) + \Phi_2(p)$$

where

$$\Phi_1(p, x_t^*) = 2\int_{y=-\infty}^{\infty} |x_t^* - y|\, p(y)\,dy$$

and

$$\Phi_2(p) = -\int_{x=-\infty}^{\infty}\int_{y=-\infty}^{x} p(x)p(y)(x-y)\,dx\,dy.$$

Note that Φ is initially zero. To see that Φ is nonnegative, we show that $\Phi_1(p, x_t^*) \geq -\Phi_2(p)$ as follows:

$$-\Phi_2(p) = \int_{x=-\infty}^{\infty}\int_{y=-\infty}^{x} p(x)p(y)(x-y)\,dx\,dy$$

[1] Given two distributions, where each distribution is viewed as a unit amount of "dirt" piled on the line, the earthmover distance (aka Wasserstein metric) is the minimum “cost” of turning one pile into the other, which is the amount of dirt that needs to be moved times the distance it has to be moved.

$$
\begin{aligned}
&= \frac{1}{2}\int_{x=-\infty}^{\infty}\int_{y=-\infty}^{\infty} p(x)p(y)|x-y|\,dx\,dy \\
&\le \frac{1}{2}\int_{x=-\infty}^{\infty}\int_{y=-\infty}^{\infty} p(x)p(y)\left(|x-x_t^*|+|y-x_t^*|\right)\,dx\,dy \\
&= \frac{1}{2}\Bigg(\int_{x=-\infty}^{\infty} p(x)|x-x_t^*|\int_{y=-\infty}^{\infty} p(y)\,dx\,dy \\
&\qquad\qquad + \int_{y=-\infty}^{\infty} p(y)|y-x_t^*|\int_{x=-\infty}^{\infty} p(x)\,dx\,dy\Bigg) \\
&= \frac{1}{2}\left(\int_{x=-\infty}^{\infty} p(x)|x-x_t^*|\,dx + \int_{y=-\infty}^{\infty} p(y)|y-x_t^*|\,dy\right) \\
&= \int_{x=-\infty}^{\infty} p(x)|x-x_t^*|\,dx \\
&= \frac{1}{2}\Phi_1(p,x_t^*) \quad \le \quad \Phi_1(p,x_t^*).
\end{aligned}
$$

Thus, to prove that $\mathcal{A}$ is 2-competitive it is sufficient to show that at all times t:

$$H_t + M_t + \left(\Phi(p_t,x_t^*) - \Phi(p_{t-1},x_{t-1}^*)\right) \le 2(H_t^* + M_t^*). \tag{1}$$

We first consider the effect on inequality (1) as the adversary moves from x_{t-1}^* to x_t^*. The only term which increases in the LHS of inequality (1) is the first term in Φ, and this increase is at most $2|x_{t-1}^* - x_t^*| = 2M_t^*$, so inequality (1) holds. For the rest of the analysis we consider the effect on inequality (1) when the algorithm $\mathcal{A}$ moves from p_{t-1} to p_t.

To make this easier we make several simplifying assumptions, and simplify our notation slightly. Without loss of generality, we assume that $f_t(x_m) = 0$ (i.e., the minimum value is 0). Indeed, for general f_t, we can assume $g_t(x) = f_t(x) - f_t(x_m)$ and prove the entire analysis for g_t, and finally add the valid inequality $f_t(x_m) \le 2f_t(x_m)$ to inequality (1) for g_t to get the corresponding inequality for f_t. (Here we use that the functions f_t are non-negative.) Also without loss of generality we will translate the points so that $x_m = 0$. To further simplify exposition, let us decompose f_t into two separate functions, $f_t^>(x)$ and $f_t^<(x)$, where the former function is 0 for all $x \le x_m$ and $f_t(x)$ otherwise, and likewise, the latter function is 0 for all $x \ge x_m$ and $f_t(x)$ otherwise. It is easy to see that $f_t(x) = f_t^<(x) + f_t^>(x)$ for all x. Hence, we can imagine that we first feed $f_t^>(\cdot)$ to the online algorithm, and then feed $f_t^<(\cdot)$ to the online algorithm, and separately show inequality (1) for each of these functions. Henceforth, we shall assume that we are dealing with the function $f_t^>(x)$. Finally we will assume without loss of generality that x_m is the leftmost point with non-zero probability mass.

For notational simplicity, we avoid overuse of subscripts and superscripts by letting d denote x_r, z denote x_t^*, p denote the original distribution p_{t-1}, and q denote the resultant distribution $p_t(\cdot)$. So by the definition of the algorithm $\mathcal{A}$, we have in our new notation, $\int_{x=0}^{d}\frac{1}{2}f''(x)\,dx = \int_{x=d}^{\infty} p(x)\,dx$. Here are some simple facts used repeatedly in our analysis.

▶ **Fact 3.1.** *For any smooth convex function f, and any values a, b and c,*

$$\int_{x=a}^{b} (c-x)f''(x)\,dx = (c-b)f'(b) - (c-a)f'(a) + f(b) - f(a)\ .$$

Proof. This is an application of integration by parts. ◀

▶ **Fact 3.2.** $\int_d^\infty p(x)\,dx = f'(d)/2$. *And hence,* $\int_{x=0}^{d} p(x)\,dx = 1 - f'(d)/2$.

Proof. By the definition of $\mathcal{A}$, it is the case that $\frac{1}{2}\int_0^d f''(x)dx = \int_d^\infty p(x)\,dx$. Then note that $\frac{1}{2}\int_0^d f''(x)dx = \frac{1}{2}(f'(d) - f'(0)) = \frac{1}{2}f'(d)$, where the second equality follows because 0 is the minimizer of f. ◀

We now proceed bounding the various terms in the inequality (1).

▶ **Lemma 3.3.** *The hit cost H_t is exactly $\int_{x=0}^d f(x)p(x)\,dx + \frac{1}{2}\int_{x=0}^d f(x)f''(x)\,dx$.*

Proof. This follows from the definition of the hit cost, and the following facts: (i) $f(x) = 0$ if $x < 0$, and (ii) the distribution $q(x)$ is simply $p(x) + \frac{1}{2}f''(x)$ for $x \in [0, d]$ and 0 if $x > d$. ◀

▶ **Lemma 3.4.** $M_t = \int_{x=d}^\infty xf(x)\,dx + \frac{f(d)}{2} - \frac{df'(d)}{2}$.

Proof. We can view the updating of the probability distribution as a two step procedure. First, all the probability mass to the right of d moves to d, and then exactly a probability mass of $\frac{1}{2}f''(x)$ moves from d to each point $x \in [0, d]$. Thus

$$\begin{aligned} M_t &= \int_0^d \frac{1}{2}f''(x)(d-x)\,dx + \int_d^\infty p(x)(x-d)\,dx \\ &= \frac{f(d)}{2} + \int_d^\infty xp(x)\,dx - \frac{df'(d)}{2} \end{aligned}$$

Here we used Fact 3.1 to simplify the first term, and Fact 3.2 to simplify the second term. ◀

▶ **Lemma 3.5.** $\Phi_1(q, z) - \Phi_1(p, z) \le 2f(z) - 2M_t$.

Proof. First consider the case that $z < d$.

$$\begin{aligned} \Phi_1(q,z) - \Phi_1(p,z) &= 2\int_0^\infty |x-z|(q(x)-p(x))\,dx \\ &= \int_0^z (z-x)f''(x)\,dx + \int_z^d (x-z)f''(x)\,dx - 2\int_d^\infty (x-z)p(x)\,dx \\ &= 2f(z) - f(d) - (z-d)f'(d) - 2\int_d^\infty xp(x)\,dx + 2z\int_d^\infty p(x)\,dx \\ &= 2f(z) - f(d) - (z-d)f'(d) - 2\int_d^\infty xp(x)\,dx + zf'(d) \\ &= 2f(z) - f(d) + df'(d) - 2\int_d^\infty xp(x)\,dx \\ &= 2f(z) - 2M_t \end{aligned}$$

The first equality is by the definition of Φ_1. The second equality is by the definition of the algorithm $\mathcal{A}$. The third equality is by application of Fact 3.1 and separating the last integral. The fourth equality is by Fact 3.2. The final equality is uses Lemma 3.4.

Now consider that case that $z \ge d$.

$$\begin{aligned} \Phi_1(q,z) - \Phi_1(p,z) &= 2\int_0^\infty |x-z|(q(x)-p(x))\,dx \\ &= \int_0^d (z-x)f''(x)\,dx - 2\int_d^z (z-x)p(x)\,dx - 2\int_z^\infty (x-z)p(x)\,dx \\ &= (z-d)f'(d) + f(d) - 2\int_d^z (z-x)p(x)\,dx - 2\int_z^\infty (x-z)p(x)\,dx \\ &= (z-d)f'(d) + f(d) - 2\int_d^\infty (x-z)p(x)\,dx - 4\int_d^z (z-x)p(x)\,dx \end{aligned}$$

$$\begin{aligned}
&\le (z-d)f'(d) + f(d) - 2\int_d^\infty (x-z)p(x)\,dx \\
&= (z-d)f'(d) + f(d) - 2\int_d^\infty xp(x)\,dx + 2\int_d^\infty zp(x)\,dx \\
&= -df'(d) + f(d) - 2\int_d^\infty xp(x)\,dx + 2zf'(d) \\
&\le 2f(z) - f(d) + df'(d) - 2\int_d^\infty xp(x)\,dx \\
&= 2f(z) - 2M_t
\end{aligned}$$

The first equality is by the definition of Φ_1. The second equality is by the definition of the algorithm $\mathcal{A}$. The third equality is an application of integration by parts. The fourth equality follows from replacing the term $2\int_z^\infty (x-z)p(x)\,dx$ by $2\int_d^\infty (x-z)p(x)\,dx - 2\int_d^z (x-z)p(x)\,dx$. The first inequality from the fact that $\int_d^z (z-x)p(x)\,dx \ge 0$ since $z \ge d$. The sixth equality uses Fact 3.2. The second inequality holds because, as f is convex, $f(z) \ge f(d) + (z-d)f'(d)$, and hence $zf'(d) \le f(z) - f(d) + df'(d)$. The final equality is uses Lemma 3.4. ◀

We now turn to analyzing $\Phi_2(q) - \Phi_2(p)$. We can express this as:

$$\begin{aligned}
&-\int_{x=0}^{d}\int_{y=0}^{x}(x-y)\left(p(x)+\frac{1}{2}f''(x)\right)\left(p(y)+\frac{1}{2}f''(y)\right)dy\,dx \\
&+\int_{x=0}^{\infty}\int_{y=0}^{x}(x-y)p(x)p(y)\,dy\,dx \\
&= -\underbrace{\frac{1}{4}\int_{x=0}^{d}\int_{y=0}^{x}(x-y)f''(x)f''(y)\,dy\,dx}_{T_1} - \underbrace{\frac{1}{2}\int_{x=0}^{d}\int_{y=0}^{x}(x-y)p(x)f''(y)\,dy\,dx}_{T_2} \\
&-\underbrace{\frac{1}{2}\int_{x=0}^{d}\int_{y=0}^{x}(x-y)f''(x)p(y)\,dy\,dx}_{T_3} + \underbrace{\int_{x=d}^{\infty}\int_{y=0}^{x}(x-y)p(x)p(y)\,dy\,dx}_{T_4} \qquad (2)
\end{aligned}$$

We now bound the terms T_1, T_2, T_3 and T_4.

▶ **Lemma 3.6.** $T_1 = \frac{1}{4}\int_0^d f(x)f''(x)\,dx$

Proof. This follows by applying Fact 3.1 to the inner integral of T_1. ◀

▶ **Lemma 3.7.** $T_2 = \frac{1}{2}\int_0^d f(x)p(x)\,dx$.

Proof. This follows by applying Fact 3.1 to the inner integral of T_2. ◀

▶ **Lemma 3.8.** $T_3 = -\frac{f'(d)}{2}\int_{x=0}^d xp(x)\,dx + \left(\frac{df'(d)}{2} - \frac{f(d)}{2}\right)\left(1 - \frac{f'(d)}{2}\right) + \frac{1}{2}\int_{x=0}^d f(x)p(x)\,dx$.

Proof.

$$\begin{aligned}
T_3 &= \frac{1}{2}\int_{x=0}^{d}\int_{y=0}^{x}(x-y)p(y)f''(x)\,dy\,dx \\
&= \frac{1}{2}\int_{y=0}^{d}\int_{x=y}^{d}(x-y)p(y)f''(x)\,dx\,dy \\
&= -\frac{1}{2}\int_{y=0}^{d}p(y)\int_{x=y}^{d}(y-x)f''(x)\,dx\,dy
\end{aligned}$$

$$\begin{aligned}
&= -\frac{1}{2}\int_{y=0}^{d} p(y)\left[(y-d)f'(d)+f(d)-f(y)\right]dy\\
&= -\frac{f'(d)}{2}\int_{y=0}^{d} yp(y)\,dy + \left(\frac{df'(d)}{2}-\frac{f(d)}{2}\right)\int_{y=0}^{d} dy + \frac{1}{2}\int_{y=0}^{d} f(y)p(y)\,dy\\
&= -\frac{f'(d)}{2}\int_{x=0}^{d} xp(x)\,dx + \left(\frac{df'(d)}{2}-\frac{f(d)}{2}\right)\left(1-\frac{f'(d)}{2}\right) + \frac{1}{2}\int_{x=0}^{d} f(x)p(x)\,dx
\end{aligned}$$

The second equality follows as the order of integration is just reversed. The fourth equality is an application of Fact 3.1. The last equality uses Fact 3.2. ◀

▶ **Lemma 3.9.** $T_4 \le \int_d^\infty xp(x)\,dx - \frac{f'(d)}{2}\int_0^d xp(x)\,dx - \frac{df'(d)^2}{4}$.

Proof.

$$\begin{aligned}
T_4 &= \int_{x=d}^{\infty}\int_{y=0}^{x}(x-y)p(x)p(y)\,dy\,dx\\
&= \int_{x=d}^{\infty}\int_{y=0}^{d}(x-y)p(x)p(y)\,dx\,dy + \int_{y=d}^{\infty}\int_{x=y}^{\infty}(x-y)p(x)p(y)\,dx\,dy \qquad (3)
\end{aligned}$$

The first expression in (3) can be rewritten as

$$\begin{aligned}
\int_{x=d}^{\infty}\int_{y=0}^{d}(x-y)p(x)p(y)\,dx\,dy &= \int_{x=d}^{\infty} xp(x)\,dx\int_{y=0}^{d}p(y)\,dy - \int_{x=d}^{\infty}p(x)\,dx\int_{y=0}^{d}yp(y)\,dy\\
&= \left(1-\frac{f'(d)}{2}\right)\int_{x=d}^{\infty}xp(x)\,dx - \frac{f'(d)}{2}\int_{y=0}^{d}yp(y)\,dy
\end{aligned}$$

The second equality follows by Fact 3.2. Similarly, for the second expression in (3), we get

$$\begin{aligned}
\int_{y=d}^{\infty}\int_{x=y}^{\infty}(x-y)p(x)p(y)\,dx\,dy &\le \left(\int_{y=d}^{\infty}p(y)\,dy\right)\int_{x=d}^{\infty}(x-d)p(x)\,dx\\
&= \left(\int_{y=d}^{\infty}p(y)\,dy\right)\int_{x=d}^{\infty}xp(x)\,dx - d\int_{x=d}^{\infty}\int_{y=d}^{\infty}p(y)p(x)\,dy\,dx\\
&= \frac{f'(d)}{2}\int_{x=d}^{\infty}xp(x)dx - \frac{df'(d)^2}{4}
\end{aligned}$$

Here, the inequality uses $(x-y)\le(x-d)$, since $y\ge d$, and the last equality uses Fact 3.2 again. Summing the expressions (and replacing the variable y by x) completes the proof. ◀

We now use Lemmas 3.3 to 3.9 to show that inequality (1) holds as follows:

$$\begin{aligned}
&H_t + M_t + \Phi(p_t) - \Phi(p_{t-1})\\
&\le \int_{x=0}^{d} f(x)p(x)\,dx + \frac{1}{2}\int_{x=0}^{d} f(x)f''(x)\,dx + 2f(z) - \int_{x=d}^{\infty} xf(x)\,dx - \frac{f(d)}{2} + \frac{df'(d)}{2}\\
&\quad -\frac{1}{4}\int_0^d f(x)f''(x)\,dx - \frac{1}{2}\int_0^d f(x)p(x)\,dx\\
&\quad + \frac{f'(d)}{2}\int_{x=0}^{d} xp(x)\,dx - \left(\frac{df'(d)}{2}-\frac{f(d)}{2}\right)\left(1-\frac{f'(d)}{2}\right) - \frac{1}{2}\int_{x=0}^{d} f(x)p(x)\,dx\\
&\quad + \int_d^\infty xp(x)\,dx - \frac{f'(d)}{2}\int_0^d xp(x)\,dx - \frac{df'(d)^2}{4}
\end{aligned}$$

When a new function $f_t(\cdot)$ arrives:

(i) Let $x_m = \text{argmin} f_t(x)$ denote the minimizer of f_t.

(ii) Move in the direction of x_m until we reach either (a) a point x s.t. $|x - x_{t-1}| = f_t(x)/2$ or (b) the minimizer x_m. Whichever happens first, set x_t to be that point.

Figure 3 The 3-competitive Memoryless Algorithm $\mathcal{M}$.

$$= 2f(z) + \frac{1}{4}\int_0^d f(x)f''(x)\,dx - \frac{f(d)f'(d)}{4}$$
$$= 2f(z) + \frac{1}{4}\left(f(d)f'(d) - \int_{y=0}^{d} (f'(y))^2\,dy\right) - \frac{f(d)f'(d)}{4}$$
$$\leq 2f(z)$$

The first equality follows by canceling identical terms. The second equality is an application of integration by parts. This proves inequality (1) and hence the 2-competitiveness of our algorithm.

3.1 Discharging the Assumptions

We now explain how to modify the algorithm and analysis if some of our simplifying assumptions do not hold. If the functions are piecewise linear, then in the algorithm we can suitably discretize the integral into a summation, and replace the second derivative at a point by the difference in slopes between consecutive points and increase the probability at each point by this difference amount. The analysis then goes through mostly unchanged. If the minimizer is at infinity, then the analysis goes through pretty much unchanged except that we can not translate so that the minimizer is at 0, and we have to explicitly keep x_m instead of 0 in the limits of the integration.

4 Memoryless Algorithm

In this section we present a simple 3-competitive memoryless algorithm $\mathcal{M}$. The action of $\mathcal{M}$ at time t depends only upon the past state x_{t-1} and the current function $f_t(x)$. The algorithm $\mathcal{M}$ is described informally in the introduction, and more formally in Figure 3. We adopt the same notation from the previous section using x_t and x_t^* to denote the locations of the algorithm and of the adversary, using H_t^* and M_t^* to denote the move and hit cost for the adversary, and we remove the expectations from the algorithm's costs, so now $H_t = f_t(x_t)$ and $M_t = |x_t - x_{t-1}|$,

▶ **Theorem 4.1.** *The online algorithm $\mathcal{M}$ is 3-competitive for the ACO problem.*

Proof. We use the potential function $\Phi(x, x^*) = 3|x - x^*|$. Clearly Φ is initially zero, and always nonnegative. Thus it will be sufficient to show that for each time step:

$$H_t + M_t + (\Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_{t-1}^*)) \leq 3(H_t^* + M_t^*). \tag{4}$$

Two simple observations are that if $x_{t-1} = x_m$ then the algorithm does not move and, secondly that for all t, $M_t \leq H_t/2$. We now argue that equation (4) always holds. Indeed, we can upperbound the change in potential by first making the adversary move and then moving the algorithm's point. Using the triangle inequality and the definition $M_t^* = |x_t^* - x_{t-1}^*|$,

$$\Phi(x_{t-1}, x_t^*) - \Phi(x_{t-1}, x_{t-1}^*) \leq 3M_t^* \quad . \tag{5}$$

Therefore, we will assume that the optimal solution has already moved to x_t^*, and show that

$$H_t + M_t + \Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*) \leq 3H_t^* \quad . \tag{6}$$

Adding equation (5) and equation (6) gives us equation (4), completing the proof. To establish eq. (6) we now consider two cases, based on the relative values of H_t and H_t^*.
Case 1: Suppose that $H_t \leq H_t^*$. We upper bound the change in potential from the algorithm moving by $3M_t$ (again using the triangle inequality) and using the fact that $M_t \leq H_t/2$, and the inequality defining the case to obtain:

$$H_t + M_t + (\Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*)) \leq H_t + H_t/2 + 3M_t \leq 3H_t \leq 3H_t^* \ .$$

Case 2: Suppose that $H_t > H_t^*$. In this case, all of the algorithm's movement must have been towards x_t^*, since it was moving in the direction of decreasing value but did not reach x_t^*. Thus, the algorithm's movement must *decrease* the potential function by $3M_t$. Furthermore, since the algorithm is not at x_m, it must be the case that $M_t = H_t/2$. We therefore have

$$H_t + M_t + (\Phi(x_t, x_t^*) - \Phi(x_{t-1}, x_t^*)) \leq H_t + H_t/2 - 3M_t \leq 0 \leq 3H_t^* \ .$$

This completes the proof. ◀

5 Lower Bounds

We first show that no memoryless deterministic algorithm can be better than 3-competitive. We then show that the competitive ratio of every algorithm is at least 1.86.

5.1 Lower Bound for Memoryless Algorithms

We show that no memoryless deterministic algorithm B can be better than 3-competitive. The first issue is that the standard definition of memorylessness, that the next state only depends on the current state and the current input, is problematic for the OCO problem. Because the state is a real number, any algorithm be converted into an algorithm in which all the memory is encoded in the very low order bits of the current state, and is thus memoryless under this standard definition. Intuitively we believe that the notion of memoryless for the setting of OCO should mean that the algorithm's responses don't depend on the scale of the line (e.g. whether distance is measured in meters or kilometers), and the algorithm's responses are bilaterally symmetric (so the algorithm's response would be the mirror of its current response if the function and the location were mirrored around the function minimizer). We formalize this in the setting that all functions are "vee-shaped", that is they have the form $f_t(x) = a|x - b|$ for some constants $a \geq 0$ and b. Our lower bound only uses such functions. In this setting, say that an algorithm is memoryless if the ratio of the distance that algorithm the moves to the distance from the previous location to the minimizer, namely $(|x_t - x_{t-1}|/(|x_{t-1} - b|)$ depends only on a, the slope of the vee-shaped function. We can assume without any real loss of generality that a memoryless algorithm always moves towards the minimizer, as any algorithm without this property cannot be $O(1)$-competitive.

Assume that the initial position is the origin of the line. The first function that arrives is $\varepsilon|x - 1|$ for some small slope ε. We consider two cases. In the first case, assume that the distance δ that B moves is less than $\varepsilon/2$. Thus B's hit cost is at least $\varepsilon(1-\delta) \geq \varepsilon(1-\varepsilon/2) = \varepsilon - \varepsilon^2/2$. In that case we continue bringing in copies of the function $\varepsilon|x-1|$. By the definition of memorylessness, B will maintain the invariant that the ratio of its hit cost to its to its movement cost is $(\varepsilon(1-\delta))/\delta \geq 2 - \varepsilon$. This continues until B gets very close to 1. Thus B's move cost is asymptotically 1, and its hit cost is at least $2-\varepsilon$. Thus B's cost is asymptotically 3. A cost of 1 is achievable by moving to the state 1 when the first function arrives.

Now consider the case that the distance δ that B moves in response to the first function is more than $\varepsilon/2$. In this case we bring many copies of the function $\varepsilon|x|$, until B has returned to very near the origin. Thus B's movement cost is approximately 2δ. By our assumption of memorylessness, $x_t = \delta(1-\delta)^{t-1}$. Thus B's hit cost is asymptotically

$$\varepsilon(1-\delta) + \sum_{t=2}^{\infty} \varepsilon\delta(1-\delta)^{t-1} = 2\varepsilon(1-\delta).$$

Thus B's total cost is at least $2\varepsilon + 2\delta(1-\varepsilon)$. Using the fact that $\delta \geq \varepsilon/2$ in this case, B's cost is at least $3\varepsilon - 2\varepsilon^2$. A cost of ε is achievable by never leaving state 0.

5.2 General Lower Bound

We now prove a lower bound on the competitive ratio of any online algorithm.

▶ **Theorem 5.1.** *There is no c-competitive algorithm for the OCO problem when $c < 1.86$.*

Proof. By Lemma 2.1, we can restrict to deterministic algorithms without loss of generality. Let $\mathcal{O}$ be an arbitrary c-competitive deterministic algorithm.

We now define our adversarial strategy. The initial position is 0. Then some number of functions of the form $\varepsilon|1-x|$ arrive. We will be interested in the limit as ε approaches 0. Then some number, possibly none, of functions of the form $\varepsilon|x|$ arrive.

For the deterministic algorithm, let $b(s)$ denote the position of $\mathcal{O}$ after s/ε functions of the type $\varepsilon|1-x|$ have arrived. Intuitively, if $b(s)$ is too small for large enough s, then it has a high hit cost on the first s/ε functions whereas the optimal solution would have moved immediately to the point 1 only incurring the moving cost. Alternately, if the position $b(s)$ is sufficiently far to the right (i.e., close to 1), then the adversary can introduce a very long sequence of requests of type $\varepsilon|x|$, forcing the algorithm to eventually move back to 0 incurring the movement cost of $b(s)$ again. In this case, the optimal solution would have stayed at 0.

Formally, the total function cost $\mathcal{O}$ at time s/ε is at least $b(s) + \int_0^s (1-b(y))\,dy$. Now, if the adversary introduces an infinite sequence of functions of the form $\varepsilon|x|$, then the best that the online algorithm can do is to move immediately to the origin incurring an additional movement cost of $b(s)$. Meanwhile, the optimal solution would have stayed at 0 throughout incurring a total cost of s. Hence, if the online algorithm is c-competitive, we must have, for all s,

$$2b(s) + \int_0^s (1-b(y))\,dy \leq cs. \tag{7}$$

Alternately, if the functions $\varepsilon|1-x|$ keep appearing forever, the online algorithm eventually moves to 1 and its total cost is therefore at least $1+\int_0^\infty (1-b(y))\,dy$ and the optimal solution would have moved to 1 at time 0 and only incurred the movement cost of 1. Hence, we also have

$$1 + \int_0^\infty (1-b(y))\,dy \leq c. \tag{8}$$

This establishes the dichotomy using which we complete our lower bound proof. Indeed, define $G(s) = \int_0^s (1-b(y))\,dy$. Then, $G'(s) = 1-b(s)$ and we can write (7) as, for all s we have

$$G'(s) \geq \frac{1}{2}\left(2 - cs + G(s)\right) \tag{9}$$

and (8) is simply

$$G(\infty) \leq c - 1\,.$$

Now, notice that in order to minimize $G(\infty)$, we may assume that (9) is satisfied at equality for all s (this can only reduce $G(s)$, which in turn reduces $G'(s)$ further), which in turn gives us a unique solution to G.

Now, writing (9) as equality and differentiating w.r.t s, we get the first-order differential equation $b(s) = 2b'(s) - c + 1$. It is a simple calculation to verify that its unique solution satisfying $b(0) = 0$ is $b(s) = (c-1)\cdot(e^{s/2}-1)$. But now, we can plug this into $G(\infty)$ to get that

$$\int_0^{2\ln\frac{c}{c-1}} (1 - b(s))\,ds + 1 = \int_0^{2\ln\frac{c}{c-1}} \left(1 - (c-1)\left(e^{s/2}-1\right)\right)\,ds + 1 \leq c.$$

Evaluation of the integral and simplification yields

$$2\ln\frac{c}{c-1} - (c-1)\left(\frac{2c}{c-1} - 2\ln\frac{c}{c-1} - 2\right) + 1 = 2c\ln\frac{c}{c-1} - 1 \leq c,$$

which is false for $c < 1.86$. ◀

We conjecture that the optimal competitive ratio for the general problem is strictly less than 2, and is achieved for the special case where all functions are of the form $\varepsilon|x|$ or $\varepsilon|x-1|$. It is implausible that our lower bound for this special case is tight. Intuitively, the optimal competitive ratio would be 2 if and only if the optimally competitive algorithm doesn't accelerate the rate of probability mass transfer, whereas it seems beneficial to accelerate the rate of probability mass transfer.

Acknowledgements. We thank Adam Wierman for his assistance and for many stimulating conversations. We thank Neal Barcelo and Michael Nugent for their assistance with the general lower bound.

References

1 Lachlan L. H. Andrew, Siddharth Barman, Katrina Ligett, Minghong Lin, Adam Meyerson, Alan Roytman, and Adam Wierman. A tale of two metrics: Simultaneous bounds on competitiveness and regret. In *COLT 2013 – The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, pages 741–763, 2013.

2 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Towards the randomized k-server conjecture: A primal-dual approach. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 40–55, 2010.

3 Yair Bartal, Béla Bollobás, and Manor Mendel. Ramsey-type theorems for metric spaces with applications to online problems. *J. Comput. Syst. Sci.*, 72(5):890–921, 2006.

4 Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. On metric ramsey-type phenomena. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC'03, pages 463–472, New York, NY, USA, 2003. ACM.

5 Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763, 1992.

6 M. Chrobak, H. Karloff, T. Payne, and S. Vishwanathan. New results on server problems. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'90, pages 291–300, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

7 Aaron Coté, Adam Meyerson, and Laura Poplawski. Randomized k-server on hierarchical binary trees. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC'08, pages 227–234, New York, NY, USA, 2008. ACM.

8 Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM J. Comput.*, 32(6):1403–1422, 2003.

9 Anna R. Karlin, Mark S. Manasse, Lyle A. McGeoch, and Susan S. Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

10 Minghong Lin, Zhenhua Liu, Adam Wierman, and Lachlan L. H. Andrew. Online algorithms for geographical load balancing. In *2012 International Green Computing Conference, IGCC 2012, San Jose, CA, USA, June 4-8, 2012*, pages 1–10, 2012.

11 Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. Online dynamic capacity provisioning in data centers. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing, Allerton Park & Retreat Center, Monticello, IL, USA, 28-30 September, 2011*, pages 1159–1163, 2011.

12 Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Trans. Netw.*, 21(5):1378–1391, 2013.

13 Minghong Lin, Adam Wierman, Alan Roytman, Adam Meyerson, and Lachlan L. H. Andrew. Online optimization with switching cost. *SIGMETRICS Performance Evaluation Review*, 40(3):98–100, 2012.

14 Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H. Low, and Lachlan L. H. Andrew. Greening geographical load balancing. In *SIGMETRICS 2011, Proceedings of the 2011 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, San Jose, CA, USA, 07-11 June 2011 (Co-located with FCRC 2011)*, pages 233–244, 2011.

15 Kai Wang, Minghong Lin, Florin Ciucu, Adam Wierman, and Chuang Lin. Characterizing the impact of the workload on the value of dynamic resizing in data centers. In *Proceedings of the IEEE INFOCOM 2013, Turin, Italy, April 14-19, 2013*, pages 515–519, 2013.

16 Adam Wierman. Personal communication, 2015.

Beating the Random Assignment on Constraint Satisfaction Problems of Bounded Degree

Boaz Barak[1], Ankur Moitra[2], Ryan O'Donnell[3], Prasad Raghavendra[4], Oded Regev[5], David Steurer[6], Luca Trevisan[4], Aravindan Vijayaraghavan[5], David Witmer[3], and John Wright[3]

1 **Microsoft Research New England, USA**
2 **MIT Mathematics Department, USA**
3 **Department of Computer Science, Carnegie Mellon University, USA**
4 **UC Berkeley, Department of Electrical Engineering and Computer Sciences, USA**
5 **Courant Institute of Mathematical Sciences, New York University, USA**
6 **Cornell University, USA**

Abstract

We show that for any odd k and any instance $\Im$ of the Max-kXOR constraint satisfaction problem, there is an efficient algorithm that finds an assignment satisfying at least a $\frac{1}{2}+\Omega(1/\sqrt{D})$ fraction of $\Im$'s constraints, where D is a bound on the number of constraints that each variable occurs in. This improves both qualitatively and quantitatively on the recent work of Farhi, Goldstone, and Gutmann (2014), which gave a *quantum* algorithm to find an assignment satisfying a $\frac{1}{2}+\Omega(D^{-3/4})$ fraction of the equations.

For arbitrary constraint satisfaction problems, we give a similar result for "triangle-free" instances; i.e., an efficient algorithm that finds an assignment satisfying at least a $\mu+\Omega(1/\sqrt{D})$ fraction of constraints, where μ is the fraction that would be satisfied by a uniformly random assignment.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases constraint satisfaction problems, bounded degree, advantage over random

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.110

1 Introduction

An instance of a Boolean constraint satisfaction problem (CSP) over n variables $x_1,\dots,x_n$ is a collection of *constraints*, each of which is some predicate P applied to a constant number of the variables. The computational task is to find an assignment to the variables that maximizes the number of satisfied predicates. In general the constraint predicates do not need to be of the same "form"; however, it is common to study CSPs where this is the case. Typical examples include: Max-kSAT, where each predicate is the OR of k variables or their negations; Max-kXOR, where each predicate is the XOR of exactly k variables or their negations; and Max-Cut, the special case of Max-2XOR in which each constraint is of the form $x_i \neq x_j$. The case of Max-kXOR is particularly mathematically natural, as it is equivalent to maximizing a homogenous degree-k multilinear polynomial over $\{\pm1\}^n$.

Given a CSP instance, it is easy to compute the expected fraction μ of constraints satisfied by a uniformly random assignment; e.g., in the case of Max-kXOR we always have $\mu=\frac{1}{2}$. Thus the question of algorithmic interest is to find an assignment that satisfies noticeably more than a μ fraction of constraints. Of course, sometimes this is simply not possible; e.g., for Max-Cut on the complete n-variable graph, at most a $\frac{1}{2} + O(1/n)$ fraction of constraints can be satisfied.[1] However, even when all or almost all constraints can be satisfied, it may still be algorithmically difficult to beat μ. For example, Håstad [7] famously proved that for every $\epsilon > 0$, given a Max-3XOR instance in which a $1 - \epsilon$ fraction of constraints can be satisfied, it is NP-hard to find an assignment satisfying a $\frac{1}{2} + \epsilon$ fraction of the constraints. Håstad showed similar "approximation resistance" results for Max-3Sat and several other kinds of CSPs.

One possible reaction to these results is to consider subconstant ϵ. For example, Håstad and Venkatesh [8] showed that for every Max-kXOR instance with m constraints, one can efficiently find an assignment satisfying at least a $\frac{1}{2} + \Omega(1/\sqrt{m})$ fraction of them.[2] (Here, and elsewhere in this introduction, the $\Omega(\cdot)$ hides a dependence on k, typically exponential.) Relatedly, Khot and Naor [9] give an efficient algorithm for Max-3XOR that satisfies a $\frac{1}{2} + \Omega(\epsilon\sqrt{(\log n)/n})$ fraction of constraints whenever the optimum fraction is $\frac{1}{2} + \epsilon$.

Another reaction to approximation resistance is to consider restricted instances. One commonly studied restriction is to assume that each variable's "degree" — i.e., the number of constraints in which it occurs — is bounded by some D. Håstad [6] showed that such instances are never approximation resistant. More precisely, he showed that for, say, Max-kXOR, one can always efficiently find an assignment satisfying at least a $\mu + \Omega(1/D)$ fraction of constraints.[3] Note that this advantage of $\Omega(1/D)$ cannot in general be improved, as the case of Max-Cut on the complete graph shows.

One may also consider further structural restrictions on instances. One such restriction is that the underlying constraint hypergraph be *triangle-free* (see Section 2 for a precise definition). For example, Shearer [12] showed that for triangle-free graphs there is an efficient algorithm for finding a cut of size at least $\frac{m}{2} + \Omega(1) \cdot \sum_i \sqrt{\deg(i)}$, where $\deg(i)$ is the degree of the ith vertex. As $\sum_i \sqrt{\deg(i)} \geq \sum_i \frac{\deg(i)}{\sqrt{D}} = \frac{2m}{\sqrt{D}}$ in m-edge degree-D bounded graphs, this shows that for *triangle-free* Max-Cut one can efficiently satisfy at least a $\frac{1}{2} + \Omega(1/\sqrt{D})$ fraction of constraints. Related results have also been shown for degree-bounded instances of Maximum Acyclic Subgraph [2], Min-Bisection [1] and Ordering k-CSPs [5, 10].

1.1 Recent developments and our work

In a recent surprising development, Farhi, Goldstone, and Gutmann [4] gave an efficient *quantum* algorithm that, for Max-3XOR instances with degree bound D, finds an assignment satisfying a $\frac{1}{2} + \Omega(D^{-3/4})$ fraction of the constraints. In addition, Farhi et al. show that if the Max-3XOR instance is "triangle-free" then an efficient quantum algorithm can satisfy a $\frac{1}{2} + \Omega(1/\sqrt{D})$ fraction of the constraints.

Farhi et al.'s result was perhaps the first example of a quantum algorithm providing a better CSP approximation guarantee than that of the best known classical algorithm (namely Håstad's [6], for Max-3XOR). As such it attracted quite some attention.[4] In this paper we

[1] Another trivial example is the Max-2XOR instance with the two constraints $x = y$ and $x \neq y$. For this reason we always assume that our Max-kXOR instances do not contain a constraint and its negation.

[2] In [8] this is stated as an approximation-ratio guarantee: if the optimum fraction is $\frac{1}{2} + \epsilon$ then $\frac{1}{2} + \Omega(\epsilon/\sqrt{m})$ is guaranteed. However inspecting their proof yields the absolute statement we have made.

[3] The previous footnote applies also to this result.

[4] As evidenced by the long list of authors on this paper; see also `http://www.scottaaronson.com/blog/?p=2155`.

show that classical algorithms can match, and in fact outperform, Farhi et al.'s quantum algorithm.

First result: Max-kXOR

We will present two results. The first result is about instances of Max-kXOR.

▶ **Theorem 1.** *There is a constant $c = \exp(-O(k))$ and a randomized algorithm running in time $\mathrm{poly}(m, n, \exp(k))$ that, given an instance $\Im$ of Max-kXOR with m constraints and degree at most D, finds with high probability an assignment $x \in \{\pm 1\}^n$ such that*

$$\left| val_{\Im}(x) - \frac{1}{2} \right| \geq \frac{c}{\sqrt{D}} \,. \tag{1}$$

Here $val_{\Im}(x)$ denotes the fraction of constraints satisfied by x. In particular, for odd k, by trying the assignment and its negation, the algorithm can output an x satisfying

$$val_{\Im}(x) \geq \frac{1}{2} + \frac{c}{\sqrt{D}} \,. \tag{2}$$

In Section 3 we give a simple, self-contained proof of Theorem 1 in the special case of Max-3XOR. For higher k we obtain it from a more general result (Theorem 7) that gives a constructive version of a theorem of Dinur, Friedgut, Kindler and O'Donnell [3]. This result shows how to attain a significant deviation from the random assignment value for multivariate low-degree polynomials with low influences. See Section 4.

We note that the deviation $\Omega(1/\sqrt{D})$ in (1) is optimal. To see why, consider any D-regular graph on n vertices, and construct a Max-2XOR instance $\Im$ as follows. For every edge (i, j) in the graph we randomly and independently include either the constraint $x_i = x_j$ or $x_i \neq x_j$. For every fixed x, the quantity $\mathrm{val}_{\Im}(x)$ has distribution $\frac{1}{m}\mathrm{Binomial}(m, \frac{1}{2})$, where $m = \frac{nD}{2}$. Hence a Chernoff-and-union-bound argument shows that with high probability all 2^n assignments will have $|\mathrm{val}_{\Im}(x) - \frac{1}{2}| \leq O(\sqrt{n/m}) = O(1/\sqrt{D})$. This can easily be extended to Max-kXOR for $k > 2$.

General CSPs

As noted earlier, the case of Max-Cut on the complete graph shows that for general CSPs, and in particular for Max-2XOR, we cannot guarantee a positive advantage of $\Omega(1/\sqrt{D})$ as in (2). In fact, a positive advantage of $\Omega(1/D)$ is the best possible, showing that the guarantee of Håstad [6] is tight in general.

A similar example can be shown for Max-2SAT: consider an instance with D^2 variables and imagine them placed on a $D \times D$ grid. For any two variables in the same row add the constraint $x \vee y$ and for any two variables in the same column add the constraint $\bar{x} \vee \bar{y}$. Then each variable participates in $O(D)$ clauses, and it can be verified that the best assignment satisfies $3/4+O(1/D)$ fraction of the clauses. We do not know if the same holds for Max-3SAT and we leave that as an open question.

Sometimes no advantage over random is possible. For instance, consider the following instance with 8 clauses on 6 variables, in which *any* assignment satisfies exactly 1/2 of the clauses:

$$\begin{aligned}
&\{\mathrm{NAE}(x_1, x_2, x_3),\\
&\ \mathrm{AE}(y_1, x_2, x_3), \mathrm{AE}(x_1, y_2, x_3), \mathrm{AE}(x_1, x_2, y_3),\\
&\ \mathrm{NAE}(x_1, y_2, y_3), \mathrm{NAE}(y_1, x_2, y_3), \mathrm{NAE}(y_1, y_2, x_3),\\
&\ \mathrm{AE}(y_1, y_2, y_3)\} \,,
\end{aligned}$$

where NAE denotes the "not all equal" constraint, and AE is the "all equal" constraint.

Second result: triangle-free instances of general CSPs

Despite the above examples, our second result shows that it is possible to recover the optimal advantage of $1/\sqrt{D}$ for triangle-free instances of *any CSP*:

▶ **Theorem 2.** *There is a constant $c = \exp(-O(k))$ and a randomized algorithm running in time $\operatorname{poly}(m, n, \exp(k))$ time that, given a triangle-free, degree-D CSP instance $\Im$ with m arbitrary constraints, each of arity between 2 and k, finds with high probability an assignment $x \in \{\pm 1\}^n$ such that*

$$val_{\Im}(x) \geq \mu + \frac{c}{\sqrt{D}}.$$

Here μ is the fraction of constraints in $\Im$ that would be satisfied in expectation by a random assignment.

This theorem is proved in Section 5. For simplicity, we state our results as achieving *randomized* algorithms and leave the question of derandomizing them (e.g., by replacing true random bits with $O(k)$-wise independence or some other such distribution) to future work.

1.2 Overview of our techniques

All three algorithms that we present in this work follow the same broad outline, while the details are different in each case. To produce an assignment that beats a random assignment, the idea is to partition the variables in to two sets (F, G) with F standing for 'Fixed' and G standing for 'Greedy' (in Section 4, these correspond to $[n] \setminus U$ and U respectively). The variables in F are assigned independent and uniform random bits and the variables in G are assigned values *greedily* based on the values already assigned to F. We will refer to constraints with exactly one variable from G as *active* constraints. The design of the *greedy* assignments and their analysis is driven by two key objectives.

1. Obtain a significant advantage over the random assignment on *active* constraints.
2. Achieve a value that is at least as good as the random assignment on inactive constraints.

The simplest example is the algorithm for Max-3XOR that we present in Section 3. First, we appeal to a *decoupling* trick due to Khot-Naor [9] to give an efficient approximation-preserving reduction from an arbitrary instance $\Im$ of Max-3XOR to a bipartite instance $\tilde{\Im}$. Specifically, the instance $\tilde{\Im}$ will contain two sets of variables $\{y_i\}_{i\in[n]}$ and $\{z_i\}_{i\in[n]}$, with every constraint having exactly one variable from $\{y_i\}_{i\in[n]}$ and two variables from $\{z_j\}_{j\in[n]}$. Notice that if we set $G = \{y_i\}_{i\in[n]}$, then objective (2) holds vacuously, i.e., every constraint in $\tilde{\Im}$ is active. The former objective (1) is achieved as a direct consequence of anticoncentration of low degree polynomials (see Fact 5). In the case of Max-kXOR, the second objective is achieved by slightly modifying the greedy assignment: we flip each of the assignments for the greedy variables with a small probability η (that corresponds to one of the extrema of the degree-k Chebyshev polynomials of the first kind).

Our algorithm for *triangle-free* instances begins by picking (F, G) to be a random partition of the variables. In this case, after fixing a random assignment to F, a natural greedy strategy would proceed as follows: Assign each variable in G a value that satisfies the maximum the number of its own active constraints.

In order to achieve objective (2), it is sufficient if for each inactive constraint its variables are assigned independently and uniformly at random. Since the instance is *triangle-free*, for every pair of variables $x_i, x_j \in G$ the active constraints of x_i and x_j are over disjoint sets of variables. This implies that the greedy assignments for variables within each inactive constraint are already independent. Unfortunately, the greedy assignment as defined above could possibly be biased, and in general much worse than a random assignment on the inactive constraints. We overcome this technical hurdle by using a modified greedy strategy defined as follows. Assign -1 to all variables in G and then for each variable $x_i \in G$, consider the change in the number of active constraints satisfied if we flip x_i from -1 to 1. The algorithm will flip the value only if this number exceeds an appropriately chosen threshold θ_i. The threshold θ_i is chosen so as to ensure that over all choices of values to F, the assignment to x_i is unbiased. Triangle-freeness implies that these assignments are independent within each inactive constraint. Putting these ideas together, we obtain the algorithm for triangle-free instances discussed in Section 5.

2 Preliminaries

Constraint satisfaction problems

We will be considering a somewhat general form of constraint satisfaction problems. An instance for us will consist of n Boolean variables and m constraints. We call the variables $x_1, \dots, x_n$, and we henceforth think of them as taking the Boolean values ± 1. Each constraint is a pair (P_ℓ, S_ℓ) (for $\ell \in [m]$) where $P_\ell : \{\pm 1\}^r \to \{0, 1\}$ is the *predicate*, and S_ℓ is the *scope*, an ordered r-tuple of distinct coordinates from $[n]$. The associated constraint is that $P_\ell(x_{S_\ell}) = 1$, where we use the notation x_S to denote variables x restricted to coordinates S. We always assume (without loss of generality) that P_ℓ depends on all r coordinates. The number r is called the *arity* of the constraint, and throughout this paper k will denote an upper bound on the arity of all constraints. Typically we think of k as a small constant.

We are also interested in the special case of Max-kXOR. By this we mean the case when all constraints are XORs of exactly k variables or their negations; in other words, when every P_ℓ is of the form $P_\ell(x_1, \dots, x_k) = \frac{1}{2} \pm \frac{1}{2} x_1 x_2 \cdots x_k$. When discussing Max-$k$XOR we will also always make the assumption that all scopes are distinct as sets; i.e., we don't have the same constraint or its negation more than once.

Hypergraph structure

We will be particularly interested in the *degree* $\deg(i)$ of each variable x_i in an instance. This is simply the number of constraints in which x_i participates; i.e., $\#\{\ell : S_\ell \ni i\}$. Throughout this work, we let D denote an upper bound on the degree of all variables.

For our second theorem, we will need to define the notion of "triangle-freeness".

▶ **Definition 3.** We say that an instance is *triangle-free* if the scopes of any two distinct constraints intersect on at most one variable ("no overlapping constraints") and, moreover, there are no three distinct constraints any two of whose scopes intersect ("no hyper-triangles"), see Figure 1.

Fourier representation

We recall that any Boolean function $f : \{\pm 1\}^n \to \mathbb{R}$ can be represented by a multilinear polynomial, or *Fourier expansion*,

$$f(x) = \sum_{S \subset [n]} \widehat{f}(S) x^S, \quad \text{where } x^S \stackrel{\text{def}}{=} \prod_{i \in S} x_i.$$

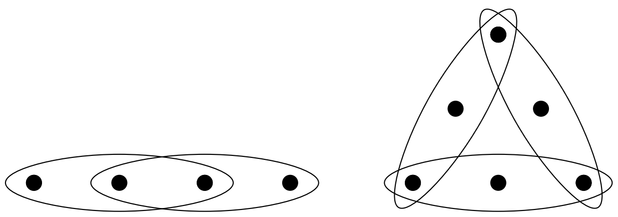


Figure 1 The two forbidden configurations for triangle-free instances.

For more details see, e.g., [11]; we recall here just a few facts we'll need. First, $\mathbb{E}[f(\boldsymbol{x})] = \widehat{f}(\emptyset)$. (Here and throughout we use **boldface** for random variables; furthermore, unless otherwise specified $\boldsymbol{x}$ refers to a uniformly random Boolean string.) Second, Parseval's identity is $\|f\|_2^2 = \mathbb{E}[f(\boldsymbol{x})^2] = \sum_S \widehat{f}(S)^2$, from which it follows that $\mathrm{Var}[f(\boldsymbol{x})] = \sum_{S\neq\emptyset} \widehat{f}(S)^2$. Third,

$$\mathrm{Inf}_i[f] = \sum_{S\ni i} \widehat{f}(S)^2 = \mathbb{E}[(\partial_i f)(\boldsymbol{x})^2],$$

where $\partial_i f$ is the *derivative* of f with respect to the ith coordinate. This can be defined by the factorization $f(x) = x_i \cdot (\partial_i f)(x') + g(x')$, where $x' = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$, or equivalently by $\partial_i f(x') = \frac{f(x',+1)-f(x',-1)}{2}$, where here (x', b) denotes $(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$. We record here a simple fact about these derivatives:

▶ **Lemma 4.** *For any predicate* $P : \{\pm1\}^r \to \{0,1\}$, $r \geq 2$, *we have* $\mathrm{Var}[(\partial_i P)(\boldsymbol{x})] \geq \Omega(2^{-r})$ *for all* i.

Proof. The function $\partial_i P(x)$ takes values in $\{-\frac{1}{2}, 0, \frac{1}{2}\}$. It cannot be constantly 0, since we assume P depends on its ith input. It also cannot be constantly $\frac{1}{2}$, else we would have $P(x) = \frac{1}{2} + \frac{1}{2}x_i$ and so P would not depend on all $r \geq 2$ coordinates. Similarly it cannot be constantly $-\frac{1}{2}$. Thus $\partial_i P(x)$ is nonconstant, so its variance is $\Omega(2^{-r})$. ◀

Given an instance and an assignment $x \in \{\pm1\}^n$, the number of constraints satisfied by the assignment is simply $\sum_\ell P_\ell(x_{S_\ell})$. This can be thought of as a multilinear polynomial $\{\pm1\}^n \to \mathbb{R}$ of degree[5] at most k. We would like to make two minor adjustments to it, for simplicity. First, we will normalize it by a factor of $\frac{1}{m}$ so as to obtain the *fraction* of satisfied constraints. Second, we will replace P_ℓ with $\overline{P}_\ell$, defined by

$$\overline{P}_\ell = P_\ell - \mathbb{E}[P_\ell] = P_\ell - \widehat{P_\ell}(\emptyset).$$

In this way, $\overline{P}_\ell(x_{S_\ell})$ represents the *advantage* over a random assignment. Thus given an instance, we define the *associated polynomial* $\mathfrak{P}(x)$ by

$$\mathfrak{P}(x) = \frac{1}{m}\sum_{\ell=1}^{m} \overline{P}_\ell(x_{S_\ell}).$$

This is a polynomial of degree at most k whose value on an assignment x represents the advantage obtained over a random assignment in terms of the fraction of constraints satisfied. In general, the algorithms in this paper are designed to find assignments $x \in \{\pm1\}^n$ with $\mathfrak{P}(x) \geq \Omega(\frac{1}{\sqrt{D}})$.

[5] We have the usual unfortunate terminology clash; here we mean degree as a polynomial.

Low-degree polynomials often achieve their expectation

Our proofs will frequently rely on the following fundamental fact from Fourier analysis, whose proof depends on the well-known "hypercontractive inequality". A proof of this fact appears in, e.g., [11, Theorem 9.24].

▶ **Fact 5.** *Let $f : \{\pm 1\}^n \to \mathbb{R}$ be a multilinear polynomial of degree at most k. Then $\mathbb{P}[f(\boldsymbol{x}) \geq \mathbb{E}[f]] \geq \frac{1}{4}\exp(-2k)$. In particular, by applying this to f^2, which has degree at most $2k$, we get*

$$\mathbb{P}\Big[|f(\boldsymbol{x})| \geq \|f\|_2\Big] \geq \exp(-O(k))$$

which implies that

$$\mathbb{E}\Big[|f(\boldsymbol{x})|\Big] \geq \exp(-O(k)) \cdot \|f\|_2 \geq \exp(-O(k)) \cdot \mathrm{stddev}[f(\boldsymbol{x})]\ .$$

3 A simple proof for Max-3XOR

We begin by proving Theorem 1 in the case of Max-3XOR, as the proof can be somewhat streamlined in this case. Given an instance of Max-3XOR we have the corresponding polynomial

$$\mathfrak{P}(x) = \sum_{|S|=3} \widehat{\mathfrak{P}}(S)x^S = \sum_{i,j,k\in[n]} a_{ijk}x_ix_jx_k,$$

where $\widehat{\mathfrak{P}}(S) \in \{\pm\frac{1}{2m}, 0\}$ depending on whether the corresponding constraint exists in the instance, and where we have introduced $a_{ijk} = \frac{1}{6}\widehat{\mathfrak{P}}(\{i,j,k\})$ for $i,j,k \in [n]$ distinct. We now use the trick of "decoupling" the first coordinate (cf. [9, Lem. 2.1]); i.e., our algorithm will consider $\widetilde{\mathfrak{P}}(y,z) = \sum_{i,j,k} a_{ijk}y_iz_jz_k$, where $y_1,\dots,y_n,z_1,\dots,z_n$ are new variables. The algorithm will ultimately produce a good assignment $y,z \in \{\pm1\}^n$ for $\widetilde{\mathfrak{P}}$. Then it will define an assignment $\boldsymbol{x} \in \{\pm1\}^n$ by using one of three "randomized rounding" schemes:

$$\text{w.p. } \tfrac{4}{9},\ \boldsymbol{x}_i = \begin{cases} y_i & \text{w.p. } \frac{1}{2} \\ z_i & \text{w.p. } \frac{1}{2}\end{cases} \forall i; \qquad \text{w.p. } \tfrac{4}{9},\ \boldsymbol{x}_i = \begin{cases} y_i & \text{w.p. } \frac{1}{2} \\ -z_i & \text{w.p. } \frac{1}{2}\end{cases} \forall i; \qquad \text{w.p. } \tfrac{1}{9},\ \boldsymbol{x}_i = -y_i\ \forall i.$$

We have that

$$\begin{aligned}\mathbb{E}[\mathfrak{P}(\boldsymbol{x})] &= \tfrac{4}{9}\sum_{i,j,k} a_{ijk}(\tfrac{y_i+z_i}{2})(\tfrac{y_j+z_j}{2})(\tfrac{y_k+z_k}{2}) + \tfrac{4}{9}\sum_{i,j,k} a_{ijk}(\tfrac{y_i-z_i}{2})(\tfrac{y_j-z_j}{2})(\tfrac{y_k-z_k}{2}) \\ &\qquad\qquad + \tfrac{1}{9}\sum_{i,j,k} a_{ijk}(-y_i)(-y_j)(-y_k) \\ &= \tfrac{1}{9}\sum_{i,j,k} a_{ijk}(y_iz_jz_k + z_iy_jz_k + z_iz_jy_k) = \frac{1}{3}\widetilde{\mathfrak{P}}(y,z). \end{aligned} \tag{3}$$

Thus in expectation, the algorithm obtains an assignment for $\mathfrak{P}$ achieving at least $\frac{1}{3}$ of what it achieves for $\widetilde{\mathfrak{P}}$.

Let us now write $\widetilde{\mathfrak{P}}(y,z) = \sum_i y_iG_i(z)$, where $G_i(z) = \sum_{j,k} a_{ijk}z_jz_k$. It suffices for the algorithm to find an assignment for z such that $\sum_i |G_i(z)|$ is large, as it can then achieve this quantity by taking $y_i = \mathrm{sgn}(G_i(z))$. The algorithm simply chooses $\boldsymbol{z} \in \{\pm1\}^n$ uniformly at random. By Parseval we have $\mathbb{E}[G_i(\boldsymbol{z})^2] = \sum_{j<k}(2a_{ijk})^2 = \frac{1}{9}\,\mathrm{Inf}_i[\mathfrak{P}]$ for each i. Applying

Fact 5 (with $k = 2$) we therefore get $\mathbb{E}[|G_i(\boldsymbol{z})|] \geq \Omega(1) \cdot \sqrt{\mathrm{Inf}_i[\mathfrak{P}]}$. Since $\mathrm{Inf}_i[\mathfrak{P}] = \deg(i)/4m^2$, we conclude

$$\mathbb{E}\left[\sum_i |G_i(\boldsymbol{z})|\right] \geq \Omega(1) \cdot \sum_i \frac{\sqrt{\deg(i)}}{m} \geq \Omega(1) \cdot \sum_i \frac{\deg(i)}{m\sqrt{D}} = \Omega(1) \cdot \frac{1}{\sqrt{D}}.$$

As $\sum_i |G_i(\boldsymbol{z})|$ is bounded by $1/2$, Markov's inequality implies that the algorithm can with high probability find a z achieving $\sum_i |G_i(z)| \geq \Omega(\frac{1}{\sqrt{D}})$ after $O(\sqrt{D})$ trials of $\boldsymbol{z}$. As stated, the algorithm then chooses y appropriately to attain $\widetilde{\mathfrak{P}}(y, z) \geq \Omega(\frac{1}{\sqrt{D}})$, and finally gets $\frac{1}{3}$ of this value (in expectation) for $\mathfrak{P}(x)$.

Derandomization

It is easy to efficiently derandomize the above algorithm. The main step is to recognize that "$(2, 4)$-hypercontractivity" is all that's needed for Fact 5 (perhaps with a worse constant); thus it holds even when the random bits are merely 4-wise independent. This is well known, but we could not find an explicit reference; hence we give the proof in the case when f is homogeneous of degree 2 (the case that's needed in the above algorithm). Without loss of generality we may assume $\mathbb{E}[f(\boldsymbol{x})] = 0$ and $\mathbb{E}[f(\boldsymbol{x})^2] = 1$. Then it's a simple exercise to check that $\mathbb{E}[f(\boldsymbol{x})^4] \leq 15$, and this only requires the bits of $\boldsymbol{x}$ to be 4-wise independent. But now

$$\mathbb{P}[f(\boldsymbol{x}) \geq 0] = \mathbb{E}[1_{\{f(\boldsymbol{x}) \geq 0\}}] \geq \mathbb{E}[.13 f(\boldsymbol{x}) + .06 f(\boldsymbol{x})^2 - .002 f(\boldsymbol{x})^4] \geq .06 - .002 \cdot 15 = .03$$

where we used the elementary fact $1_{\{t \geq 0\}} \geq .13t + .06t^2 - .002t^4$ for all $t \in \mathbb{R}$. Thus indeed the algorithm can find a z achieving $\sum_i |G_i(z)| \geq \Omega(\frac{1}{\sqrt{D}})$ by enumerating all strings in a 4-wise independent set; it is well known this can be done in polynomial time. Following this, the algorithm chooses string y deterministically. Finally, it is clear that each of the three different randomized rounding schemes only requires 3-wise independence, and a deterministic algorithm can simply try all three and choose the best one.

4 A general result for bounded-influence functions

One can obtain our Theorem 1 for higher odd k by generalizing the proof in the preceding section. Constructing the appropriate "randomized rounding" scheme to decouple the first variable becomes slightly more tricky, but one can obtain the identity analogous to (3) through the use of Chebyshev polynomials. At this point the solution becomes very reminiscent of the Dinur et al. [3] work. Hence in this section we will simply directly describe how one can make [3] algorithmic.

The main goal of [3] was to understand the "Fourier tails" of bounded degree-k polynomials. One of their key technical results was the following theorem, showing that if a degree-k polynomial has all of its influences small, it must deviate significantly from its mean with noticeable probability:

▶ **Theorem 6.** *([3, Theorem 3].) There is a universal constant C such that the following holds. Suppose $g : \{\pm 1\}^n \to \mathbb{R}$ is a polynomial of degree at most k and assume* $\mathrm{Var}[g] = 1$. *Let $t \geq 1$ and suppose that* $\mathrm{Inf}_i[g] \leq C^{-k} t^{-2}$ *for all $i \in [n]$. Then*

$$\mathbb{P}[|g(\boldsymbol{x})| \geq t] \geq \exp(-Ct^2 k^2 \log k).$$

In the context of Max-kXOR, this theorem already nearly proves our Theorem 1. The reason is that in this context, the associated polynomial $\mathfrak{P}(x)$ is given by

$$\mathfrak{P}(x) = \frac{1}{2m} \sum_{\ell=1}^{m} b_\ell \prod_{j \in S_\ell} x_j, \quad \text{where } b_\ell \in \{-1, 1\}.$$

Hence $\mathrm{Var}[\mathfrak{P}] = 1/4m$ and $\mathrm{Inf}_i[\mathfrak{P}] = \deg(x_i)/4m^2 \le D/4m^2$. Taking $g = 2\sqrt{m} \cdot \mathfrak{P}$ and $t = \exp(-O(k)) \cdot \sqrt{m/D}$, Theorem 6 immediately implies that

$$\mathbb{P}\Big[|\mathfrak{P}(\boldsymbol{x})| \ge \exp(-O(k)) \cdot \frac{1}{\sqrt{D}}\Big] \ge \exp(-O(m/D)). \tag{4}$$

This already shows the desired existential result, that there *exists* an assignment beating the random assignment by $\exp(-O(k)) \cdot \frac{1}{\sqrt{D}}$. The only difficulty is that the low probability bound in (4) does not imply we can find such an assignment efficiently.

However this difficulty really only arises because [3] had different goals. In their work, it was essential to show that g achieves a slightly large value on a completely *random* input.[6] By contrast, we are at liberty to show g achieves a large value however we like — semi-randomly, greedily — so long as our method is algorithmic. That is precisely what we do in this section of the paper. Indeed, in order to "constructivize" [3], the only fundamental adjustment we need to make is at the beginning of the proof of their Lemma 1.3: when they argue that "$\mathbb{P}[|\ell(\boldsymbol{x})| \ge t'] \ge \exp(-O(t'^2))$ for the degree-1 polynomial $\ell(x)$", we can simply greedily choose an assignment x with $|\ell(x)| \ge t'$.

Our constructive version of Theorem 6 follows. It directly implies our Theorem 1, as described above.

▶ **Theorem 7.** *There is a universal constant C and a randomized algorithm such that the following holds. Let $g : \{\pm 1\}^n \to \mathbb{R}$ be a polynomial with degree at most k and $\mathrm{Var}[g] = 1$ be given. Let $t \ge 1$ and suppose that $\mathrm{Inf}_i[g] \le C^{-k} t^{-2}$ for all $i \in [n]$. Then with high probability the algorithm outputs an assignment x with $|g(x)| \ge t$. The running time of the algorithm is $\mathrm{poly}(m, n, \exp(k))$, where m is the number of nonzero monomials in g.*[7]

The algorithm AdvRand achieving Theorem 7 is given below. It is derived directly from [3], and succeeds with probability that is inverse polynomial in n. The success probability is then boosted by running the algorithm multiple times. We remark that $\eta_0^{(k)}, \eta_1^{(k)}, \ldots, \eta_k^{(k)}$ denote the $k+1$ extrema in $[-1,1]$ of the kth Chebyshev polynomial of the first kind $T_k(x)$, and are given by $\eta_j^{(k)} = \cos(j\pi/k)$ for $0 \le j \le k$. We now describe the algorithm below, for completeness. In the rest of the section, we will assume without loss of generality that k is odd (for even k, we just think of the polynomial as being of degree $k+1$, with the degree $(k+1)$ part being 0).

AdvRand: Algorithm for Advantage over Average for degree k polynomials

Input: a degree k-function g

Output: an assignment x

```
1. Let $1 \le s \le \log_2 k$ be a scale such that the mass (i.e., sum of squares of coefficients) of
   the Fourier transform of $g$ on levels between $2^{s-1}$ and $2^s$ is at least $1/\log k$.
2. For every $i \in [n]$, put $i$ in set $U$ with probability $2^{-s}$. For every $i \notin U$, set $x_i \in \{-1,1\}$
   uniformly at random and let $y$ be the assignment restricted to the variables in $[n] \setminus U$.
3. Let $g_y$ be the restriction obtained. For every $j \in U$, set $x_j = \mathrm{sign}(\widehat{g_y}(\{j\}))$.
4. Pick $r \in \{0,1,\ldots,k\}$ uniformly at random, and let $\eta = \eta_r^{(k)}/2$.
5. For each coordinate $j \in U$, flip $x_j$ independently at random with probability $(1-\eta)/2$.
6. Output $x$.
```

[6] Also, their efforts were exclusively focused on the parameter k, with quantitative dependencies on t not mattering. Our focus is essentially the opposite.

[7] For simplicity in our algorithm, we assume that exact real arithmetic can be performed efficiently.

We now give the analysis of the algorithm, following [3]. The second step of the algorithm performs a *random restriction*, that ensures that g_y has a lot of mass on the first-order Fourier coefficients. The key lemma (that follows from the proof of Lemma 1.3 and Lemma 4.1 in [3]) shows that we can find an assignment that obtains a large value for a polynomial with sufficient "smeared" mass on the first-order Fourier coefficients.

▶ **Lemma 8.** *Suppose $g : \{\pm 1\}^N \to \mathbb{R}$ has degree at most k, $t \geq 1$, and $\sum_{i\in[N]} |\widehat{g}(\{i\})| \geq 2t(k+1)$. Then a randomized polynomial time algorithm outputs a distribution over assignments $\boldsymbol{x} \in \{-1,1\}^N$ such that*

$$\mathop{\mathbb{P}}_{\boldsymbol{x}}[|g(\boldsymbol{x})| \geq t] \geq \exp(-O(k)).$$

The algorithm proving Lemma 8 corresponds to Steps (3-6) of the Algorithm AdvRand.

Proof. We sketch the proof, highlighting the differences to Lemma 1.3 of [3]. First we observe that by picking the assignment $x_i^* = \operatorname{sign}(\widehat{g}(\{i\}))$, we can maximize the linear portion as

$$\sum_{i\in[N]} \widehat{g}(\{i\}) x_i^* = \sum_{i\in[N]} |\widehat{g}(\{i\})| \geq 2t(k+1).$$

From this point on, we follow the proof of Lemma 1.3 in [3] with their initial point x_0 being set to x^*. Let $\boldsymbol{z} \leftarrow_\eta \{\pm 1\}^N$ be a random string generated by independently setting each coordinate $\boldsymbol{z}_j = -1$ with probability $(1-\eta)/2$ (as in step 5 of the algorithm), and let

$$(T_\eta g)(x^*) = \mathop{\mathbb{E}}_{\boldsymbol{z} \leftarrow_\eta \{\pm 1\}^n} [g(x^* \cdot \boldsymbol{z})].$$

Lemma 1.3 of [3], by considering $(T_\eta g)(x^*)$ as a polynomial in η and using the extremal properties of Chebyshev polynomials (Corollary 2.8 in [3]), shows that there exists $\eta \in \{\frac{\eta_0^{(k)}}{2}, \frac{\eta_1^{(k)}}{2}, \ldots, \frac{\eta_k^{(k)}}{2}\}$ such that

$$\mathop{\mathbb{E}}_{\boldsymbol{z} \leftarrow_\eta \{\pm 1\}^n} \Big[|g(x^* \cdot \boldsymbol{z})| \Big] \geq 2t(k+1) \cdot \frac{1}{(2k+2)} = t. \tag{5}$$

Consider $g(x^* \cdot \boldsymbol{z})$ as a polynomial in $\boldsymbol{z}$, with degree at most k. As in [3], we will now use the hypercontractivity to give a lower bound on the probability (over random $\boldsymbol{z}$) that $|g(x^* \cdot \boldsymbol{z})|$ exceeds the expectation. Note that our choice of $\eta \in [-\frac{1}{2}, \frac{1}{2}]$ and hence the bias is in the interval $[\frac{1}{4}, \frac{3}{4}]$. Using Lemma 2.5 in [3] (the analogue of Fact 5 for biased measures), it follows that

$$\mathop{\mathbb{P}}_{\boldsymbol{z}} \Big[|g(x^* \cdot \boldsymbol{z})| \geq t \Big] \geq \tfrac{1}{4} \exp(-2k).$$

Hence when $\boldsymbol{x}$ is picked according to $\mathcal{D}$, with probability at least $1/(k+1)$ the algorithm chooses an η such that (5) holds, and then a random $\boldsymbol{z}$ succeeds with probability $\exp(-O(k))$, thereby giving the required success probability. ◀

We now sketch the proof of the constructive version of Theorem 3 in [3], highlighting why algorithm AdvRand works.

Proof of Theorem 7. The scale s is chosen such that the Fourier coefficients of g of order $[2^{s-1}, 2^s]$ have mass at least $1/\log k$. The algorithm picks set U randomly by choosing each variable with probability 2^{-s}, and g_y is the restriction of g to the coordinates in U obtained by setting the other variables randomly to $\boldsymbol{y} \in \{-1,1\}^{[N]\setminus U}$.

Let $\gamma_i = \sum_{S \cap U = \{i\}} \widehat{g}(S)^2$. Fixing U and y, let the indices $T = \{i \in U : \widehat{g}_y(\{i\})^2 \leq (2e)^{2k}\gamma_i\}$. The proof of Theorem 3 in [3] shows that a constant fraction of the first order Fourier coefficients are large; in particular after Steps 1 and 2 of the algorithm,

$$\mathop{\mathbb{P}}_{U,\boldsymbol{y}}\left[\sum_{i \in T} \widehat{g}_{\boldsymbol{y}}(\{i\})^2 \geq \frac{1}{100 \log k}\right] \geq \exp(-O(k)) \ . \tag{6}$$

Further, for $i \in T$, we have $|\widehat{g}_y(\{i\})| \leq (2e)^k \sqrt{\gamma_i} \leq (2e)^k \sqrt{\mathrm{Inf}_i(g)}$. Hence, when the above event in (6) is satisfied we have

$$\begin{aligned}\sum_{i \in U} |\widehat{g}_y(\{i\})| &\geq \frac{1}{\max_{i \in T} |\widehat{g}_y(\{i\})|} \cdot \sum_{i \in T} \widehat{g}_y(\{i\})^2 \\ &\geq \frac{1}{(2e)^k \sqrt{\max_i \mathrm{Inf}_i(g)}} \cdot \frac{1}{100 \log k} \geq 2t(k+1).\end{aligned}$$

Hence, applying Lemma 8 with g_y we get that

$$\mathop{\mathbb{P}}_{\boldsymbol{x} \in \mathcal{D}}\Big[|g(x)| \geq t\Big] \geq \exp(-O(k)), \tag{7}$$

where $\mathcal{D}$ is the distribution over assignments x output by the algorithm. Repeating this algorithm $\exp(O(k))$ times, we get the required high probability of success. ◀

5 Triangle-free instances

In this section we present the proof of Theorem 2, which gives an efficient algorithm for beating the random assignment in the case of arbitrary triangle-free CSPs (recall Definition 3). We now restate Theorem 2 and give its proof. As in the proof of Theorem 7, we can easily move from an expectation guarantee to a high probability guarantee by first applying Markov's inequality, and then repeating the algorithm $\exp(k)\,\mathrm{poly}(n, m)$ times; hence we will prove the expectation guarantee here.

▶ **Theorem 9.** *There is a* $\mathrm{poly}(m, n, \exp(k))$*-time randomized algorithm with the following guarantee. Let the input be a triangle-free instance over n Boolean variables, with m arbitrary constraints each of arity between 2 and k. Assume that each variable participates in at most D constraints. Let the associated polynomial be $\mathfrak{P}(x)$. Then the algorithm outputs an assignment $\boldsymbol{x} \in \{\pm 1\}^n$ with*

$$\mathbb{E}[\mathfrak{P}(\boldsymbol{x})] \geq \exp(-O(k)) \cdot \sum_{i=1}^{n} \frac{\sqrt{\deg(i)}}{m} \geq \exp(-O(k)) \cdot \frac{1}{\sqrt{D}}.$$

Proof. Let (F, G) be a partition of $[n]$, with F standing for "Fixed" and G standing for "Greedy". Eventually the algorithm will choose the partition randomly, but for now we treat it as fixed. We will write the two parts of the algorithm's random assignment $\boldsymbol{x}$ as $(\boldsymbol{x}_F, \boldsymbol{x}_G)$. The bits $\boldsymbol{x}_F$ will first be chosen independently and uniformly at random. Then the bits $\boldsymbol{x}_G$ will be chosen in a careful way which will make them uniformly random, but not completely independent.

To make this more precise, define a constraint (P_ℓ, S_ℓ) to be *active* if its scope S_ℓ contains exactly one coordinate from G. Let us partition these active constraints into groups

$$N_j = \{\ell : (P_\ell, S_\ell) \text{ is active and } S_\ell \ni j\}, \quad j \in G.$$

For each coordinate $j \in G$, we'll define $A_j \subset F$ to be the union of all active scopes involving j (but excluding j itself); i.e.,

$$A_j = \bigcup \{S_\ell \setminus \{j\} : \ell \in N_j\}.$$

This set A_j may be empty. Our algorithm's choice of $\boldsymbol{x}_G$ will have the following property:

$\forall j \in G$, *the distribution of* $\boldsymbol{x}_j$ *is uniformly random, and depends only on* $(\boldsymbol{x}_i : i \in A_j)$. (†)

From property (†) we may derive:

▶ **Claim 9.1.** *For every* inactive *constraint* (P_ℓ, S_ℓ)*, the random assignment bits* $\boldsymbol{x}_{S_\ell}$ *are uniform and* independent*.*

Proof of Claim. First consider the coordinates $j \in S_\ell \cap G$. By the property (†), each such $\boldsymbol{x}_j$ depends only on $(\boldsymbol{x}_i : i \in A_j)$; further, these sets A_j are disjoint precisely because of the "no hyper-triangles" part of triangle-freeness. Thus indeed the bits $(\boldsymbol{x}_j : j \in S_\ell \cap G)$ are uniform and mutually independent. The remaining coordinates $S_\ell \cap F$ are also disjoint from all these $(A_j)_{j \in S_\ell \cap G}$, by the "no overlapping constraints" part of the triangle-free property. Thus the remaining bits $(\boldsymbol{x}_i : i \in S_\ell \cap F)$ are uniform, independent, and independent of the bits $(\boldsymbol{x}_j : j \in S_\ell \cap G)$, completing the proof of the claim. ◀

An immediate corollary of the claim is that all inactive constraints, $\overline{P}_\ell$ contribute nothing, in expectation, to $\mathbb{E}[\mathfrak{P}(\boldsymbol{x})]$. Thus it suffices to consider the contribution of the active constraints. Our main goal will be to show that the bits $\boldsymbol{x}_G$ can be chosen in such a way that

$$\forall j \in G \quad \mathbb{E}\Big[\sum_{\ell \in N_j} \overline{P}_\ell(\boldsymbol{x}_{S_\ell})\Big] \geq \exp(-O(k)) \cdot \sqrt{|N_j|} \tag{8}$$

and hence

$$\mathbb{E}[\mathfrak{P}(\boldsymbol{x})] \geq \frac{1}{m} \cdot \exp(-O(k)) \cdot \sum_{j \in G} \sqrt{|N_j|}. \tag{9}$$

Given (9) it will be easy to complete the proof of the theorem by choosing the partition (F, G) randomly.

So towards showing (8), fix any $j \in G$. For each $\ell \in N_j$ we can write $\overline{P}_\ell(x_{S_\ell}) = x_j Q_\ell(x_{S_\ell \setminus \{j\}}) + R_\ell(x_{S_\ell \setminus \{j\}})$, where $Q_\ell = \partial_j \overline{P}_\ell = \partial_j P_\ell$. Since the bits $\boldsymbol{x}_i$ for $i \in S_\ell \setminus \{j\} \subset F$ are chosen uniformly and independently, the expected contribution to (8) from the R_ℓ polynomials is 0. Thus we just need to establish

$$\mathbb{E}\Big[\boldsymbol{x}_j \cdot \sum_{\ell \in N_j} \boldsymbol{Q}_\ell\Big] \geq \exp(-O(k)) \cdot \sqrt{|N_j|}, \qquad \text{where } \boldsymbol{Q}_\ell \overset{\text{def}}{=} Q_\ell(\boldsymbol{x}_{S_\ell \setminus \{j\}}). \tag{10}$$

We now finally describe how the algorithm chooses the random bit $\boldsymbol{x}_j$. Naturally, we will choose it to be $+1$ when $\sum_{\ell \in N_j} \boldsymbol{Q}_\ell$ is "large" and -1 otherwise. Doing this satisfies the second aspect of property (†), that $\boldsymbol{x}_j$ should depend only on $(\boldsymbol{x}_i : i \in A_j)$. To satisfy the first aspect of property (†), that $\boldsymbol{x}_j$ is equally likely ± 1, we are essentially forced to define

$$\boldsymbol{x}_j = \operatorname{sgn}\Big(\sum_{\ell \in N_j} \boldsymbol{Q}_\ell - \theta_j\Big), \tag{11}$$

where θ_j is defined to be a *median* of the random variable $\sum_{\ell \in N_j} \boldsymbol{Q}_\ell$.

(Actually, we have to be a little careful about this definition. For one thing, if the median θ_j is sometimes achieved by the random variable, we would have to carefully define sgn(0) to be sometimes $+1$ and sometimes -1 so that $\boldsymbol{x}_j$ is equally likely ± 1. For another thing, we are assuming here that the algorithm can efficiently *compute* the medians θ_j. We will describe how to handle these issues in a technical remark after the proof.)

Having described the definition (11) of $\boldsymbol{x}_j$ satisfying property (†), it remains to verify the inequality (10). Notice that by the "no overlapping constraints" aspect of triangle-freeness, the random variables $\boldsymbol{Q}_\ell$ are actually mutually independent. Further, Lemma 4 implies that each has variance $\Omega(2^{-k})$; hence the variance of $\boldsymbol{Q} \overset{\text{def}}{=} \sum_{\ell \in N_j} \boldsymbol{Q}_\ell$ is $\exp(-O(k)) \cdot |N_j|$. Thus inequality (10) is equivalent to

$$\mathbb{E}[\operatorname{sgn}(\boldsymbol{Q} - \theta_j)\boldsymbol{Q}] \geq \exp(-O(k)) \cdot \operatorname{stddev}[\boldsymbol{Q}] = \exp(-O(k)) \cdot \operatorname{stddev}[\boldsymbol{Q} - \theta_j].$$

Now

$$\mathbb{E}[\operatorname{sgn}(\boldsymbol{Q} - \theta_j)\boldsymbol{Q}] = \mathbb{E}[\operatorname{sgn}(\boldsymbol{Q} - \theta_j)(\boldsymbol{Q} - \theta_j + \theta_j)] = \mathbb{E}[|\boldsymbol{Q} - \theta_j|] + \mathbb{E}[\boldsymbol{x}_j \cdot \theta_j]. \tag{12}$$

We have $\mathbb{E}[\boldsymbol{x}_j \cdot \theta_j] = 0$ since $\mathbb{E}[\boldsymbol{x}_j] = 0$. And as for $\mathbb{E}[|\boldsymbol{Q} - \theta_j|]$, it is indeed at least $\exp(-O(k)) \cdot \operatorname{stddev}[\boldsymbol{Q}]$ by Fact 5, since $\boldsymbol{Q}$ is a degree-$(k-1)$ function of uniform and independent random bits. Thus we have finally established (8), and therefore (9).

To conclude, we analyze what happens when the algorithm initially chooses a uniformly *random* partition $(\boldsymbol{F}, \boldsymbol{G})$ of $[n]$. In light of (9), it suffices to show that for each $i \in [n]$ we have

$$\mathbb{E}\Big[\mathbf{1}[i \in \boldsymbol{G}] \cdot \sqrt{|\boldsymbol{N}_i|}\Big] \geq \exp(-O(k)) \cdot \sqrt{\deg(i)}. \tag{13}$$

We have $\mathbb{P}[i \in \boldsymbol{G}] = \frac{1}{2}$; conditioning on this event, let us consider the random variable $|\boldsymbol{N}_i|$; i.e., the number of active constraints involving variable x_i. A constraint scope S_ℓ containing i becomes active if and only if all the other indices in S_ℓ go into $\boldsymbol{F}$, an event that occurs with probability 2^{-k+1} (at least). Furthermore, these events are independent across the scopes containing i because of the "no overlapping constraints" property of triangle-freeness. Thus (conditioned on $i \in \boldsymbol{G}$), each random variable $|\boldsymbol{N}_i|$ is the sum $\boldsymbol{A}_1 + \cdots + \boldsymbol{A}_{\deg(i)}$ independent indicator random variables, each with expectation at least 2^{-k+1}. Thus we indeed have $\mathbb{E}[\sqrt{|\boldsymbol{N}_i|}] \geq \exp(-O(k))\sqrt{\deg(i)}$ as needed to complete the proof of (13). This follows from the well known fact that $\mathbb{E}[\sqrt{\text{Binomial}(d,p)}] \geq \Omega(\min(\sqrt{dp}, dp))$. (Alternatively, this follows from the fact that $\boldsymbol{A}_1 + \cdots + \boldsymbol{A}_{d_i}$ is at least its expectation $d_i 2^{-k+1}$ with probability at least $\exp(-O(k))$, by Fact 5. Here we would use that the $\boldsymbol{A}_j$'s are degree-$(k-1)$ functions of independent random bits defining $(\boldsymbol{F}, \boldsymbol{G})$). The proof is complete. ◀

▶ Remark. Regarding the issue of algorithmically obtaining the medians in the above proof: In fact, we claim it is unnecessary for the algorithm to compute the median θ_j of each $\boldsymbol{Q}_j$ precisely. Instead, our algorithm will (with high probability) compute a number $\widetilde{\theta}_j$ and a probabilistic way of defining $\operatorname{sgn}(0) \in \{\pm 1\}$ such that, when $\boldsymbol{x}_j$ is defined to be $\operatorname{sgn}(\boldsymbol{Q} - \widetilde{\theta}_j)$, we have $|\mathbb{E}[\boldsymbol{x}_j]| \leq \delta$, where $\delta = 1/\operatorname{poly}(m, n, \exp(k))$ is sufficiently small. First, let us briefly say why this is sufficient. The above proof relied on $\mathbb{E}[\boldsymbol{x}_j] = 0$ in two places. One place was in the last term of (12), where we used $\mathbb{E}[\boldsymbol{x}_j \cdot \theta_j] = 0$. Now in the approximate case, we'll have $|\,\mathbb{E}[\boldsymbol{x}_j \cdot \widetilde{\theta}_j]| \leq \delta m$, and by taking δ appropriately small this will contribute negligibly to the overall theorem. The other place that $\mathbb{E}[\boldsymbol{x}_j] = 0$ was used was in deducing from Claim 9.1, that the inactive constraints contributed nothing to the algorithm's expected value. When we merely have $|\mathbb{E}[\boldsymbol{x}_j]| \leq \delta$ (but still have the independence used in the claim), it's easy to see from Fourier considerations that each inactive constraint still contributes at most $2^k\delta$ to the overall expectation, and again this is negligible for the theorem as a whole if $\delta = 1/\operatorname{poly}(m, n, \exp(k))$ is sufficiently small. Finally, it is not hard to show that the algorithm can compute an appropriate $\widetilde{\theta}_j$ and probabilistic definition of sgn(0) in $\operatorname{poly}(m, n, \exp(k))$ time (with high probability), just by sampling to find a good approximate median $\widetilde{\theta}_j$ and then also estimating $\mathbb{P}[\boldsymbol{Q}_j = \widetilde{\theta}_j]$ to handle the definition of sgn(0).

Acknowledgments. We thank Scott Aaronson for bringing the paper of Farhi et al. [4] to (some of) the authors' attention. RO, DW, and JW were supported by NSF grants CCF-0747250 and CCF-1116594. DW was also supported by the NSF Graduate Research Fellowship Program under grant DGE-1252522; JW was also supported by a Simons Graduate Fellowship. OR, DS, and AV acknowledge the support of the Simons Collaboration on Algorithms and Geometry. OR was also supported by NSF grant CCF-1320188. DS was also supported by a Sloan fellowship, a Microsoft Research Faculty Fellowship, and by the NSF. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

1 Noga Alon. On the edge-expansion of graphs. *Combin. Probab. Comput.*, 6(2):145–152, 1997.

2 Bonnie Berger and Peter W. Shor. Approximation alogorithms for the maximum acyclic subgraph problem. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'90, pages 236–243, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

3 Irit Dinur, Ehud Friedgut, Guy Kindler, and Ryan O'Donnell. On the Fourier tails of bounded functions over the discrete cube. *Israel J. Math.*, 160:389–412, 2007.

4 Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem, 2014. arXiv:1412.6062.

5 Venkatesan Guruswami and Yuan Zhou. Approximating bounded occurrence ordering csps. In Anupam Gupta, Klaus Jansen, José Rolim, and Rocco Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 7408 of *Lecture Notes in Computer Science*, pages 158–169. Springer Berlin Heidelberg, 2012.

6 Johan Håstad. On bounded occurrence constraint satisfaction. *Inform. Process. Lett.*, 74(1-2):1–6, 2000.

7 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

8 Johan Håstad and S. Venkatesh. On the advantage over a random assignment. *Random Structures Algorithms*, 25(2):117–149, 2004.

9 Subhash Khot and Assaf Naor. Linear equations modulo 2 and the L_1 diameter of convex bodies. *SIAM J. Comput.*, 38(4):1448–1463, 2008.

10 Konstantin Makarychev. Local search is better than random assignment for bounded occurrence ordering k-csps. In *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 – March 2, 2013, Kiel, Germany*, pages 139–147, 2013.

11 Ryan O'Donnell. *Analysis of Boolean Functions.* Cambridge University Press, 2014.

12 James B. Shearer. A note on bipartite subgraphs of triangle-free graphs. *Random Structures Algorithms*, 3(2):223–226, 1992.

Improved Bounds in Stochastic Matching and Optimization

Alok Baveja*[1], Amit Chavan[2], Andrei Nikiforov[1], Aravind Srinivasan†[2], and Pan Xu‡[2]

1 **School of Business, Rutgers, The State University of New Jersey, Camden, NJ 08102, USA**
`{baveja@crab|andnikif@camden}.rutgers.edu`

2 **Department of Computer Science, University of Maryland, College Park, MD 20742, USA**
`{amitc|srin|panxu}@cs.umd.edu`

Abstract

We consider two fundamental problems in stochastic optimization: approximation algorithms for stochastic matching, and sampling bounds in the black-box model. For the former, we improve the current-best bound of 3.709 due to Adamczyk, Grandoni, and Mukherjee [1], to 3.224; we also present improvements on Bansal, Gupta, Li, Mestre, Nagarajan, and Rudra [2] for hypergraph matching and for relaxed versions of the problem. In the context of stochastic optimization, we improve upon the sampling bounds of Charikar, Chekuri, and Pál [3].

1998 ACM Subject Classification F.2.2 Computations on discrete structures

Keywords and phrases stochastic matching, approximation algorithms, sampling complexity

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.124

1 Introduction

Stochastic optimization deals with problems where there is uncertainty inherent in the input [14]; this classical sub-area of optimization has received much attention in computer science over the last decade, especially from the viewpoints of approximation algorithms and of (efficiently) handling various models for the input (see, e.g., [3, 5, 10, 11, 12, 13, 15, 16]). We make progress on two basic problems in this regard. First, matching is well-known to be a bedrock of combinatorial optimization – a problem that has also played a key role in the advancement of new algorithmic paradigms including parallel algorithms, randomized algorithms, and, more recently, online algorithms in sponsored-search advertising. However, we do not yet have a full algorithmic understanding even for various basic *stochastic* versions of the problem. We advance this goal by improving upon the bounds of [2] and [1] for the matching problem in graphs and in uniform hypergraphs. Second, a fundamental model in this field is the *black-box* model: we assume that the input-distribution is represented by a black box, from which we can sample inputs independently any number of times (in addition to other assumptions on the input's structure). Thus, a key question is the number of samples

* Supported in part by grants from (a) United States Department of Transportation (Via the University Transportation Research Center Program) – Grant# 49198-25-26 and (b) British Council's UKIERI research program.
† Supported in part by NSF Awards CNS 1010789 and CCF 1422569 and a research award from Adobe, Inc.
‡ Supported in part by NSF Awards CNS 1010789 and CCF 1422569.

needed to solve various stochastic-optimization problems in this model, as a function of, e.g., the desired accuracy ϵ and the confidence (probability of successfully estimating to within ϵ).

Informally, the basic stochastic-matching problem is as follows [2, 4]. We are given a graph $G = (V, E)$ with a weight $w_e \geq 0$ and a probability $p_e \in [0, 1]$ for each edge e; each vertex v also has a positive integral "patience" t_v. Our goal is to construct a matching of maximum weight; however, there are a few catches. First, the edges are only present probabilistically: each edge e is present *independently* with probability p_e, and the presence (or lack thereof) of any edge e can only be ascertained by probing for it – adaptively, in any order we choose. However, if we choose to probe $e = (u, v)$ and find that it is present, we are forced to add it to our matching: in particular, all edges incident on e are removed immediately if e is found to be present. Furthermore, the edges incident upon any vertex v can only be probed for up to t_v times; i.e., we cannot exceed the hard constraint of the patience of any vertex. Under these constraints, the goal is to find a matching of maximum expected weight, where the expectation is taken both over the stochastic existence of the edges, and over any internal randomization of our algorithm. Intriguingly, it is not yet known if this problem is NP-hard. The state of the art in terms of approximation is mainly from the work of [2], who present a 3–approximation for bipartite graphs, and a 4–approximation for general graphs. Recently, these bounds have been improved to 2.845 and 3.709 for bipartite and general graphs respectively [1]. We present the following two improvements for the general graphs, with Theorem 2 being a bicriteria result that allows the patience constraints to be violated by at most 1:

▶ **Theorem 1.** *There is a* 3.224*–approximation algorithm for the weighted stochastic matching problem on a general graph.*

▶ **Theorem 2.** *There is a* 2.675*–approximation algorithm for the weighted stochastic matching problem on a general graph if patience constraints are allowed to be violated by* 1*.*

In essence, the LP-based approach of [2] uses a dependent-rounding algorithm of [7] to first guarantee that the patience constraints are satisfied with probability one within the context of their randomized algorithm; the probing is done on top of this setup. In contrast, we randomly permute the edges and then probe them in this order, with probing probabilities suggested by the LP – of course, not probing infeasible edges in the process. An edge is infeasible if a neighboring edge has already been placed in the matching, or if one of the two end-points has had its patience exhausted. While it is not too hard to incorporate the matching constraints here, the patience constraints are far more complex to handle well: e.g., direct use of Chernoff-type bounds will not help. We work to identify extremal input-instances for our algorithm and combine this with rigorous computer-aided calculations in order to conduct our analyses. Theorem 2 follows from a new attenuation idea. The algorithms themselves are quite simple to implement; the main feature of our work is a detailed analysis of the worst-case settings for our algorithms. All calculations and proofs omitted from this preliminary version will appear in the full version.

Theorem 3 and Theorem 4 improve upon the $(k + 1)$–approximation of [2] for weighted matching in k-uniform hypergraphs. Both of these algorithms use first to classify the hyperedges as "small" or "large" based on the LP values, and treat each group separately. The difference is as follows. The algorithm of Theorem 3 attenuates the small edges to boost the performance of large edges; the algorithm of Theorem 4 uses a "weighted permutation" of the hyperedges such that each large edge has a higher chance to fall behind a small edge. Although Theorem 4 is asymptotically better, we present both theorems since their ideas can be useful elsewhere. Note that the LP-based methods of [2] and ours cannot in general do better than $k - 1 + 1/k$ [6]; hence, we are close to optimal for LP-based approaches.

▶ **Theorem 3.** *There is a $(k+\frac{1}{2}+o(1))$–approximation algorithm for the stochastic matching problem on a k-uniform hypergraph, where the "$o(1)$" term is a function of k that vanishes asymptotically.*

▶ **Theorem 4.** *For any given $\epsilon > 0$, there is a $(k+\epsilon+o(1))$–approximation algorithm for the stochastic matching problem on a k-uniform hypergraph, where the "$o(1)$" term is a function of k that vanishes asymptotically.*

Finally, we significantly improve upon the sample complexity of [3] for stochastic optimization in the black-box model. Since the bounds are somewhat technical, we defer discussion of the actual parameters to Section 5: please see Theorems 13 and 14 for statements of the state-of-the-art and of our improvement. The analysis of [3] has different worst-case settings, but we show that the values of the parameters are very different in these different regimes. This enables a careful analysis of how many samples the approach really needs. This black-box model is quite general, and an improved sample complexity translates to more-efficient implementations of the several applications of the work of [3] (see, e.g., [8, 9, 17]).

Preliminaries. We will often consider a uniformly random permutation π on a set of items $\{e_1, \dots, e_n\}$. We can assume that π is chosen as follows: for each item e, we pick independently and uniformly at random a real number $\pi(e) = a_e \in [0,1]$, and then sort these in increasing order to obtain π. Note that we abuse notation by letting π denote both the permutation and the reals chosen; however, this choice will be clear from the context.

We make use of the following form of the Chernoff bound:

▶ **Definition 5** (Chernoff Bound.). Let $X_1, \dots, X_n$ be n independent random variables with $0 \le X_i \le 1$. Let $X = X_1 + \dots + X_n$ and $\mu = \mathbb{E}[X]$. Then for any $\varepsilon > 0$,

$$\Pr[X \ge (1+\varepsilon)\mu] \le \exp\left(-\frac{\varepsilon^2}{2+\varepsilon}\mu\right), \text{ and}$$
$$\Pr[X \le (1-\varepsilon)\mu] \le \exp\left(-\frac{\varepsilon^2}{2}\mu\right)$$

2 Stochastic Matching

We consider the following stochastic matching problem. The input is an undirected graph $G = (V, E)$ with a weight w_e and a probability value p_e on each edge $e \in E$. In addition, there is an integer value t_v – the *patience* – for each vertex $v \in V$. Initially, each vertex $v \in V$ has patience t_v. At any step in the algorithm, only an edge $e(u,v) \in E$ such that $t_u > 0$ and $t_v > 0$ can be probed. Upon probing such an edge e, one of the following happens: (1) with probability p_e, e exists; u and v get matched and are removed from G along with their incident edges, or (2) with probability $(1-p_e)$, e does not exist; e is removed, and t_u and t_v are reduced by 1. (All these edge-existence events are independent.) We seek to find an adaptive strategy for probing edges; its performance is measured by the expected weight of the matched edges. We prove Theorem 1 now.

Consider the following natural LP relaxation [2]: for any vertex $v \in V$, $\partial(v)$ denotes the edges incident to v. The LP variable y_e denotes the probability that edge $e(u,v)$ gets probed in the adaptive strategy, and $x_e = y_e p_e$ denotes the probability that e gets matched in the strategy.

$$\text{maximize} \sum_{e \in E} w_e x_e \tag{2.1}$$

$$\text{subject to} \sum_{e \in \partial(v)} x_e \leq 1 \qquad \forall v \in V \tag{2.2}$$

$$\sum_{e \in \partial(v)} y_e \leq t_v \qquad \forall v \in V \tag{2.3}$$

$$x_e = y_e p_e \geq 0, \; y_e \leq 1 \qquad \forall e \in E \tag{2.4}$$

▶ **Lemma 6** ([2]). *The optimal value for the LP* (2.1) *is an upper bound on the performance of any adaptive algorithm for stochastic matching.*

We use (x, y) to denote the optimal solution to the LP in equation (2.1). For an edge $e(u, v)$, it is called *safe* at the time it is considered if: (1) neither u nor v is matched, and (2) both of $t_u > 0$ and $t_v > 0$. Our algorithm, denoted by SM_1, first fixes a uniformly random permutation π on the set of edges E. It then inspects the edges one by one in the order of π. If an edge e is safe, the algorithm probes it (independently) with probability y_e, otherwise it skips to the next one. For ease of analysis, we state our algorithm SM_1 in a slightly different but equivalent way in Algorithm 1.

Algorithm 1: SM_1: Stochastic Matching

```
1 Choose a random permutation $\pi$ on $E$.
2 For each edge $e \in E$, generate a random bit $Y_e = 1$ independently with probability $y_e$.
  Let $E'$ be the set of edges with $Y_e = 1$.
3 Follow the random order $\pi$ to inspect edges in $E'$
4     If an edge $e$ is safe, then probe it; otherwise, skip it.
```

To analyze the performance of our algorithm, we conduct an edge-by-edge analysis. Recall that the LP variable $x_e = y_e p_e$ denotes the probability that e is matched in the LP (2.1), and the optimal value of the LP is exactly $\sum_{e \in E} w_e p_e y_e$. The expected weight of the matching found by our algorithm is $\mathbb{E}[\mathsf{SM}_1] = \sum_{e \in E} w_e p_e \Pr[e \in E'] \Pr[e \text{ gets probed} | e \in E']$, which is $\sum_{e \in E} w_e p_e y_e \Pr[e \text{ gets probed} | e \in E'] \geq \sum_{e \in E} w_e p_e y_e \lambda$, assuming $\Pr[e \text{ gets probed} | e \in E'] \geq \lambda$. This gives us a λ-approximation algorithm.

We now start to discuss how to compute the value of λ. Focus on a specific edge $e = e(u, v)$, let $E(u)$ be the set of edges incident to u *excluding* e itself, i.e. $E(u) = \partial(u) \setminus \{e\}$. Conditioning on $\pi(e) = x$ with $0 < x < 1$ and $Y_e = 1$, let $\mathcal{P}_u$ be the probability that e is *not blocked* by any of edges in $E(u)$ in the algorithm SM_1. Here we say e is blocked by some edge f in $E(u)$ if f gets matched or patience constraint on u gets tight resulting from probing f $(t_u = 0)$. We assume without loss of generality that $|E(u)| \geq t_u$, otherwise the patience constraint for node u will be redundant.

A little thought gives us the following lower bound on $\mathcal{P}_u$:

$$\mathcal{P}_u \geq P_u = \sum_{S \subseteq E(u), |S| \leq t_u - 1} x^{|S|} \prod_{f \in S} y_f (1 - p_f) \prod_{f \notin S} (1 - x y_f) \tag{2.5}$$

To see why this is true, let Y'_f (for any $f \in E(u)$) be the indicator random variable that is 1 iff f gets matched when probed, i.e., $\Pr[Y'_f = 1] = p_f$. For each $S \subseteq E(u)$, such that $|S| \leq t_u - 1$, we associate an event E_S that says: "(1) each edge $f \in S$ falls before e in π

with $Y_f = 1$ and $Y'_f = 0$; and (2) each edge $f \notin S$ either falls after e in π or $Y_f = 0$". We can see that each E_S is a sufficient condition for e being not blocked by any edge of S. Thus $\mathcal{P}_u$ should be at least the probability that one or more of E_S happen, which is exactly P_u.

In the following paragraphs, we carefully investigate the configuration of edges that minimizes the value of P_u. We denote such adversarial configurations as the **worst-case structure (WS)** of $E(u)$. For each of these configurations, we have the constraints: (i) $\sum_{f \in E(u)} y_f p_f \leq 1$, (ii) $\sum_{f \in E(u)} y_f \leq t_u$ and (iii) $0 \leq y_f \leq 1$ for each $f \in E(u)$. Here we view x as a (given) parameter.

▶ **Lemma 7.** *In WS, there will be at most one edge with $p_f = 1$ and at most one edge with $0 < p_f < 1$. All other edges must have $p_f = 0$.*

Proof. We prove by contradiction. Assume there are two edges, say $p_1 = p_2 = 1$ in WS. Then $y_1 + y_2 \leq 1$ since $\sum_i y_i p_i \leq 1$. We perturb the current configuration as follows: merge the two edges into a single edge e_3 where $y_3 = y_1 + y_2$ and $p_3 = 1$. Notice that after this perturbation, both of the values $\sum_{f \in E(u)} y_f p_f$ and $\sum_{f \in E(u)} y_f$ remain unchanged. Thus both of matching and patience constraints are maintained at u, and our perturbation gives a feasible configuration.

The change brought by this perturbation to the value P_u is as follows: for each non-zero term in P_u associated with some $S \subseteq E(u)$ where $e_1 \notin S, e_2 \notin S$, the term $(1 - xy_1)(1 - xy_2)$ will be replaced with $(1 - x(y_1 + y_2))$, which results in a strictly lower value of P_u. This is a contradiction.

Now assume there are two edges a, b with $0 < p_a, p_b < 1$ in WS. Consider the following perturbation: for some small $\varepsilon \neq 0$, set $p'_a = p_a + \varepsilon/y_a$ and $p'_b = p_b - \varepsilon/y_b$. After this perturbation, both of $\sum_{f \in E(u)} y_f p_f$ and $\sum_{f \in E(u)} y_f$ remains unchanged and the perturbed configuration is still feasible.

Let $f(\varepsilon)$ be the value of P_u after this update. In the expression of P_u, the terms contributing to ε^2 must be those associated with S where $a, b \in S$. Notice that

$$(1 - p'_a)(1 - p'_b) = (1 - p_a - \varepsilon/y_a)(1 - p_b + \varepsilon/y_b)$$

has a negative coefficient of ε^2, implying that the second derivative $f'' < 0$. Therefore we can always find a non-zero value of ε to make P_u strictly smaller. Again a contradiction. ◀

Let $E_1(u)$ and $E_0(u)$ be the set of edges in WS which have $p_f = 1$ and $p_f = 0$ respectively. Let (y_a, p_a) be the potential edge taking a floating $0 < p_a < 1$ value. Lemma 7 tells us $E_1(u)$ contains at most one edge in WS. Let $A = \sum_{f \in E_1(u)} y_f$.

Based on Lemma 7, we can update the expression of P_u as

$$P_u = (1 - xA)(1 - xy_a)\Pr[Z_u \leq t_u - 1] + (1 - xA)xy_a(1 - p_a)\Pr[Z_u \leq t_u - 2] \tag{2.6}$$

where $Z_u = \sum_{f \in E_0(u)} Z_f$ and $\{Z_f | f \in E_0(u)\}$ are independent Bernoulli random variables with $\Pr[Z_f = 1] = xy_f, \forall f \in E_0(u)$. Here are two useful lemmas; the proofs will appear in the full version.

▶ **Lemma 8.** *In WS, $p_a = 0$.*

From Lemma 8, we can claim that there is no edge f that takes fractional p_f value. Thus we can further simplify the expression of P_u in equation (2.6) as

$$P_u = (1 - xA)\Pr[Z_u \leq t_u - 1] \tag{2.7}$$

▶ **Lemma 9.** *In WS, $A = 1$ and Z_u follows Poisson distribution with mean $x(t_u - 1)$.*

At this point, we have all the essentials to prove Theorem 1.

Proof. We have $\Pr[e \text{ gets probed } | Y_e = 1] = \int_0^1 \mathcal{P}_u \mathcal{P}_v dx \geq \int_0^1 P_u P_v dx$, i.e., at least

$$H(t_u, t_v) \doteq \int_0^1 (1-x)^2 \Pr[Z_u \leq t_u - 1] \Pr[Z_v \leq t_v - 1] dx,$$

where Z_u and Z_v follow Poisson distributions with means $\mathbb{E}[Z_u] = x(t_u - 1)$ and $\mathbb{E}[Z_v] = x(t_v - 1)$ respectively. The rest of the analysis splits into the following three cases.

- We can numerically verify that $H(t_u, t_v)$ achieves its minimum value of $0.31016 = 1/3.224$ at $t_u = t_v = 2$ when $1 \leq t_u, t_v \leq 20$.
- For $t_u, t_v \geq 20$, by applying the Chernoff bound, we get

  $$H(t_u, t_v) \geq \int_0^1 (1-x)^2 \left[1 - \exp\left(\frac{-\epsilon^2}{2+\epsilon} x(t_u - 1)\right)\right] \left[1 - \exp\left(\frac{-\epsilon^2}{2+\epsilon} x(t_v - 1\right)\right] dx,$$

  where $\epsilon = \epsilon(x) = \frac{1}{x} - 1$; by plugging in $t_u = t_v = 20$, we can verify numerically that this integral is at least 0.316324.
- Similarly, for $1 \leq t_u \leq 20$ while $t_v \geq 20$, we can verify numerically (by checking all integers $1 \leq t_u \leq 20$) that with $\epsilon = \frac{1}{x} - 1$,

  $$H(t_u, t_v) \geq \int_0^1 (1-x)^2 \Pr(Z_u \leq t_u - 1) \left[1 - \exp\left(\frac{-\epsilon^2}{2+\epsilon} x(20 - 1)\right)\right] dx \geq 0.312253.$$

This establishes the key claim that $\Pr[e \text{ gets probed } | Y_e = 1] \geq 0.3106$ for each $e \in E$. ◀

3 Stochastic Matching with Relaxed Patience

In this section, we consider the variant of the stochastic matching problem in which patience constraints are allowed to be violated by at most 1, and prove Theorem 2. From the analysis of Section 2, we observe that the edges with a large $y_e p_e$ value are probed with a much higher probability than those with small ones. This indicates that small edges are the ones that bottleneck the performance of our algorithm. Our high level idea here is to *attenuate* such "large" edges in order to improve the performance of the small ones. The process of attenuation carefully calculates a value $h_e \in (0, 1]$, called as the *attenuation factor*, for each $e \in E$. Instead of probing an edge e with probability y_e as in algorithm SM_1, our new algorithm probes it with probability $h_e y_e$. We will show that such a strategy balances the performance of large and small edges and improves the overall performance of SM_1.

The overall picture of our algorithm, denoted SM_2, is as follows. First we label each edge $e \in E$ as "large" if $y_e p_e > 1/2$ or "small" if $y_e p_e \leq 1/2$. Similar to SM_1, we follow a random permutation π on the set of edges E to inspect each edge. If an edge e is safe when considered, we probe it with probability $h_e y_e$; otherwise we skip it. Here $h_e = h$ if e is large and $h_e = 1$ otherwise, where $h \geq 1/2$ is a parameter which we optimize later. For ease of analysis, we state the algorithm SM_2 in an alternate but essentially equivalent way in Algorithm 2. In the spirit of Section 2, we conduct an edge-by-edge analysis. The full analysis will appear in the full version.

Algorithm 2: SM$_2$: Stochastic Matching with relaxed patience

```
1 Choose a random permutation $\pi$ of $E$.
2 For each edge $e \in E$, set $h_e = h$ if $y_e p_e > 1/2$, set $h_e = 1$ otherwise.
3 For each edge $e \in E$, generate a random bit $Y_e = 1$ with probability $h_e y_e$. Let $E'$ be
  the set of edges with $Y_e = 1$.
4 Follow the random order $\pi$ to inspect edges in $E'$
5     If an edge $e$ is safe, probe it; otherwise, skip it.
```

4 Stochastic Hypergraph Matching

We now consider stochastic matching in a k-uniform hypergraph, i.e., a hypergraph where all edges have size k. The standard LP can be obtained by naturally extending the LP in (2.1) to the one below:

$$\max \sum_{e \in E} w_e x_e : \sum_{e \in \partial(v)} x_e \leq 1, \forall v \in V, x_e = y_e p_e \geq 0, y_e \leq 1, \forall e \in E \tag{4.1}$$

Note that we do not consider the patience parameter at a vertex, as in Section 2. Here $\partial(v)$ denotes the set of hyperedges incident to v.

4.1 An algorithm achieving $(k + 1/2 + o(1))$ approximation ratio

Let (x, y) be an optimal solution to the LP (4.1). At a high level, our algorithm proceeds according to the outline below. Let $c \geq 1/2$ be a parameter, which will be optimized at $1/2$ later.

1. Divide the edges into two sets, the "small" edge set $E_S = \{e | y_e p_e \leq c\}$, and the "large" edge set $E_L = E \setminus E_S$.
2. Choose a random permutation π on E_S.
3. Sample each edge $e \in E_S$ with probability y_e, independent of other edges. Let E'_S be the set of edges sampled.
4. Follow the random order π to inspect if each small edge $e \in E'_S$ is safe or not. If e is safe, probe it with probability h_e; otherwise, skip it. Here $0 < h_e \leq 1$ is a parameter to fix later.
5. After inspecting all small edges, remove all the unsafe large edges from E_L, and probe the rest with probability 1 (in arbitrary order).

Roughly speaking, an edge e being "safe" means none of the edges in the neighborhood of e are matched. Later, we will give a definition that is both stronger and *exactly* computable. Based on the new definition, we compute an *attenuation factor* h_e for each $e \in E_S$, such that at the end of the algorithm, e is probed with probability *exactly equal* to y_e/λ. Here, $\lambda \geq 1$ is our target approximation ratio. All that remains is to analyze the performance of each large edge $e \in E_L$ and show that e is probed with probability at least y_e/λ. That gives us a λ–approximation algorithm.

We redefine the notion of a small edge e being safe. Suppose π is the random order on E_S and $\pi(e) = x, 0 < x < 1$. Let $N_S[e]$ be the set of small edges in the neighborhood of e. For each $f \in N_S[e]$, let X_f, Y_f, Z_f be three random variables such that: $X_f = 1$ if f falls before e in π, $Y_f = 1$ if $f \in E'_S$ and $Z_f = 1$ if f exists in the hypergraph when probed. Note that the collection of random variables $\{X_f, Y_f, Z_f | f \in N_S[e]\}$ are mutually independent.

For each $f \in N_S[e]$, let A_f be the event that $(X_f + Y_f + Z_f \leq 2)$ and $\mathsf{S}_e = \wedge_{f \in N_S[e]} A_f$. *We* say e is safe iff S_e happens . Lemma 10 computes the probability that a small edge e is safe in our algorithm.

▶ **Lemma 10.**

$$\Pr[\mathsf{S}_e] = \int_0^1 \Pr[\mathsf{S}_e | \pi(e) = x] dx = \int_0^1 \prod_{f \in N_S[e]} (1 - x y_f p_f) dx. \tag{4.2}$$

Proof. By definition, $\Pr[X_f = 1 | \pi(e) = x] = x$. Note that $\Pr[Y_f = 1] = y_f$, $\Pr[Z_f = 1] = p_f$, and the two values are independent of $\pi(e)$. Thus, given $\pi(e) = x$, A_f will occur with probability $(1 - x y_f p_f)$. Since the A_f are independent for $f \in N_S[e]$, the proof is completed. ◀

Here are two interesting points for the event S_e: (1) When S_e happens, e must be safe according to our initial definition, i.e., none of the edges in its neighborhood get matched; the contrary is not true. Thus the new definition is more strict. (2) On checking e in the algorithm, we might not know if S_e occurs or not due to some missing Z_f for $f \in N_S[e]$. For instance, some $f \in N_S[e]$ gets blocked by some small edge $f' \in N_S[f]$ while $X_f = Y_f = 1$. In this case, we do not know the value of Z_f since f will not be probed. In order to continue our algorithm, we simulate Z_f by generating a random bit $Z_f = 1$ with probability p_f and $Z_f = 0$ otherwise. Notice that if $Z_f = 1$, we will view e as not safe and will not probe it, even though it might be safe according to our initial definition.

The full description and analysis of algorithm in Theorem 3 will appear in the full version.

4.2 An algorithm achieving $(k + \epsilon + o(1))$ approximation ratio

In this section, we present a randomized algorithm that achieves an approximation ratio of $(k + \epsilon + o(1))$ for stochastic matching on a k-uniform hypergraph, where ϵ is given in advance.

Let (x, y) be an optimal solution to the LP (4.1). W.L.O.G we assume $1/\epsilon = N$ where N is an integer. Let a be a constant such that $1 - 1/N < a < 1$. We say an edge e is large if $y_e p_e > 1/N$, otherwise we call e small. For each small edge e, we draw a random real number x_e uniformly from $[0, 1]$. For each large edge e, we draw a random real number x_e from $[0, \delta]$ with density a and from $(\delta, 1]$ with density $(1 - a\delta)/(1 - \delta)$, where $\delta = \min(1, N(1 - a^{1/(N-1)})$. Then we derive a random permutation π by sorting $\{x_e, e \in E\}$ in increasing order. Assuming N is sufficiently large, the value δ is at most $1/N + o(1/N)$. Notice that N, a and δ are all fixed constants. Based on π, we sketch our randomized algorithm below:

Algorithm 3: SM_4: Stochastic Matching on a k-uniform hyergraph

```
1 Initially all edges are safe.
2 Follow the random order $\pi$ to check each edge $e \in E$ is safe or not.
3     If $e$ is safe, then probe it with probability $y_e$; otherwise, skip it.
```

The lemmas below are useful for the proof of Theorem 4.

▶ **Lemma 11.** *For any $c > 1/N$ and $0 < x < \delta$, we have*

$$1 - axc > (1 - x/N)^{cN}$$

Proof. Define $F(x) = 1 - axc - (1 - x/N)^{cN}$. We can verify that: (1) $F(0) = 0$, and (2) $F'(x) > 0$ for any $0 \leq x < \delta$. This gives the desired result. ◀

Consider an edge $e = (v_1, v_2, \cdots, v_k)$. Suppose $y_e p_e = c_e < 1-1/N$ and $x_e = x, 0 < x < \delta$. For each $1 \leq i \leq k$, let $\partial'(v_i)$ denote the set of edges incident to v_i excluding e itself. Denote by $\mathcal{S}_i$ the event that none of edges in $\partial'(v_i)$ come before e and get matched.

▶ **Lemma 12.**

$$\Pr[\mathcal{S}_i] \geq (1 - x/N)^{(1-c_e)N}$$

The proof of Lemma 12 will appear in the full version. Now we start to prove Theorem 4.

Proof.

1. Consider a small edge e, say $e = (v_1, v_2, \cdots, v_k)$ and $x_e = x$. From Lemma 12, we see $\Pr[\mathcal{S}_i] \geq (1 - x/N)^N$ for each $1 \leq i \leq k$. Thus by applying the FKG inequality, we get $\Pr\left[\bigwedge_i \mathcal{S}_i\right] \geq (1 - x/N)^{kN}$, which is followed by

$$\Pr[\text{ } e \text{ is checked as safe }] \geq \int_0^\delta (1 - x/N)^{kN} dx = \frac{1}{k + 1/N} - O(k_0^k/k)$$

where $k_0 = (1 - \delta/N)^N < 1$ is a constant.

2. Consider a large edge e, say $e = (v_1, v_2, \cdots, v_k)$ and $x_e = x$. From Lemma 12, we see $\Pr[\mathcal{S}_i] \geq (1 - x/N)^{N-1}$ for each $1 \leq i \leq k$. Thus by applying FKG, we see when $x \leq \delta$, $\Pr\left[\bigwedge_i \mathcal{S}_i\right] \geq (1 - x/N)^{k(N-1)}$, which is followed by

$$\Pr[\text{ } e \text{ is checked as safe }] \geq \int_0^\delta a(1-x/N)^{k(N-1)} dx = \frac{aN}{N-1}\frac{1}{k + 1/(N-1)} - O(k_0^k/k) > \frac{1}{k}$$

where $k_0 = (1 - \delta/N)^{N-1} < 1$ is a constant; we use the fact that $a > 1 - 1/N$ to get the last inequality above. ◀

5 Sample Complexity of Black-Box Stochastic Optimization

In this section, we consider the following *two-stage* stochastic minimization program

$$\min_{x \in X} f(x) = c(x) + \mathop{\mathbb{E}}_{\omega}[q(x, \omega)]. \tag{5.1}$$

An important context in which this problem arises is *two-stage stochastic optimization with recourse*. In this model, a *first-stage* decision $x \in X$ has to be made while having only probabilistic information about the future, represented by the probability distribution π on Ω. Then, after a particular future scenario $\omega \in \Omega$ is realized, a *recourse action* $r \in R$ may be taken to ensure that the requirements of the scenario ω are satisfied. In the two-stage model, $c(x)$ denotes the cost of taking the first stage action x. Given a particular scenario ω and a first-stage action x, the cost of the second stage $q(x, \omega)$ is represented as

$$q(x, \omega) = \min_{r \in R}\{\text{cost}_\omega(x, r) | (x, r) \text{ is a feasible solution for scenario } \omega\}.$$

A natural approach to solve problems modeled by equation (5.1) is to take some number, N, of independent samples $\omega_1, \ldots, \omega_N$ from the distribution π, and to approximate the function f by the sample-average function

$$\hat{f}(x) = c(x) + \frac{1}{N}\sum_{i=1}^{N} q(x, \omega_i). \tag{5.2}$$

One might then hope that for a suitably chosen sample size N, a good solution $\hat{x}$ to equation (5.2) would be a good solution to f; more precisely, $\hat{x} \in X$ is an α-approximate minimizer of the function f defined in (5.1), if for all $x \in X$, $f(\hat{x}) \leq \alpha f(x)$. This approach, called the *sample average approximation* method (SAA), was considered by Charikar, Chekuri, and Pál [3], who considered a setting with the following properties:

(P1) Non-negativity. $c(x) \geq 0$ and $q(x,\omega) \geq 0$ for each $x \in X$ and $\omega \in \Omega$.
(P2) Empty First stage. We assume there is an empty first stage action, $0 \in X$ with $c(0) = 0$, $q(x,\omega) \leq q(0,\omega)$ for each $x \in X, \omega \in \Omega$.
(P3) Bounded Inflation Factor. For each $x \in X, \omega \in \Omega$, we have $q(0,\omega) - q(x,\omega) \leq \lambda c(x)$.

In such a setting, a key result of [3] is:

▶ **Theorem 13** ([3])**.** *There is a constant $K_0 > 0$ such that the following holds. Any exact minimizer $\bar{x}$ of the function $\hat{f}$ defined in* (5.2) *constructed with $K_0 \cdot \frac{\lambda^2}{\varepsilon^4} \log|X| \log\frac{1}{\delta}$ samples is, with probability at least $1-\delta$, an $(1+O(\varepsilon))$-approximate minimizer of the function f defined in* (5.1)*.*

Our result on improved sample complexity states as follows. The proof will appear in the full version.

▶ **Theorem 14.** *There is a constant $K_1 > 0$ such that the following holds. Any exact minimizer $\bar{x}$ of the function $\hat{f}$ defined in* (5.2) *constructed with $N = K_1 \cdot \log\frac{|X|}{\delta} \cdot \max\left[\frac{\lambda^2}{\varepsilon^2}, \frac{\lambda}{\varepsilon^3}\right]$ samples is, with probability at least $1-\delta$, an $(1+O(\varepsilon))$-approximate minimizer of the function f defined in* (5.1)*.*

This improvement, in turn, improves the runtime of the several applications that employ this sampling framework; see, e.g., [8, 9, 17].

Acknowledgements. We thank the conference referees for their helpful suggestions.

References

1 Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. Improved approximation algorithms for stochastic matching. *CoRR*, abs/1505.01439, 2015.
2 Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012.
3 Moses Charikar, Chandra Chekuri, and Martin Pál. Sampling bounds for stochastic optimization. In *Proceedings of the 8th International Workshop on Approximation, Randomization and Combinatorial Optimization Problems, and Proceedings of the 9th International Conference on Randamization and Computation: Algorithms and Techniques*, APPROX'05/RANDOM'05, pages 257–269. Springer-Verlag, 2005.
4 Ning Chen, Nicole Immorlica, Anna R Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. *Automata, Languages and Programming*, pages 266–278, 2009.
5 Brian C Dean, Michel X Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.
6 Zoltán Füredi, Jeff Kahn, and Paul D. Seymour. On the fractional matching polytope of a hypergraph. *Combinatorica*, 13(2):167–180, 1993.

7 Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM (JACM)*, 53(3):324–360, 2006.

8 Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *SODA*, pages 942–951, 2008.

9 Anupam Gupta and Amit Kumar. A constant-factor approximation for stochastic steiner forest. In *STOC*, pages 659–668, 2009.

10 Anupam Gupta, R Ravi, and Amitabh Sinha. An edge in time saves nine: Lp rounding approximation algorithms for stochastic network design. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 218–227. IEEE, 2004.

11 Nicole Immorlica, David Karger, Maria Minkoff, and Vahab S Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 691–700. Society for Industrial and Applied Mathematics, 2004.

12 Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

13 Jeff Linderoth, Alexander Shapiro, and Stephen Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241, 2006.

14 Andrzej Ruszczynski and Alexander Shapiro. Stochastic programming. *Handbooks in operations research and management science*, 2003.

15 Alexander Shapiro. Monte Carlo sampling methods. *Handbooks in operations research and management science*, 10:353–425, 2003.

16 David B Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *Journal of the ACM (JACM)*, 53(6):978–1012, 2006.

17 A. Srinivasan. Approximation algorithms for stochastic and risk-averse optimization. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 1305–1313, 2007.

Fully Dynamic Bin Packing Revisited[*]

Sebastian Berndt[1], Klaus Jansen[†2], and Kim-Manuel Klein[2]

1 **Institute of Theoretical Computer Science, Universität zu Lübeck, Germany,** `berndt@tcs.uni-luebeck.de`

2 **Department of Computer Science, Christian-Albrechts-University to Kiel, Germany,** `{kj,kmk}@informatik.uni-kiel.de`

Abstract

We consider the *fully dynamic bin packing* problem, where items arrive and depart in an online fashion and repacking of previously packed items is allowed. The goal is, of course, to minimize both the number of bins used as well as the amount of repacking. A recently introduced way of measuring the repacking costs at each timestep is the *migration factor,* defined as the total size of repacked items divided by the size of an arriving or departing item. Concerning the trade-off between number of bins and migration factor, if we wish to achieve an asymptotic competitive ratio of $1+\epsilon$ for the number of bins, a relatively simple argument proves a lower bound of $\Omega(1/\epsilon)$ on the migration factor. We establish a fairly close upper bound of $O(1/\epsilon^4 \log 1/\epsilon)$ using a new dynamic rounding technique and new ideas to handle small items in a dynamic setting such that no amortization is needed. The running time of our algorithm is polynomial in the number of items n *and* in $1/\epsilon$. The previous best trade-off was for an asymptotic competitive ratio of $5/4$ for the bins (rather than $1+\epsilon$) and needed an amortized number of $O(\log n)$ repackings (while in our scheme the number of repackings is independent of n and non-amortized).

1998 ACM Subject Classification G.2.1 Combinatorial algorithms

Keywords and phrases online, bin packing, migration factor, robust, AFPTAS

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.136

1 Introduction

For the classical bin packing problem, we are given a set I of items with a size function $s\colon I \to (0,1]$ and need to pack them into as few unit sized bins as possible. In practice, the complete instance is often not known in advance, which has lead to the definition of a variety of *online* versions of the bin packing problem. First, in the classical *online bin packing* [35], items arrive over time and have to be packed on arrival. Second, in *dynamic bin packing* [8], items also depart over time. This dynamic model is often used for instance in

- the placement and movement of virtual machines onto different servers for cloud computing [3, 4, 22, 23, 32, 36],
- the development of guaranteed quality of service channels over certain multi-frequency time division multiple access systems [28],
- the placement of processes, which require different resources, onto physical host machines [33, 34],
- the resource allocation in a cloud network where the cost depends upon different parameters [9, 26].

[*] A full version of this paper is available under `http://arxiv.org/abs/1411.0960`.

[†] Supported by DFG Project "Entwicklung und Analyse von effizienten polynomiellen Approximationsschemata für Scheduling- und verwandte Optimierungsprobleme," Ja 612/14-1.

Name	Deletion	Repacking
Online Bin Packing	✗	✗
Relaxed Online Bin Packing	✗	✓
Dynamic Bin Packing	✓	✗
Fully Dynamic Bin Packing	✓	✓

Figure 1 Overview of online models.

Third and fourth, we may allow already packed items to be slightly rearranged, leading to online bin packing with repacking (known as *relaxed online bin packing*) [14] and dynamic bin packing with repacking (known as *fully dynamic bin packing*) [16]. See Figure 1 for a short overview on the different models.

The amount of repacking can be measured in different ways. We can either count the total number of moved items at each timestep or the sum of the sizes of the moved items at each timestep. If one wants to count the number of moved items, one typically counts a group of tiny items as a single move. A *shifting move* [14] thus involves either a single large item or a bundle of small items in the same bin of total size s with $1/10 \leq s \leq 1/5$. Such a bundle may consists of up to $\Omega(n)$ (very small) items. If an algorithm measures the repacking by shifting moves, an occurring tiny item may lead to a large amount of repacking. In order to guarantee that a tiny item i with size $s(i)$ only leads to a small amount of repacking, one may allow to repack items whose size adds up to at most $\beta \cdot s(i)$. The term β is called the *migration factor* [29]. Note that shifting moves and migration factor are incomparable in the sense that a small migration factor does not imply a small number of shifting moves and vice versa.

In order to measure the quality of an online algorithm, we compare the costs incurred by an online algorithm with the costs incurred by an optimal offline algorithm. An *online algorithm* receives as input a *sequence* of items $I = (i_1, i_2, i_3, \ldots)$ and decides at each timestep t, where to place the item i_t without knowing future items $i_{t+1}, i_{t+2}, \ldots$. We denote by $I(t) = (i_1, i_2, \ldots, i_t)$ the instance containing the first t items of the instance I and by $\textsc{opt}(I(t))$ the minimal number of bins needed to pack all items in $I(t)$. Note that the packings corresponding to $\textsc{opt}(I(t))$ and $\textsc{opt}(I(t+1))$ may differ significantly, as those packings do not need to be consistent. For an online algorithm A, we denote by $A(I(t))$ the number of bins generated by the algorithm on the input sequence $I(t)$. Note that A must make its decision online, while $\textsc{opt}(I(t))$ is the optimal value of the offline instance. The quality of an algorithm for the online bin packing problem is typically measured by its *asymptotic competitive ratio*. An online algorithm A is called an *asymptotic α-competitive algorithm*, if there is a function f with $\lim_{n\to\infty} \sup \left\{ \frac{f(I)}{\textsc{opt}(I)} \mid \textsc{opt}(I) = n \right\} = 0$ such that $A(I(t)) \leq \alpha\, \textsc{opt}(I(t)) + f(I(t))$ for all instances I and all $t \leq |I|$. The minimum α such that A is an asymptotic α-competitive algorithm is called the *asymptotic competitive ratio of A*, denoted by $r^{\text{on}}_{\infty}(A)$, i. e., the ratio is defined as $r^{\text{on}}_{\infty}(A) = \min\{\alpha \mid A \text{ is an asymptotic } \alpha\text{-competitive algorithm}\}$. The online algorithm A thus has a double disadvantage: It does not know future items and we compare its quality to the optimal offline algorithm which may produce arbitrary different packings at time t and time $t + 1$. In order to remedy this situation, one may also compare the solution generated by A to a non-repacking optimal offline algorithm. This non-repacking optimal offline algorithm knows the complete instance, but is not allowed to repack, i.e., the solutions at time t and time $t + 1$ must be consistent.

In this work, we present new results in fully dynamic bin packing where we measure the quality of an algorithm against a repacking optimal offline algorithm and achieve an asymptotic competitive ratio of $1+\epsilon$. The amount of repacking is bounded by $\mathcal{O}(1/\epsilon^4 \log(1/\epsilon))$. While we measure the amount of repacking in terms of the migration factor, we also prove that our algorithm uses at most $\mathcal{O}(1/\epsilon^4 \log(1/\epsilon))$ shifting moves. Our algorithm runs in time polynomial in the instance size and in $1/\epsilon$.

1.1 Previous Results on Online Variants of Bin Packing

Online Bin Packing

The classical version of the online bin packing problem was introduced by Ullman [35]. In this classical model items arrive over time and have to be packed at their arrival, while *one is not allowed to repack already packed items.* Ullman gave the very first online algorithm FirstFit for the problem and proved that it its absolute competitive ratio is at most 2. The next algorithm NextFit was given by Johnson [19], who proved that its absolute competitive is also at most 2. The analysis of FirstFit was refined by Johnson, Demers, Ullman, Garey and Graham [20], who proved that its asymptotic competitive ratio is at most $17/10$. A revised version of FirstFit, called Revised FirstFit was shown to have asymptotic competitive ratio of at most $5/3$ by Yao [39]. A series of developments of so called *harmonic algorithms* for this problem was started by Lee and Lee [25] and the best known algorithm of this class which has asymptotic competitive ratio at most 1.58889 was given by Seiden [30].

The lower bound on the absolute approximation ratio of $3/2$ also holds for the asymptotic competitive ratio as shown by Yao [39]. This lower bound was first improved independently by Brown [5] and Liang [27] to 1.53635 and subsequently to 1.54014 by van Vliet [37] and finally to 1.54037 by Balogh, Békési and Galambos [1].

Relaxed Online Bin Packing Model

In contrast to the classical online bin packing problem, Gambosi, Postiglione and Talamo [14] considered the online case where one is *allowed to repack items.* They called this model the *relaxed online bin packing model* and proved that the lower bound on the competitive ratio in the classical online bin packing model can be beaten. They presented an algorithm that uses 3 *shifting moves* and has an asymptotic competitive ratio of at most $3/2$, and an algorithm that uses at most 7 shifting moves and has an asymptotic competitive ratio of $4/3$. In another work, Ivković and Lloyd [15] gave an algorithm that uses $\mathcal{O}(\log n)$ *amortized* shifting moves and achieves an asymptotic competitive ratio of $1+\epsilon$. In this amortized setting, shifting moves can be saved up for later use and the algorithm may repack the whole instance sometimes. Epstein and Levin [11] used the measure of the migration factor to give an algorithm that has an asymptotic competitive ratio of $1+\epsilon$ and a migration factor of $2^{\mathcal{O}((1/\epsilon)\log^2(1/\epsilon))}$. This result was improved by Jansen and Klein [18] who achieved polynomial migration of $\mathcal{O}(1/\epsilon^4)$ to achieve an asymptotic competitive ratio of $1+\epsilon$.

Concerning lower bounds on the migration factor, Epstein and Levin [11] showed that no optimal solution can be maintained while having a constant migration factor (independent of $1/\epsilon$). Furthermore, Balogh, Békési, Galambos and Reinelt [2] proved that a lower bound on the asymptotic competitive ratio of 1.3877 holds, if one is only allowed to repack a *constant number of items.*

Dynamic Bin Packing

An extension to the classical online bin packing model was given by Coffman, Garey and Johnson [8], called the *dynamic bin packing* model. In addition to the insertion of items,

items also depart over time. *No repacking is allowed* in this model. It is easily seen that no algorithm can achieve a constant asymptotic competitive ratio in this setting. In order to measure the performance of an online algorithm A in this case, they compared the *maximum number of bins used by* A with the *maximum number of bins used by an optimal offline algorithm*, i. e., an algorithm A in this dynamic model is called an *asymptotic* α*-competitive algorithm*, if there is a function f with $\lim_{n\to\infty} \sup\left\{\frac{f(I)}{\text{max-OPT}(I)} \mid \text{max-}\textsc{opt}(I) = n\right\} = 0$ where max-$\textsc{opt}(I) = \max_t \textsc{opt}(I(t))$ such that $\max_t A(I(t)) \leq \alpha \cdot \max_t \textsc{opt}(I(t)) + f(I)$ for all instances I. The minimum of all such α is called the *asymptotic competitive ratio of* A. Coffman, Garey and Johnson modified the FirstFit algorithm and proved that its asymptotic competitive ratio is at most 2.897. Furthermore, they showed a lower bound of 2.5 on the asymptotic competitive ratio when the performance of the algorithm is compared to a repacking optimal offline algorithm, i. e., $\max_t \textsc{opt}(I(t))$.

In the case that the performance of the algorithm is compared to an optimal non-repacking offline algorithm, Coffman, Garey and Johnson showed a lower bound of 2.388. This bound on the non-repacking optimum was improved to 2.428 by Chan, Lam and Wong [6], to 2.5 by Chan, Wong and Yung [7] and finally to $8/3 \approx 2.666$ by Wong, Yung and Burcea [38].

Fully Dynamic Bin Packing

We consider the dynamic bin packing when repacking of already packed items is allowed. This model was first investigated by Ivković and Lloyd [16] and is called *fully dynamic bin packing*. In this model, items arrive and depart in an online fashion and limited repacking is allowed. The quality of an algorithm is measured by the asymptotic competitive ratio as defined in the classical online model (no maximum is taken as in the dynamic bin packing model). Ivković and Lloyd developed an algorithm that uses amortized $\mathcal{O}(\log n)$ many shifting moves (see definition above) to achieve an asymptotic competitive ratio of $5/4$.

Related Results on the Migration Factor

Since the introduction of the migration factor, several problems were considered in this model and different algorithms for these problems have been developed. Following the terminology of Sanders, Sivadasan and Skutella [29] we sometimes use the term *approximation ratio* instead of competitive ratio. Hence, we also use the terms asymptotic polynomial time approximation scheme (APTAS) and asymptotic fully polynomial time approximation scheme (AFPTAS) in the context of online algorithms. If the migration factor of an algorithm A only depends upon the approximation ratio ϵ and not on the size of the instance, we call A *robust*.

In the case of online bin packing, Epstein and Levin [11] developed the first robust APTAS for the problem using a migration factor of $2^{\mathcal{O}((1/\epsilon^2)\log(1/\epsilon))}$. They also proved that there is no online algorithm for this problem that has a constant migration factor and that maintains an optimal solution. The APTAS by Epstein and Levin was later improved by Jansen and Klein [18], who developed a robust AFPTAS for the problem with migration factor $\mathcal{O}(1/\epsilon^4)$. In their paper, they developed new linear program (LP)/integer linear program (ILP) techniques, which we make use of to obtain polynomial migration. It was shown by Epstein and Levin [12] that their APTAS for bin packing can be generalized to packing d-dimensional cubes into unit cubes. Sanders, Sivadasan and Skutella [29] developed a robust polynomial time approximation scheme (PTAS) for the scheduling problem on identical machines with a migration factor of $2^{\mathcal{O}((1/\epsilon)\log^2(1/\epsilon))}$. Skutella and Verschae [31] studied the problem of maximizing the minimum load given n jobs and m identical machines. They also considered a dynamic setting, where jobs may also depart. They showed that there is no robust PTAS for this machine covering problem with constant migration. The main reason for the nonexistence is due to very small jobs. By using an amortized migration factor, they developed a PTAS for the problem with amortized migration of $2^{\mathcal{O}((1/\epsilon)\log^2(1/\epsilon))}$.

1.2 Our Contributions

Main Result

In this work, we investigate the *fully dynamic bin packing* model. We measure the amount of repacking by the *migration factor*; but our algorithm uses a bounded number of shifting moves as well. Since the work of Ivković and Lloyd from 1998 [16], no progress was made on the fully dynamic bin packing problem concerning the asymptotic competitive ratio of $5/4$. It was also unclear whether the number of shifting moves (respectively migration factor) must depend on the number of packed items n. In this paper we give positive answers for both of these concerns. We develop an algorithm that provides at each time step t an approximation guarantee of $(1+\epsilon)\,\mathrm{OPT}(I(t)) + \mathcal{O}(1/\epsilon \log(1/\epsilon))$. The algorithm uses a migration factor of $\mathcal{O}(1/\epsilon^4 \cdot \log(1/\epsilon))$ by repacking at most $\mathcal{O}(1/\epsilon^3 \cdot \log(1/\epsilon))$ bins. Hence, the generated solution can be arbitrarily close to the optimum solution, and for every fixed ϵ the provided migration factor is constant (it does not depend on the number of packed items). The running time is polynomial in n and $1/\epsilon$. In case that no deletions are used, the algorithm has a migration factor of $\mathcal{O}(1/\epsilon^3 \cdot \log(1/\epsilon))$, which beats the best known migration factor of $\mathcal{O}(1/\epsilon^4)$ by Jansen and Klein [18]. Since the number of repacked bins is bounded, so is the number of shifting moves as it requires at most $O(1/\epsilon)$ shifting moves to repack a single bin. Furthermore, we prove that there is no asymptotic approximation scheme for the online bin packing problem with a migration factor of $o(1/\epsilon)$ even in the case that no items depart (and even if $\mathcal{P} = \mathcal{NP}$).

Technical Contributions

We use the following techniques to achieve our results:

- In order to obtain a lower bound on the migration factor in Section 2, we construct a series of instances that provably need migration of $\Omega(1/\epsilon)$ in order to have an asymptotic approximation ratio of $1+\epsilon$.
- In Section 3, we show how to handle large items in a fully dynamic setting. The fully dynamic setting involves more difficulties in the rounding procedure, in contrast to the setting where large items may not depart, treated in [18]. A simple adaption of the dynamic techniques developed in [18] does not work (see introduction of Section 3). We modify the offline rounding technique by Karmarkar and Karp [24] such that a feasible rounding structure can be maintained when items are inserted or removed. This way, we can make use of the LP-techniques developed in Jansen and Klein [18].
- In Section 4, we explain how to deal with small items in a dynamic setting. In contrast to the setting where departure of items is not allowed, the fully dynamic setting provides major challenges in the treatment of small items. An approach is thus developed where small items of similar size are packed near each other. We describe how this structure can be maintained as new items arrive or depart. Note that the algorithm of Ivković and Lloyd [16] relies on the ability to manipulate up to $\Omega(n)$ very small items in constant time. See also their updated work for a thorough discussion of this issue [17].
- In order to unify the different approaches for small and large items in Section 4.2, we develop an advanced structure for the packing. We give novel techniques and ideas to manage this mixed setting of small and large items. The advanced structure makes use of a potential function, which bounds the number of bins that need to be reserved for incoming items.

Several proofs in this extended abstract are removed due to space constraints. The full version[1] contains all of these proofs.

2 Lower Bound

We start our investigation by analyzing the connection between the approximation ratio and the migration factor of an algorithm. Intuitively, one would expect that a small migration factor c leads to a worse approximation ratio r_c. Whether the dependency between those term is logarithmic, linear, quadratic or something completely different is unclear. A simple argument shows that the dependency is at least linear, i. e., there is no robust asymptotic approximation scheme for bin packing with migration factor of $o(1/\epsilon)$, even if $\mathcal{P} = \mathcal{NP}$. This improves upon the lower bound given by Epstein and Levin [11], which states that no algorithm for online bin packing, that maintains an optimal solution can have constant migration.

▶ **Theorem 1.** *For a fixed migration factor $\gamma > 0$, there is no robust approximation algorithm for bin packing with asymptotic approximation ratio better than $1 + \frac{1}{6\lceil\gamma\rceil+5}$.*

▶ **Corollary 2.** *There is no robust asymptotic approximation scheme for bin packing with a migration factor $\gamma \leq 1/6(1/\epsilon - 11) = \Theta(1/\epsilon)$.*

3 Dynamic Rounding

The goal of this section is to give a robust AFPTAS for the case that only large items arrive and depart. In the first subsection we present a general rounding structure. In the second subsection we give operations that modify the rounding in a way that the general structure is preserved. We give the final algorithm for the insertion and departure of large items in Section 3.3. Finally, the correctness is proved by using the LP/ILP techniques developed in [18].

In [18], the last two authors developed a dynamic rounding technique based on an offline rounding technique from Fernandez de la Vega and Lueker [13]. However, a simple adaption of these techniques does not work in the dynamic case where items may also depart. In the case of the offline rounding by Fernandez de la Vega and Lueker, items are sorted and then collected in groups of the same cardinality. As a new item arrives in an online fashion, this structure can be maintained by inserting the new item to its corresponding group. By shifting the largest item of each group to the left, the cardinality of each group (except for the first one) can be maintained. However, shifting items to the right whenever an item departs leads to difficulties in the LP/ILP techniques. As the rounding for a group may increase, patterns of the existing LP/ILP solution might become infeasible. We overcome these difficulties by developing a new dynamic rounding structure and operations based on the offline rounding technique by Karmarkar and Karp [24]. We felt that this dynamic rounding technique is easier to analyze since the structure can essentially be maintained by shifting items, instead of the use of multiple complex operations as in [18].

A bin packing instance consists of a set of *items* $I = \{i_1, i_2, \ldots, i_n\}$ with *size function* $s : I \to [0, 1] \cap \mathbb{Q}$. A feasible solution is a partition $B^1, \ldots, B^k$ of I such that $\sum_{i \in B^j} s(i) \leq 1$ for $j = 1, \ldots, k$. We call a partition $B^1, \ldots, B^k$ a *packing* and a single set B^j is called a *bin*. The goal is to find a solution with a minimal number of bins. If the item i is packed into the

[1] The full version is available at `http://arxiv.org/abs/1411.0960`.

bin B^j, we write $B(i) = j$. The smallest value $k \in \mathbb{N}$ such that a packing with k bins exists is denoted by $\textsc{opt}(I, s)$ or if the size function is clear by $\textsc{opt}(I)$. A trivial, yet useful lower bound on $\textsc{opt}(I, s)$ is given by the value $\textsc{size}(I, s) = \sum_{i \in I} s(i)$.

3.1 Rounding

First, we divide the set of items into *small* ones and *large* ones. An item i is called *small* if $s(i) < \epsilon/14$, otherwise it is called *large*. Instance I is partitioned accordingly into a set of large items I_L and a set of small items I_S. We treat small items and large items differently. Small items can be packed using an algorithm presented in Section 4.1 while large items will be assigned using an ILP. In this section we discuss how to handle large items.

To obtain a LP formulation of fixed (independent of $|I|$) dimension, we use a rounding technique based on the offline AFPTAS by Karmarkar and Karp [24]. Here, large items are put into categories such that each item in category ℓ has size $\in (2^{-(\ell+1)}, 2^{-\ell}]$. In each of those categories, the items are sorted by their size and the first $2^\ell \cdot k$ items are put into the first group, the second $2^\ell \cdot k$ items into the second group and so on. The size of every item in a group is rounded to the maximal size in this group. Depending on a suitable choice of k, there are at most $\mathcal{O}(1/\epsilon \log(1/\epsilon))$ groups and the additive error produced by the rounding is bounded by ϵ.

In order to use this rounding technique for our dynamic setting, we generalize their rounding by having groups of size $2^\ell \cdot k$ (the A-block) and groups of size $2^\ell(k-1)$ (the B-block). This generalized rounding has a certain structure that is maintained throughout the algorithm and guarantees an approximate solution for the original instance. As in [24], we characterize the set of large items more precisely by their sizes. We say that an item i is in *size category* ℓ if $s(i) \in (2^{-(\ell+1)}, 2^{-\ell}]$. Denote the set of all size categories by W. As every large item has size at least $\epsilon/14$, the number of size categories is bounded by $|W| \leq \log(1/\epsilon) + 5$. Next, items of the same size category are characterized by their *block*, which is either A or B and their *position* $r \in \mathbb{N}$ in this block. Therefore, we partition the set of large items into a set of groups $G \subseteq W \times \{A, B\} \times \mathbb{N}$. A group $g \in G$ thus consists of a triple (ℓ, X, r) with size category $\ell \in W$, block $X \in \{A, B\}$ and position $r \in \mathbb{N}$. The *rounding function* $R : I_L \mapsto G$ maps each large item $i \in I_L$ to a *group* $g \in G$. By $g[R]$ we denote the set of items being mapped to the group g, i.e., $g[R] = \{i \in I_L \mid R(i) = g\}$.

Let $q(\ell, X)$ be the maximal $r \in \mathbb{N}$ such that $|(\ell, X, r)[R]| > 0$, i.e., $q(\ell, X)$ is the last position in block X (with respect to the size category ℓ). If (ℓ, X_1, r_1) and (ℓ, X_2, r_2) are two different groups, we say that (ℓ, X_1, r_1) is *left* of (ℓ, X_2, r_1), if $X_1 = A$ and $X_2 = B$ or $X_1 = X_2$ and $r_1 < r_2$. We say that (ℓ, X_1, r_1) is *right* of (ℓ, X_2, r_2) if it is not left of it.

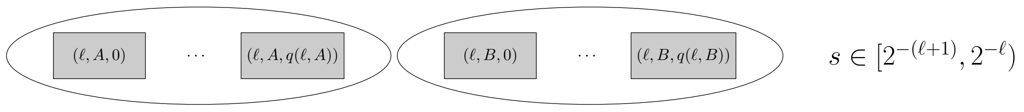


Figure 2 Grouping in $(\ell, A, \cdot)$ and $(\ell, B, \cdot)$.

Given an instance (I, s) and a rounding function R, we define the rounded size function s^R by rounding the size of every large item i up to the size of the largest item in its group, i.e. $s^R(i) = \max_{i'} \{s(i') \mid R(i') = R(i)\}$. We denote by $\textsc{opt}(I, s^R)$ the value of an optimal solution of the rounded instance (I, s^R).

Depending on a parameter k, we define the following properties for a rounding function R. Property (a) guarantees that the items are categorized correctly according to their sizes.

Property (b) guarantees that items of the same size category are sorted by their size and properties (c) and (d) define the number of items in each group.

(a) For each $i \in (\ell, X, r)[R]$ we have $2^{-(\ell+1)} < s(i) \leq 2^{-\ell}$.

(b) For each $i \in (\ell, X, r)[R]$ and each $i' \in (\ell, X, r')[R]$ and $r < r'$, we have $s(i) \geq s(i')$.

(c) For each $\ell \in W$ and $1 \leq r \leq q(\ell, A)$ we have that $|(\ell, A, r)[R]| = 2^{\ell}k$ and $|(\ell, A, 0)[R]| \leq 2^{\ell}k$.

(d) For each $\ell \in W$ and $0 \leq r \leq q(\ell, B) - 1$ we have $|(\ell, B, r)[R]| = 2^{\ell}(k-1)$ and $|(\ell, B, q(\ell, B))[R]| \leq 2^{\ell}(k-1)$.

The following lemma shows that the number of groups is bounded and the rounding function does in fact yield a $(1+\epsilon)$-approximation for a suitable choice of k.

▶ **Lemma 3.** *If* $\text{SIZE}(I_L, s) > {}^{8}/_{\epsilon} \cdot (\lceil \log({}^{1}/_{\epsilon}) \rceil + 5)$ *and* $k = \left\lfloor \frac{\text{SIZE}(I_L, s) \cdot \epsilon}{2(\lfloor \log({}^{1}/_{\epsilon}) \rfloor + 5)} \right\rfloor$*, the number of non-empty groups is bounded by* $\mathcal{O}({}^{1}/_{\epsilon} \log({}^{1}/_{\epsilon}))$.

▶ **Lemma 4.** *Given an instance* (I_L, s) *with items greater than* $\epsilon/14$ *and a rounding function* R *fulfilling properties (a) to (d), then* $\text{OPT}(I_L, s^R) \leq (1+\epsilon) OPT(I_L, s)$.

3.2 Rounding Operations

Let us consider the case where large items arrive and depart in an online fashion. Formally this is described by a sequence of pairs $(i_1, A_1), \ldots, (i_n, A_n)$ where $A_i \in \{\text{Insert}, \text{Delete}\}$. At each time $t \in \{1, \ldots, n\}$ we need to pack the item i_t into the corresponding packing of $i_1, \ldots, i_{t-1}$ if $A_i = \text{Insert}$ or remove the item i_t from the corresponding packing of $i_1, \ldots, i_{t-1}$ if $A_i = \text{Delete}$. We will denote the instance $i_1, \ldots, i_t$ at time t by $I(t)$ and the corresponding packing by B_t. We will also round our items and denote the rounding function at time t by R_t. The large items of $I(t)$ are denoted by $I_L(t)$. At time t we are allowed to repack several items with a total size of $\beta \cdot s(i_t)$ but we intend to keep the migration factor β as small as possible. The term $\text{repack}(t) = \sum_{i, B_{t-1}(i) \neq B_t(i)} s(i)$ denotes the sum of the items which are moved at time t, the *migration factor* β of an algorithm is then defined as $\max_t \{{}^{\text{repack}(t)}/_{s(i_t)}\}$. As the value of SIZE will also change over the time, we define the value $\kappa(t)$ as

$$\kappa(t) = \frac{\text{SIZE}(I_L(t), s) \cdot \epsilon}{2(\lfloor \log({}^{1}/_{\epsilon}) \rfloor + 5)}.$$

As shown in Lemma 3, we will make use of the value $k(t) := \lfloor \kappa(t) \rfloor$.

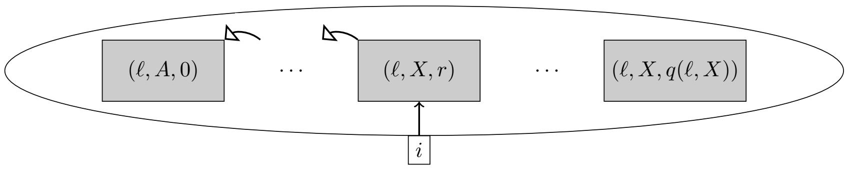
We present operations that transform the current rounding R_t, the current packing B_t with its corresponding LP/ILP solutions into a new rounding R_{t+1}, a new packing B_{t+1} and new corresponding LP/ILP solutions for the new instance $I(t+1)$. At every time t the rounding R_t maintains properties (a) to (d). Therefore the rounding provides an asymptotic approximation ratio of $1+\epsilon$ (Lemma 4) while maintaining only $\mathcal{O}({}^{1}/_{\epsilon} \log({}^{1}/_{\epsilon}))$ many groups (Lemma 3). We will now present a way how to adapt this rounding to a dynamic setting, where items arrive or depart online.

Our rounding R_t is manipulated by different *operations*, called the *insert, delete, shiftA* and *shiftB* operation. Some ideas behind the operations are inspired by the operations described by Epstein and Levin [11]. The insert operation is performed whenever a large item arrives and the delete operation is performed whenever a large item departs. The shift operations are used to modify the number of groups that are contained in the A and B block. As we often need to filter the largest items of a group g in a rounding R, we denote this item by $\lambda(g, R)$. Due to space constraints we do not describe how the LP/ILP solutions are modified by the operations (see full version). Intuitively, an insert operation finds the group

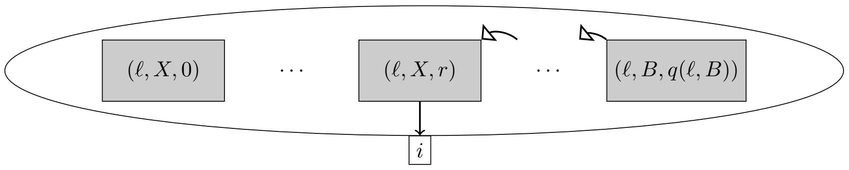
g, where the new item needs to be placed and inserts it into g. In order to maintain the size of g, the largest item from g is shifted to its left neighbour g'. The largest item of g' is then shifted to its left neighbour and so on. The deletion algorithm removes the item from its group g. In order to maintain the size of g, the largest item from its right neighbour g' is shifted into g. The largest item of the right neighbour of g' is then shifted into g' and so on.

- Insert: To insert item i_t, find the corresponding group (ℓ, X, r) with
  - $s(i_t) \in (2^{-(\ell+1)}, 2^{-\ell}]$,
  - $\min_i \{s(i) \mid i \in (\ell, X, r-1)\} > s(i_t)$ and
  - $s(\lambda((\ell, X, r+1), R)) \leq s(i_t)$.

  We will then insert i_t into (ℓ, X, r) and get the rounding R' by shifting the largest element of (ℓ, X, r) to $(\ell, X, r-1)$ and the largest item of $(\ell, X, r-1)$ to $(\ell, X, r-2)$ and so on until the first group $(\ell, A, 0)$ is reached.



(a) Insert i into $(\ell, X, \cdot)$



(b) Delete i from $(\ell, X, \cdot)$

Figure 3 Insert and Delete.

- Delete: To delete an item i_t that is in group (ℓ, X, r), we remove i_t from this group and move the largest item from $(\ell, X, r+1)$ into (ℓ, X, r) and the largest item from $(\ell, X, r+2)$ into $(\ell, X, r+1)$ and so on until the last group $(\ell, B, q(\ell, B))$ is reached.

To control the number of groups in A and B we introduce operations shiftA and shiftB that increase or decrease the number of groups in A respectively B. An operation shiftA increases the number of groups in A by 1 and decreases the number of groups in B by 1 by shifting a group from block B to block A. The operation shiftB moves a group From block A to block B. The ability to modify the blocks is needed later on, as the value k used by the rounding changes over time.

- shiftA: In order to move a group from B to A, shift exactly 2^ℓ items from $(\ell, B, q(\ell, B))$ to $(\ell, B, q(\ell, B) - 1)$. Then shift exactly 2^ℓ items from $(\ell, B, q(\ell, B) - 1)$ to $(\ell, B, q(\ell, B) - 2)$ and so on until $(\ell, B, 0)$ is reached. The group $(\ell, B, 0)$ has now the same size as the groups in $(\ell, A, \cdot)$. We transfer $(\ell, B, 0)$ to block A. Note that the total size of the 2^ℓ items is bounded by 1.
- shiftB: In order to move a group from A to B, shift 2^ℓ items from $(\ell, A, q(\ell, A))$ to $(\ell, A, q(\ell, A) - 1)$. Then shift exactly 2^ℓ items from $(\ell, A, q(\ell, A) - 1)$ to $(\ell, A, q(\ell, A) - 2)$ and so on until $(\ell, A, 0)$ is reached. The group $(\ell, A, q(\ell, A))$ has now the same size as the groups in $(\ell, B, \cdot)$. We transfer $(\ell, A, q(\ell, A))$ to block B.

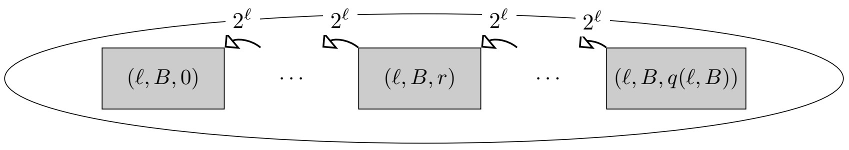


Figure 4 shiftA.

The next lemma shows that all of these operations maintain the desired properties of the rounding.

▶ **Lemma 5.** *Let R be a rounding function fulfilling properties (a) to (d). Applying any of the operations insert, delete, shiftA or shiftB on R results in a rounding function R' fulfilling properties (a) to (d).*

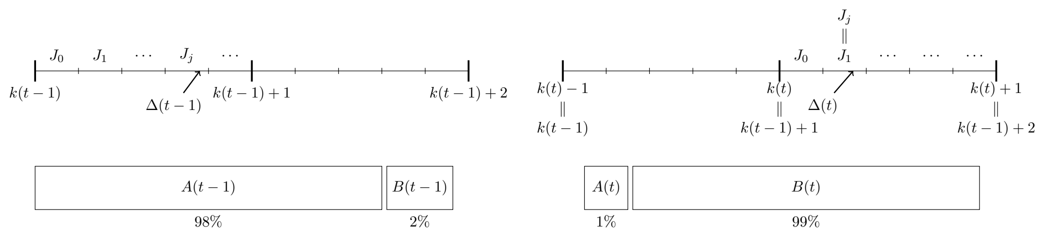
According to Lemma 3 the rounded instance (I, s^R) has $\mathcal{O}(1/\epsilon \log(1/\epsilon))$ different item sizes (given a suitable k). Using the LP formulation of Eisemann [10], the resulting LP called $LP(I, s^R)$ has $m = \mathcal{O}(1/\epsilon \log(1/\epsilon))$ constraints. We say a packing B *corresponds* to a rounding R and an integral solution y of the ILP if all items in (I, s^R) are packed by B according to y. The operations also modify these ILP solutions.

▶ **Lemma 6.** *Applying any of the operations insert, delete, shiftA or shiftB on a rounding R and ILP solution y with corresponding packing B defines a new rounding R' and a new integral solution y'. The solution y' is a feasible solution of the linear program $LP(I, s^{R'})$.*

3.3 Algorithm for Dynamic Bin Packing

We will use the operations from the previous section to obtain a dynamic algorithm for bin packing with respect to large items. The operations insert and delete are designed to process the input depending of whether an item is to be inserted or removed. Keep in mind that the parameter $k(t) = \lfloor \kappa(t) \rfloor = \left\lfloor \frac{\text{SIZE}(I_L(t)) \cdot \epsilon}{2(\lfloor \log(1/\epsilon) \rfloor + 5)} \right\rfloor$ changes over time as $\text{SIZE}(I_L(t))$ may increase or decrease. In order to fulfill the properties (c) and (d), we need to adapt the number of items per group whenever k changes. The shiftA and shiftB operations are thus designed to manage the dynamic number of items in the groups as k changes. Note that a group in the A-block with parameter k has by definition the same number of items as a group in the B-block with parameter $k-1$ if they are in the same size category. If k increases, the former A block is treated as the new B block in order to fulfill the properties (c) and (d) while a new empty A block is introduced. To be able to rename the blocks, the B block needs to be empty. Accordingly the A block needs to be empty if k decreases in order to treat the old B block as new A block. Hence we need to make sure that there are no groups in the B-block if k increases and vice versa, that there are no groups in the A-block if k decreases.

We denote the number of all groups in the A-blocks at time t by $A(t)$ and the number of groups in B-blocks at time t by $B(t)$. To make sure that the B-block (respectively the A-block) is empty when k increases (decreases) the ratio $\frac{A(t)}{A(t)+B(t)}$ needs to correlate to the fractional digits of $\kappa(t)$ at time t denoted by $\Delta(t)$. For example, if $\Delta(t) = 0.98$, nearly every group must be in an A-block, as the B-blocks need to be empty if $k(t)$ increases (see Figure 5(a) for a sketch of the situation). Note that the term $\frac{A(t)}{A(t)+B(t)}$ is 0 if the A-block is empty and the term is 1 if the B-block is empty. This way, we can make sure that as soon as $k(t)$ increases, the number of B-blocks is close to 0 and as soon as $k(t)$ decreases, the number of A-blocks is close to 0. Therefore, the blocks can be renamed whenever $k(t)$



(a) Before Insert **(b)** After Insert

Figure 5 Comparison of the situation before and after an Insert Operation.

changes. Hence we partition the interval $[0,1)$ into exactly $A(t)+B(t)$ smaller intervals $J_i = \left[\frac{i}{A(t)+B(t)}, \frac{i+1}{A(t)+B(t)}\right)$. We will make sure that $\Delta(t) \in J_i$ iff $\frac{A(t)}{A(t)+B(t)} \in J_i$.

The algorithm inserts an item via the insert operation and then uses shiftA and shiftB operations to adjust the number of A- and B-blocks. Recall that a shiftA operation reduces the number of groups in the B-block by 1 and increases the number of groups in the A-block by 1 (shiftB works vice versa).

In the following algorithm we also make use of an algorithm called IMPROVE, which was developed in [18] to reduce the number of used bins. Using IMPROVE(x) on a packing B with approximation guarantee $\max_i B(i) \le (1+\bar{\epsilon})\,\mathrm{OPT} + C$ for some $\bar{\epsilon} = \mathcal{O}(\epsilon)$ and some additive term C yields a new packing B' with approximation guarantee $\max_i B(i) \le (1+\bar{\epsilon})\,\mathrm{OPT} + C - x$. We use the operations in combination with IMPROVE to obtain a fixed approximation guarantee. The similar deletion algorithm can be found in the full version.

▶ **Algorithm 1** (AFPTAS for large items).

```
Algorithm: Insertion
if SIZE(I(t)) < $(m+2)(1/\delta+2)$ or SIZE(I(t)) < $8(1/\delta+1)$ then
    use offline Bin Packing;
else
    IMPROVE(2); insert(i);
    // Shifting to the correct interval
    Let $J_i$ be the interval containing $\Delta(t)$;
    Let $J_j$ be the interval containing $\frac{A(t)}{A(t)+B(t)}$;
    Set $d = i - j$;
    if $k(t) > k(t-1)$ then // Modulo $A(t)+B(t)$ when $k$ increases
        $d = d + (A(t)+B(t))$;
    for $p := 0$ to $|d|-1$ do // Shifting $d$ groups from $B$ to $A$
        if $i+p = A(t) + B(t)$ then
            Rename(A, B);
        IMPROVE(1); shiftA;
```

Note that as exactly $d = i - j$ groups are shifted from A to B (or B to A) we have by definition that $\Delta(t) \in \left[\frac{A(t)}{A(t)+B(t)}, \frac{A(t)+1}{A(t)+B(t)}\right)$ at the end of the algorithm. The following lemmas prove that the algorithm works as expected by moving only a constant number of groups.

▶ **Lemma 7.** *At most* 11 *groups are shifted from A to B (or B to A) in Algorithm 1.*

▶ **Lemma 8.** *Every rounding function R_t produced by Algorithm 1 fulfills properties* (a) *to* (d) *with parameter* $k(t) = \left\lfloor \frac{\textsc{size}(I_L(t))\cdot\epsilon}{2(\lceil\log(1/\epsilon)\rceil+5)} \right\rfloor$.

Using analysing techniques developed in [18] we prove the following theorem.

▶ **Theorem 9.** *Algorithm 1 is an AFPTAS with migration factor $\mathcal{O}(\frac{1}{\epsilon^3}\cdot\log(1/\epsilon))$ for the fully dynamic bin packing problem with respect to large items.*

If no deletions are present, we can use a simple FirstFit algorithm (as described by Jansen and Klein [18]) to pack the small items into the bins. This does not change the migration factor or the running time of the algorithm and we obtain a robust AFPTAS with $\mathcal{O}(\frac{1}{\epsilon^3}\cdot\log(1/\epsilon))$ migration for the case that no items depart. This improves the best known migration factor of $\mathcal{O}(\frac{1}{\epsilon^4})$ [18].

4 Handling Small Items

In this section we present methods for dealing with arbitrary small items in a dynamic setting. First, we present a robust AFPTAS with migration factor of $\mathcal{O}(1/\epsilon)$ for the case that only small items arrive and depart. In Section 4.2 we generalize these techniques to a setting where small items arrive into a packing, that already contains large items which can not be rearranged. Finally we state the AFPTAS for the general fully dynamic bin packing problem. In a robust setting without departing items, small items can easily be treated by packing them greedily via the classical FirstFit algorithm of Johnson et al. [21] (see Epstein and Levin [11] or Jansen and Klein [18]). However, in a setting where items may also depart, small items need much more attention. We show in the full version that the FirstFit algorithm does not work in this dynamic setting.

▶ **Lemma 10.** *Using the* FirstFit *algorithm to pack small items may lead to an arbitrarily bad approximation.*

4.1 Only Small Items

We consider a setting where only small items exist, i.e., items with a size less than $\epsilon/14$. One way to overcome the problematic instances, like those in Lemma 10, is to make sure that the small items remain ordered. If one has m bins $b_1, b_2, \ldots, b_m$, one can make sure that the larger small items are in bins with lower indices as the smaller ones. A possible way to maintain such a property works as follows: Whenever a small items i arrives, find the bin with the smallest index that contains a set J of small items such that $\sum_{j\in J} s(j) \geq s(i)$. Remove those items in J from the bin and add i in this place. Insert all of the items in J in the same way. The deletion algorithm is very similar, but searches for a set J from smaller sizes, that replaces the departing item i. In order to bound the migration of this operation, we declare every $1/\epsilon$-th bin as buffer bin and terminate the procedure at the buffer bin. If the moved items do not fit into the buffer bin, we declare the buffer bin as normal and open a new buffer bin containing the remaining items. As only a fraction of ϵ of the bins are buffer bins, this worsen the approximation ratio only by ϵ. The migration remains bounded as at most $1/\epsilon$ bins are changed.

Formally, we divide the set of small items into different size intervals S_j where $S_j = \left[\frac{\epsilon}{2^{j+1}}, \frac{\epsilon}{2^j}\right)$ for $j \geq 1$. Let $b_1, \ldots, b_m$ be the used bins of our packing. We say a size category S_j is bigger than a size category S_k if $j < k$, i. e., the item sizes contained in S_j are larger (note that a size category S_j with large index j is called small). We say a bin b is filled completely

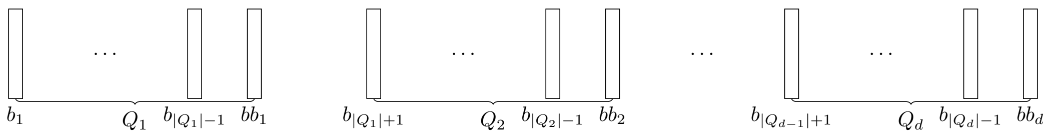


Figure 6 Distribution of bins with small items into queues.

if it has less than $\frac{\epsilon}{2^j}$ remaining space, where S_j is the biggest size category appearing in b, i.e., no item from the categories $S_1, S_2, \ldots, S_j$ fit into b. Furthermore we label bins b as *normal* or as *buffer bins* and partition all bins $b_1, \ldots, b_m$ into *queues* $Q_1, \ldots, Q_d$. A queue is a consecutive subsequence of bins $b_i, b_{i+1} \ldots, b_{i+c}$ where bins $b_i, \ldots, b_{i+c-1}$ are normal bins and bin b_{i+c} is a buffer bin. We denote the number of bins in Q_i by $|Q_i|$. The buffer bin of queue Q_i is denoted by bb_i. See Figure 6 for a sketch of the situation.

We will maintain a special form for the packing of small items such that the following properties are always fulfilled. For the sake of simplicity, we assume that $1/\epsilon$ is integral.

1. For every item $i \in b_d$ with size $s(i) \in S_j$ for some $j, d \in \mathbb{N}$, there is no item $i' \in b_{d'}$ with size $s(i') \in S_{j'}$ such that $d' > d$ and $j' > j$. This means: Items are ordered from left to right by their size intervals.
2. Every normal bin is filled completely.
3. The length of each queue is at least $1/\epsilon$ and at most $2/\epsilon$ except for the last queue Q_d.

Note that property (1) implies that all items in the same size interval S_j are packed into consecutive bins $b_x, b_{x+1}, \ldots, b_{x+c}$. Items in the next smaller category S_{j+1} are then packed into bins $b_{x+c}, b_{x+c+1}, \ldots$ and so on. We denote by $b_{S(j)}$ the last bin in which an item of category S_j appears.

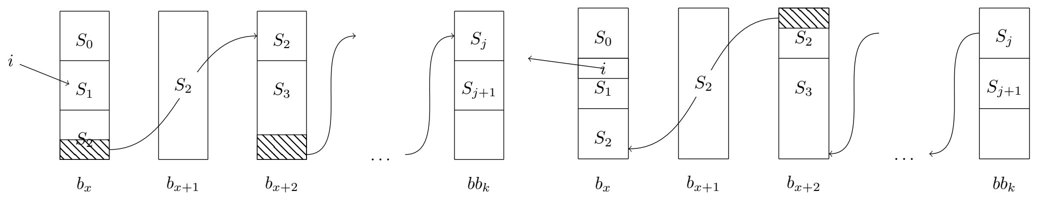
The following lemma guarantees that a packing with properties (1) to (3) is close to the optimum solution.

▶ **Lemma 11.** *If properties* (1) *to* (3) *hold, then at most* $(1 + \mathcal{O}(\epsilon))\,\textsc{opt}(I, s) + 2$ *bins are used in the packing for every* $\epsilon \leq 1/3$.

We will now describe the operations that are applied whenever a small item is inserted or removed from the packing. The operations are designed such that properties (1) to (3) are maintained. Lemma 11 thus guarantees a good approximation ratio at every step of the algorithm. The operations are applied recursively such that some items from each size interval are shifted from left to right (insert) or right to left (delete). The recursion halts if the first buffer bin is reached. Therefore, the free space in the buffer bins will change over time. Since the recursion always halts at the buffer bin, the algorithm is applied on a single queue Q_k.

The following Insert/Delete operation is defined for a whole set $J = \{i_1, \ldots, i_n\}$ of items. If an item i of size interval S_ℓ has to be inserted or deleted, the algorithm is called with Insert($\{i\}, b_{S(\ell)}, Q_k$) respectively Delete($\{i\}, b_x, Q_k$), where b_x is the bin containing item i and Q_k is the queue containing bin $b_{S(\ell)}$ or b_x. Recall that $S_j = \left[\frac{\epsilon}{2^{j+1}}, \frac{\epsilon}{2^j}\right)$ is a fixed interval for every $j \geq 1$ and define $S_{\leq j}, S_{>j}$ as $S_{\leq j} = \bigcup_{i=1}^{j} S_i$ and $S_{>j} = \bigcup_{i>j} S_i$.

Figure 7a shows an example call of Insert($\{i\}$,b_x,Q_k). Item i with $s(i) \in S_1$ is put into the corresponding bin b_x into the size interval S_1. As b_x now contains too many items, some items from the smallest size interval S_2 (marked by the dashed lines) are put into the last bin b_{x+2} containing items from S_2. Those items in turn push items from the smallest size interval S_3 into the last bin containing items of this size and so on. This process terminates if either no items need to be shifted to the next bin or the buffer bin bb_k is reached.



(a) Insert($\{i\}$,b_x,Q_k) with $s(i) \in S_1$ **(b)** Delete($\{i\}$,b_x,Q_k) with $s(i) \in S_1$

Figure 7 Example calls of Insert and Delete.

▶ **Algorithm 2** (Insert/Delete: only small items).

- ***Insert***(J, b_x, Q_k)***:***
  - *For $J = \{i_1, \ldots, i_n\}$, find the smallest ℓ such that $s(i_j) \in S_{\leq \ell}$ for all $j \geq 1$ and insert those items into bin b_x. (By Lemma 12 the total size of J is bounded by $\mathcal{O}(^1/_\epsilon)$ times the size of the item which triggered the first Insert operation.)*
  - *Remove just as many items $J' = \{i'_1, \ldots, i'_m\}$ of the smaller size intervals $S_{>\ell}$ appearing in bin b_x (starting by the smallest) such that all items J fit into the bin b_x. If there are not enough items of smaller categories to insert all items from J, insert the remaining items from J into b_{x+1}.*
  - *Partition J' into $J'_{\ell+1}, J'_{\ell+2}, \ldots$ such that $J_{\ell+i}$ contains the items in the respective size interval $S_{\ell+i}$. Put $J'_{\ell+i}$ recursively into bin $b_{S(\ell+i)}$ (i. e., call Insert$(J'_{\ell+i}, b_{S(\ell+i)}, Q_k)$ for each $i \geq 1$). If the buffer bin bb_k is left of $b_{S(\ell+i)}$ call Insert$(J'_{\ell+i}, bb_k, Q_k)$ instead.*
- ***Delete***(J, b_x, Q_k)***:***
  - *For $J = \{i_1, \ldots, i_n\}$, find the smallest ℓ such that $s(i_j) \in S_{\leq \ell}$ for all $j \geq 1$ and remove those from bin b_x (By Lemma 12 the total size of J is bounded by $\mathcal{O}(^1/_\epsilon)$ times the size of the item which triggered the first Delete operation.)*
  - *Let $S_{\ell'}$ be the smallest size interval appearing in b_x. Insert as many items $J' = \{i'_1, \ldots, i'_m\}$ from $b_{S(\ell')}$ such that b_x is filled completely. If there are not enough items from the size category $S_{\ell'}$, choose items from the next size category $S_{\geq \ell'+1}$ in bin b_{x+1}.*
  - *Partition J' into $J'_{\ell+1}, J'_{\ell+2}, \ldots$ such that $J_{\ell+i}$ contains the items in the respective size interval $S_{\ell+i}$. Remove items $J'_{\ell+i}$ from bin $b_{S(\ell+i)}$ recursively (i. e., call Delete$(J'_{\ell+i}, b_{S(\ell+i)}, Q_k)$ for each $i \geq 1$). If the buffer bin bb_k is left of $b_{S(\ell+i)}$, call Delete$(J'_{\ell+i}, bb_k, Q_k)$ instead.*

Using the above operations, the normal bins are filled completely. However, the size of the items in the buffer bins changes. In the following we describe how to handle buffer bins that are being emptied or filled completely. If a buffer bin is filled completely, we add a new empty buffer bin and split the queue if it contains too many bins. Similarly, if a buffer bin is emptied completely, we remove it and label the last bin of the queue as buffer bin. If the queue now contains too few bins, we merge the queue with its successor. There is thus no need in introducing a new buffer bin, as one can simply use the buffer bin of the successor as the new buffer bin.

▶ **Algorithm 3** (Handle filled/emptied buffer bins).

- **Case 1: The buffer bin of Q_i is filled completely by an insert operation.**
  - *Label the filled bin as a normal bin and add a new empty buffer bin to the end of Q_i.*
  - *If $|Q_i| > ^2/_\epsilon$, split Q_i into two queues Q'_i, Q''_i with $|Q''_i| = |Q'_i| + 1$. The buffer bin of Q''_i is the newly added buffer bin. Add an empty buffer bin to Q'_i such that $|Q'_i| = |Q''_i|$.*
- **Case 2: The buffer bin of Q_i is being emptied due to a delete operation.**

- *Remove the now empty bin.*
- *If $|Q_i| \geq |Q_{i+1}|$ and $|Q_i| > 1/\epsilon$, choose the last bin of Q_i and label it as new buffer bin of Q_i.*
- *If $|Q_{i+1}| > |Q_i|$ and $|Q_{i+1}| > 1/\epsilon$, choose the first bin of Q_{i+1}, move it to Q_i and label it buffer bin.*
- *If $|Q_{i+1}| = |Q_i| = 1/\epsilon$, merge queues Q_i and Q_{i+1}. As Q_{i+1} already contains a buffer bin, there is no need to label another bin as buffer bin for the merged queue.*

Creating and deleting buffer bins this way guarantees that property (3) is never violated since queues never exceed the length of $2/\epsilon$ and never fall below $1/\epsilon$.

It remains to prove that the migration of the operations is bounded and that the properties are invariant under those operations.

▶ **Lemma 12.**

(i) *Let B be a packing that fulfills properties (1) to (3). Applying Algorithm 2 on B yields a packing B' that also fulfills properties (1) to (3).*

(ii) *The migration factor of a single operation is bounded by $\mathcal{O}(1/\epsilon)$ for all $\epsilon \leq 2/7$.*

4.2 Handling the General Setting

In the scenario that there are mixed item types (small and large items), we need to be more careful in the creation and the deletion of buffer bins. To maintain the approximation guarantee, we have to make sure that as long as there are bins containing only small items, the remaining free space of all bins can be bounded. Packing small items into empty bins and leaving bins with large items untouched does not lead to a good approximation, as the free space of the bins containing only large items is not used. To tackle this problem we developed new techniques and ideas using an even more refined distribution of the bins. A bin that contains no small items is called a heap bin. The heap bins are then used as buffer bins. In order to measure the number of buffer bins that are about to get filled, we developed a potential function. The potential function is related to the number of heap bins and the free space of the last buffer bin that contains large items. This relation allows us to extend Algorithm 2 to the general case of both small and large items. The developed techniques involve a complex structure of the packing and require an intricate analysis. See the full version for details.

Combining all the results from the current and the previous section, we finally obtain our central result.

▶ **Theorem 13.** *There is a AFPTAS with a migration factor of at most $\mathcal{O}(1/\epsilon^4 \cdot \log 1/\epsilon)$ for the fully dynamic bin packing problem.*

Acknowledgements. We would like to thank Till Tantau for his valuable comments and suggestions to improve the presentation of the paper.

References

1 J. Balogh, J. Békési, and G. Galambos. New lower bounds for certain classes of bin packing algorithms. In *Workshop on Approximation and Online Algorithms(WAOA)*, volume 6534 of *LNCS*, pages 25–36, 2010.

2 J. Balogh, J. Békési, G. Galambos, and G. Reinelt. Lower bound for the online bin packing problem with restricted repacking. *SIAM Journal on Computing*, 38(1):398–410, 2008.

3 A. Beloglazov and R. Buyya. Energy efficient allocation of virtual machines in cloud data centers. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGrid 2010*, pages 577–578, 2010.

4 N. Bobroff, A. Kochut, and K.A. Beaty. Dynamic placement of virtual machines for managing SLA violations. In *Integrated Network Management, IM 2007. 10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 119–128, 2007.

5 D.J. Brown. A lower bound for on-line one-dimensional bin packing algorithms. Technical Report R-864, Coordinated Sci Lab Univ of Illinois Urbana, 1979.

6 J.W. Chan, T. Lam, and P.W.H. Wong. Dynamic bin packing of unit fractions items. *Theoretical Computer Science*, 409(3):521–529, 2008.

7 J.W. Chan, P.W.H. Wong, and F.C.C. Yung. On dynamic bin packing: An improved lower bound and resource augmentation analysis. *Algorithmica*, 53(2):172–206, 2009.

8 E.G. Coffman, M.R. Garey, and D.S. Johnson. Dynamic bin packing. *SIAM Journal on Computing*, 12(2):227–258, 1983.

9 Khuzaima D., Shahin K., and Alejandro L. On the online fault-tolerant server consolidation problem. In *26th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA'14*, pages 12–21, 2014.

10 K. Eisemann. The Trim Problem. *Management Science*, 3(3):279–284, 1957.

11 L. Epstein and A. Levin. A robust APTAS for the classical bin packing problem. *Mathematical Programming*, 119(1):33–49, 2009.

12 L. Epstein and A. Levin. Robust approximation schemes for cube packing. *SIAM Journal on Optimization*, 23(2):1310–1343, 2013.

13 W. Fernandez de la Vega and G.S. Lueker. Bin packing can be solved within $1+\epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

14 G. Gambosi, A. Postiglione, and M. Talamo. Algorithms for the relaxed online bin-packing model. *SIAM Journal on Computing*, 30(5):1532–1551, 2000.

15 Z. Ivković and E.L. Lloyd. Partially dynamic bin packing can be solved within $1+\epsilon$ in (amortized) polylogarithmic time. *Information Processing Letter*, 63(1):45–50, 1997.

16 Z. Ivković and E.L. Lloyd. Fully dynamic algorithms for bin packing: Being (mostly) myopic helps. *SIAM Journal on Computing*, 28(2):574–611, 1998.

17 Z. Ivković and E.L. Lloyd. Fully dynamic bin packing. In *Fundamental Problems in Computing*, pages 407–434. Springer, 2009.

18 K. Jansen and K. Klein. A robust AFPTAS for online bin packing with polynomial migration. In *International Colloquium on Automata, Languages, and Programming(ICALP)*, pages 589–600, 2013.

19 D.S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8(3):272–314, 1974.

20 D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, and R.L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3(4):299–325, 1974.

21 D.S. Johnson, A.J. Demers, J.D. Ullman, M.R. Garey, and R.L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3(4):299–325, 1974.

22 G. Jung, K.R. Joshi, M.A. Hiltunen, R.D. Schlichting, and C. Pu. Generating adaptation policies for multi-tier applications in consolidated server environments. In *2008 International Conference on Autonomic Computing, ICAC 2008, June 2-6, 2008, Chicago, Illinois, USA*, pages 23–32, 2008.

23 G. Jung, K.R. Joshi, M.A. Hiltunen, R.D. Schlichting, and C. Pu. A cost-sensitive adaptation engine for server consolidation of multitier applications. In *Middleware 2009,*

ACM/IFIP/USENIX, 10th International Middleware Conference, Proceedings, pages 163–183, 2009.

24 N. Karmarkar and R.M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 312–320. IEEE Computer Society, 1982.

25 C.C. Lee and D. Lee. A simple on-line bin-packing algorithm. *Journal of the ACM (JACM)*, 32(3):562–572, 1985.

26 Y. Li, X. Tang, and W. Cai. On dynamic bin packing for resource allocation in the cloud. In *26th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA'14*, pages 2–11, 2014.

27 F.M. Liang. A lower bound for on-line bin packing. *Information processing letters*, 10(2):76–79, 1980.

28 J.M. Park, Uday R. Savagaonkar, E.K.P. Chong, H.J. Siegel, and S.D. Jones. Efficient resource allocation for qos channels in mf-tdma satellite systems. In *MILCOM 2000. 21st Century Military Communications Conference Proceedings*, volume 2, pages 645–649. IEEE, 2000.

29 P. Sanders, N. Sivadasan, and M. Skutella. Online scheduling with bounded migration. *Mathematics of Operations Research*, 34(2):481–498, 2009.

30 S.S. Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5):640–671, 2002.

31 M. Skutella and J. Verschae. A robust PTAS for machine covering and packing. In *European Symposium on Algorithms(ESA)*, volume 6346 of *LNCS*, pages 36–47, 2010.

32 S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, pages 10–10, 2008.

33 A.L. Stolyar. An infinite server system with general packing constraints. *Operations Research*, 61(5):1200–1217, 2013.

34 A.L. Stolyar and Y. Zhong. A large-scale service system with packing constraints: Minimizing the number of occupied servers. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, pages 41–52. ACM, 2013.

35 J.D. Ullman. *The Performance of a Memory Allocation Algorithm.* Technical report. Princeton University, 1971.

36 A. Verma, P. Ahuja, and A. Neogi. pmapper: Power and migration cost aware application placement in virtualized systems. In *Middleware 2008, ACM/IFIP/USENIX 9th International Middleware Conference, Proceedings*, pages 243–264, 2008.

37 A. Vliet. An improved lower bound for on-line bin packing algorithms. *Information Processing Letters*, 43(5):277–284, 1992.

38 Prudence WH Wong, Fencol CC Yung, and Mihai Burcea. An 8/3 lower bound for online dynamic bin packing. In *Algorithms and Computation*, pages 44–53. Springer, 2012.

39 A.C. Yao. New algorithms for bin packing. *Journal of the ACM (JACM)*, 27(2):207–227, 1980.

Approximate Hypergraph Coloring under Low-discrepancy and Related Promises

Vijay V. S. P. Bhattiprolu[*], **Venkatesan Guruswami**[†], **and Euiwoong Lee**[‡]

School of Computer Science, Carnegie Mellon University
Gates Hillman Center, 5000 Forbes Avenue, Pittsburgh, PA-15213, USA
{vpb,guruswami,euiwoonl}@cs.cmu.edu

Abstract

A hypergraph is said to be χ-colorable if its vertices can be colored with χ colors so that no hyperedge is monochromatic. 2-colorability is a fundamental property (called Property B) of hypergraphs and is extensively studied in combinatorics. Algorithmically, however, given a 2-colorable k-uniform hypergraph, it is NP-hard to find a 2-coloring miscoloring fewer than a fraction 2^{-k+1} of hyperedges (which is trivially achieved by a random 2-coloring), and the best algorithms to color the hypergraph properly require $\approx n^{1-1/k}$ colors, approaching the trivial bound of n as k increases.

In this work, we study the complexity of approximate hypergraph coloring, for both the maximization (finding a 2-coloring with fewest miscolored edges) and minimization (finding a proper coloring using fewest number of colors) versions, when the input hypergraph is promised to have the following stronger properties than 2-colorability:

- Low-discrepancy: If the hypergraph has a 2-coloring of discrepancy $\ell \ll \sqrt{k}$, we give an algorithm to color the hypergraph with $\approx n^{O(\ell^2/k)}$ colors.

  However, for the maximization version, we prove NP-hardness of finding a 2-coloring miscoloring a smaller than $2^{-O(k)}$ (resp. $k^{-O(k)}$) fraction of the hyperedges when $\ell = O(\log k)$ (resp. $\ell = 2$). Assuming the Unique Games conjecture, we improve the latter hardness factor to $2^{-O(k)}$ for almost discrepancy-1 hypergraphs.
- Rainbow colorability: If the hypergraph has a $(k-\ell)$-coloring such that each hyperedge is polychromatic with all these colors (this is stronger than a $(\ell+1)$-discrepancy 2-coloring), we give a 2-coloring algorithm that miscolors at most $k^{-\Omega(k)}$ of the hyperedges when $\ell \ll \sqrt{k}$, and complement this with a matching Unique Games hardness result showing that when $\ell = \sqrt{k}$, it is hard to even beat the 2^{-k+1} bound achieved by a random coloring.
- Strong Colorability: We obtain similar (stronger) Min- and Max-2-Coloring algorithmic results in the case of $(k+\ell)$-strong colorability.

1998 ACM Subject Classification G.2.1 Combinatorics, G.2.2 Graph Theory, F.2.2 Nonnumerical Algorithms and Problems, G.1.6 Optimization

Keywords and phrases Hypergraph Coloring, Discrepancy, Rainbow Coloring, Stong Coloring, Algorithms, Semidefinite Programming, Hardness of Approximation

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.152

[*] Supported by NSF grant CCF-1115525.
[†] Supported in part by NSF grant CCF-1115525.
[‡] Supported by a Samsung Fellowship and NSF grant CCF-1115525.

1 Introduction

Coloring (hyper)graphs is one of the most important and well-studied tasks in discrete mathematics and theoretical computer science. A k-uniform hypergraph $G = (V, E)$ is said to be χ-colorable if there exists a coloring $c : V \mapsto \{1, \ldots, \chi\}$ such that no hyperedge is monochromatic, and such a coloring c is referred to as a proper χ-coloring. Graph and hypergraph coloring has been the focus of active research in both fields, and has served as the benchmark for new research paradigms such as the probabilistic method (Lovász local lemma [16]) and semidefinite programming (Lovász theta function [27]).

While such structural results are targeted towards special classes of hypergraphs, given a general χ-colorable k-uniform hypergraph, the problem of reconstructing a χ-coloring is known to be a hard task. Even assuming 2-colorability, reconstructing a proper 2-coloring is a classic NP-hard problem for $k \geq 3$. Given the intractability of proper 2-coloring, two notions of *approximate coloring* of 2-colorable hypergraphs have been studied in the literature of approximation algorithms. The first notion, called *Min-Coloring*, is to minimize the number of colors while still requiring that every hyperedge be non-monochromatic. The second notion, called *Max-2-Coloring* allows only 2 colors, but the objective is to maximize the number of non-monochromatic hyperedges.[1]

Even with these relaxed objectives, the promise that the input hypergraph is 2-colorable seems grossly inadequate for polynomial time algorithms to exploit in a significant way. For Min-Coloring, given a 2-colorable k-uniform hypergraph, the best known algorithm uses $O(n^{1-\frac{1}{k}})$ colors [13, 3], which tends to the trivial upper bound n as k increases. This problem has been actively studied from the hardness side, motivating many new developments in constructions of probabilistically checkable proofs. Coloring 2-colorable hypergraphs with $O(1)$ colors was shown to be NP-hard for $k \geq 4$ in [18] and $k = 3$ in [15]. An exciting body of recent work has pushed the hardness beyond poly-logarithmic colors [14, 17, 25, 22]. In particular, [25] shows quasi-NP-hardness of $2^{(\log n)^{\Omega(1)}}$-coloring a 2-colorable hypergraphs (very recently the exponent was shown to approach $1/4$ in [22]).

The hardness results for Max-2-Coloring show an even more pessimistic picture, wherein the naive *random assignment* (randomly give one of two colors to each vertex independently to leave a $(\frac{1}{2})^{k-1}$ fraction of hyperedges monochromatic in expectation), is shown to have the best guarantee for a polynomial time algorithm when $k \geq 4$ (see [21]).

Given these strong intractability results, it is natural to consider what further relaxations of the objectives could lead to efficient algorithms. For maximization versions, Austrin and Håstad [6] prove that (almost[2]) 2-colorability is useless (in a formal sense that they define) for any Constraint Satisfaction Problem (CSP) that is a relaxation of 2-coloring [37]. Therefore, it seems more natural to find a stronger promise on the hypergraph than mere 2-colorability that can be significantly exploited by polynomial time coloring algorithms for the objectives of Min-Coloring and Max 2-Coloring. This motivates our main question *"how strong a promise on the input hypergraph is required for polynomial time algorithms to perform significantly better than naive algorithms for Min-Coloring and Max-2-Coloring?"*

There is a very strong promise on k-uniform hypergraphs which makes the task of proper 2-coloring easy. If a hypergraph is k-partite (i.e., there is a k-coloring such that each hyperedge has each color exactly once), then one can properly 2-color the hypergraph in

[1] The maximization version is also known as Max-Set-Splitting, or more specifically Max k-Set-Splitting when considering k-uniform hypegraphs, in the literature.

[2] We say a hypergraph is *almost* χ-colorable for a small constant $\epsilon > 0$, there is a χ-coloring that leaves at most ϵ fraction of hyperedges monochromatic.

polynomial time. The same algorithm can be generalized to hypergraphs which admit a c-balanced coloring (i.e., c divides k and there is a k-coloring such that each hyperedge has each color exactly $\frac{k}{c}$ times). This can be seen by random hyperplane rounding of a simple SDP, or even simpler by solving a homogeneous linear system and iterating [2], or by a random recoloring method analyzed using random walks [29]. In fact, a proper 2-coloring can be efficiently achieved assuming that the hypergraph admits a fair partial 2-coloring, namely a pair of disjoint subsets A and B of the vertices such that for every hyperedge e, $|e \cap A| = |e \cap B| > 0$ [29].

The promises on structured colorings that we consider in this work are natural relaxations of the above strong promise of a perfectly balanced (partial) coloring.

- A hypergraph is said to have *discrepancy* ℓ when there is a 2-coloring such that in each hyperedge, the difference between the number of vertices of each color is at most ℓ.
- A χ-coloring ($\chi \leq k$) is called *rainbow* if every hyperedge contains each color at least once.
- A χ-coloring ($\chi \geq k$) is called *strong* if every hyperedge contains k different colors.

These three notions are interesting in their own right, and have been independently studied. Discrepancy minimization has recently seen different algorithmic ideas [8, 28, 33] to give constructive proofs of the classic six standard deviations result of Spencer [36]. Rainbow coloring admits a natural interpretation as a partition of V into the maximum number of disjoint vertex covers, and has been actively studied for geometric hypergraphs due to its applications in sensor networks [11]. Strong coloring is closely related to graph coloring by definition, and is known to capture various other notions of coloring [1]. It is easy to see that ℓ-discrepancy ($\ell < k$), χ-rainbow colorability ($2 \leq \chi \leq k$), and χ-strong colorability ($k \leq \chi \leq 2k-2$) all imply 2-colorability. For odd k, both $(k+1)$-strong colorability and $(k-1)$-rainbow colorability imply discrepancy-1, so strong colorability and rainbow colorability seem stronger than low discrepancy.

Even though they seem very strong, previous works have mainly focused on hardness with these promises. The work of Austrin et al. [5] shows NP-hardness of finding a proper 2-coloring under the discrepancy-1 promise. The work of Bansal and Khot [9] shows hardness of $O(1)$-coloring even when the input hypergraph is promised to be almost k-partite (under the Unique Games Conjecture); Sachdeva and Saket [34] establish NP-hardness of $O(1)$-coloring when the graph is almost $k/2$-rainbow colorable; and Guruswami and Lee [19] establish NP-hardness when the graph is perfectly (not almost) $\frac{k}{2}$-rainbow colorable, or admits a 2-coloring with discrepancy 2. These hardness results indicate that it is still a nontrivial task to exploit these strong promises and outperform naive algorithms.

1.1 Our Results

In this work, we prove that our three promises, unlike mere 2-colorability, give enough structure for polynomial time algorithms to perform significantly better than naive algorithms. We also study these promises from a hardness perspective to understand the asymptotic threshold at which beating naive algorithms goes from easy to UG/NP-Hard. In particular assuming the UGC, for Max-2-Coloring under ℓ-discrepancy or $k-\ell$-rainbow colorability, this threshold is $\ell = \Theta(\sqrt{k})$.

▶ **Theorem 1.** *There is a randomized polynomial time algorithm that produces a* 2*-coloring of a* k*-uniform hypergraph* H *with the following guarantee. For any* $0 < \epsilon < \frac{1}{2}$ *(let* $\ell = k^{\epsilon}$*), there exists a constant* $\eta > 0$ *such that if* H *is* $(k-\ell)$*-rainbow colorable or* $(k+\ell)$*-strong colorable, the fraction of monochromatic edges in the produced* 2*-coloring is* $O((\frac{1}{k})^{\eta k})$ *in expectation.*

Our results indeed show that this algorithm significantly outperforms the random assignment even when ℓ approaches $\sqrt{k}$ asymptotically. See Theorem 16 and Theorem 22 for the precise statements.

For the ℓ-discrepancy case, we observe that when $\ell < \sqrt{k}$, the framework of the second and the third authors [20] yields an approximation algorithm that marginally (by an additive factor much less than 2^{-k}) outperforms the random assignment, but we do not formally prove this here.

The following hardness results suggest that this gap between low-discrepancy and rainbow/strong colorability might be intrinsic. Let the term *UG-hardness* denote NP-hardness assuming the Unique Games Conjecture.

▶ **Theorem 2.** *For sufficiently large odd k, given a k-uniform hypergraph which admits a 2-coloring with at most a $(\frac{1}{2})^{6k}$ fraction of edges of discrepancy larger than 1, it is UG-hard to find a 2-coloring with a $(\frac{1}{2})^{5k}$ fraction of monochromatic edges.*

▶ **Theorem 3.** *For even $k \geq 4$, given a k-uniform hypergraph which admits a 2-coloring with no edge of discrepancy larger than 2, it is NP-hard to find a 2-coloring with a $k^{-O(k)}$ fraction of monochromatic edges.*

▶ **Theorem 4.** *For k sufficiently large, given a k-uniform hypergraph which admits a 2-coloring with no edge of discrepancy larger than $O(\log k)$, it is NP-hard to find a 2-coloring with a $2^{-O(k)}$ fraction of monochromatic edges.*

▶ **Theorem 5.** *For k such that $\chi := k - \sqrt{k}$ is an integer greater than 1, and any $\epsilon > 0$, given a k-uniform hypergraph which admits a χ-coloring with at most ϵ fraction of non-rainbow edges, it is UG-hard to find a 2-coloring with a $(\frac{1}{2})^{k-1}$ fraction of monochromatic edges.*

For Min-Coloring, all three promises lead to an $\tilde{O}(n^{\frac{1}{k}})$-coloring that is decreasing in k. These results are also notable in the sense that our promises are helpful not only for structured SDP solutions, but also for combinatorial *degree reduction* algorithms.

▶ **Theorem 6.** *Consider any k-uniform hypergraph $H = (V, E)$ with n vertices and m edges. For any $\ell < O(\sqrt{k})$, If H has discrepancy-ℓ, $(k - \ell)$-rainbow colorable, or $(k + \ell)$-strong colorable, one can color H with $\tilde{O}((\frac{m}{n})^{\frac{\ell^2}{k^2}}) \leq \tilde{O}(n^{\frac{\ell^2}{k}})$ colors.*

These results significantly improve the current best $\tilde{O}(n^{1-\frac{1}{k}})$ colors that assumes only 2-colorability. Our techniques give slightly better results depending on the promise — see Theorem 28. Table 1.1 summarizes our results.

1.2 Techniques

Our algorithms for Max-2-Coloring are straightforward applications of semidefinite programming, namely, we use natural vector relaxations of the promised properties, and round using a random hyperplane. The analysis however, is highly non-trivial and boils down to approximating a multivariate Gaussian integral. In particular, we show a (to our knowledge, new) upper bound on the Gaussian measure of simplicial cones in terms of simple properties of these cones. We should note that this upper bound is sensible only for simplicial cones that are well behaved with respect to the these properties. (The cones we are interested in are those given by the intersection of hyperplanes whose normal vectors constitute a solution to our vector relaxations). We believe our analysis to be of independent interest as similar approaches may work for other k-CSP's.

Table 1.1 Summary of our algorithmic and hardness results with valid ranges of ℓ. Two results with † are implied in [20]. The numbers of the first row indicate upper bounds on the fraction of monochromatic edges in a 2-coloring produced by our algorithms. $\delta := \delta(k, \ell) > 0$ is a small constant. The second row shows lower bounds on the fraction of monochromatic edges achieved by polynomial time algorithms. For the UG-hardness results, note that the input hypergraph does not have all edges satisfying the promises but almost edges satisfying them. The third row shows upper bounds upto log factors, on the number of colors the algorithms use to properly 2-color the hypergraph.

Promise	ℓ-Discrepancy	$(k-\ell)$-Rainbow	$(k+\ell)$-Strong
Max-2-Color Algorithm	$1/2^{k-1} - \delta,\ \ell < \sqrt{k}^{\dagger}$	$1/k^{\Omega(k)},\ \ell \ll \sqrt{k}$	$1/k^{\Omega(k)},\ \ell \ll \sqrt{k}$
Max-2-Color Hardness	UG: $1/2^{5k},\ \ell = 1.$ NP: $1/k^{O(k)},\ \ell = 2.$ NP: $1/2^{O(k)},\ \ell = \Omega(\log k)$ UG: $1/2^{k-1},\ \ell \geq \sqrt{k}^{\dagger}$	UG: $1/2^{k-1},\ \ell = \Omega(\sqrt{k})$	
Min-Color Algorithm	$n^{\ell^2/k},\quad \ell = O(\sqrt{k})$	$n^{\ell^2/k},\quad \ell = O(\sqrt{k})$	$n^{\ell^2/k},\quad \ell = O(\sqrt{k})$

1.2.1 Gaussian Measure of Simplicial Cones

As can be seen via an observation of Kneser [26], the Gaussian measure of a simplicial cone is equal to the fraction of spherical volume taken up by a spherical simplex (a spherical simplex is the intersection of a simplicial cone with a ball centered at the apex of the cone). This however, is a very old problem in spherical geometry, and while some things are known, like a nice differential formula due to Schlafli (see [35]), closed forms upto four dimensions (see [30]), and a complicated power series expansion due to Aomoto [4], it is likely hopeless to achieve a closed form solution or even an asymptotic formula for the volume of general spherical simplices.

Zwick [39] considered the performance of hyperplane rounding in various 3-CSP formulations, and this involved analyzing the volume of a 4-dimensional spherical simplex. Due to the complexity of this volume function, the analysis was tedious, and non-analytic for many of the formulations. His techniques were based on the Schlafli differential formula, which relates the volume differential of a spherical simplex to the volume functions of its codimension-2 faces and dihedral angles. However, to our knowledge not much is known about the general volume function in even 6 dimensions. This suggests that Zwick's techniques are unlikely to be scalable to higher dimensions.

On the positive side, an asymptotic expression is known in the case of symmetric spherical simplices, due to H. E. Daniels [32] who gave the analysis for regular cones of angle $\cos^{-1}(1/2)$. His techniques were extended by Rogers [31] and Boeroeczky and Henk [12] to the whole class of regular cones.

We combine the complex analysis techniques employed by Daniels with a lower bound on quadratic forms in the positive orthant, to give an upper bound on the Gaussian measure of a much larger class of simplicial cones.

1.2.2 Column Subset Selection

Informally, the cones for which our upper bound is relevant are those that are high dimensional in a strong sense, i.e. the normal vectors whose corresponding hyperplanes form the cone, must be such that no vector is too close to the linear span of any subset of the remaining vectors.

When the normal vectors are solutions to our rainbow colorability SDP relaxation, this need not be true. However, this can be remedied. We consider the column matrix of these normal vectors, and using spectral techniques, we show that there is a reasonably large subset of columns (vectors) that are well behaved with respect to condition number. We are then able to apply our Gaussian Measure bound to the cone given by this subset, admittedly in a slightly lower dimensional space.

2 Approximate Max-2-Coloring

In this section we show how the properties of $(k+\ell)$-strong colorability and $(k-\ell)$-rainbow colorability in k-uniform hypergraphs allow one to 2-color the hypergraph, such that the respective fractions of monochromatic edges are small. For $\ell = o(\sqrt{k})$, these guarantees handsomely beat the naive random algorithm (color every vertex blue or red uniformly and independently at random), wherein the expected fraction of monochromatic edges is $1/2^{k-1}$.

Our algorithms are straightforward applications of semidefinite programming, namely, we use natural vector relaxations of the above properties, and round using a random hyperplane. The analysis however, is quite involved.

2.1 Semidefinite Relaxations

Our SDP relaxations for low-discrepancy, rainbow-colorability, and strong-colorability are the following. Given that $\langle v_i, v_j \rangle = \frac{-1}{\chi-1}$ when unit vectors $v_1, \ldots, v_\chi$ form a χ-regular simplex centered at the origin, it is easy to show that they are valid relaxations. Due to space constraints, we defer the proofs of feasibility to the full version [10].

Discrepancy ℓ.

$$\begin{aligned} \forall\, e \in E,\ \left\| \sum_{i \in e} u_i \right\|_2 &\leq \ell \\ \forall\, i \in [n],\ \|u_i\|_2 &= 1 \\ \forall\, i \in [n],\ u_i &\in \mathbb{R}^n \end{aligned} \tag{2.1}$$

$(k-\ell)$-Rainbow Colorability.

$$\begin{aligned} \forall\, e \in E,\ \left\| \sum_{i \in e} u_i \right\|_2 &\leq \ell \\ \forall\, e \in E,\ \forall\, i < j \in e,\ \langle u_i, u_j \rangle &\geq \frac{-1}{k-\ell-1} \\ \forall\, i \in [n],\ \|u_i\|_2 &= 1 \\ \forall\, i \in [n],\ u_i &\in \mathbb{R}^n \end{aligned} \tag{2.2}$$

$(k+\ell)$-Strong Colorability.

$$\begin{aligned} \forall\, e \in E,\ \forall\, i < j \in e, \quad \langle u_i, u_j \rangle &= \frac{-1}{k+\ell-1} \\ \forall\, i \in [n],\ \|u_i\|_2 &= 1 \\ \forall\, i \in [n],\ u_i &\in \mathbb{R}^n \end{aligned} \tag{2.3}$$

Our rounding scheme is the same for all the above relaxations.

Rounding Scheme. Pick a standard n-dimensional Gaussian random vector r. For any $i \in [n]$, if $\langle v_i, r \rangle \geq 0$, then vertex i is colored blue, and otherwise it is colored red.

2.2 Setup of Analysis

We now setup the framework for analyzing all the above relaxations.
Consider a standard n-dimensional Gaussian random vector r, i.e. each coordinate is independently picked from the standard normal distribution $\mathcal{N}(0,1)$. The following are well known facts (the latter being due to Renyi),

▶ **Lemma 7.** *$r/\|r\|_2$ is uniformly distributed over the unit sphere in $\mathbb{R}^n$.*

Note. Lemma 7 establishes that our rounding scheme is equivalent to random hyperplane rounding.

▶ **Lemma 8.** *Consider any $j < n$. The projections of r onto the pairwise orthogonal unit vectors $e_1, \dots, e_j$ are independent and have distribution $\mathcal{N}(0,1)$.*

Next, consider any k-uniform hypergraph $H = (V = [n], E \subseteq \binom{V}{k})$ that is feasible for any of the aforementioned formulations. Our goal now, is to analyze the expected number of monochromatic edges. To obtain this expected fraction with high probability, we need only repeat the rounding scheme polynomially many times, and the high probability of a successful round follows by Markov's inequality. Thus we are only left with bounding the probability that a particular edge is monochromatic.

To this end, consider any edge $e \in E$ and let the vectors corresponding to the vertices in e be $u'_1, \dots, u'_k$. Consider a k-flat $\mathcal{F}$ (subspace of $\mathbb{R}^n$ congruent to $\mathbb{R}^k$), containing $u'_1, \dots, u'_k$. Applying Lemma 8 to the standard basis of $\mathcal{F}$, implies that the projection of r into $\mathcal{F}$ has the standard k-dimensional Gaussian distribution. Now since projecting r onto $\mathrm{Span}(u'_1, \dots u'_k)$ preserves the inner products $\{\langle r, u'_i \rangle\}_i$, we may assume without loss of generality that $u'_1, \dots, u'_k$ are vectors in $\mathbb{R}^k$, and the rounding scheme corresponds to picking a random k-dimensional Gaussian vector r, and proceeding as before.

Let U be the $k \times k$ matrix whose columns are the vectors $u'_1, \dots, u'_k$ and μ represent the Gaussian measure in $\mathbb{R}^k$. Then the probability of e being monochromatic in the rounding is given by,

$$\mu\Big(\Big\{x \in \mathbb{R}^k \ \big|\ U^T x \geq 0\Big\}\Big) + \mu\Big(\Big\{x \in \mathbb{R}^k \ \big|\ U^T x < 0\Big\}\Big) = 2\mu\Big(\Big\{x \in \mathbb{R}^k \ \big|\ U^T x \geq 0\Big\}\Big) \tag{2.4}$$

In other words, this boils down to analyzing the Gaussian measure of the cone given by $U^T x \geq 0$. We thus take a necessary detour.

2.3 Gaussian Measure of Simplicial Cones

In this section we show how to bound the Gaussian measure of a special class of simplicial cones. This is one of the primary tools in our analysis of the previously introduced SDP relaxations. We first state some preliminaries.

2.3.1 Preliminaries

Simplicial Cones and Equivalent Representations. A simplicial cone in $\mathbb{R}^k$, is given by the intersection of a set of k linearly independent halfspaces. For any simplicial cone with apex at position vector p, there is a unique set (upto changes in lengths), of k linearly independent vectors, such that the direct sum of $\{p\}$ with their positive span produces the cone. Conversely, a simplicial cone given by the direct sum of $\{p\}$ and the positive span of k linearly independent vectors, can be expressed as the intersection of a unique set of k halfspaces with apex at p. We shall refer to the normal vectors of the halfspaces above, as simply normal vectors of the cone, and we shall refer to the spanning vectors above, as simplicial vectors. We represent a simplicial cone C with apex at p, as (p, U, V) where U is a column matrix of unit vectors $u_1, \ldots, u_k$ (normal vectors), V is a column matrix of unit vectors $v_1, \ldots, v_k$ (simplicial vectors) and

$$C = \left\{ x \in \mathbb{R}^k \;\middle|\; u_1^T x \geq p_1, \ldots, u_k^T x \geq p_k \right\} = \left\{ p + x_1 v_1 + \cdots + x_k v_k \;\middle|\; x \geq 0, x \in \mathbb{R}^k \right\}$$

Switching Between Representations. Let $C \equiv (0, U, V)$ be a simplicial cone with apex at the origin. It is not hard to see that any v_i is in the intersection of exactly $k-1$ of the k halfspaces determined by U, and it is thus orthogonal to exactly $k-1$ vectors of the form u_j. We may assume without loss of generality that for any v_i, the only column vector of U not orthogonal to it, is u_i. Thus clearly $V^T U = D$ where D is some non-singular diagonal matrix. Let $A_U = U^T U$ and $A_V = V^T V$, be the gram matrices of the vectors. A_U and A_V are positive definite symmetric matrices with diagonal entries equal to one (they comprise of the pairwise inner products of the normal and simplicial vectors respectively). Also, clearly,

$$V = U^{-T} D, \qquad A_V = D A_U^{-1} D \tag{2.5}$$

One then immediately obtains: $(A_V)_{ij} = \frac{a_{ij}}{\sqrt{a_{ii} a_{jj}}}$, and $(A_U)_{ij} = \frac{-a'_{ij}}{\sqrt{a'_{ii} a'_{jj}}}$. where a_{ij} and a'_{ij} are the cofactors of the $(i,j)^{th}$ entries of A_U and A_V respectively.

Formulating the Integral. Let $C \equiv (0, U, V)$ be a simplicial cone with apex at the origin, and for $x \in \mathbb{R}^k$, let dx denote the differential of the standard k-dimensional Lebesgue measure. Then the Gaussian measure of C is given by,

$$\begin{aligned}
\frac{1}{\pi^{k/2}} \int\limits_{U^T x \geq 0} e^{-||x||_2^2}\,\mathrm{dx} \quad &= \frac{\det(V)}{\pi^{k/2}} \int\limits_{\mathbb{R}_+^k} e^{-||Vx||_2^2}\,\mathrm{dx} \quad (x \leftarrow Vx) \\
= \frac{\det(V)}{\pi^{k/2}} \int\limits_{\mathbb{R}_+^k} e^{-||U^{-T}Dx||_2^2}\,\mathrm{dx} \;\; (\text{Eq. (2.5)}) \quad &= \frac{\det(V)}{\pi^{k/2}\det(D)} \int\limits_{\mathbb{R}_+^k} e^{-||U^{-T}x||_2^2}\,\mathrm{dx} \;\; (x \leftarrow Dx) \\
= \frac{1}{\pi^{k/2}\det(U)} \int\limits_{\mathbb{R}_+^k} e^{-||U^{-T}x||_2^2}\,\mathrm{dx} \quad &= \frac{1}{\pi^{k/2}\sqrt{\det(A_U)}} \int\limits_{\mathbb{R}_+^k} e^{-x^T A_U^{-1} x}\,\mathrm{dx}
\end{aligned} \tag{2.6}$$

For future ease of use, we give a name to some properties.

▶ **Definition 9.** The *para-volume* of a set of vectors (resp. a matrix U), is the volume of the parallelotope determined by the set of vectors (resp. the column vectors of U).

▶ **Definition 10.** The *sum-norm* of a set of vectors (resp. a matrix U), is the length of the sum of the vectors (resp. the sum of the column vectors of U).

Walkthrough of Symmetric Case Analysis. We next state some simple identities that can be found in say, [32], some of which were originally used by Daniels to show that the Gaussian measure of a symmetric cone in $\mathbb{R}^k$ of angle $\cos^{-1}(1/2)$ (between any two simplicial vectors) is $\frac{(1+o(1))\ e^{k/2-1}}{\sqrt{2}^{k+1}\sqrt{k}^{k-1}\sqrt{\pi}^k}$. We state these identities, while loosely describing the analysis of the symmetric case, to give the reader an idea of their purpose.
First note that the gram matrices S_U and S_V, of the symmetric cone of angle $\cos^{-1}(1/2)$ are given by:

$$S_U = (1+1/k)\mathrm{I} - \mathbf{1}\mathbf{1}^T/k \qquad S_V = (\mathrm{I} + \mathbf{1}\mathbf{1}^T)/2$$

Thus $x^T S_U^{-1} x$ is of the form,

$$\alpha\,||x||_1^2 + \beta\,||x||_2^2 \tag{2.7}$$

The key step is in linearizing the $||x||_1^2$ term in the exponent, which allows us to separate the terms in the multivariate integral into a product of univariate integrals, and this is easier to analyze.

▶ **Lemma 11** (Linearization). $\sqrt{\pi}e^{-s^2} = \int_{-\infty}^{\infty} e^{-t^2+2its}\ \mathrm{dt}$.

▶ **Observation 12.** *Let $f : (-\infty,\infty) \mapsto \mathbb{C}$ be a continuous complex function. Then,*

$$\left|\int_{-\infty}^{\infty} f(t)\ \mathrm{dt}\right| \leq \int_{-\infty}^{\infty} |f(t)|\ \mathrm{dt}.$$

On applying Lemma 11 to Eq. (2.6) in the symmetric case, one obtains a product of identical univariate complex integrals. Specifically, by Eq. (2.6), Eq. (2.7), and Lemma 11, we have the expression,

$$\int_{\mathbb{R}_+^k} e^{-\beta||x||_2^2-\alpha||x||_1^2}\ \mathrm{dx}\ =\ \int_{-\infty}^{\infty} e^{-t^2}\left(\int_0^{\infty} e^{-\beta s^2+2it\sqrt{\alpha}s}\ \mathrm{ds}\right)^k$$

The inner univariate complex integral is not readily evaluable. To circumvent this, one can change the line of integration so as to shift mass form the inner integral to the outer integral. Then we can apply the crude upper bound of Observation 12 to the inner integral, and by design, the error in our estimate is small.

▶ **Lemma 13** (Changing line of integration). *Let $g(t)$ be a real valued function for real t. If, when interpreted as a complex function in the variable $t = a+ib$, $g(a+ib)$ is an entire function, and furthermore, $\lim_{a\to\infty} g(a+ib) = 0$ for some fixed b, then we have, $\int_{-\infty}^{\infty} g(t)\ \mathrm{dt} = \int_{-\infty}^{\infty} g(a+ib)\ \mathrm{da}$.*

Squared $\mathbf{L_1}$ Inequality. Motivated by the above linearization technique, we prove the following lower bound on quadratic forms in the positive orthant:

▶ **Lemma 14.** *Consider any $k \times k$ matrix A, and $x \in \mathbb{R}^k_+$, such that x is in the column space of A. Let $A^\dagger$ denote the Moore-Penrose pseudo-inverse of A. Then, $x^T A^\dagger x \geq \frac{||x||_1^2}{\text{sum}(A)}$.*

Proof. Consider any x in the positive orthant and column space of A. Let $v_1, \ldots, v_q$ be the eigenvectors of A corresponding to it's non-zero eigenvalues. We may express x in the form $x = \sum_i \beta_i v_i$, so that

$$||x||_1 = \langle \mathbf{1}, x\rangle = \sum_{i\in[q]} \beta_i \langle \mathbf{1}, v_i\rangle \Rightarrow ||x||_1^2 = (\sum_{i\in[q]} \beta_i \langle \mathbf{1}, v_i\rangle)^2.$$

We also have

$$x^T A^\dagger x = x^T (\sum_{i\in[q]} \lambda_i^{-1} v_i v_i^T) x = \sum_{i\in[q]} \lambda_i^{-1} \beta_i^2.$$

Now by Cauchy-Schwartz,

$$\left(\sum_i \lambda_i \langle \mathbf{1}, v_i\rangle^2\right)\left(\sum_{i\in[q]} \lambda_i^{-1}\beta_i^2\right) \geq ||x||_1^2 .$$

Therefore, we have

$$x^T A^\dagger x \geq \frac{||x||_1^2}{\sum_{i\in[q]} \lambda_i \langle \mathbf{1}, v_i\rangle^2} = \frac{||x||_1^2}{\mathbf{1}^T A \mathbf{1}} = \frac{||x||_1^2}{\text{sum}(A)}.$$

◀

Equipped with all necessary tools, we may now prove our result.

2.3.2 Our Gaussian Measure Bound

Let $C \equiv (0, U, V)$ be a simplicial cone with apex at the origin. We now show an upper bound on the Gaussian measure of C that depends surprisingly on only the para-volume and sum-norm of U. Since Gaussian measure is at most 1, it is evident when viewing our bound that it can only be useful for simplicial cones wherein the sum-norm of their normal vectors is $O(\sqrt{k})$, and the para-volume of their normal vectors is not too small.

▶ **Theorem 15.** *Let $C \equiv (0, U, V)$ be a simplicial cone with apex at the origin. Let $\ell = ||\sum_i u_i||_2$ (i.e. sum-norm of the normal vectors), then the Gaussian measure of C is at most $\left(\frac{e}{2\pi k}\right)^{k/2} \frac{\ell^k}{\sqrt{\det(A_U)}}$*

Proof. By the sum-norm property, the sum of entries of A_U is ℓ^2. Also by the definition of a simplicial cone, U, and cosequently A_U, must have full rank. Thus we may apply Lemma 14 over the entire positive orthant. We proceed to analyze the multivariate integral in Eq. (2.6), by first applying Lemma 14 and then linearizing the exponent using Lemma 11. Post-linearization, our approach is similar to the presentation of Boeroeczky and Henk [12]. We have,

$$\int_{\mathbb{R}^k_+} e^{-x^T A_U^{-1} x}\,\mathrm{dx} \quad \leq \quad \int_{\mathbb{R}^k_+} e^{-||x||_1^2/\ell^2}\,\mathrm{dx} \qquad\qquad \text{(by Lemma 14)}$$

$$= \ell^k \int_{\mathbb{R}_+^k} e^{-||y||_1^2}\,\mathrm{dy} \qquad \text{(Subst. } y \leftarrow x/\ell)$$

$$= \frac{\ell^k}{\sqrt{\pi}} \int_{\mathbb{R}_+^k} \int_{-\infty}^{\infty} e^{-t^2 + 2it\sum_{i\in[k]} y_i}\,\mathrm{dt}\,\mathrm{dy} \qquad \text{(by Lemma 11)}$$

$$= \frac{\ell^k}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-t^2}\left(\int_0^\infty e^{2its}\,\mathrm{ds}\right)^k \mathrm{dt} \qquad \text{Let } g(t) = e^{-t^2}\left(\int_0^\infty e^{2its}\,\mathrm{ds}\right)^k$$

$$|g(a+ib)| \le e^{-a^2+b^2}\left(\int_0^\infty e^{-2bs}\,\mathrm{ds}\right)^k$$

$$\Rightarrow \lim_{a\to\infty} g(a+ib) \to 0,\ \forall b > 0$$

$$= \frac{\ell^k}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-a^2+b^2-2abi}\left(\int_0^\infty e^{-2bs+2asi}\,\mathrm{ds}\right)^k \mathrm{da} \qquad \forall\ b>0,\text{ by Lemma 13}$$

$$= \frac{e^{k/2}\,\ell^k}{\sqrt{\pi}(2k)^{k/2}} \int_{-\infty}^{\infty} e^{-a^2}\left(2be^{-ia/b}\int_0^\infty e^{-2bs+2asi}\,\mathrm{ds}\right)^k \mathrm{da} \qquad \text{Fixing } b = \sqrt{k/2}$$

$$= \frac{e^{k/2}\,\ell^k}{\sqrt{\pi}(2k)^{k/2}} \left|\int_{-\infty}^{\infty} e^{-a^2}\left(2be^{-ia/b}\int_0^\infty e^{-2bs+2asi}\,\mathrm{ds}\right)^k \mathrm{da}\right| \qquad \text{Expr. is real, +ve}$$

$$\le \frac{e^{k/2}\,\ell^k}{\sqrt{\pi}(2k)^{k/2}} \int_{-\infty}^{\infty} e^{-a^2}\left(2b\int_0^\infty e^{-2bs}\,\mathrm{ds}\right)^k \mathrm{da} \qquad \text{By Observation 12}$$

$$= \frac{e^{k/2}\,\ell^k}{\sqrt{\pi}(2k)^{k/2}} \int_{-\infty}^{\infty} e^{-a^2}\,\mathrm{da} \;=\; \frac{e^{k/2}\,\ell^k}{(2k)^{k/2}}$$

Lastly, the claim follows by substituting the above in Eq. (2.6). ◀

2.4 Analysis of Hyperplane Rounding given Strong Colorability

In this section we analyze the performance of random hyperplane rounding on k-uniform hypergraphs that are $(k+\ell)$-strongly colorable.

▶ **Theorem 16.** *Consider any $(k+\ell)$-strongly colorable k-uniform hypergraph $H = (V,E)$. The expected fraction of monochromatic edges obtained by performing random hyperplane rounding on the solution of Relaxation 2.3, is $O\Big(\ell^{k-1/2}\big(\frac{e}{2\pi}\big)^{k/2}\frac{1}{k^{(k-1)/2}}\Big)$.*

Proof. Let U be any $k\times k$ matrix whose columns are unit vectors $u_1,\ldots,u_k \in Re^k$ that satisfy the edge constraints in Relaxation 2.3. Recall from Section 2.2, that to bound the probability of a monochromatic edge we need only bound the expression in Eq. (2.4) for U of the above form. By Relaxation 2.3, the gram matrix $A_U = U^TU$, is exactly, $A_U = (1+\alpha)\mathrm{I} - \alpha\mathbf{1}\mathbf{1}^T$ where $\alpha = \frac{1}{k+\ell-1}$. By matrix determinant lemma (determinant formula for rank one updates), we know

$$\det(A_U) = (1+\alpha)^k\left(1 - \frac{k\alpha}{1+\alpha}\right) \ge \left(\frac{\ell}{k+\ell}\right) = \Omega\left(\frac{\ell}{k}\right)$$

Further, Relaxation 2.3 implies the length of $\sum_i u_i$, is at most ℓ. The claim then follows by combining Eq. (2.4) with Theorem 15. ◀

Note. Being that any edge in the solution to the strong colorability relaxation corresponds to a symmetric cone, Theorem 16 is directly implied by prior work on the volume of symmetric spherical simplices. It is in the next section, where the true power of Theorem 15 is realized.

Remark. As can be seen from the asymptotic volume formula of symmetric spherical simplices, $\sqrt{\pi k/(2e)}$ is a sharp threshold for ℓ, i.e. when $\ell > (1+o(1))\sqrt{\pi k/(2e)}$, hyperplane rounding does worse than the naive random algorithm, and when $\ell < (1-o(1))\sqrt{\pi k/(2e)}$, hyperplane rounding beats the naive random algorithm.

2.5 Analysis of Hyperplane Rounding given Rainbow Colorability

In this section we analyze the performance of random hyperplane rounding on k-uniform hypergraphs that are $(k-\ell)$-rainbow colorable.

Let U be the $k \times k$ matrix whose columns are unit vectors $u_1, \ldots, u_k \in \mathbb{R}^k$ satisfying the edge constraints in Relaxation 2.2. We need to bound the expression in Eq. (2.4) for U of the above form. While we'd like to proceed just as in Section 2.4, we are limited by the possibility of U being singular or the parallelotope determined by U having arbitrarily low volume (as u_1 can be chosen arbitrarily close to the span of $u_2, \ldots, u_k$ while still satisfying $||\sum_i u_i||_2 \leq \ell$).

While U can be bad with respect to our properties of interest, we will show that some subset of the vectors in U are reasonably well behaved with respect to para-volume and sum-norm.

2.5.1 Finding a Well Behaved Subset

We'd like to find a subset of U with high para-volume, or equivalently, a principal submatrix of A_U with reasonably large determinant. To this end, we express the gram matrix $A_U = U^T U$ as the sum of a symmetric skeleton matrix B_U and a residue matrix E_U. Formally, $E_U = A_U - B_U$ and $B_U = (1+\beta)I - \beta \mathbf{1}\mathbf{1}^T$ where $\beta = \frac{1}{k-\ell-1}$.

$$B_U = (1+\beta)I - \beta \mathbf{1}\mathbf{1}^T \qquad \text{where} \quad \beta = \frac{1}{k-\ell-1}$$

We have (assuming $\ell = o(k)$), $\text{sum}(A_U) \leq \ell^2$ and $\text{sum}(B_U) = k - k(k-1)\beta = \frac{-\ell}{1-o(1)}$. Let $s \leftarrow \text{sum}(E_U) \leq \ell^2 - \text{sum}(B_U) = \ell^2 + \frac{\ell}{1-o(1)}$.

We further observe that E_U is symmetric, with all diagonal entries zero. Also since $u_1, \ldots, u_k$ satisfy Relaxation 2.2, all entries of E_U are non-negative.

By an averaging argument, at most ck^δ columns of E_U have column sums greater than $s/(ck^\delta)$ for some parameters δ, c to be determined later. Let $S \subseteq [k]$ be the set of indices of the columns having the lowest $k - ck^\delta$ column sums. Let $\tilde{k} \leftarrow |S| = k - ck^\delta$, and let A_S, B_S, E_S be the corresponding matrices restricted to S (in both columns and rows).

2.5.1.1 Spectrum of B_S and E_S

▶ **Observation 17.** *For a square matrix X, let $\lambda_{\min}(X)$ denote its minimum eigenvalue. The eigenvalues of B_S are exactly $(1+\beta)$ with multiplicity $(\tilde{k}-1)$, and $(1+\beta-\tilde{k}\beta)$ with* *multiplicity* 1. *Thus* $\lambda_{\min}(B_S) = 1+\beta-\tilde{k}\beta$. *This is true since* B_S *merely shifts all eigenvalues of* $-\beta\mathbf{1}\mathbf{1}^T$ *by* $1+\beta$.

While we don't know as much about the spectrum of E_S, we can still say some useful things.

▶ **Observation 18.** *Since* E_S *is non-negative, by Perron-Frobenius theorem, its spectral radius is equal to its max column sum, which is at most* $s/(ck^\delta)$. *Thus* $\lambda_{\min}(E_S) \geq -s/(ck^\delta)$.

Now that we know some information about the spectra of B_S and E_S, the next natural step is to consider the behaviour of spectra under matrix sums.

2.5.1.2 Spectral properties of Matrix sums

The following identity is well known.

▶ **Observation 19.** *If* X *and* Y *are symmetric matrices with eigenvalues* $x_1 > x_2 > \cdots > x_m$ *and* $y_1 > y_2 > \cdots > y_m$ *and the eigenvalues of* $A+B$ *are* $z_1 > z_2 > \cdots > z_m$, *then*

$$\forall\ 0 \leq i+j \leq m, \quad z_{m-i-j} \geq x_{m-i} + y_{m-j}.$$

In particular, this implies $\lambda_{\min}(X+Y) \geq \lambda_{\min}(X) + \lambda_{\min}(Y)$.

We may finally analyze the spectrum of A_S.

2.5.1.3 Properties of A_S

▶ **Observation 20** (Para-Volume). *Let the eigenvalues of* A_S *be* $a_1 > a_2 > \cdots > a_{\tilde{k}}$ *By Observation 17, Observation 18, and Observation 19 we have (Assuming* $\ell < ck^{\delta/2}$*),*

$$\begin{aligned} \lambda_{\min}(A_S) = a_{\tilde{k}} &\geq 1+\beta-\tilde{k}\beta - \frac{s}{ck^\delta} = \frac{c}{k^{1-\delta}} - \frac{\ell^2}{ck^\delta} - o(1) \\ a_2, a_3, \ldots, a_{\tilde{k}-1} &\geq 1+\beta - \frac{s}{ck^\delta} \qquad = 1 - \frac{\ell^2}{ck^\delta} - o(1) \end{aligned}$$

Consequently,

$$\det(A_S) \geq \left(\frac{c}{k^{1-\delta}} - \frac{\ell^2}{ck^\delta} - o(1)\right)\left(1 - \frac{\ell^2}{ck^\delta} - o(1)\right)^{\tilde{k}} \geq \left(\frac{c}{k^{1-\delta}} - \frac{\ell^2}{ck^\delta} - o(1)\right)e^{-k}$$

In particular, note that A_S *is non-singular and has non-negligible para-volume when*

$$\frac{\ell^2}{ck^\delta} = \frac{c}{2k^{1-\delta}}, \quad i.e.\ \ell \approx ck^{\delta-1/2} \quad or,\ \delta \approx \frac{1}{2}\frac{\log(\ell/c)}{\log k}$$

▶ **Observation 21** (Sum-Norm). *Since* E_U *is non-negative,* $\text{sum}(E_S) \leq \text{sum}(E_U) = s$. *Also we know that the sum of entries of* A_S *is*

$$\text{sum}(B_S) + \text{sum}(E_S) = \tilde{k}(1+\beta) - \tilde{k}(\tilde{k}-1)\beta + s \leq ck^\delta + s \tag{2.8}$$

2.5.2 The Result

We are now equipped to prove our result.

▶ **Theorem 22.** *For* $\ell < \sqrt{k}/100$, *consider any* $(k-\ell)$*-rainbow colorable* k*-uniform hypergraph* $H = (V,E)$. *Let* $\theta = 1/2 + \log(\ell)/\log(k)$ *and* $\eta = 19(1-\theta)/40$. *The expected fraction of monochromatic edges obtained by performing random hyperplane rounding on the solution of Relaxation 2.2, is at most*

$$\frac{1}{2.1^k\, k^{\eta k}}$$

Proof. Let U be any $k \times k$ matrix whose columns are unit vectors $u_1, \ldots, u_k \in \mathbb{R}^k$ that satisfy the edge constraints in Relaxation 2.3. Recall from Section 2.2, that to bound the probability of a monochromatic edge we need only bound the expression in Eq. (2.4) for U of the above form.

By Section 2.5.1, we can always choose a matrix U_S whose columns $\tilde{u}_1, \ldots, \tilde{u}_{\tilde{k}}$ are from the set $\{u_1, \ldots, u_k\}$, such that the gram matrix $A_S = U_S^T U_S$ satisfies Eq. (2.8) and Observation 20. Clearly the probability of all vectors in U being monochromatic is at most the probability of all vectors in U_S being monochromatic.

Thus just as in Section 2.2, to find the probability of U_S being monochromatic, we may assume without loss of generality that we are performing random hyperplane rounding in $\mathbb{R}^{\tilde{k}}$ on any $\tilde{k}$-dimensional vectors $\tilde{u}_1, \ldots, \tilde{u}_{\tilde{k}}$ whose gram (pairwise inner-product) matrix is the aforementioned A_S.

Specifically, by combining Eq. (2.8) and Observation 20 with Theorem 15, our expression is at most:

$$\left(\frac{e}{2\pi}\right)^{\tilde{k}/2} \left(\frac{ck^\delta + s}{k}\right)^{\tilde{k}/2} \frac{1}{\sqrt{det(A_U)}} \le 3.2^{\tilde{k}/2} \left(\frac{(1 - o(1))c}{k^{1-\delta}}\right)^{\tilde{k}/2} \le \frac{1}{2.1^k \, k^{(1-c)(1-\delta)k}}$$

assuming $c = 1/20$, $\delta \ge 1/2$ and $\ell < \sqrt{k}/100$ (constraint on ℓ ensures that non-singularity conditions of Observation 20 are satisfied). The claim follows. ◀

Remark. Yet again we see a threshold for ℓ, namely, when $\ell < \sqrt{k}/100$, hyperplane rounding beats the naive random algorithm, and for $\ell = \Omega(\sqrt{k})$, it fails to do better. In fact, as we'll see in the next section, assuming the UGC, we show a hardness result when $\ell = \Omega(\sqrt{k})$.

3 Hardness of Max-2-Coloring under Low Discrepancy

In this section we consider the hardness of Max-2-Coloring when promised discrepancy as low as one. As noted in Section 2.5, our analysis requires the configuration of vectors in an edge to be well behaved with respect to sum-norm and para-volume. While in the discrepancy case, we can ensure good sum-norm, the vectors in an edge can have arbitrarily low para-volume. While in the rainbow case we can remedy this by finding a reasonably large well behaved subset of vectors, this is not possible in the case of discrepancy.

Indeed, consider the following counterexample: Start in 2 dimensions with $k/3$ copies each of any u_1, u_2, u_3 such that $u_1 + u_2 + u_3 = 0$. Lift all vectors to 3-dimensions by assigning every vector a third coordinate of value exactly $1/k$. This satisfies Relaxation 2.1, yet every superconstant sized subset has para-volume zero.

Confirming that this is not an artifact of our techniques and the problem is in fact hard, we show in this section via a reduction from Max-Cut, that assuming the Unique Games conjecture, it is NP-Hard to Max-2-Color much better than the naive random algorithm that miscolors 2^{-k+1} fraction of edges, even in the case of discrepancy-1 hypergraphs.

3.1 Reduction from Max-Cut

Let $k = 2t + 1$. Let $G = (V, E)$ be an instance of Max-Cut, where each edge has weight 1. Let $n = |V|$ and $m = |E|$. We produce a hypergraph $H = (V', E')$ where $V' = V \times [k]$. For each $u \in V$, let $\mathsf{cloud}(u) := \{u\} \times [k]$. For each edge $(u, v) \in E$, we add $N := 2\binom{k}{t}\binom{k}{t+1}$ hyperedges

$$\{U \cup V : U \subseteq \mathsf{cloud}(u), V \subseteq \mathsf{cloud}(v), |U| + |V| = k, ||U| - |V|| = 1\},$$

each with weight $\frac{1}{N}$. Call these hyperedges *created by* (u, v). The sum of weights is m for both G and H.

3.1.1 Completeness

Given a coloring $C : V \mapsto \{B, W\}$ that cuts at least $(1 - \alpha)m$ edges of G, we color H so that for every $v \in V$, each vertex in $\mathsf{cloud}(v)$ is given the same color as v. If $(u, v) \in E$ is cut, all hyperedges created by (u, v) will have discrepancy 1. Therefore, the total weight of hyperedges with discrepancy 1 is at least $(1 - \alpha)m$.

3.1.2 Soundness

Given a coloring $C' : V' \mapsto \{B, W\}$ such that the total weight of non-monochromatic hyperedges is $(1 - \beta)m$, $v \in V$ is given the color that appears the most in its cloud (k is odd, so it is well-defined). Consider $(u, v) \in E$. If no hyperedge created by (u, v) is monochromatic, it means that u and v should be given different colors by the above majority algorithm (if they are given the same color, say white, then there are at least $t + 1$ white vertices in both clouds, so we have at least one monochromatic hyperedge).

This means that for each $(u, v) \in E$ that is uncut by the above algorithm (lost weight 1 for Max-Cut objective), at least one hyperedge created by (u, v) is monochromatic, and we lost weight at least $\frac{1}{N}$ there for our problem. This means that the total weight of cut edges for Max-Cut is at least $(1 - \beta N)m$.

3.1.3 The Result

▶ **Theorem 23** ([24]). *Let $G = (V, E)$ be a graph with $m = |E|$. For sufficiently small $\epsilon > 0$, it is UG-hard to distinguish the following cases.*

- *There is a 2-coloring that cuts at least $(1 - \epsilon)|E|$ edges.*
- *Every 2-coloring cuts at most $(1 - (2/\pi)\sqrt{\epsilon})|E|$ edges.*

Our reduction shows that

▶ **Theorem 24.** *Given a hypergraph $H = (V, E)$, it is UG-hard to distinguish the following cases.*

- *There is a 2-coloring where at least $(1 - \epsilon)$ fraction of hyperedges have discrepancy 1.*
- *Every 2-coloring cuts (in a standard sense) at most $(1 - (2/\pi)\frac{\sqrt{\epsilon}}{N})$ fraction of hyperedges.*

$N = 2\binom{k}{t}\binom{k}{t+1} \leq (2/\pi)2^k \cdot 2^k \leq (2/\pi)2^{2k}$. If we take $\epsilon = 2^{-6k}$ for large enough k, we cannot distinguish

- There is a 2-coloring where at least $(1 - 2^{-6k})$ fraction of hyperedges have discrepancy 1.
- Every 2-coloring cuts (in a standard sense) at most $(1 - 2^{-5k})$ fraction of hyperedges.

This proves Theorem 2.

3.2 NP-Hardness

In this subsection, we show that given a hypergraph which admits a 2-coloring with discrepancy at most 2, it is NP-hard to find a 2-coloring that has less than $k^{-O(k)}$ fraction of monochromatic hyperedges. Note that while the inapproximability factor is worse than the previous subsection, we get NP-hardness and it holds when the input hypergraph is promised to have *all* hyperedges have discrepancy at most 2. The reduction and the analysis closely follow from the more general framework of Guruswami and Lee [19] except that we prove a better reverse hypercontractivity bound for our case.

3.2.1 Q-Hypergraph Label Cover

An instance of Q-Hypergraph Label Cover is based on a Q-uniform hypergraph $H = (V, E)$. Each hyperedge-vertex pair (e, v) such that $v \in e$ is associated with a projection $\pi_{e,v} : [R] \to [L]$ for some positive integers R and L. A labeling $l : V \to [R]$ *strongly satisfies* $e = \{v_1, \ldots, v_Q\}$ when $\pi_{e,v_1}(l(v_1)) = \cdots = \pi_{e,v_Q}(l(v_Q))$. It *weakly satisfies* e when $\pi_{e,v_i}(l(v_i)) = \pi_{e,v_j}(l(v_j))$ for some $i \neq j$. The following are two desired properties of instances of Q-Hypergraph Label Cover.

- Regular: every projection is d-to-1 for $d = R/L$.
- Weakly dense: any subset of V of measure at least ϵ vertices induces at least $\frac{\epsilon^Q}{2}$ fraction of hyperedges.
- T-smooth: for all $v \in V$ and $i \neq j \in [R]$, $\Pr_{e \in E : e \ni v}[\pi_{e,v}(i) = \pi_{e,v}(j)] \leq \frac{1}{T}$.

The following theorem asserts that it is NP-hard to find a good labeling in such instances.

▶ **Theorem 25** ([19]). *For all integers $T, Q \geq 2$ and $\eta > 0$, the following is true. Given an instance of Q-Hypergraph Label Cover that is regular, weakly-dense and T-smooth, it is NP-hard to distinguish between the following cases.*

- *Completeness: There exists a labeling l that strongly satisfies every hyperedge.*
- *Soundness: No labeling l can weakly satisfy η fraction of hyperedges.*

3.2.2 Distributions

We first define the distribution $\overline{\mu}'$ for each *block*. $2Q$ points $x_{q,i} \in \{1,2\}^d$ for $1 \leq q \leq Q$ and $1 \leq i \leq 2$ are sampled by the following procedure.

- Sample $q' \in [Q]$ uniformly at random.
- Sample $x_{q',1}, x_{q',2} \in \{1,2\}^d$ i.i.d.
- For $q \neq q'$, $1 \leq j \leq d$, sample a permutation $((x_{q,1})_j, (x_{q,2})_j) \in \{(1,2),(2,1)\}$ uniformly at random.

3.2.3 Reduction and Completeness

We now describe the reduction from Q-Hypergraph Label Cover. Given a Q-uniform hypergraph $H = (V, E)$ with Q projections from $[R]$ to $[L]$ for each hyperedge (let $d = R/L$), the resulting instance of $2Q$-Hypergraph Coloring is $H' = (V', E')$ where $V' = V \times \{1,2\}^R$. Let $\mathsf{cloud}(v) := \{v\} \times \{1,2\}^R$. The set E' consists of hyperedges generated by the following procedure.

- Sample a random hyperedge $e = (v_1, \ldots, v_Q) \in E$ with associated projections, $\pi_{e,v_1}, \ldots, \pi_{e,v_Q}$ from E.
- Sample $(x_{q,i})_{1 \leq q \leq Q, 1 \leq i \leq 2} \in \{1,2\}^R$ in the following way. For each $1 \leq j \leq L$, independently sample $((x_{q,i})_{\pi^{-1}_{e,v_q}(j)})_{q,i}$ from $((\{1,2\}^d)2Q, \overline{\mu}')$.
- Add a hyperedge between $2Q$ vertices $\{(v_q, x_{q,i})\}_{q,i}$ to E'. We say this hyperedge is *formed from* $e \in E$.

Given the reduction, completeness is easy to show.

▶ **Lemma 26.** *If an instance of Q-Hypergraph Label Cover admits a labeling that strongly satisfies every hyperedge $e \in E$, there is a coloring $c : V' \to \{1,2\}$ of the vertices of H' such that every hyperedge $e' \in E'$ has at least $(Q-1)$ vertices of each color.*

Proof. Let $l : V \to [R]$ be a labeling that strongly satisfies every hyperedge $e \in E$. For any $v \in V, x \in \{1,2\}^R$, let $c(v,x) = x_{l(v)}$. For any hyperedge $e' = \{(v_q, x_{q,i})\}_{q,i} \in E'$, $c(v_q, x_{q,i}) = (x_{q,i})_{l(v_q)}$, and all but one q satisfies $\left\{(x_{q,1})_{l(v_q)}, (x_{q,2})_{l(v_q)}\right\} = \{1,2\}$. Therefore, the above strategy ensures that every hyperedge of E' contains at least $(Q-1)$ vertices of each color. ◀

3.2.4 Soundness

▶ **Lemma 27.** *There exists $\eta := \eta(Q)$ such that if $I \subseteq V'$ of measure $\frac{1}{2}$ induces less than $Q^{-O(Q)}$ fraction of hyperedges in H', the corresponding instance of Q-Hypergraph Label Cover admits a labeling that weakly satisfies a fraction η of hyperedges.*

Proof. Consider a vertex v and hyperedge $e \in E$ that contains v with a permutation $\pi = \pi_{e,v}$. Let $f : \{1,2\}^R \mapsto [0,1]$ be a *noised* indicator function of $I \cap \mathsf{cloud}(v)$ with $\mathbb{E}_{x \in \{1,2\}^R}[f(x)] \geq \frac{1}{2} - \epsilon$ for small $\epsilon > 0$ that will be determined later. We define the inner product

$$\langle f, g \rangle = \mathbb{E}_{x \in \{1,2\}^R}[f(x)g(x)].$$

f admits the Fourier expansion

$$\sum_{S \subseteq [R]} \hat{f}(S)\chi_S$$

where

$$\chi_S(x_1, \ldots, x_k) = \prod_{i \in S}(-1)^{x_i}, \qquad \hat{f}(S) = \langle f, \chi_S \rangle.$$

In particular, $\hat{f}(\emptyset) = \mathbb{E}[f(x)]$, and

$$\sum_S \hat{f}(S)^2 = \mathbb{E}[f(x)^2] \leq \mathbb{E}[f(x)] \tag{3.1}$$

A subset $S \subseteq [R]$ is said to be *shattered* by π if $|S| = |\pi(S)|$. For a positive integer J, we decompose f as the following:

$$f^{\mathsf{good}} = \sum_{S:\text{ shattered}} \hat{f}(S)\chi_S$$
$$f^{\mathsf{bad}} = f - f^{\mathsf{good}}.$$

By adding a suitable noise and using smoothness of Label Cover, for any $\delta > 0$, we can assume that $||f^{\mathsf{bad}}||^2 \leq \delta$. See [19] for the details.

Each time a $2Q$-hyperedge is sampled is formed from e, two points are sampled from each cloud. Let x, y be the points in $\mathsf{cloud}(v)$. Recall that they are sampled such that for each $1 \leq j \leq L$,

- With probability $\frac{1}{Q}$, for each $i \in \pi^{-1}(j)$, x_i and y_i are independently sampled from $\{1,2\}$.
- With probability $\frac{Q-1}{Q}$, for each $i \in \pi^{-1}(j)$, (x_i, y_i) are sampled from $\{(1,2),(2,1)\}$.

We can deduce the following simple properties.

1. $\mathbb{E}_{x,y}[\chi_{\{i\}}(x)\chi_{\{i\}}(y)] = -\frac{Q-1}{Q}$. Let $\rho := -\frac{Q-1}{Q}$.
2. $\mathbb{E}_{x,y}[\chi_{\{i\}}(x)\chi_{\{j\}}(y)] = 0$ if $i \neq j$.
3. $\mathbb{E}_{x,y}[\chi_S(x)\chi_T(y)] = 0$ unless $\pi(S) = \pi(T) = \pi(S \cap T)$.

We are interested in lower bounding

$$\mathbb{E}_{x,y}[f(x)f(y)] \geq \mathbb{E}[f^{\mathsf{good}}(x)f^{\mathsf{good}}(y)] - 3\|f^{\mathsf{bad}}(x)\|^2\|f\|^2 \geq \mathbb{E}[f^{\mathsf{good}}(x)f^{\mathsf{good}}(y)] - 3\delta.$$

By the property 3.,

$$\begin{aligned}
\mathbb{E}[f^{\mathsf{good}}(x)f^{\mathsf{good}}(y)] &= \sum_{S\text{: shattered}} \hat{f}(S)^2\rho^{|S|} \\
&= \mathbb{E}[f]^2 + \sum_{S\text{: shattered}} \hat{f}(S)^2\rho^{|S|} \\
&\geq \mathbb{E}[f]^2 + \rho(\sum_{|S|>1} \hat{f}(S)^2) \qquad \text{since } \rho \text{ is negative} \\
&\geq \mathbb{E}[f]^2 + \rho(\mathbb{E}[f] - \mathbb{E}[f]^2) \qquad \text{by (3.1)} \\
&\geq \mathbb{E}[f]^2(1+\rho) - \epsilon \qquad \text{since } \mathbb{E}[f] \geq \frac{1}{2} - \epsilon \Rightarrow \mathbb{E}[f] - \mathbb{E}[f]^2 \leq \mathbb{E}[f]^2 + \epsilon \\
&\geq \frac{\mathbb{E}[f]^2}{Q} - \epsilon.
\end{aligned}$$

By taking ϵ and δ small enough, we can ensure that

$$\mathbb{E}[f(x)f(y)] \geq \zeta := \frac{1}{5Q}. \tag{3.2}$$

The soundness analysis of Guruswami and Lee [19] ensures ((3.2) replaces their Step 2) that there exists $\eta := \eta(Q)$ such that if the fraction of hyperedges induced by I is less than $Q^{-O(Q)}$, the Hypergraph Label Cover instance admits a solution that satisfies η fraction of constraints. We omit the details. ◀

3.2.5 Corollary to Max-2-Coloring under discrepancy $O(\log k)$

The above NP-hardness, combined with the reduction techinque from Max-Cut in Section 3.1, shows that given a k-uniform hypergraph, it is NP-hard to distinguish whether it has discrepancy at most $O(\log k)$ or any 2-coloring leaves at least $2^{-O(k)}$ fraction of hyperedges monochromatic. Even though the direction reduction from Max-Cut results in a similar inapproximability factor with discrepancy even 1, this result does not rely on the UGC and hold even *all* edges (compared to *almost* in Section 3.1) have discrepancy $O(\log k)$.

Let $r = \Theta(\frac{k}{\log k})$ so that $s = \frac{k}{r} = \Theta(\log k)$ is an integer. Given a r-uniform hypergraph, it is NP-hard to distinguish whether it has discrepancy at most 2 or any 2-coloring leaves at least $r^{-O(r)}$ fraction of hyperedges monochromatic. Given a r-uniform hypergraph, the reduction replaces each vertex v with $\mathsf{cloud}(v)$ that contains $(2s-1)$ new vertices. Each hyperedge $(v_1, \ldots, v_r)$ is replaced by $d := (\binom{2s-1}{s})^r \leq (2^s)^r = 2^k$ hyperedges

$$\{\cup_{i=1}^r V_i : V_i \subset \mathsf{cloud}(v_i), |V_i| = s\}.$$

If the given r-uniform hypergraph has discrepancy at most 2, the resulting k-uniform hypergraph has discrepancy at most $2s = O(\log k)$.

If the resulting k-uniform hypergraph admits a coloring that leaves α fraction of hyperedges monochromatic, giving v the color that appears more in $\mathsf{cloud}(v)$ is guaranteed to leaves at most $d\alpha$ fraction of hyperedges monochromatic. Therefore, if any 2-coloring of the input r-uniform hypergraph leaves at least $r^{-O(r)}$ fraction of hyperedges monochromatic, any 2-coloring of the resulting k-uniform hypergraph leaves at least $\frac{r^{-O(r)}}{d} = 2^{-O(k)}$ fraction of hyperedges.

3.3 Hardness of Max-2-Coloring under almost $(k - \sqrt{k})$-colorability

Let k be such that $\ell := \sqrt{k}$ be an integer and let $\chi := k - \ell$. We prove the following hardness result for any $\epsilon > 0$ assuming the Unique Games Conjecture: given a k-uniform hypergraph such that there is a χ-coloring that have at least $(1 - \epsilon)$ fraction of hyperedges rainbow, it is NP-hard to find a 2-coloring that leaves at most $(\frac{1}{2})^{k-1}$ fraction of hyperedges monochromatic.

The main technique for this result is to show the existence of a *balanced pairwise independence distribution* with the desired support. Let μ be a distribution on $[\chi]^k$. μ is called balanced pairwise independent if for any $i \neq j \in [k]$ and $a, b \in [\chi]$,

$$\Pr_{(x_1,\ldots,x_k)\sim\mu}[x_i = a, x_j = b] = \frac{1}{\chi^2}.$$

For example, the uniform distribution on $[\chi]^k$ is a balanced pairwise distribution. We now consider the following distribution μ to sample $(x_1, \ldots, x_k) \in [\chi]^k$.

- Sample $S \subseteq [k]$ with $|S| = \chi$ uniformly at random. Let $S = \{s_1 < \cdots < s_\chi\}$.
- Sample a permutation $\pi : [\chi] \mapsto [\chi]$.
- Sample $y \in [\chi]$.
- For each $i \in [k]$, if $i = s_j$ for some $j \in [\chi]$, output $x_i = \pi(\chi)$. Otherwise, output $x_i = y$.

Note that for any supported by $(x_1, \ldots, x_k)$, we have $\{x_1, \ldots, x_k\} = [\chi]$. Therefore, μ is supported on *rainbow strings*. We now verify pairwise independence. Fix $i \neq j \in [k]$ and $a, b \in [\chi]$.

- If $a = b$, by conditioning on wheter i, j are in S or not,

$$\begin{aligned}\Pr_\mu[x_i = a, x_j = b] =& \Pr[x_i = a, x_j = b | i, j \in S] \Pr[i, j \in S] + \\ & \Pr[x_i = a, x_j = b | i \in S, j \notin S] \Pr[i \in S, j \notin S] + \\ & \Pr[x_i = a, x_j = b | i \notin, j \in S] \Pr[i \notin, j \in S] + \\ & \Pr[x_i = a, x_j = b | i, j \notin S] \Pr[i, j \notin S] \\ =& 0 \cdot (\frac{\chi(\chi-1)}{k(k-1)}) + 2 \cdot (\frac{1}{\chi^2}) \cdot (\frac{l\chi}{k(k-1)}) + (\frac{1}{\chi}) \cdot (\frac{\ell(\ell-1)}{k(k-1)}) \\ =& \frac{2\ell\chi + \chi(\ell^2 - \ell)}{\chi^2 k(k-1)} = \frac{\chi k + \chi\sqrt{k}}{\chi^2 k(k-1)} = \frac{\sqrt{k}(\sqrt{k}+1)}{\chi k(\sqrt{k}+1)(\sqrt{k}-1)} \\ =& \frac{1}{\chi(k - \sqrt{k})} = \frac{1}{\chi^2}.\end{aligned}$$

- If $a \neq b$, by the same conditioning,

$$\begin{aligned}\Pr_\mu[x_i = a, x_j = b] =& (\frac{1}{\chi(\chi-1)}) \cdot (\frac{\chi(\chi-1)}{k(k-1)}) + 2 \cdot (\frac{1}{\chi^2}) \cdot (\frac{\ell\chi}{k(k-1)}) + 0 \cdot (\frac{\ell(\ell-1)}{k(k-1)}) \\ =& \frac{\chi^2 + 2l\chi}{\chi^2 k(k-1)} = \frac{\chi + 2\ell}{\chi k(k-1)} = \frac{k + \sqrt{k}}{\chi k(k-1)} = \frac{1}{\chi^2}.\end{aligned}$$

Given such a balanced pairwise independent distribution supported on rainbow strings, a standard procedure following the work of Austrin and Mossel [7] shows that it is UG-hard to outperform the random 2-coloring. We omit the details.

4 Approximate Min-Coloring

In this section, we provide approximation algorithms for the Min-Coloring problem under strong colorability, rainbow colorability, and low discrepancy assumptions. Our approach is standard, namely, we first apply degree reduction algorithms followed by the usual paradigm pioneered by Karger, Motwani and Sudan [23], for coloring bounded degree (hyper)graphs. Consequently, our exposition will be brief.

In the interest of clarity, all results henceforth assume the special cases of Discrepancy-1, or $(k-1)$-rainbow colorability, or $(k+1)$-strong colorability. All arguments generalize easily to the cases parameterized by ℓ.

4.1 Approximate Min-Coloring in Bounded Degree Hypergraphs

4.1.1 The Algorithm

INPUT: k-uniform hypergraph $H = ([n], E)$ with max-degree t and m edges, having Discrepancy 1, or being $(k-1)$-rainbow colorable, or being $(k+1)$-strong colorable.

```
1. Let $u_1, \ldots, u_n$ be a solution to the SDP relaxation from Section 2.1 corresponding to the
   assumption on the hypergraph.
2. Let $H_1$ be a copy of $H$, and let $\gamma, \tau$ be parameters to be determined shortly.
3. Until no vertex remains in the hypergraph, Repeat:
   a. Find an independent set $\mathcal{I}$ in the residual hypergraph, of size at least $\gamma n$ by repeating
      the below process until $|\mathcal{I}| \geq \gamma n$:
      (A) Pick a random vector $r$ from the standard multivariate normal distribution.
      (B) For all $i$, if $\langle u_i, r\rangle \geq \tau$, add vertex $i$ to $\mathcal{I}$.
      (C) For every edge $e$ completely contained in $\mathcal{I}$, delete any single vertex in $e$, from $\mathcal{I}$.
   b. Color $\mathcal{I}$ with a new color and remove $\mathcal{I}$ and all edges involving vertices in $\mathcal{I}$, from $H_1$.
```

4.1.2 Analysis

First note that by Lemma 8, for any fixed vector a, $\langle a, r\rangle$ has the distribution $\mathcal{N}(0,1)$. Note that all SDP formulations in Section 2.1 satisfy,

$$\left\| \sum_{j\in[k]} u_{i_j} \right\|_2 \leq 1 \tag{4.1}$$

Now consider any edge $e = (i_1, \ldots, i_k)$. In any fixed iteration of the inner loop, the probability of e being contained in $\mathcal{I}$ at Step (B), is at most the probability of

$$\langle r, \sum_{j\in[k]} u_{i_j}\rangle \geq k\tau$$

However, by Lemma 8 and Eq. (4.1), the inner product above is dominated by the distribution $\mathcal{N}(0,1)$. Thus in any fixed iteration of the inner loop, let H_1 have n_1 vertices and m_1 edges, we have

$$\begin{aligned}\mathbf{E}[\mathcal{I}] &\geq n_1\Phi(\tau) - m_1\Phi(k\tau)\\ &\geq n_1 e^{-\tau^2/2} - \frac{n_1 t}{k} e^{-k^2\tau^2/2}\\ &= \Omega(\gamma n_1) \qquad\qquad \text{setting, } \tau^2 = \frac{2\log t}{k^2-1}, \quad \text{and} \quad \gamma = t^{-1/(k^2-1)}\end{aligned}$$

Now by applying Markov's inequality to the vertices not in $\mathcal{I}$, we have, $\mathbf{Pr}[|\mathcal{I}| < \gamma n_1] \leq 1 - \Omega(\gamma)$. Thus for a fixed iteration of the outer loop, with high probability, the inner loop doesn't repeat more than $O(\log n_1/\gamma)$ times.

Lastly, the outermost loop repeats $O(\log n/\gamma)$ times, using one color at each iteration. Thus with high probability, in polynomial time, the algorithm colors H with $t^{1/(k^2-1)}\log n$ colors.

Important Note. We can be more careful in the above analysis for the rainbow and strong colorability cases. Specifically, the crux boils down to finding the gaussian measure of the cone given by $\{x \mid U^T x \geq \tau\}$ instead of zero. Indeed, on closely following the proof of Theorem 15 we obtain for strong and rainbow coloring respectively (assuming max-degree n^k),

$$n^{\frac{1}{k}(1-\frac{3\beta}{2})}\log n \qquad \text{and} \qquad n^{\frac{1}{k}(1-\frac{5\beta}{4})}\log n, \qquad \text{where } \beta = \frac{\log k}{\log n}$$

While these improvements are negligible for small k, they are significant as k approaches $n^{2/3}$.

4.2 Degree Reduction Schemes under Promise

Wigderson [38] and Alon et al. [3] studied degree reduction in the cases of 3-colorable graphs and 2-colorable hypergraphs, respectively. Assuming our proposed structures, we are able to combine some simple combinatorial ideas with counterparts of the observations made by Wigderson and Alon et al., to obtain degree reduction approximation schemes. Such approximation schemes are likely not possible assuming only 2-colorability. Due to space constraints we defer these degree reduction algorithms to the full version [10].

4.3 Main Min-Coloring Result

Combining results from Section 4.1.2 with our degree reduction approximation schemes from the Section 4.2, we obtain the Min-Coloring results.

▶ **Theorem 28.** *Consider any k-uniform hypergraph $H = (V, E)$ with n vertices. In $n^{c+O(1)}$ time, one can color H with*

$$\min\left\{\left(\frac{n}{c\log n}\right)^{\alpha},\ n^{\frac{1}{k}(1-\frac{3\beta}{2})}, \left(\frac{m}{n}\right)^{\frac{1}{k^2}}\right\}\log n \text{ colors,} \quad \text{if } H \text{ is } (k+1)\text{-strongly colorable.}$$

$$\min\left\{\left(\frac{n}{c}\right)^{\alpha},\ n^{\frac{1}{k}(1-\frac{5\beta}{4})}, \left(\frac{m}{n}\right)^{\frac{1}{k^2}}\right\}\log n \text{ colors,} \quad \text{if } H \text{ is } (k-1)\text{-rainbow colorable.}$$

$$\min\left\{\left(\frac{n}{c}\right)^{\alpha},\ \left(\frac{m}{n}\right)^{\frac{1}{k^2}}\right\}\log n \text{ colors,} \quad \text{if } H \text{ has discrepancy } 1.$$

$$\text{where,} \quad \alpha = \frac{1}{k+2-o(1)}, \quad \beta = \frac{\log k}{\log n}$$

▶ Remark. In all three promise cases the general polytime min-coloring guarantee parameterized by ℓ, is roughly $n^{\ell^2/k}$. Thus, the threshold value of ℓ, for which standard min-coloring techniques improve with k, is $o(\sqrt{k})$.

References

1 Geir Agnarsson and Magnús M Halldórsson. Strong colorings of hypergraphs. In *Approximation and Online Algorithms*, pages 253–266. Springer, 2005.

2 Noga Alon. Personal communication, 2014.

3 Noga Alon, Pierre Kelsen, Sanjeev Mahajan, and Ramesh Hariharan. Approximate hypergraph coloring. *Nordic Journal of Computing*, 3(4):425–439, 1996.

4 Kazuhiko Aomoto et al. Analytic structure of schläfli function. *Nagoya Math. J*, 68:1–16, 1977.

5 Per Austrin, Venkatesan Guruswami, and Johan Håstad. $(2+\epsilon)$-SAT is NP-hard. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*. IEEE, 2014.

6 Per Austrin and Johan Håstad. On the usefulness of predicates. *ACM Trans. Comput. Theory*, 5(1):1:1–1:24, May 2013.

7 Per Austrin and Elchanan Mossel. Approximation resistant predicates from pairwise independence. *computational complexity*, 18(2):249–271, 2009.

8 Nikhil Bansal. Constructive algorithms for discrepancy minimization. In *Foundations of Computer Science (FOCS), 2014 IEEE 51st Annual Symposium on*, FOCS'10, pages 3–10. IEEE, 2010.

9 Nikhil Bansal and Subhash Khot. Inapproximability of hypergraph vertex cover and applications to scheduling problems. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming*, ICALP'10, pages 250–261, 2010.

10 V. Bhattiprolu, V. Guruswami, and E. Lee. Approximate Hypergraph Coloring under Low-discrepancy and Related Promises. *Arxiv*, 2015. arXiv:1506.06444.

11 Béla Bollobás, David Pritchard, Thomas Rothvoß, and Alex Scott. Cover-decomposition and polychromatic numbers. *SIAM Journal on Discrete Mathematics*, 27(1):240–256, 2013.

12 Károly Böröczky Jr and Martin Henk. Random projections of regular polytopes. *Archiv der Mathematik*, 73(6):465–473, 1999.

13 Hui Chen and Alan M Frieze. Coloring bipartite hypergraphs. In *Proceedings of the 5th international conference on Integer Programming and Combinatorial Optimization*, IPCO'96, pages 345–358, 1996.

14 Irit Dinur and Venkatesan Guruswami. PCPs via low-degree long code and hardness for constrained hypergraph coloring. In *Foundations of Computer Science (FOCS), 2014 IEEE 54th Annual Symposium on*, FOCS 13, pages 340–349, 2013.

15 Irit Dinur, Oded Regev, and Clifford D. Smyth. The hardness of 3-Uniform hypergraph coloring. *Combinatorica*, 25(1):519–535, 2005.

16 Paul Erdos and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and finite sets*, 10(2):609–627, 1975.

17 Venkatesan Guruswami, Johan Håstad, Prahladh Harsha, Srikanth Srinivasan, and Girish Varma. Super-polylogarithmic hypergraph coloring hardness via low-degree long codes. In *Proceedings of the 46th annual ACM Symposium on Theory of Computing*, STOC'14, 2014.

18 Venkatesan Guruswami, Johan Håstad, and Madhu Sudan. Hardness of approximate hypergraph coloring. *SIAM Journal on Computing*, 31(6):1663–1686, 2002.

19 Venkatesan Guruswami and Euiwoong Lee. Strong inapproximability results on balanced rainbow-colorable hypergraphs. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'15, pages 822–836, 2015.

20 Venkatesan Guruswami and Euiwoong Lee. Towards a characterization of approximation resistance for symmetric CSPs. *To Appear, The 18th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2015.

21 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, July 2001.

22 Sangxia Huang. $2^{(\log N)^{1/4-o(1)}}$ hardness for hypergraph coloring. *Electronic Colloquium on Computational Complexity (ECCC)*, 15-062, 2015.

23 David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM (JACM)*, 45(2):246–265, 1998.

24 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for Max-Cut and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007.

25 Subhash Khot and Rishi Saket. Hardness of coloring 2-colorable 12-uniform hypergraphs with $exp(\log^{\Omega(1)} n)$ colors. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 206–215. IEEE, 2014.

26 Hellmuth Kneser. Der simplexinhalt in der nichteuklidischen geometrie. *Deutsche Math*, 1:337–340, 1936.

27 László Lovász. On the shannon capacity of a graph. *Information Theory, IEEE Transactions on*, 25(1):1–7, 1979.

28 Shachar Lovett and Raghu Meka. Constructive discrepancy minimization by walking on the edges. In *Foundations of Computer Science (FOCS), 2014 IEEE 53rd Annual Symposium on*, FOCS 12, pages 61–67, 2012.

29 Colin McDiarmid. A random recolouring method for graphs and hypergraphs. *Combinatorics, Probability and Computing*, 2(03):363–365, 1993.

30 Jun Murakami and Masakazu Yano. On the volume of a hyperbolic and spherical tetrahedron. *Communications in analysis and geometry*, 13(2):379, 2005.

31 CA Rogers. An asymptotic expansion for certain schläfli functions. *Journal of the London Mathematical Society*, 1(1):78–80, 1961.

32 Claude Ambrose Rogers. *Packing and covering.* Number 54 in Cambridge Tracts in Mathematics and Mathematical Physics. Cambridge University Press, 1964.

33 Thomas Rothvoß. Constructive discrepancy minimization for convex sets. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 140–145. IEEE, 2014.

34 S. Sachdeva and R. Saket. Optimal inapproximability for scheduling problems via structural hardness for hypergraph vertex cover. In *Proceedings of the 28th annual IEEE Conference on Computational Complexity*, CCC'13, pages 219–229, 2013.

35 Ludwig Schläfli. On the multiple integral $\int \dots \int$ dx dy... dz, whose limits are $p1 = a1x + b1y + \cdots + h1z > 0, p2 > 0, ..., pn > 0$, and $x2 + y2 + \cdots + z2 < 1$. *Quart. J. Math*, 2(1858):269–300, 1858.

36 Joel Spencer. Six standard deviations suffice. *Transactions of the American Mathematical Society*, 289(2):679–706, 1985.

37 Cenny Wenner. Parity is positively useless. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: The 17th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 433–448. Schloss Dagstuhl, 2014.

38 Avi Wigderson. Improving the performance guarantee for approximate graph coloring. *Journal of the ACM (JACM)*, 30(4):729–735, 1983.

39 Uri Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 98 of *SODA'98*, pages 201–210, 1998.

Stochastic and Robust Scheduling in the Cloud*

Lin Chen[1], Nicole Megow[2], Roman Rischke[1], and Leen Stougie[3]

1 **Department of Mathematics, Technische Universität Berlin, Germany**
`{lchen,rischke}@math.tu-berlin.de`

2 **Center for Mathematics, Technische Universität München, Germany**
`nicole.megow@tum.de`

3 **Department of Econometrics and Operations Research, Vrije Universiteit Amsterdam & CWI, The Netherlands**
`stougie@cwi.nl`

Abstract

Users of cloud computing services are offered rapid access to computing resources via the Internet. Cloud providers use different pricing options such as (i) time slot reservation in advance at a fixed price and (ii) on-demand service at a (hourly) pay-as-used basis. Choosing the best combination of pricing options is a challenging task for users, in particular, when the instantiation of computing jobs underlies uncertainty.

We propose a natural model for two-stage scheduling under uncertainty that captures such resource provisioning and scheduling problem in the cloud. Reserving a time unit for processing jobs incurs some cost, which depends on when the reservation is made: a priori decisions, based only on distributional information, are much cheaper than on-demand decisions when the actual scenario is known. We consider both stochastic and robust versions of scheduling unrelated machines with objectives of minimizing the sum of weighted completion times $\sum_j w_j C_j$ and the makespan $\max_j C_j$. Our main contribution is an $(8+\epsilon)$-approximation algorithm for the min-sum objective for the stochastic polynomial-scenario model. The same technique gives a $(7.11+\epsilon)$-approximation for minimizing the makespan. The key ingredient is an LP-based separation of jobs and time slots to be considered in either the first or the second stage only, and then approximately solving the separated problems. At the expense of another ϵ our results hold for any arbitrary scenario distribution given by means of a black-box. Our techniques also yield approximation algorithms for robust two-stage scheduling.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.1 Combinatorics, G.3 Probability and Statistics

Keywords and phrases Approximation Algorithms, Robust Optimization, Stochastic Optimization, Unrelated Machine Scheduling, Cloud Computing

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.175

1 Introduction

Users of cloud computing services are offered rapid access to computing resources such as processing power, storage capacity, or network bandwidth via the Internet. Cloud providers, e.g. Amazon EC2, use different pricing options such as *on-demand* and *reserved instances* [1]. In the reservation option, a user pays a priori a fixed amount to reserve resources in advance, whereas on-demand instances are charged on a (e.g. hourly) pay-as-used basis. Users of cloud computing services face the challenging task of choosing the best combination of pricing

* Supported by the German Science Foundation (DFG) under contract ME 3825/1.

options when provisioning resources [4] – in particular, if instances of computing jobs underlie uncertainty.

In this paper, we propose the following general model for *two-stage scheduling with reservation cost under uncertainty* that captures such resource provisioning and scheduling problem in the cloud. In the first stage, we are given distributional information about scheduling scenarios, and in the second stage the actual scenario is revealed. The task is to construct a schedule for the realized scenario. Using a time unit of processing in the schedule incurs some fixed cost, independent of the used capacity (number of machines), but dependent on when the time unit is reserved: low if it is reserved in the first stage, not knowing the actual scenario, and high in the second stage, given full information. Such a cost structure applies, for example, when reserving a time unit on a server gives access to all processors on this server. In the *stochastic setting*, the overall goal is to minimize total expected payment (in both stages) plus scheduling cost. In the *robust setting*, the overall goal is to minimize the maximum, over all scenarios, of payment (in both stages) plus scheduling cost.

This setting opens up a whole new class of scheduling problems with its own particular challenges. As a first problem in this class we focus on scheduling preemptive jobs with release dates on unrelated machines, the most general machine model in scheduling, such as to minimize the total weighted completion time and makespan. The corresponding single-stage, single-scenario versions of these problems are fundamental classical scheduling problems. We give constant approximation algorithms for both objectives in the stochastic and the robust model. Our results for the stochastic setting hold in the most general random model, the so-called black-box model.

Problem Definition. In the underlying deterministic problem we are given a set of jobs $J = \{1, \dots, n\}$ and a set of machines $M = \{1, \dots, m\}$. Each job $j \in J$ is specified by a release date $r_j \geq 0$, before which j cannot be processed, a machine-dependent processing time $p_{ij} \in \mathbb{N}$, the processing time when executing j completely on machine $i \in M$, and a weight $w_j \geq 1$. In a feasible schedule each machine runs at most one job at the time and no job runs at more than one machine at the same time. A job may be preempted at any time and may resume processing on the same or any other machine. We assume that time is discretized into unit time slots. For ease of exposition let completion time C_j of a job $j \in J$ be the end of the unit-size time slot in which it actually completes. For every time slot, in which at least one machine is processing, a fixed *reservation cost* c is paid. The objective is to minimize the sum of weighted completion times $\sum_{j \in J} w_j C_j$ or the makespan $C_{\max} := \max_j C_j$ plus total reservation cost.

In the *two-stage* version of this problem, the actual job set to be processed is one of a set $\mathcal{S}$ of possible scenarios. Any time slot can be reserved either in the first stage at cost c, and can be used in every scenario, or in the second stage, for a specific scenario, at cost λc, where $\lambda \geq 1$ is an inflation factor. We assume λ to be defined by the scenario as well. The inflation factor together with the job set, $(\lambda_k \geq 1, J_k \subseteq J)$, make up a scenario $k \in \mathcal{S}$.

In the stochastic setting, we consider two models with respect to randomness. In the *polynomial-scenario model*, the distribution of $\mathcal{S}$ is given explicitly, i.e., each scenario $k \in \mathcal{S}$ is associated with a probability $\pi_k \in [0, 1]$ with $\sum_{k \in \mathcal{S}} \pi_k = 1$. In the *black-box model*, we have efficient access to an oracle that provides samples according to the unknown probability distribution with possibly exponentially many and dependent scenarios. In the robust setting, we restrict to the model with an explicit description of $\mathcal{S}$, called *discrete-scenario model.*

Related Work. Preemptively scheduling unrelated machines to minimize the sum of (weighted) completion times, $R \,|\, pmtn, (r_j) \,|\, \sum(w_j)C_j$, is $\mathcal{APX}$-hard [23] and admits a $(2+\epsilon)$-approximation [18]. The makespan problem $R \,|\, r_j, pmtn \,|\, C_{\max}$ can be solved in polynomial time [15].

Somewhat related to our scheduling problem with reservation cost is the problem of scheduling with variable time-slot cost [27, 14, 6]. Our second-stage problem involves a special case with time slots having cost either 0 or some fixed amount.

With respect to the stochastic problem, our work is closer to two- or multistage stochastic versions of scheduling problems, see e.g. [3, 2], than to traditional *dynamic stochastic scheduling* [17], in which the algorithm's decision on processing a job or not crucially influences the information release. However, the former involve different scheduling problems than ours, and more importantly performance quality is assessed by computational experiments instead of rigorous worst-case analysis. The only work on approximation algorithms for a two-stage scheduling problem we are aware of is by Shmoys and Sozio [21]. They present a $(2+\epsilon)$-approximation for two-stage stochastic throughput maximization on a single machine in which jobs can be deferred in the first stage gaining some profit.

The study of approximation algorithms for two-stage stochastic optimization problems was initiated in [8] with a polynomial-scenario model for a service-provision problem. Subsequently, next to [21] above, various two-stage stochastic versions of combinatorial optimization problems such as set cover, network design, maximum weight matching, etc. were studied, see [25] for a nice overview on the earlier work. General frameworks for solving several two-stage stochastic combinatorial optimization problems approximately have been proposed in [11] and [22]. The cost-sharing based approach in [11] yields a 2-approximation for a two-stage stochastic scheduling problem without release dates on identical parallel machines [16]. It is not clear how to apply it when there are release dates.

In the black-box model, we adopt the *Sample Average Approximation (SAA) method* proposed in [13]. It replaces the distribution on the random parameters by its empirical distribution defined by samples from it. Under certain conditions, good approximate solutions are obtained by drawing only a polynomial number of samples and solving the resulting SAA problem instead [5, 26].

In a two-stage setting, robust versions of multiple-scenario combinatorial optimization problems have been studied for covering and network design problems in [7, 9, 12]. We are not aware of any relevant scheduling work.

Our Contribution. We develop approximation algorithms for the stochastic and robust two-stage variants of classical scheduling problems. Our results rely on a new scheduling-tailored *time slot and job-set separation* procedure, which separates jobs into those processing exclusively on either first-stage reserved slots or second-stage reserved slots. It is inspired by [22] in which the idea of separating clients was introduced in the context of covering and network design problems. The separation in our setting is achieved through solving a generalization of the time-indexed unrelated machine scheduling LP [20] followed by an application of the slow-motion technique, proposed in [19] for min-sum single machine scheduling and extended later, among others, to unrelated machines scheduling in [18]. After separating, our rounding is applied independently to both separated instances. The two (possibly overlapping) solutions are merged to a feasible joint solution. Carefully balancing the change in reservation and scheduling cost by exploiting properties of slow-motion and α-points, the resulting procedure is proven to be an $(8+\epsilon)$-approximation algorithm for the two-stage polynomial-scenario stochastic version of $R \,|\, pmtn, r_j \,|\, \sum_j w_jC_j$ (Sec. 2.3).

Our time slot and job-set separation procedure is based on a general result, which is interesting on its own in the polynomial-scenario model: Given a ρ-approximation for the special case in which slots are reserved only in the first stage, there is an 8ρ-approximation for the two-stage model (Sec. 2.2). For this special case, we give a $\rho = (3+\epsilon)$-approximation (Sec. 2.1).

Our techniques also yield a $64/9 + \epsilon \approx 7.11 + \epsilon$ algorithm for the two-stage stochastic version of the makespan problem $R \,|\, r_j, pmtn \,|\, C_{\max}$ (Sec. 2.3).

Adopting the SAA framework, mentioned before, we apply our algorithms to arrive at a sampling-based $(8 + \epsilon)$-approximation algorithm for the min-sum problem and a $(64/9 + \epsilon)$-approximation for minimizing the makespan (Sec. 3). We notice that the work of [5, 26] leads to a first-stage reservation decision. It is not obvious in our model how to construct a good second-stage solution given a set of slots for free from the first-stage solution. In fact, considering this problem independently from the first-stage, it is unclear if a constant approximation exists. But even when considering the two stages jointly, the difficult part is to show how a second-stage solution for some scenario (not necessarily in the sample set) can be found and bounded by the SAA solution for the sample set.

Finally, we argue that our algorithms can be adopted for the min-max robust optimization model with a discrete set of scenarios (Sec. 4). For the min-$\sum w_j C_j$ problem, certain randomized steps of our algorithm must be replaced by deterministic ones losing a factor 2 in the approximation guarantee. For the robust makespan problem we derive a 2-approximation.

In this paper, we consider the most interesting and most general variants of the considered problems. For several special cases we can improve results, omitting details in this paper. E.g., when all jobs in all scheduling scenarios are released at time 0, then obviously the (first-stage) reservation interval will be $[0, t]$ for some t. It is not difficult to see that our considered objective functions (as well as others such as minimizing the ℓ_p-norm of machine loads) of the two-stage problems without release dates are convex in t. Hence, we find the optimal t simply by a combination of binary search for t and known approximation algorithms for the single-stage single-scenario problems to determine the total cost for a given t. Thus, in the absence of release dates, the two-stage problem is not harder than the underlying deterministic problem. This changes drastically when jobs have arbitrary release dates. Further improved results for other special cases, such as less general machine environments or a constant number of scenarios, will be given in the full version of the paper.

2 Polynomial-Scenario Model for Min-Sum Objective

Consider the two-stage stochastic version of $R \,|\, r_j, pmtn \,|\, \sum w_j C_j$ in the polynomial-scenario model, in which the set of scenarios $\mathcal{S}$ and its distribution are fully specified. For each scenario $k \in \mathcal{S}$ the triple (π_k, λ_k, J_k) describes its probability of occurring π_k, the inflation factor λ_k, and the set of jobs J_k.

We use a natural LP that generalizes and further relaxes the time-indexed LP for unrelated-machine scheduling [20]. To facilitate the exposition, we will present an LP with exponentially many variables and constraints and derive our algorithms based on its solution, even though we cannot expect to solve it in polynomial time. However, using the standard technique of time-intervals of geometrically increasing size [20] we obtain polynomial-time algorithms loosing only a small factor.

To keep notation amenable, we re-index jobs, such that each job j refers to a unique job-scenario combination, and we let $J := J_1 \cup \ldots \cup J_N$. We choose the time horizon $T = \max_{k \in \mathcal{S}, j \in J_k}\{r_j\} + \max_{k \in \mathcal{S}}\{\sum_{j \in J_k} \max_{i \in M} p_{ij}\}$, an obvious upper bound on any completion time in a reasonable schedule. Variables x_t and x_{kt} represent the first and second stage reservation decisions for time slot $[t, t+1)$. Let y_{ijt} be the amount of time job j is processed in interval $[t, t+1)$ on machine i.

$$\min \sum_{t=0}^{T-1} cx_t + \sum_{k=1}^{N} \pi_k \left(\sum_{j \in J_k} w_j C_j^{LP} + \lambda_k c \sum_{t=0}^{T-1} x_{kt} \right) \tag{1a}$$

$$\text{s.t.} \sum_{j \in J_k} y_{ijt} \le x_t + x_{kt} \qquad \forall i \in M, k \in \mathcal{S}, 0 \le t \le T-1 \tag{1b}$$

$$\sum_{i \in M} y_{ijt} \le x_t + x_{kt} \qquad \forall j \in J, k \in \mathcal{S}, 0 \le t \le T-1 \tag{1c}$$

$$x_t + x_{kt} \le 1 \qquad \forall k \in \mathcal{S}, 0 \le t \le T-1 \tag{1d}$$

$$\sum_{t=r_j}^{T-1} \sum_{i \in M} \frac{y_{ijt}}{p_{ij}} = 1 \qquad \forall j \in J \tag{1e}$$

$$\sum_{t=r_j}^{T-1} \sum_{i \in M} (t+1) \cdot \frac{y_{ijt}}{p_{ij}} = C_j^{LP} \qquad \forall j \in J \tag{1f}$$

$$x_t, x_{kt}, y_{ijt} \in [0,1] \qquad \forall i \in M, j \in J, k \in \mathcal{S}, 0 \le t \le T-1 \tag{1g}$$

Constraints (1b),(1d),(1e),(1g) are self-explaining. With (1c) we ensure that no job is processed for more than the total amount reserved in $[t, t+1)$ guaranteeing non-parallelism. For (1f), consider an arbitrary schedule with $t_j = \max_t\{t | y_{ijt} > 0, i \in M\}$, then the *true completion time* of job j in this schedule is $t_j + 1$, while C_j^{LP} offers a lower bound. Thus, even if we enforce all variables to be integral, the LP still gives a relaxation of our problem.

Given an LP solution (x_t, x_{kt}, y_{ijt}), we let $LP^r = \sum_t cx_t + c\sum_{k,t} \pi_k \lambda_k x_{kt}$ denote the reservation cost and we let $LP^s = \sum_{k, j \in J_k} \pi_k w_j C_j^{LP}$ denote the scheduling cost.

2.1 An Algorithm for First-Stage Reservation Only

Consider the special case of the two-stage problem in which all reservation must be done in the first stage; as if all inflation factors λ_k are excessive. We refer to it as *problem with first-stage reservation only*. A lower bound is given by LP_1 obtained from the above LP by setting $x_{kt} = 0$, for all k, t. W.l.o.g. we omit the π_k pre-multiplication in the objective function by assuming it to be incorporated into the weights $w_j, j \in J_k$.

We describe a procedure for rounding a fractional solution (x_t, y_{ijt}) of LP_1 to a feasible integer-value solution. We first round the first-stage decision on reserving slots x_t by maintaining a feasible LP solution, and then we determine the actual schedule. In the first step, it is important to maintain a fractional scheduling solution in which the true completion time of a job j, i.e., $\max\{t+1 \,|\, y_{ijt} > 0, i \in M\}$, does not diverge too much from C_j^{LP}. To that end, we utilize the idea of *slow-motion*, proposed in [19] for single machine scheduling, and extended to unrelated machines scheduling in [18].

For $\alpha \in [0,1]$, let $C_j(\alpha)$ denote the earliest point in time in the LP-solution in which job j has completed an α-fraction of its total processing requirement. We use the following link between $C_j(\alpha)$ an C_j^{LP} adopted from [10], which is used for the analysis of randomized algorithms.

▶ **Lemma 1** ([10]). $\int_0^1 C_j(\alpha) d\alpha = \sum_t \sum_i y_{ijt}/p_{ij} \cdot (t + 1/2) = C_j^{LP} - 1/2$.

For deterministic algorithms, however, we use the following relation.

▶ **Lemma 2** ([24]). $C_j^{LP} \ge \alpha + (1-\alpha) \cdot C_j(\alpha)$.

Slow-Motion. Given a fractional solution (x_t, y_{ijt}) for LP$_1$ and $\beta \geq 1$, we construct a new β-expanded solution $(\beta x_t, \beta y_{ijt})$ that we obtain by mapping every time point t to βt. Then βx_t indicates the amount of reservation for the interval $[\beta t, \beta(t+1))$, and βy_{ijt} the amount of job j scheduled during $[\beta t, \beta(t+1))$.

Obviously $(\beta x_t, \beta y_{ijt})$ is a new feasible solution to LP$_1$, which over-schedules each job by a fraction of $\beta - 1$. We simply delete the over-scheduled part and apply a lemma given in [19] that upper bounds the completion times of jobs in the expanded solution. We directly adopt their result to our requirement of job completion times being rounded up to integer values.

▶ **Lemma 3** ([19]). *The completion time of job j in the expanded solution is at most $\lceil \beta C_j(1/\beta) \rceil$.*

Rounding time slot reservation. Given any fractional solution (x_t, y_{ijt}), e.g. the expanded LP$_1$ solution, we show how to round the fractional reservation x_t to 0 or 1 so that the number of slots reserved will not be much higher than $\sum_t x_t$ but sufficiently large to accommodate all workload. We reassign the workload to reserved slots ensuring that the completion times remain relatively small.

We first apply a standard rounding technique, which we call *accumulated reservation:* Let $X_t = \sum_{h=0}^{t} x_h$, for $t \in \{0, 1, \dots, T-1\}$ and $X_{-1} = 0$. We set $\bar{x}_t = 1$, i.e., we reserve time slot $[t, t+1)$, if $\lfloor X_{t-1} \rfloor \leq \lfloor X_t \rfloor - 1$, and set $\bar{x}_t = 0$ otherwise. In total, we reserve $\lfloor \sum_t x_t \rfloor$ slots this way. To ensure sufficiently reserved time capacity we do an *extra reservation:* if $\bar{x}_t = 1$ and $\bar{x}_{t+1} = 0$ for some t, we reserve additionally the slot $[t+1, t+2)$. The number of extra reserved time slots is no more than the number of accumulatively reserved slots.

▶ **Proposition 4.** *Given a fractional solution x_t, the total cost for rounding to an integral time slot reservation $\bar{x}_t$ is $c \sum_{t=0}^{T-1} \bar{x}_t \leq 2c \lfloor \sum_{t=0}^{T-1} x_t \rfloor \leq 2LP^r$.*

Our reservation policy creates intervals $I_1, I_2, \dots$ of consecutive reserved time slots, each of them starting with a set of accumulatively reserved time slots and ending with a single extra time slot.

▶ **Lemma 5.** *Every interval $I_h = [\underline{t}_h, \bar{t}_h + 2)$ has enough capacity to accommodate all workload y_{ijt} assigned to time units $[\bar{t}_{h-1} + 1, \bar{t}_{h-1} + 2), \dots, [\underline{t}_{h+1} - 1, \underline{t}_{h+1})$.*

Proof. Consider interval I_h. Its last time slot $[\bar{t}_h + 1, \bar{t}_h + 2)$ is the extra reserved time slot. The total number (capacity) of the time slots in I_h is $\lfloor X_{\bar{t}_h} \rfloor - \lfloor X_{\bar{t}_{h-1}} \rfloor + 1$. By definition of our accumulative reservation and according to constraints (1b) and (1c), the total workload in terms of y_{ijt} in the interval $[\bar{t}_{h-1} + 1, \underline{t}_{h+1})$ is bounded by

$$\sum_{t=\bar{t}_{h-1}+1}^{\underline{t}_{h+1}-1} x_t = \sum_{t=0}^{\underline{t}_{h+1}-1} x_t - \sum_{t=0}^{\bar{t}_{h-1}} x_t \leq X_{\underline{t}_{h+1}-1} - \lfloor X_{\bar{t}_{h-1}} \rfloor \leq \lfloor X_{\bar{t}_h} \rfloor - \lfloor X_{\bar{t}_{h-1}} \rfloor + 1.$$

◀

The lemma implies that none of the workload y_{ijt}, fractionally assigned to time slots up to time slot $[\bar{t}_h, \bar{t}_h + 1)$, needs to be done later than $\bar{t}_h + 2$ if appropriately reassigned. In particular, this holds for the last reserved interval, i.e., all jobs in all scenarios will have been processed. Even stronger, workload assigned to the time slots $[\bar{t}_h + 1, \bar{t}_h + 2), \dots, [\underline{t}_{h+1} - 1, \underline{t}_{h+1})$ can be done within interval I_h, unless the release date of some job j is larger than $\bar{t}_h + 1$, preventing it to be scheduled in I_h.

Reassigning workload. Given a (not necessarily feasible) solution with fractional scheduling variables y_{ijt} and integer-valued reserved time slots $\bar{x}_t$, we describe a reassignment procedure to arrive at a feasible solution in which all workload $\bar{y}_{ijt}$ is assigned to reserved slots.

In increasing order of t we claim a total fraction x_t from time slot $[\underline{t}_h, \underline{t}_h + 1)$ if $\bar{t}_{h-1}+1 \leq t < \underline{t}_h$ for some h. All of the y_{ijt} is moved into this claimed space and added to $\bar{y}_{ij\underline{t}_h}$. Otherwise if $[t, t+1) \in I_h$, and $t \neq \bar{t}_h + 1$, then we claim $\frac{\lfloor X_t \rfloor - X_{t-1}}{X_t - X_{t-1}} x_t$ from $[t, t+1)$ and $\frac{X_t - \lfloor X_t \rfloor}{X_t - X_{t-1}} x_t$ from $[t+1, t+2)$. All of the y_{ijt} is moved in equal proportions $\frac{\lfloor X_t \rfloor - X_{t-1}}{X_t - X_{t-1}} y_{ijt}$ and $\frac{X_t - \lfloor X_t \rfloor}{X_t - X_{t-1}} y_{ijt}$ into the claimed space and added, respectively, to $\bar{y}_{ijt}$ and $\bar{y}_{ij(t+1)}$.

This assignment leaves (unclaimed) capacity $\lceil X_{\bar{t}_h} \rceil - X_{\bar{t}_h}$ of the extra reserved time slot $[\bar{t}_h + 1, \bar{t}_h + 2)$, for each h. As a second reassignment step we remove all y_{ijt} with $\bar{t}_{h-1} + 1 \leq t < \underline{t}_h$ and j with $r_j \leq \bar{t}_{h-1} + 1$ from $\bar{y}_{ij\underline{t}_h}$ and add it to $\bar{y}_{ij\tau}$ where $\tau = \bar{t}_{h-1} + 1$. This is feasible by Lemma 5.

▶ **Lemma 6.** *Applying the reservation and reassignment procedures to a feasible solution (x_t, y_{ijt}) of LP_1 increases the completion time of any job j by at most 1.*

Proof. For every t for which $y_{ijt} > 0$, there are two cases:
Case 1: $[t, t+1) \in I_h$, and $t \neq \bar{t}_h + 1$ for some h. Then by the assignment procedure y_{ijt} is moved into $[t, t+1)$ and $[t+1, t+2)$, whence that part of job j finishes at most 1 time unit later than in the unrounded solution. In particular, this holds for the last t such that $y_{ijt} > 0$.
Case 2: $\bar{t}_{h-1}+1 \leq t < \underline{t}_h$ for some h. If $r_j \leq \bar{t}_{h-1}+1$, then y_{ijt} is moved into $[\bar{t}_{h-1}+1, \bar{t}_{h-1}+2)$ and finishes earlier, or if $r_j > \bar{t}_{h-1} + 1$, then y_{ijt} is moved into $[\underline{t}_h, \underline{t}_h + 1)$. However, since $p_{ij} \geq 1$ (viz. integer), by the shifted-reservation policy job j cannot have been completed before $\underline{t}_h$, i.e., there must be a $t \geq \underline{t}_h$ and/or another i such that $y_{ijt} > 0$. ◀

The one-stage reservation algorithm. Given an optimal solution to LP$_1$, we apply slow-motion to expand the solution and the time slot reservation to obtain integral reservations to which we reassign the workload. It remains to specify the actual schedule for workload $\bar{y}_{ijt}$ within a time slot $[t, t+1)$ by ensuring that a job is not scheduled in parallel on multiple machines. This is essentially $R|pmtn|C_{max}$ in each time slot, which is polynomial-time solvable [15].

▶ **Theorem 7.** *The one-stage reservation algorithm is a* 3.5*-approximation for two-stage scheduling on unrelated machines with first-stage reservation only.*

Proof (Sketch). Consider an optimal solution to LP$_1$. By slow-motion we derive a β-expanded solution with a reservation cost of βLP^r, and job j completes at time $\lceil \beta C_j(1/\beta) \rceil$ by Lemmas 1 and 3. Applying the rounding of time slot reservation and then reassigning workload, Proposition 4 shows that the reservation cost becomes $2\beta LP^r$, while Lemma 6 ensures that job j completes at $\lceil \beta C_j(1/\beta) \rceil + 1$. Thus, by choosing the expansion parameter β at random according to the density function $f(\alpha) = 3\alpha^2$ where $\alpha = 1/\beta \in [0, 1]$, the total cost for reservation and scheduling can be bounded by:

$$\begin{aligned} 2LP^r \int_0^1 \frac{1}{\alpha} f(\alpha) d\alpha \quad &+ \quad \sum_j w_j \int_0^1 (\lceil C_j(\alpha)/\alpha \rceil + 1) f(\alpha) d\alpha \\ &\leq \quad 3(LP^r + LP^s) + 1/2 \sum_j w_j. \end{aligned}$$

Obviously, $\sum_j w_j \leq \sum_j w_j C_j^{LP}$, since $C_j^{LP} \geq 1$, which implies that the algorithm produces a solution with objective value bounded by $3LP^r + 3.5LP^s$. ◀

A refinement of the algorithm and its analysis give an improved bound.

▶ **Theorem 8.** *There exists a $(3+\epsilon)$-approximation algorithm for the two-stage scheduling problem on unrelated machines with first-stage reservation only.*

2.2 A Generic Algorithm for Two-Stage Scheduling

We first give a simple algorithm that allows a black-box application of the one-stage reservation algorithm above to obtain the following general result.

▶ **Theorem 9.** *Given a ρ-approximation algorithm for the two-stage problem with only first-stage reservation, there exists an 8ρ-approximation algorithm for the two-stage problem.*

The crucial ingredient is to separate the time slots and jobs to be considered for only first-stage or only second-stage reservation.

▶ **Lemma 10.** *Given an optimal solution (x_t, x_{kt}, y_{ijt}) to the LP with objective value $LP^r + LP^s$, there exists a feasible solution $(x'_t, x'_{it}, y'_{hjt})$ satisfying the following* **separation property***:*

- *Any time unit is reserved either in the first stage or in the second stage, or not at all; i.e., for all t, $x'_t > 0 \Rightarrow x'_{kt} = 0\ \forall k$.*
- *A job is scheduled either completely in slots reserved in the first stage, or completely in slots reserved in the second stage, i.e., $J = J_I \cup J_{II}$, s.t. $J_I = \{j \mid x'_t = 0 \Rightarrow y'_{ijt} = 0\ \forall ijt\}$ and $J_{II} = \{j \mid \sum_k x'_{kt} = 0 \Rightarrow y'_{ijt} = 0\,\forall ijt\}$.*
- *The objective value is at most $2LP^r + 4LP^s$.*

Proof. We first double the number of time units: for every time unit $[t, t+1)$ we obtain two time units $[2t, 2t+1)$ and $[2t+1, 2t+2)$. We reserve x_t of the even slot $[2t, 2t+1)$, and x_{kt} of the odd slot $[2t+1, 2t+2)$. We split y_{ijt} accordingly, such that for each of the slots $[2t, 2t+1)$ and $[2t+1, 2t+2)$ constraints (1b) and (1c) are satisfied. Notice that in this way we have blown up the scheduling cost by a factor of 2, while the reservation cost remains the same. Furthermore, notice that every job is processed at least half either in odd slots or in even slots. Thus by doubling again each slot and reserving in each of the two the same fraction, we can enforce a job to be either entirely scheduled in slots that are reserved in the first stage, or in slots reserved in the second stage. ◀

Proof (Thm. 9). Let $(x'_t, x'_{it}, y'_{ijt})$ be a feasible LP solution that satisfies the separation property and has objective value $Z' \le 2LP^r + 4LP^s$. We show that the existence of a ρ-approximation algorithm for the problem with first-stage reservation only implies the existence of an algorithm that produces a feasible schedule for the two-stage problem with total cost at most $2\rho Z'$.

Since jobs are divided into J_I and J_{II} in the solution $(x'_t, x'_{it}, y'_{ijt})$, we denote by Z'_I and Z'_{II} their contributions in Z' respectively: $Z' = Z'_I + Z'_{II}$. Consider scheduling J_I with only first-stage reservation and let Z_I be the optimal value of the corresponding LP. Similarly, let Z_{II} be the optimal value of the LP for scheduling J_{II} with only second-stage reservation. Clearly, $Z_I \le Z'_I$ and $Z_{II} \le Z'_{II}$. The ρ-approximation algorithm for the problem with only first-stage reservation for J_I returns a feasible schedule of cost at most ρZ_I. The problem of reserving and scheduling jobs only in the second stage can be separated into N single scenario problems, each of which is like a first-stage reservation problem, we thus also get a feasible schedule of cost at most ρZ_{II} for J_{II}.

Now we need to merge the two schedules for J_I and J_{II}. Notice that the two schedules may overlap in the sense that some slot is reserved in both schedules. To handle this we further double the two schedules. For the schedule of J_I, we double the time units and put whatever is scheduled in $[t, t+1)$ into the even slot $[2t, 2t+1)$, while for the schedule of J_{II}, we put whatever is scheduled in $[t, t+1)$ into the odd slot $[2t+1, 2t+2)$. Now the total cost of the merged solution is bounded by $2\rho(Z_I + Z_{II}) \le 2\rho Z'$. ◀

Directly applying Theorem 9 gives us a $8 \cdot (3+\epsilon) = 24 + \epsilon'$-approximation.

2.3 A Refined Two-Stage Algorithm

▶ **Theorem 11.** *There is an $(8+\epsilon)$-approximation algorithm for the two-stage stochastic variant of $R \,|\, r_j, pmtn \,|\, \sum w_j C_j$ in the polynomial-scenario model.*

Proof (Idea). Given an optimal LP solution, we first apply slow-motion to get a β-expanded solution. Then we apply time slot and job-set separation (Lemma 10) and obtain jobs and slots to be covered by either first-stage or second-stage reservation only. Then, we apply the technique of accumulative and extra reservation and reassign the workload *separately* on the slots reserved in the first and second stage. Here the last procedure must be carried out with caution so that after we separately round first and second stage reservation, they do not overlap. A careful analysis gives the result. ◀

We conclude this section by remarking that our techniques lead to the following result for the makespan objective.

▶ **Theorem 12.** *There is a $(64/9+\epsilon)$-approximation algorithm for the two-stage stochastic variant of $R \,|\, r_j, pmtn \,|\, C_{\max}$ in the polynomial-scenario model.*

3 The Black-Box Model

We now show that at the expense of another ϵ our results for the two-stage stochastic min-sum and makespan problem hold for any arbitrary scenario distribution given by means of a black-box. Besides that, the problem is as before.

Given a first-stage reservation $\bar{x} \in \{0,1\}^T$, a lower bound on the second-stage cost for a scenario S_k is as follows:

$$q(\bar{x}, S_k) = \min \left\{ \sum_{j \in J_k} w_j C_j^{LP} + \lambda_k c \sum_{t=0}^{T-1} x_{kt} \;\middle|\; \text{(1b) - (1g)} \wedge x_t = \bar{x}_t \;\forall t \right\}.$$

Let $c(x)$ denote the cost of a (possibly fractional) reservation $x \in [0,1]^T$. Then the following gives a lower bound on our two-stage stochastic scheduling problem.

$$\min_{x \in [0,1]^T} f(x) = c(x) + E_{S \in \mathcal{S}} \left[q(x, S) \right]. \qquad (2)$$

For an unknown distribution in the black-box model we cannot solve this problem efficiently. However, using the SAA method [13] we can approximate it. We draw a number N of independent samples $S_1, \ldots, S_N$ from the black-box and solve the following sample average problem:

$$\min_{x \in [0,1]^T} \hat{f}(x) = c(x) + \frac{1}{N} \cdot \sum_{k=1}^{N} q(x, S_k). \qquad (3)$$

Notice that (3) is exactly the LP of Section 2 with all N scenarios having probability $1/N$, and can thus be solved efficiently. It remains to determine the number of samples N that is needed to guarantee a certain approximation. We can show that our LP in (2) can be cast as a stochastic LP of type required in [26] for obtaining such a result. To that end, we must be given $\lambda := \max_{k\in S}\lambda_k$.

▶ **Lemma 13** ([26]). *There is a polynomially bounded number N such that any optimal solution x^{LP} to the sample average problem* (3) *with N samples satisfies $f(x^{LP}) \le (1+\epsilon)\min_x f(x)$ with high probability.*

Based on this lemma we can obtain a good integral first-stage solution. We draw N samples and solve problem (3). Let $(x_t^{LP}, x_{kt}^{LP}, y_{ijt}^{LP})$ be an optimal (fractional) solution. Applying our rounding technique (Sec. 2) we derive a solution $(\bar{x}_t, \bar{x}_{kt}, \bar{y}_{ijt})$ with $(\bar{x}_t, \bar{x}_{kt}) \in \{0,1\}$. We fix $\bar{x}_t$ as first-stage reservation.

The difficult part is to find a second-stage solution for some scenario (that is not necessarily in the sample set) and bound it by the LP solution for the sample set. The key is that our rounding procedure for the first stage reservation x only depends on x itself and is independent of the scheduling solution. Given $\bar{x}_t$ and a scenario S, we solve the resulting second-stage problem as follows: we solve the problem $\min_{x\in[0,1]^T} c(x^{LP}) + q(x^{LP}, S)$, which is again exactly the LP of Section 2 with a single scenario S, after fixing first-stage reservation at $x_t = x_t^{LP}$. Let (x'_{kt}, y'_{ijt}) be the optimal solution. Plugging in x_t^{LP} and applying our rounding procedure on $(x_t^{LP}, x'_{kt}, y'_{ijt})$, we get a feasible schedule of total cost at most $(\rho+\epsilon)(c(x^{LP}) + q(x^{LP}, S))$, with $\rho = 8$ for the min-sum objective and $\rho = 64/9$ for the makespan. And, most importantly, the first stage reservation $\bar{x}_t$ is consistent with our first-stage reservation. Using Lemma 13 we have in expectation total cost of at most $(\rho+\epsilon)f(x^{LP}) \le (\rho + O(\epsilon))\min_x f(x) \le (\rho+\epsilon')Z^*$.

▶ **Theorem 14.** *In the black-box model, there is a $(\rho+\epsilon)$-approximation algorithm for two-stage stochastic variant of $R\,|\,r_j, pmtn\,|\sum w_jC_j$ $(\rho = 8)$ and $R\,|\,r_j, pmtn\,|\,C_{\max}$ $(\rho = 64/9)$, respectively.*

4 Two-Stage Robust Scheduling

In the robust setting, we restrict to the model with an explicit description of the scenario set $\mathcal{S}$. The objective is now to minimize the worst-case total cost instead of the expected total cost. Notice that the LP relaxations, that our algorithms in Sec. 2 rely on, can be easily adopted.

Our approximation algorithms for the stochastic model are risk-averse, i.e., the performance guarantee holds for every scenario. Therefore, the techniques used for the stochastic model also apply to the discrete-scenario robust model. For the min-$\sum w_jC_j$ problem, certain randomized steps of our algorithm must be replaced by deterministic ones losing a factor 2 in the approximation guarantee. Such an adaptation is not needed for the robust makespan problem and we directly obtain again a $(7.11+\epsilon)$-approximation algorithm. However, the makespan problem is much easier and we provide a simple 2-approximation algorithm.

▶ **Theorem 15.** *For two-stage discrete-scenario robust scheduling with reservation cost, there is a ρ-approximation algorithm for the scheduling problems $R\,|\,r_j, pmtn\,|\sum w_jC_j$ $(\rho = 16+\epsilon)$ and $R\,|\,r_j, pmtn\,|\,C_{\max}$ $(\rho = 2)$, respectively.*

5 Conclusion

Inspired by the resource provisioning problem of cloud users, we propose an optimization model that reflects two-stage decision processes in which computing resources must be reserved under uncertainty about the set of computational tasks. It leads to a new class of scheduling problems. We present first results that suggest higher approximation complexity than their single stage, single scenario versions. The quest for better approximations is left for future research.

We also leave open the approximability of the equivalent non-preemptive scheduling problems with release dates. Notice that the second-stage problem would not admit a constant approximation (large inflation factor, 2-partition), unless P=NP, when considering it independently of the first stage problem. However, a two-stage algorithm may yield a constant-factor approximation.

Another interesting variation of the problem arises if a user may reserve machines individually, possibly at machine-dependent rates. We note that, even if reservation costs are uniform over the machines, our proposed LP relaxation has a non-constant integrality gap in this case.

References

1 Amazon EC2 Pricing Options: `https://aws.amazon.com/ec2/pricing/`.

2 Talal Al-Khamis and Rym M'Hallah. A two-stage stochastic programming model for the parallel machine scheduling problem with machine capacity. *Computers & OR*, 38(12):1747–1759, 2011.

3 Gerard M. Campbell. A two-stage stochastic program for scheduling and allocating cross-trained workers. *JORS*, 62(6):1038–1047, 2011.

4 Sivadon Chaisiri, Bu-Sung Lee, and Dusit Niyato. Optimization of resource provisioning cost in cloud computing. *IEEE Trans. Serv. Comput.*, 5(2):164–177, 2012.

5 Moses Charikar, Chandra Chekuri, and Martin Pál. Sampling bounds for stochastic optimization. In *Proc. of APPROX and RANDOM 2005*, pages 257–269, 2005.

6 Lin Chen, Nicole Megow, Roman Rischke, Leen Stougie, and José Verschae. Optimal algorithms and a PTAS for cost-aware scheduling. *To appear in Proc. of MFCS 2015*, 2015.

7 Kedar Dhamdhere, Vineet Goyal, R. Ravi, and Mohit Singh. How to pay, come what may: Approximation algorithms for demand-robust covering problems. In *Proc. of FOCS 2005*, pages 367–376, 2005.

8 Shane Dye, Leen Stougie, and Asgeir Tomasgard. The stochastic single resource service-provision problem. *Naval Res. Logist.*, 50(8):869–887, 2003.

9 Uriel Feige, Kamal Jain, Mohammad Mahdian, and Vahab S. Mirrokni. Robust combinatorial optimization with exponential scenarios. In *Proc. of IPCO 2007*, pages 439–453, 2007.

10 Michel X. Goemans. Improved approximation algorthims for scheduling with release dates. In *Proc. of SODA 1997*, pages 591–598, 1997.

11 Anupam Gupta, Martin Pál, R. Ravi, and Amitabh Sinha. Sampling and cost-sharing: Approximation algorithms for stochastic optimization problems. *SIAM J. Comput.*, 40(5):1361–1401, 2011.

12 Rohit Khandekar, Guy Kortsarz, Vahab S. Mirrokni, and Mohammad R. Salavatipour. Two-stage robust network design with exponential scenarios. *Algorithmica*, 65(2):391–408, 2013.

13 Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM J. Optim.*, 12(2):479–502, 2001.

14 Janardhan Kulkarni and Kamesh Munagala. Algorithms for cost aware scheduling. In *Proc. of WAOA 2012*, pages 201–214, 2013.

15 Eugene L. Lawler and Jacques Labetoulle. On preemptive scheduling of unrelated parallel processors by linear programming. *J. ACM*, 25(4):612–619, 1978.

16 Stefano Leonardi, Nicole Megow, Roman Rischke, Leen Stougie, Chaitanya Swamy, and José Verschae. Scheduling with time-varying cost: Deterministic and stochastic models. Presentation at the 11th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP 2013), 2013.

17 Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 2008.

18 Maurice Queyranne and Maxim Sviridenko. A $(2+\epsilon)$-approximation algorithm for the generalized preemptive open shop problem with minsum objective. *J. Algorithms*, 45(2):202–212, 2002.

19 Andreas S. Schulz and Martin Skutella. Random-based scheduling: New approximations and LP lower bounds. In *Proc. of APPROX and RANDOM 1997*, pages 119–133, 1997.

20 Andreas S. Schulz and Martin Skutella. Scheduling unrelated machines by randomized rounding. *SIAM J. Discrete Math.*, 15(4):450–469, 2002.

21 David B. Shmoys and Mauro Sozio. Approximation algorithms for 2-stage stochastic scheduling problems. In *Proc. of IPCO 2007*, pages 145–157. Springer, 2007.

22 David B. Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *J. ACM*, 53(6):978–1012, 2006.

23 René A. Sitters. Approximability of average completion time scheduling on unrelated machines. In *Proc. of ESA 2008*, pages 768–779, 2008.

24 Martin Skutella. List scheduling in order of α-points on a single machine. In Evripidis Bampis, Klaus Jansen, and Claire Kenyon, editors, *Efficient Approximation and Online Algorithms*, volume 3484 of *LNCS*, pages 250–291. Springer, 2006.

25 Chaitanya Swamy and David B. Shmoys. Approximation algorithms for 2-stage stochastic optimization problems. *SIGACT News*, 37(1):33–46, 2006.

26 Chaitanya Swamy and David B. Shmoys. Sampling-based approximation algorithms for multistage stochastic optimization. *SIAM J. Comput.*, 41(4):975–1004, 2012.

27 G. Wan and X. Qi. Scheduling with variable time slot costs. *Naval Research Logistics*, 57:159–171, 2010.

On Approximating Node-Disjoint Paths in Grids

Julia Chuzhoy*[1] and David H. K. Kim†[2]

1 **Toyota Technological Institute at Chicago**
6045 S. Kenwood Ave., Chicago, Illinois 60637, USA
`cjulia@ttic.edu`

2 **Department of Computer Science, University of Chicago**
1100 East 58th Street, Chicago, Illinois 60615, USA
`hongk@cs.uchicago.edu`

Abstract

In the Node-Disjoint Paths (NDP) problem, the input is an undirected n-vertex graph G, and a collection $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ of pairs of vertices called *demand pairs*. The goal is to route the largest possible number of the demand pairs (s_i, t_i), by selecting a path connecting each such pair, so that the resulting paths are node-disjoint. NDP is one of the most basic and extensively studied routing problems. Unfortunately, its approximability is far from being well-understood: the best current upper bound of $O(\sqrt{n})$ is achieved via a simple greedy algorithm, while the best current lower bound on its approximability is $\Omega(\log^{1/2-\delta} n)$ for any constant δ. Even for seemingly simpler special cases, such as planar graphs, and even grid graphs, no better approximation algorithms are currently known. A major reason for this impasse is that the standard technique for designing approximation algorithms for routing problems is LP-rounding of the standard multicommodity flow relaxation of the problem, whose integrality gap for NDP is $\Omega(\sqrt{n})$ even on grid graphs.

Our main result is an $O(n^{1/4} \cdot \log n)$-approximation algorithm for NDP on grids. We distinguish between demand pairs with both vertices close to the grid boundary, and pairs where at least one of the two vertices is far from the grid boundary. Our algorithm shows that when all demand pairs are of the latter type, the integrality gap of the multicommodity flow LP-relaxation is at most $O(n^{1/4} \cdot \log n)$, and we deal with demand pairs of the former type by other methods. We complement our upper bounds by proving that NDP is APX-hard on grid graphs.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Node-disjoint paths, approximation algorithms, routing and layout

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.187

1 Introduction

In the classical Node-Disjoint Paths (NDP) problem, the input is an undirected n-vertex graph $G = (V, E)$, and a collection $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ of pairs of vertices, called *source-destination*, or *demand*, pairs, that we would like to route. In order to route a pair (s_i, t_i), we need to select some path P connecting s_i to t_i. The goal is to route the largest possible number of the demand pairs on node-disjoint paths: that is, every vertex of G may participate in at most one path in the solution.

NDP is one of the most basic and extensively studied routing problems. When the number of the demand pairs k is bounded by a constant, Robertson and Seymour [27, 29] have

* Supported in part by NSF grant CCF-1318242.
† Supported in part by NSF grant CCF-1318242.

shown an efficient algorithm for the problem, as part of their seminal Graph Minors project. However, when k is a part of the input, the problem is known to be NP-hard [17]. Even though the NDP problem, together with its many variants, has been extensively studied, its approximability is still poorly understood. The best currently known upper bound on the approximation factor is $O(\sqrt{n})$ [22], achieved by the following simple greedy algorithm: start with graph G and an empty solution. While G contains any path connecting any demand pair, choose the shortest such path P, add P to the solution, and delete all vertices of P from G. Surprisingly, this elementary algorithm is the best currently known approximation algorithm for NDP, even for restricted special cases of the problem, where the input graph G is a planar graph, or even just a grid. On the negative side, it is known that there is no $O(\log^{1/2-\delta} n)$-approximation algorithm for NDP for any constant δ, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{poly}\log n})$ [5, 4]. Perhaps the biggest obstacle in breaking the $O(\sqrt{n})$-approximation barrier for the problem is the fact that the integrality gap of the standard multicommodity flow LP-relaxation for NDP is $\Omega(\sqrt{n})$, even in grid graphs. In the LP-relaxation, instead of connecting the demand pairs by paths, we try to send as much flow as possible between the demand pairs, subject to the constraint that each vertex carries at most one flow unit. The $O(\sqrt{n})$-approximation greedy algorithm described above can be cast as an LP-rounding algorithm for the multicommodity flow LP, and therefore, the integrality gap of the LP is $\Theta(\sqrt{n})$. So far, rounding this LP relaxation has been the main method used in designing approximation algorithms for a variety of routing problems, and it appears that new techniques are needed in order to improve the $O(\sqrt{n})$-approximation factor for NDP.

In this paper we break the $O(\sqrt{n})$-barrier on the approximation factor for NDP on grid graphs [1], by providing an $O(n^{1/4} \cdot \log n)$-approximation algorithm. Our algorithm distinguishes between two types of demand pairs: an (s_i, t_i) pair is *bad* if both s_i and t_i are close to the grid boundary, and it is good otherwise. Interestingly, the standard integrality gap examples for the multicommodity flow LP relaxation usually involve a grid graph, and bad demand pairs. Our algorithm deals with bad and good demand pairs separately, and in particular it shows that if all demand pairs are good, then the integrality gap of the LP relaxation becomes $O(n^{1/4} \cdot \log n)$ (but unfortunately it still remains polynomial in n - see Section 6). We complement these results by showing that NDP is APX-hard even on grid graphs. We believe that understanding the approximability of NDP on grid graphs is an important first step towards understanding the approximability of the NDP problem in general, as grids seem to be the simplest graphs, for which the approximability of the NDP problem is widely open, and the integrality gap of the multicommodity flow LP is $\Omega(\sqrt{n})$. We hope that some of the techniques introduced in this paper will be helpful in breaking the $O(\sqrt{n})$-approximation barrier in general planar graphs.

NDP in grid graphs has been studied in the past, often in the context of VLSI layout. Aggarwal, Kleinberg and Williamson [1] consider a special case, where the set of the demand pairs is a permutation — that is, every vertex of the grid participates in exactly one demand pair. They show that for any permutation, one can route $\Omega(\sqrt{n}/\log n)$ demand pairs. They also show that with spacing d, every permutation contains a set of $\Omega(\sqrt{nd}/\log n)$ pairs that can be routed on node-disjoint paths. Our algorithm for routing on grids is inspired by their work.

Cutler and Shiloach [16] studied NDP in grids in the following three settings. They assume that all source vertices appear on the top row R_1 of the grid, and all destination

[1] Since n denotes, by convention, the number of vertices in the input graph, the size of the grid is $(\sqrt{n} \times \sqrt{n})$.

vertices appear on some other row R_ℓ of the grid, sufficiently far from the top and the bottom rows (for example, $\ell = \lceil n/2 \rceil$). In the packed-packed setting, the sources are a set of k consecutive vertices of R_1, and the destinations are a set of k consecutive vertices of R_ℓ. They show a necessary and a sufficient condition for when all demand pairs can be routed in the packed-packed instance. The second setting is the packed-spaced setting. Here, the sources are again a set of k consecutive vertices of R_1, but the distance between every consecutive pair of the destination vertices on R_ℓ is at least d. For this setting, the authors show that if $d \geq k$, then all demand pairs can be routed. We extend their algorithm to a more general setting, where the destination vertices may appear anywhere in the grid, as long as the distance between any pair of the destination vertices, and any destination vertex and the boundary of the grid, is at least $\Omega(k)$. This extension of the algorithm of [16] is used as a basic building block in both our algorithm, and the APX-hardness proof. We note that Robertson and Seymour [28] provided sufficient conditions for the existence of node-disjoint routing of a given set of demand pairs in the more general setting of graphs drawn on surfaces, and they provide an algorithm whose running time is $\text{poly}(n) \cdot f(k)$ for finding the routing, where $f(k)$ is at least exponential in k. Their result implies the existence of the routing on grids, when the destination vertices are sufficiently spaced from each other and from the grid boundaries. However, we are not aware of an algorithm for finding the routing, whose running time is polynomial in n and k, and we provide such an algorithm here. The third setting studied by Cutler and Shiloach is the spaced-spaced setting, where the distance between any pair of source vertices, and any pair of destination vertices is at least d. The authors note that they could not come up with a better algorithm for this setting, than the one provided for the packed-spaced case.

Other Related Work

A problem closely related to NPD is the Edge-Disjoint Paths (EDP) problem. It is defined similarly, except that now the paths chosen to the solution are allowed to share vertices, and are only required to be edge-disjoint. It is easy to show, by using a line graph of the EDP instance, that NDP is more general than EDP. The approximability status of EDP is very similar to that of NDP: there is an $O(\sqrt{n})$-approximation algorithm [13], and it is known that there is no $O(\log^{1/2-\delta} n)$-approximation algorithm for any constant δ, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{poly}\log n})$ [5, 4]. As in the NDP problem, we can use the standard multicommodity flow LP-relaxation of the problem, in order to obtain the $O(\sqrt{n})$-approximation algorithm, and the integrality gap of the LP-relaxation is $\Omega(\sqrt{n})$ even on planar graphs. However, for even-degree planar graphs, Kleinberg [19], building on the work of Chekuri, Khanna and Shepherd [12, 11], has shown an $O(\log^2 n)$-approximation LP-rounding algorithm. Aumann and Rabani [8] showed an $O(\log^2 n)$-approximation algorithm for EDP on grid graphs, and Kleinberg and Tardos [21, 20] showed $O(\log n)$-approximation algorithms for wider classes of nearly-Eulerian uniformly high-diameter planar graphs, and nearly-Eulerian densely embedded graphs. Recently, Kawarabayashi and Kobayashi [18] gave an $O(\log n)$-approximation algorithm for EDP when the input graph is either 4-edge-connected planar or Eulerian planar. It appears that the restriction of the graph G to be Eulerian, or near-Eulerian, makes the EDP problem significantly simpler, and in particular improves the integrality gap of the LP-relaxation. The analogue of the grid graph for the EDP problem is the wall graph (see Figure 1): the integrality gap of the standard LP relaxation for EDP on wall graphs is $\Omega(\sqrt{n})$, and to the best of our knowledge, no better than $O(\sqrt{n})$-approximation algorithm for EDP on walls is known. Our $O(n^{1/4} \cdot \log n)$-approximation algorithm for NDP on grids can be extended to the EDP problem on wall graphs (see Section 7).

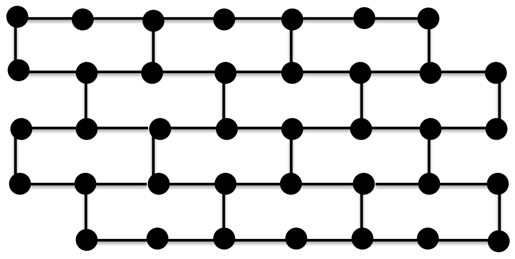


Figure 1 A wall graph.

A variation of the NPD and EDP problems, where small congestion is allowed, has been a subject of extensive study. In the NDP with congestion (NDPwC) problem, the input is the same as in the NDP problem, and we are additionally given a non-negative integer c. The goal is to route as many of the demand pairs as possible with congestion at most c: that is, every vertex may participate in at most c paths in the solution. EDP with Congestion (EDPwC) is defined similarly, except that now the congestion bound is imposed on edges and not vertices. The classical randomized rounding technique of Raghavan and Thompson [25] gives a constant-factor approximation for both problems, if the congestion c is allowed to be as high as $\Theta(\log n / \log\log n)$. A recent line of work [12, 24, 3, 26, 14, 15, 10, 9] has lead to an $O(\text{poly}\log k)$-approximation for both NDPwC and EDPwC problems, with congestion $c = 2$. For planar graphs, a constant-factor approximation with congestion 2 is known [30]. All these algorithms perform LP-rounding of the standard multicommodity flow LP-relaxation of the problem.

Organization

We start with Preliminaries in Section 2, and show a generalization of the algorithm of Cutler and Shiloah [16] for routing with well-separated destinations in Section 3. In Section 4 we provide an $O(n^{1/4} \cdot \log n)$-approximation algorithm for NDP on grids, and we provide the APX-hardness proof in Section 5. We discuss the integrality gap of the multicommodity flow LP-relaxation when all terminals are far from the grid boundary in Section 6, and we sketch the extension of our $O(n^{1/4} \log n)$-approximation algorithm to EDP on wall graphs in Section 7.

2 Preliminaries

We consider the NDP problem in two-dimensional grids: The input is an $(N \times N)$-grid graph $G = (V, E)$, and a collection $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of pairs of vertices, called demand, or source-destination, pairs. The goal is to find a largest cardinality collection $\mathcal{P}$ of paths, where each path in $\mathcal{P}$ connects some demand pair (s_i, t_i), and every vertex of G participates in at most one path in $\mathcal{P}$. The vertices in the set $\{s_1, t_1, \ldots, s_k, t_k\}$ are called *terminals*. By convention, we denote $n = |V|$, so $n = N^2$.

We assume that the grid rows are indexed $R_1, \ldots, R_N$ in the top-to-bottom order, and the columns are indexed $C_1, \ldots, C_N$ in the left-to-right order. We denote by $v(i, j)$ the unique vertex in $R_i \cap C_j$. Given a vertex $v \in V$, let $\text{col}(v)$ denote the column, and $\text{row}(v)$ denote the row in which v lies. The boundary of the grid is $\Gamma(G) = R_1 \cup R_N \cup C_1 \cup C_N$. We call R_1, R_N, C_1, C_N the *boundary edges* of the grid. Given any integers $1 \leq i \leq i' \leq N$, $1 \leq j \leq j' \leq N$, we denote by $G[i : i', j : j']$ the sub-graph of G, induced by the set

$\{v(i'', j'') \mid i \le i'' \le i', j \le j'' \le j'\}$ of vertices. We sometimes say that $G[i : i', j : j']$ is the sub-grid of G, spanned by rows $R_i, \dots, R_{i'}$ and columns $C_j, \dots, C_{j'}$.

Given a path P in G, and a set S of vertices of G, we say that P is *internally disjoint* from S, if no vertex of S serves as an inner vertex of P. We will use the following simple observation.

▶ **Observation 1.** *Let G be a $(h \times w)$-grid, with $w, h > 2$, and let $k \le \min\{w-2, h-2\}$ be an integer. Then for any pair L, L' of opposing boundary edges of G, for any pair $S \subseteq V(L)$, $T \subseteq V(L')$ of vertex subsets on these boundary edges, with $|S| = |T| = k$, there is a set $\mathcal{P}$ of k node-disjoint paths, connecting the vertices of S to the vertices of T in G, such that all paths in $\mathcal{P}$ are internally disjoint from $V(L \cup L')$. Moreover, the path set $\mathcal{P}$ can be found efficiently.*

Proof. Let G' be the sub-graph of G, obtained by deleting all vertices of $(L \cup L') \setminus (S \cup T)$ from G. It is enough to show that there is a set $\mathcal{P}$ of k disjoint paths connecting the vertices of S to the vertices of T in G'.

Assume without loss of generality that L is the top and L' is the bottom boundary edge of G. Assume for contradiction that such a set $\mathcal{P}$ of paths does not exist. Then from Menger's theorem, there is a set Z of at most $k-1$ vertices, such that in $G' \setminus Z$, there is no path from a vertex of $S \setminus Z$ to a vertex of $T \setminus Z$. However, the vertices of S lie on k distinct columns of G, so at least one such column, say C, does not contain a vertex of Z. Similarly, there is some column C' of G that contains a vertex of T, and $V(C') \cap Z = \emptyset$. Finally, since there are at least $k+2$ rows in G, there is some row $R \neq R_1, R_h$, that contains no vertex of Z. Altogether, $(C \cup R \cup C') \cap G'$ lie in the same connected component of $G' \setminus Z$, and this connected component contains a vertex of S and a vertex of T, a contradiction. The set $\mathcal{P}$ of paths can be found efficiently by computing the maximum single-commodity flow between the vertices of S and the vertices of T in G', and using the integrality of flow. ◀

Consider the input grid graph G. The L_∞-distance between two vertices $v(i, j)$ and $v(i', j')$ is defined as $d_\infty(v(i, j), v(i', j')) = \max(|i - i'|, |j - j'|)$. The distance between a set $S \subseteq V(G)$ of vertices and a vertex $v \in V(G)$ is $d_\infty(v, S) = \min_{u \in S}\{d_\infty(v, u)\}$.

Multicommodity Flow LP Relaxation

For each demand pair $(s_i, t_i) \in \mathcal{M}$, let $\mathcal{P}_i$ be the set of all paths connecting s_i to t_i in G, and let $\mathcal{P} = \bigcup_{i=1}^k \mathcal{P}_i$. In order to define the multicommodity flow LP-relaxation of NDP, every path $P \in \mathcal{P}$ is assigned a variable $f(P)$ representing the amount of flow that is sent on P, and for each demand pair (s_i, t_i), we introduce variable x_i, whose value is the total amount of flow sent from s_i to t_i. The LP-relaxation is then defined as follows.

$$\begin{array}{lll}
\text{(LP-flow)} \quad \max & \sum_{i=1}^k x_i & \\
\text{s.t.} & \sum_{P \in \mathcal{P}_i} f(P) = x_i & \forall 1 \le i \le k \\
& \sum_{P : v \in P} f(P) \le 1 & \forall v \in V \\
& f(P) \ge 0 & \forall 1 \le i \le k, \forall P \in \mathcal{P}_i
\end{array}$$

Even though this LP-relaxation has exponentially many variables, it can be efficiently solved by standard techniques, e.g. by using an equivalent polynomial-size edge-based formulation.

It is well known that the integrality gap of (LP-flow) is $\Omega(\sqrt{n})$ even in grid graphs. Indeed, let G be an $(N \times N)$-grid, and let $k = N - 2$. We let the sources $s_1, \dots, s_k$ appear

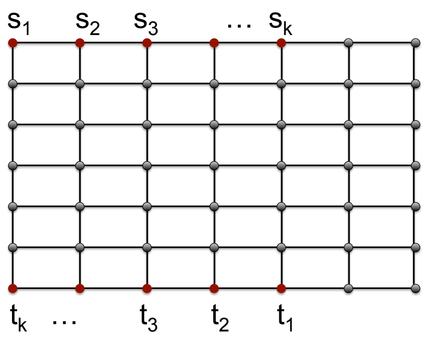


Figure 2 Integrality gap example.

consecutively on row R_1, starting from $v(1,1)$ in this order, and the destinations appear consecutively on row R_N starting from $v(N,1)$, in the opposite order: $t_k,\ldots,t_1$ (see Figure 2). It is easy to see that there is a solution to (LP-flow) of value $k/3=\Omega(N)$: for each pair (s_i,t_i), we send 1/3 flow unit on the path P_i, where P_i is an s_i–t_i path lying in the union of columns C_i, C_{N-i-1} and row R_i+1. On the other hand, it is easy to see that the value of any integral solution is 1, since any pair of paths connecting the demand pairs have to cross. Since the number of vertices in G is $n=N^2$, this gives a lower bound of $\Omega(\sqrt{n})$ on the integrality gap of (LP-flow).

3 Routing with Well-Separated Destinations

In this section we generalize the results of Cutler and Shiloach [16], by proving the following theorem.

▶ **Theorem 2.** *Let H be the $(N\times N)$-grid, and let $\mathcal{M}=\{(s_1,t_1),\ldots,(s_k,t_k)\}$ be a set of $k\geq 4$ demand pairs in H, such that: (i) $s_1,\ldots,s_k$ are all distinct, and they appear on the first row of H; (ii) for all $1\leq i\neq j\leq k$, $d_\infty(t_i,t_j)>4k+4$; and (iii) for all $1\leq i\leq k$, $d_\infty(t_i,V(\Gamma(H)))>4k+4$. Then there is an efficient algorithm that routes all demand pairs in $\mathcal{M}$ in graph H.*

The rest of this section is devoted to proving Theorem 2. For each destination vertex t_j, we define a sub-grid B_j of H of size $((2k+3)\times(2k+3))$, centered at t_j, that is, if $t_j=v(i,i')$, then B_j is a sub-grid of G spanned by rows $R_{i-(k+1)},\ldots,R_{i+(k+1)}$ and columns $C_{i'-(k+1)},\ldots,C_{i'+(k+1)}$ of H.

We call the resulting sub-grids $B_1,\ldots,B_k$ *boxes*. Notice that all boxes are disjoint from each other, due to the spacing of the destination terminals. We start with a high-level intuitive description of our algorithm. For each box B_j, we can associate an interval $I(B_j)\subseteq(1,N)$ with B_j, as follows: If C_{i_1},C_{i_2} are the columns of H containing the first and the last columns of B_j, respectively, then $I(B_j)=(i_1,i_2)$. We say that the resulting set $\mathcal{I}=\{I(B_j)\}_{j=1}^k$ of intervals is aligned, if for all $i\neq j$, either $I(B_i)=I(B_j)$, or $I(B_i)\cap I(B_j)=\emptyset$. For simplicity, assume first that all intervals in $\mathcal{I}$ are aligned, and let $\{I_1,I_2,\ldots,I_r\}$ be the set of all distinct intervals in $\mathcal{I}$, ordered in their natural left-to-right order. For each $1\leq h\leq r$, let $\mathcal{B}_h$ be the set of all boxes B_j with $I(B_j)=I_h$, and let $\mathcal{B}=\{B_j\mid 1\leq j\leq k\}$. We define a "snake-like" ordering of the boxes in $\mathcal{B}$ as follows. For all $1\leq h<h'\leq r$, the boxes of $\mathcal{B}_h$ appear before all boxes of $\mathcal{B}_{h'}$ in this ordering. Within each set $\mathcal{B}_h$, if h is odd, then the boxes of $\mathcal{B}_h$ are ordered in the order of their position in H from top to bottom, and otherwise they are ordered in the order of their position in H from bottom to top. We then define a set $\mathcal{P}$ of

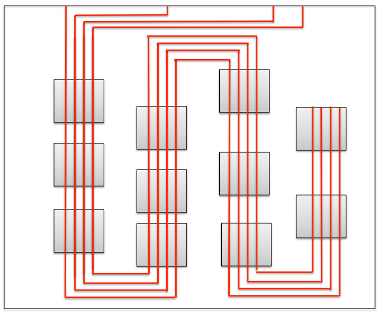


Figure 3 Traversing the boxes.

k paths, that start from the sources $s_1, \dots, s_k$, and visit all boxes in $\mathcal{B}$ in this order (see Figure 3). We will make sure that when the paths of $\mathcal{P}$ traverse any box B_j, the path $P_j \in \mathcal{P}$ that originates at s_j visits the vertex t_j. In order to accomplish this, we need the following lemma.

▶ **Lemma 3.** *Let B be the $((2k+3)\times(2k+3))$ grid, $t = v(k+2, k+2)$ the vertex in the center of the grid, and $1 \le j \le k$ any integer. Let $X = \{x_1, \dots, x_k\}$ be any set of k vertices on the top boundary edge L of B and $Y = \{y_1, \dots, y_k\}$ any set of k vertices on the bottom boundary edge L' of B, both sets ordered from left to right. Then we can efficiently find k disjoint paths $P'_1, \dots, P'_k$ in B, such that for $1 \le i \le k$, path P'_i connects x_i to y_i; all paths are internally disjoint from $V(L \cup L')$; and path P'_j contains t.*

Proof. Let $U = \{u_1, \dots, u_k\}$ be any set of k vertices on row R_{k+2} of B, ordered from left to right, such that $u_j = t$. Let $B' \subseteq B$ be the grid spanned by the top $k+2$ rows of B, and $B'' \subseteq B$ the grid spanned by the bottom $k+2$ rows of B. Note that $B' \cap B'' = R_{k+2}$.

From Observation 1, there is a set $\mathcal{P}_1$ of k node-disjoint paths in B', connecting the vertices of X to the vertices of U, and there is a set $\mathcal{P}_2$ of k node-disjoint paths in B'', connecting the vertices of U to the vertices of Y. Moreover, the paths in $\mathcal{P}_1 \cup \mathcal{P}_2$ are internally disjoint from $V(R_{k+2} \cup L \cup L')$. By concatenating the paths in $\mathcal{P}_1$ and $\mathcal{P}_2$, we obtain a set $\mathcal{P}'$ of k node-disjoint paths in B, connecting the vertices of X to the vertices of Y, such that the paths in $\mathcal{P}'$ are internally disjoint from $L \cup L'$. The intersection of each path in $\mathcal{P}'$ with the row R_{k+2} is exactly one vertex. Since graph B is planar, the paths in $\mathcal{P}'$ cross the row R_{k+2} in the same left-to-right order in which their endpoints appear on L and L'. Therefore, for $1 \le i \le k$, the ith path connects x_i to y_i, and the jth path contains the vertex t. ◀

Since in general the intervals in $\mathcal{I}$ may not be aligned, we need to define the ordering between the boxes, and the set of paths traversing them more carefully. We start by defining an ordering of the destination vertices $\{t_j\}_{j=1}^k$, which will define an ordering of their corresponding boxes.

We draw vertical lines in the grid at every column whose index is an integral multiple of $(3k+2)$, and let $\{V_1, V_2, \dots\}$ denote the sets of vertices of the resulting vertical strips of width $3k+2$, that is, for $1 \le m \le \lceil N/(3k+2) \rceil$,

$$V_m = \{v(j, j') \mid (m-1)(3k+2) < j' \le \min\{m(3k+2), N\}; 1 \le j \le N\}.$$

We assign every terminal t_j to the unique set V_m containing t_j. We then define a collection $\mathcal{S}$ of vertical strips of H as follows: For each set V_m, such that at least one terminal is assigned to V_m, we add $H[V_m]$ to $\mathcal{S}$. We assume that the set of strips $\mathcal{S} = \{S_1, \dots, S_p\}$ is indexed in their natural left-to-right order. Abusing the notation, we will denote $V(S_m)$ by V_m, for $1 \le m \le p$.

Consider some vertical strip S_m, and let $t_i, t_j \in V_m$, for $j \neq i$. Then the horizontal distance between t_i and t_j, $|\operatorname{col}(t_i) - \operatorname{col}(t_j)| \le 3k+2$, and since $d_\infty(t_i, t_j) > 4k+4$, t_i and t_j must be at a vertical distance at least $4k+4$. Therefore, we can order the destination terminals assigned to the same vertical strip in the increasing or decreasing row index. We define the ordering of all destination terminals as follows: (1) for every $1 \le m < m' \le p$, every terminal $t_i \in V_m$ precedes every terminal $t_j \in V_{m'}$; and (2) for $t_i, t_j \in V_m$, with $\operatorname{row}(t_j) > \operatorname{row}(t_i)$, if m is odd then t_i precedes t_j, and if m is even, then t_j precedes t_i. Let $\mathcal{B} = \{B_j \mid 1 \le j \le k\}$ be the set of boxes corresponding to the destination vertices. The ordering of the destination vertices now imposes an ordering on $\mathcal{B}$. We re-index the boxes B_j according to this ordering, and we denote by $t(B_j)$ the unique destination terminal lying in B_j. We will say that a box B_j belongs to strip S_m iff the corresponding terminal $t(B_j) \in V_m$. (Note that B_j is not necessarily contained in S_m). The following observation is immediate.

▶ **Observation 4.** *If box B_j belongs to strip S_m, then at least $k+2$ vertices from the top boundary of B_j, and at least $k+2$ vertices from the bottom boundary of B_j belong to V_m.*

In order to complete the construction of the set $\mathcal{P}$ of paths routing all demand pairs, we define, for $1 \le i \le k$, a set $\mathcal{P}_i$ of k disjoint paths, with the following properties:

P1. Paths in $\mathcal{P}_1$ connect $\{s_i\}_{i=1}^k$ to some set of k vertices on the top boundary of B_1;

P2. For $i > 1$:
- if B_{i-1} and B_i belong to the same strip S_m, and m is odd, then paths in $\mathcal{P}_i$ connect k vertices on the bottom row of B_{i-1} to k vertices on the top row of B_i;
- if B_{i-1} and B_i belong to the same strip S_m, and m is even, then paths in $\mathcal{P}_i$ connect k vertices on the top row of B_{i-1} to k vertices on the bottom row of B_i;
- if B_{i-1} belongs to strip S_m and B_i to strip S_{m+1}, and m is odd, then paths in $\mathcal{P}_i$ connect k vertices on the bottom row of B_{i-1} to k vertices on the bottom row of B_i;
- if B_{i-1} belongs to strip S_m and B_i to strip S_{m+1}, and m is even, then paths in $\mathcal{P}_i$ connect k vertices on the top row of B_{i-1} to k vertices on the top row of B_i; and

P3. All paths in $\bigcup_{i=1}^k \mathcal{P}_i$ are disjoint from each other, and each path is internally disjoint from $\bigcup_{B \in \mathcal{B}} V(B)$.

▶ **Theorem 5.** *There is an efficient algorithm to find the collections $\mathcal{P}_1, \ldots, \mathcal{P}_k$ of paths with properties (3)–(3).*

We prove Theorem 5 below, and we first complete the proof of Theorem 2 here. Assume that we are given the path sets $\mathcal{P}_1, \ldots, \mathcal{P}_k$ with properties (3)–(3). For each box B_j, let $X_j \subseteq V(B_j)$ be the set of k vertices that serve as endpoints of the paths of $\mathcal{P}_j$, and let $Y_j \subseteq V(B_j)$ be the set of k vertices that serve as endpoints of the paths in $\mathcal{P}_{j+1}$. (For $j = k$, we choose the set Y_k of k vertices on the top or the bottom boundary of B_k (opposing the boundary edge where the vertices of X_k lie) arbitrarily). We construct the set $\mathcal{P}$ of paths gradually, by starting with $\mathcal{P} = \mathcal{P}_1$, and performing k iteration. We assume that at the beginning of iteration i, set $\mathcal{P}$ contains k disjoint paths, connecting the k source vertices to the vertices of X_i. This is clearly true at the beginning of the first iteration. The ith iteration is executed as follows. Assume that $t(B_i) = t_r$, and let $u \in X_i$ be the vertex where the path of $\mathcal{P}$ originating at s_r terminates. From Lemma 3, we can find a set $\mathcal{Q}_i$ of paths inside B_i, connecting the vertices of X_i to the vertices of Y_i, that are internally disjoint from the top and the bottom boundary edges of B_i, such that the path originating at u contains

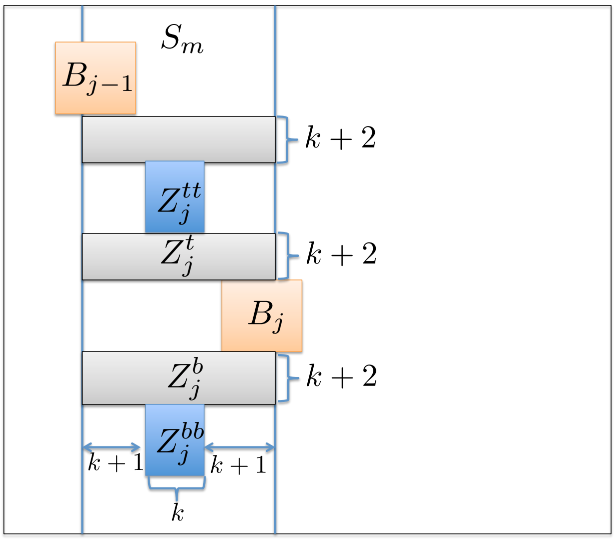


Figure 4 Graphs Z_j^b, Z_j^t, Z_j^{bb} and Z_j^{tt}.

the vertex t_r. We then concatenate the paths in $\mathcal{P}$ with the paths in $\mathcal{Q}_i$, and, if $i < k$, with the paths in $\mathcal{P}_{i+1}$, to obtain the new set $\mathcal{P}$ of paths, and continue to the next iteration. After k iterations, we obtain a collection of k node-disjoint paths that traverse all boxes B_j, such that for each $1 \leq i \leq k$, the path originating from s_i contains the vertex t_i. It now remains to prove Theorem 5.

Proof of Theorem 5. For each box B_j, for $1 \leq j \leq k$, we define four sub-graphs of H, $Z_j^t, Z_j^b, Z_j^{tt}, Z_j^{bb}$, that will be used in order to route the sets $\mathcal{P}_j$, $\mathcal{P}_{j+1}$ of paths.

Consider some box B_j, and assume that it belongs to strip S_m. Let C_ℓ, C_r be the columns of H that serve as the left and the right boundaries of S_m, respectively. Let R_t, R_b be the rows of H containing the top and the bottom row of B_j, respectively. Intuitively, Z_j^t is the sub-grid of strip S_m, containing the $k+1$ rows immediately above row R_t, in addition to the row R_t, and Z_j^b is defined similarly below B_j. Formally, Z_j^t is the sub-grid of H spanned by columns $C_\ell, \ldots, C_r$, and rows $R_{t-k-1}, \ldots, R_t$, so Z_j^t contains $k+2$ rows and $3k+2$ columns. Similarly, Z_j^b is the sub-grid of H spanned by columns $C_\ell, \ldots, C_r$, and rows $R_b, \ldots, R_{b+k+1}$, so Z_j^b contains $k+2$ rows and $3k+2$ columns (see Figure 4).

We now turn to define the grids Z_j^{tt} and Z_j^{bb}. Graph Z_j^{tt} is defined as follows. Assume w.l.o.g. that m is odd (recall that S_m is the strip containing $t(B_j)$). If B_j is not the topmost box that belongs to S_m, then let R_a be the row of H containing the bottom row of Z_{j-1}^b; otherwise let $R_a = R_{2k+1}$ if $j > 1$ and $R_a = R_{k+1}$ if $j = 1$. Let $R_{a'}$ be the row of H containing the top row of Z_j^t. We would like Z_j^{tt} to be the grid containing the segments of the middle k columns of S_m, between rows R_a and $R_{a'}$. Formally, we let Z_j^{tt} be the sub-grid of H spanned by rows $R_a, \ldots, R_{a'}$, and columns $C_{\ell+k+2}, \ldots, C_{\ell+2k+1}$.

We define the graph Z_j^{bb} similarly. If B_j is not the bottommost box of S_m, then let R_c be the row of H containing the top row of Z_{j+1}^t, and otherwise let $R_c = R_{N-k-1}$. Let $R_{c'}$ be the row of H containing the bottom row of Z_j^b. Graph Z_j^{bb} is the sub-grid of H spanned by rows $R_{c'}, \ldots, R_c$, and columns $C_{\ell+k+2}, \ldots, C_{\ell+2k+1}$.

Notice that if B_j is not the topmost box of S_m, then $Z_j^{tt} = Z_{j-1}^{bb}$, and if B_j is not the bottommost box of B_m, then $Z_j^{bb} = Z_{j+1}^{tt}$. We need the following observation.

▶ **Observation 6.** *For all $1 \le q \le k$, $B_q \cap Z_j^{tt}, B_q \cap Z_j^{bb} = \emptyset$. Moreover, if $q \neq j$, then additionally $B_q \cap Z_j^b, B_q \cap Z_j^t = \emptyset$.*

Proof. We prove for Z_j^t and Z_j^{tt}. The proofs for Z_j^b and Z_j^{bb} are symmetric.

Consider some box B_q with $q \neq j$, and assume for contradiction that $B_q \cap Z_j^t \neq \emptyset$. Then the vertical distance between $t(B_q)$ and $t(B_j)$ is less than $4k+4$, and so the horizontal distance between them must be greater than $4k+4$. However, $t(B_j)$ lies in the strip S_m, and, since B_q intersects Z_j^t, the horizontal distance between $t(B_q)$ and the left or the right column of S_m is at most $k+1$, and so the total horizontal distance between $t(B_q)$ and $t(B_j)$ is at most $4k+4$, a contradiction.

Consider now some box B_q, for $1 \le q \le k$, and assume for contradiction that $B_q \cap Z_j^{tt} \neq \emptyset$. If B_j is the topmost box in S_m, then B_q cannot belong to S_m. If B_j is not the topmost box of S_m, then B_q cannot belong to S_m due to the definition of Z_j^{tt}. Therefore, $t(B_q)$ lies in either S_{m+1} or S_{m-1}. But since B_q is a box of width $2k+3$, with $t(B_q)$ lying in $(k+2)$th column of B_q, it is impossible for B_q to intersect Z_j^{tt}. ◀

We are now ready to define the sets $\mathcal{P}_i$ of paths. In order to do so, we define a collection $\{H_1, \ldots, H_k\}$ of disjoint sub-graphs of H, and each such sub-graph H_i will be used to route the set $\mathcal{P}_i$ of paths. We start by letting H_1 be the union of three graphs, Z_1^t, Z_1^{tt}, and the sub-grid of H spanned by the top $k+1$ rows of H. We denote this latter graph by H_1'. Recall that the terminal $t(B_1)$ lies in strip S_1. Let A_1 be the set of k vertices on the top boundary of Z_1^{tt}, A_2 the set of k vertices on the bottom row of Z_1^{tt}, and let A_3 be any set of k vertices on the top row of B_1, that lie in S_1 (from Observation 4, such a set exists). From Observation 1, we can construct three sets of paths: set $\mathcal{P}_1'$ in H_1', connecting each source vertex to some vertex of A_1; set $\mathcal{P}_1''$ in Z_1^{tt} connecting the vertices of A_1 to the vertices of A_2 (the paths in $\mathcal{P}_1''$ are just the columns of Z_1^{tt}), and set $\mathcal{P}_1'''$ in Z_1^t, connecting the vertices of A_2 to the vertices of A_3. We let $\mathcal{P}_1$ be obtained by concatenating the paths in $\mathcal{P}_1', \mathcal{P}_1''$, and $\mathcal{P}_1'''$.

Consider now some index $1 < j \le k$, and assume that B_{j-1} belongs to some strip S_m. We assume w.l.o.g. that m is odd (the case where m is even is dealt with similarly), and we show how to construct the set $\mathcal{P}_j$ of paths. We consider two cases. The first case is when B_j also lies in S_m. We then let H_j be the union of Z_{j-1}^b, Z_{j-1}^{bb} and Z_j^t. The set $\mathcal{P}_j$ of paths will be contained in H_j, and it is defined as follows. Let A_1 be any set of k vertices on the bottom row of B_{j-1}, that lie in V_m (this set exists due to Observation 4); let A_2 and A_3 be the vertices of the top and the bottom rows of Z_{j-1}^{bb}, respectively, and let A_4 be any set of k vertices on the top row of B_j that lie in V_m. As before, using Observation 1, we can construct three sets of paths: set $\mathcal{P}_j'$ in Z_{j-1}^b, connecting each vertex of A_1 to some vertex of A_2; set $\mathcal{P}_j''$ in Z_{j-1}^{bb} connecting the vertices of A_2 to the vertices of A_3 (the paths in $\mathcal{P}_j''$ are just the columns of Z_{j-1}^{bb}), and set $\mathcal{P}_j'''$ in Z_j^t, connecting the vertices of A_3 to the vertices of A_4. We let $\mathcal{P}_j$ be obtained by concatenating the paths in $\mathcal{P}_j', \mathcal{P}_j'$, and $\mathcal{P}_j'''$.

Finally, assume that B_j belongs to S_{m+1}. Let C_ℓ and C_r be the columns of H that serve as the left boundary of S_m and the right boundary of S_{m+1}, respectively. Let H_j' be the sub-grid of H, spanned by columns $C_\ell, \ldots, C_r$, and rows $R_{N-k-1}, \ldots, R_N$. We let H_j be the union of $Z_{j-1}^b, Z_{j-1}^{bb}, H_j', Z_j^b$ and Z_j^{bb}. Using methods similar to those described above, it is easy to find a set $\mathcal{P}_j$ of k disjoint paths in H_j, connecting k vertices on the bottom row of B_{j-1} to k vertices on the bottom row of B_j.

The case where m is even is dealt with similarly. The only difference is that in the case where B_j belongs to S_{m+1}, we use rows $R_{k+2}, \ldots, R_{2k+1}$ to define H_j', instead of rows $R_{N-k+1}, \ldots, R_N$, to avoid collision with the graph H_1'.

From the construction of the graphs H_i, it is easy to see that all such graphs are mutually disjoint, and therefore we obtain the desired sets $\mathcal{P}_1, \ldots, \mathcal{P}_k$ of paths with properties (3)–(3). ◀

4 An $\tilde{O}(n^{1/4})$-Approximation Algorithm

We assume that we are given the $(N \times N)$ grid graph $G = (V, E)$, so $n = |V| = N^2$, and a collection $\mathcal{M} = \{(s_i, t_i)\}_{i=1}^k$ of demand pairs. We say that a demand pair (s_i, t_i) is *bad* if both $d_\infty(s_i, \Gamma(G)), d_\infty(t_i, \Gamma(G)) \leq 4\sqrt{N} + 4$, and we say that it is good otherwise. Let $\mathcal{M}', \mathcal{M}'' \subseteq \mathcal{M}$ denote the sets of the good and the bad demand pairs in $\mathcal{M}$, respectively. We find an approximate solution to each of the two sub-problems, defined by $\mathcal{M}'$ and $\mathcal{M}''$, separately, and take the better of the two solutions. The following two subsections describe these two algorithms.

4.1 Routing the Good Pairs

Our first algorithm provides an $O(n^{1/4} \log n)$-approximation for the special case when all demand pairs are good. We start with a high-level overview of the algorithm. The algorithm is based on LP-rounding of (LP-flow), and so it proves that the integrality gap of (LP-flow) for this special case is $O(n^{1/4} \log n)$. The first step of the algorithm is to reduce the problem to the following special case: We are given a grid A of size $(\Theta(m) \times \Theta(m))$, where $m \leq N/8$ is some integer, and two disjoint sub-grids Q, Q' of A, of size $(m \times m)$ each, such that the minimum L_∞-distance between a vertex in Q and a vertex in Q' is $\Omega(m)$. We are also given a set $\mathcal{M}(Q, Q')$ of demand pairs, where for each pair $(s, t) \in \mathcal{M}(Q, Q')$, $s \in Q$, $t \in Q'$, and $d_\infty(s, \Gamma(Q)) > 4\sqrt{N} + 4$ (where N is the size of the side of our original grid G). We refer to the resulting routing problem as *2-square routing*. We show that an α-approximation algorithm to the 2-square routing problem immediately implies an $O(\alpha \log n)$-approximation to the original problem. We note that a similar reduction to the 2-square routing problem has been used in the past, e.g. in [1]. It is now enough to design an $O(\sqrt{m}) = O(\sqrt{N}) = O(n^{1/4})$-approximation algorithm for the 2-square routing problem. Let OPT' be the optimal solution to this problem, and let $\mathcal{M}^* \subseteq \mathcal{M}(Q, Q')$ be the subset of the demand pairs routed in OPT'. Notice that $|\mathsf{OPT}'| \leq 4m$, since each path in the optimal solution must contain at least one vertex of $\Gamma(Q)$. We define a partition $\mathcal{X}$ of Q into sub-squares of size $(\Theta(\sqrt{m}) \times \Theta(\sqrt{m}))$, and show an efficient algorithm to find a subset $\tilde{\mathcal{M}} \subseteq \mathcal{M}(Q, Q')$ of $\Omega(|\mathsf{OPT}'|/\sqrt{m})$ demand pairs, with $|\tilde{\mathcal{M}}| \leq \sqrt{m}$, so that the following holds. Let S' and T' denote the sets of the source and the destination vertices, participating in the pairs in $\tilde{\mathcal{M}}$, respectively. Then (i) for each square $X \in \mathcal{X}$, $|V(X) \cap S'| \leq 1$; (ii) all vertices in T' can be simultaneously routed to $\Gamma(Q') \setminus \Gamma(G)$ on node-disjoint paths; and (iii) every vertex of A participates in at most one demand pair. Set $\tilde{\mathcal{M}}$ is found by setting up an appropriate instance of the maximum flow problem. It is then easy to route all vertices in T' to $\Gamma(Q)$ on paths that are node-disjoint and internally disjoint from Q. We then use Theorem 2 to complete the routing inside Q. We now turn to describe the algorithm more formally.

Let (f, x) be the optimal solution to the linear program (LP-flow) on instance $(G, \mathcal{M}')$, and let $\mathsf{OPT}_{\mathsf{LP}}$ be its value. We show an algorithm that routes $\Omega(\mathsf{OPT}_{\mathsf{LP}}/(n^{1/4} \cdot \log n))$ demand pairs. The algorithm consists of two steps. In the first step, we reduce the problem to routing between two square sub-grids of G. We note that a similar reduction has been used in prior work, e. g. by Aggarwal et al. [1]. In the second step, we show an approximation algorithm for the resulting sub-problem.

Reduction to the 2-Square Problem

In this step, we reduce the problem of routing on G with a general set $\mathcal{M}'$ of good demand pairs, to a problem where we are given two disjoint sub-grids (or squares) Q_1, Q_2 of G, and every demand pair (s_j, t_j) has $s_j \in Q_1$ and $t_j \in Q_2$, or vice versa.

We start by partitioning the set $\mathcal{M}'$ of the demand pairs into $\lceil \log N \rceil$ subsets, $\mathcal{M}_1, \ldots, \mathcal{M}_{\lceil \log N \rceil}$, where

$$\mathcal{M}_h = \left\{(s_j, t_j) \in \mathcal{M}' \mid 2^{h-1} \le d_\infty(s_j, t_j) < 2^h\right\}.$$

For each $1 \le h \le \lceil \log N \rceil$, let $F_h = \sum_{(s_j,t_j) \in \mathcal{M}_h} x_j$, where x_j is the amount of flow sent from s_j to t_j in the solution to (LP-flow). We let h^* be the index maximizing F_{h^*}, so $F_{h^*} \ge \mathsf{OPT}_{\mathsf{LP}} / \lceil \log N \rceil$. From now on, we focus on routing the pairs in $\mathcal{M}_{h^*}$, and we will route $\Omega(F_{h^*}/n^{1/4})$ such pairs.

Assume first that $h^* \le 6$. In this case, we partition the grid into sub-grids of size at most (256×256) with a random offset, as follows. Select an integer $0 \le z < 256$ uniformly at random, and use the set $\mathcal{C} = \{C_{z+256i}\}_{i=0}^{\lfloor (N-z)/256 \rfloor}$ of columns and the set $\mathcal{R} = \{R_{z+256i}\}_{i=0}^{\lfloor (N-z)/256 \rfloor}$ of rows to partition the grid into sub-grids. Let $\mathcal{Q}$ be the resulting collection of sub-grids. We define a new LP-solution as follows: start with the original LP-solution; for every demand pair $(s_j, t_j) \notin \mathcal{M}_{h^*}$, set $x_j = 0$, and $f(P) = 0$ for all paths $P \in \mathcal{P}_j$. For every demand pair $(s_j, t_j) \in \mathcal{M}_{h^*}$, if s_j or t_j lie on a row of $\mathcal{R}$ or a column of $\mathcal{C}$, or if they belong to different sub-grids in $\mathcal{Q}$, set $x_j = 0$ and $f(P) = 0$ for all paths $P \in \mathcal{P}_j$. Since for each pair $(s_j, t_j) \in \mathcal{M}_{h^*}$, $d_\infty(s_j, t_j) < 64$, it is easy to see that the expected value of the resulting LP-solution is $W = \Omega(F_{h^*}) = \Omega(\mathsf{OPT}_{\mathsf{LP}}/\log N) = \Omega(\mathsf{OPT}_{\mathsf{LP}}/\log n)$. By trying all possible values $0 \le z < 256$, we can find a partition $\mathcal{Q}$ of G, and a corresponding LP-solution, whose value is at least W. Notice that for each sub-grid $Q \in \mathcal{Q}$, the number of vertices of Q is bounded by 256^2, and so the total amount of flow routed between the demand pairs contained in Q is bounded by 256^2. For each sub-grid $Q \in \mathcal{Q}$, if there is any demand pair $(s_j, t_j) \in \mathcal{M}_{h^*}$ with $s_j, t_j \in Q$, and a non-zero value x_j in the current LP-solution, we select any such pair and route it via any path P contained in Q, which is disjoint from the boundary of Q. It is easy to see that the total number of the demand pairs routed is $\Omega(W) = \Omega(\mathsf{OPT}_{\mathsf{LP}}/\log n)$. From now on, we assume that $h^* > 6$.

For convenience, we denote h^* by h from now on. Let $m = 2^h/16$. We partition the grid into a collection $\mathcal{Q} = \{Q_{p,q} \mid 1 \le p \le \lfloor N/m \rfloor, 1 \le q \le \lfloor N/m \rfloor\}$ of disjoint sub-grids, or squares, as follows. First, partition G into $\lfloor N/m \rfloor$ disjoint vertical strips $V_1, \ldots, V_{\lfloor N/m \rfloor}$, each containing m consecutive columns of G, except for the last strip, that may contain between m and $2m-1$ columns. Next, partition each vertical strip V_p into $\lfloor N/m \rfloor$ disjoint sub-grids, where each sub-grid contains m consecutive rows of V_p, except possibly for the last sub-grid, that may contain between m and $2m-1$ rows. The width and the hight of each such sub-grid is then between m and $2m-1$, where $m \le N/16$. Notice that for each such grid $Q_{p,q} \in \mathcal{Q}$, if L is the left boundary edge of $Q_{p,q}$, and L' is the left boundary edge of G, then either $L \subseteq L'$, or L and L' are separated by at least $m-1$ columns. The same holds for the other three boundary edges. We need the following observation.

▶ **Observation 7.** *Let $(s_j, t_j) \in \mathcal{M}_h$ be a demand pair, and assume that $s_j \in Q_{p,q}$ and $t_j \in Q_{p',q'}$. Then:*

$$5 \le |p - p'| + |q - q'| \le 34.$$

Proof. We first show that $|p - p'| + |q - q'| \ge 5$. Indeed, assume otherwise. Then both the horizontal and the vertical distances between s_j and t_j are less than $8m = 8 \cdot 2^h/16 = 2^{h-1}$, while $d_\infty(s_j, t_j) \ge 2^{h-1}$, a contradiction.

Assume now for contradiction that $|p-p'|+|q-q'|>34$. Then $d_\infty(s_j,t_j)>16m=2^h$, contradicting the fact that $d_\infty(s_j,t_j)<2^h$. ◀

We say that a pair $(Q_{p,q},Q_{p',q'})$ of squares in $\mathcal{Q}$ is *interesting* iff $5\leq|p-p'|+|q-q'|\leq 34$. Let $\mathcal{Z}$ be the set of all interesting pairs of squares in $\mathcal{Q}$. We associate an NDP instance with each such pair $Z=(Q_{p,q},Q_{p',q'})$, as follows. Let $\mathcal{M}(Z)\subseteq\mathcal{M}_h$ be the set of all demand pairs $(s_j,t_j)\in\mathcal{M}_h$ where $s_j\in Q_{p,q}$ and $t_j\in Q_{p',q'}$, or vice versa. We also define a box $A(Z)$, that contains $Q_{p,q}\cup Q_{p',q'}$, and adds a margin of m around them, if possible. More precisely, let ℓ be the smallest integer, such that $R_\ell\cap(Q_{p,q}\cup Q_{p',q'})\neq\emptyset$, and let ℓ' be the largest integer, such that $R_{\ell'}\cap(Q_{p,q}\cup Q_{p',q'})\neq\emptyset$. Similarly, let b and b' be the smallest and the largest integers, respectively, such that $C_b\cap(Q_{p,q}\cup Q_{p',q'}),C_{b'}\cap(Q_{p,q}\cup Q_{p',q'})\neq\emptyset$. We then let $A(Z)$ be the sub-grid of G spanned by rows $R_{\max\{1,\ell-m\}},\ldots,R_{\min\{\ell'+m,N\}}$, and by columns $C_{\max\{1,b-m\}},\ldots,C_{\min\{b'+m,N\}}$. For every interesting pair of squares $Z\in\mathcal{Z}$, we now define an instance of the NDP problem on graph $A(Z)$, with the set $\mathcal{M}(Z)$ of demand pairs. Let $F(Z)$ be the total amount of flow routed between the demand pairs in $\mathcal{M}(Z)$ in the current LP-solution F_h to our original problem (notice that in our LP-solution, the fractional routing of the demand pairs in $\mathcal{M}(Z)$ is not necessarily contained in $A(Z)$). From the above discussion, $\sum_{Z\in\mathcal{Z}}F(Z)=\Omega(\mathsf{OPT}_{\mathsf{LP}}/\log N)$. We will show an algorithm that routes, for each $Z\in\mathcal{Z}$, $\Omega(F(Z)/n^{1/4})$ demand pairs in $\mathcal{M}(Z)$ integrally, in graph $A(Z)$. However, it is possible that for two pairs $Z,Z'\in\mathcal{Z}$, $A(Z)\cap A(Z')\neq\emptyset$, and the two routings may interfere with each other. We resolve this problem in the following step.

From Observation 7, it is easy to see that for each interesting pair of squares $Z\in\mathcal{Z}$, the number of pairs $Z'\in\mathcal{Z}$ with $A(Z)\cap A(Z')\neq\emptyset$ is bounded by some constant c. We construct a graph H, whose vertex set is $V(H)=\{v_Z\mid Z\in\mathcal{Z}\}$, and there is an edge $(v_Z,v_{Z'})$ iff $A(Z)\cap A(Z')\neq\emptyset$. As observed above, the maximum vertex degree in this graph is bounded by some constant c, and so we can color H with $c+1$ colors. Let $U_i\subseteq V(H)$ be the set of vertices of color i. We select a color class i^*, maximizing the value $F^{i^*}=\sum_{v_Z\in U_{i^*}}F(Z)$. Clearly, $F^{i^*}=\Omega(\mathsf{OPT}_{LP}/\log N)$. For every pair $v_Z,v_{Z'}$ of vertices in U_{i^*}, we now have $A(Z)\cap A(Z')=\emptyset$. In order to obtain an $O(n^{1/4}\log n)$-approximation algorithm for the special case where all demand pairs are good, it is now enough to prove the following theorem.

▶ **Theorem 8.** *There is an efficient algorithm, that, for every interesting pair $Z\in\mathcal{Z}$ of squares, routes $\Omega(F(Z)/n^{1/4})$ demand pairs of $\mathcal{M}(Z)$ inside the grid $A(Z)$.*

The Rounding Algorithm

From now on we focus on proving Theorem 8. We assume that we are given an interesting pair $Z=(Q,Q')$ of squares, where the width and the height of each square is bounded by $2m-1$. We are also given a collection $\mathcal{M}(Z)$ of demand pairs, that, for convenience, we denote by $\mathcal{M}$ from now on. For each demand pair $(s_j,t_j)\in\mathcal{M}$, we can assume without loss of generality that $s_j\in Q$ and $t_j\in Q'$. Recall that we have a fractional solution (f,x) that routes $F^*=F(Z)$ flow units between the demand pairs in $\mathcal{M}$, in the grid G. Additionally, we are given a square $A=A(Z)$, containing Q and Q', as defined above. Recall that for any pair $v\in Q$, $v'\in Q'$ of vertices, $d_\infty(v,v')\geq 5m$.

From our definition of good demand pairs, it is possible that for a pair $(s_j,t_j)\in\mathcal{M}$, $d_\infty(s_j,\Gamma(G))\leq 4\sqrt{N}+4$, or $d_\infty(t_j,\Gamma(G))\leq 4\sqrt{N}+4$, but not both. We say that (s_j,t_j) is a type-1 pair if $d_\infty(s_j,\Gamma(G))\leq 4\sqrt{N}+4$, and we say that it is a type-2 demand pair otherwise. Let F_1 be the total flow in the LP-solution between the type-1 demand pairs, and F_2 the total flow between type-2 demand pairs. We assume without loss of generality that

$F_1 \leq F_2$, so $F_2 \geq F^*/2$. From now on we focus on routing type-2 demand pairs. Abusing the notation, we use $\mathcal{M}$ to denote the set of all type-2 demand pairs.

We next define a sub-grid Q^+ of A, obtained by adding a margin of m around the grid Q, if possible. Specifically, let $R_\ell, R_{\ell'}$ be the rows of G, containing the top and the bottom rows of Q, respectively. Similarly, let $C_b, C_{b'}$ be the columns of G, containing the left and the right columns of Q, respectively. We let Q^+ be the sub-grid of G, spanned by rows $R_{\max\{1,\ell-m\}}, \ldots, R_{\min\{N,\ell'+m\}}$ and columns $C_{\max\{1,b-m\}}, \ldots, C_{\min\{N,b'+m\}}$. From our definition of A, $Q^+ \subseteq A$. Moreover, since $m \leq N$, and since we have assumed that all demand pairs are type-2 good pairs, all source vertices corresponding to the demand pairs in $\mathcal{M}$ are within L_∞ distance at least $4\sqrt{m}+5$ from the boundary of Q^+. We start with the following simple observation.

▶ **Observation 9.** *Let L' be a boundary edge of Q', such that $L' \not\subseteq \Gamma(G)$, and let $Y \subseteq V(L')$ be any set of its vertices. Then there is a boundary edge L of Q^+, and a set $\mathcal{P}$ of $|Y|$ disjoint paths in graph A, connecting every vertex of Y to a distinct vertex of L, such that the paths in $\mathcal{P}$ are internally disjoint from $Q^+ \cup Q'$.*

Proof. If the top boundary edge $\tilde{L}$ of Q^+ is separated by at least m rows from the top boundary edge of G, then set $L = \tilde{L}$; otherwise, let L be the bottom boundary edge of Q^+ - notice that it must be separated by at least m rows from the bottom boundary edge of G. Let $X \subseteq V(L)$ be any set of $|Y|$ vertices, and let A' be the graph obtained from A, by deleting all vertices in $Q^+ \setminus X$ and $Q' \setminus Y$ from it. It is enough to show that there is a set $\mathcal{P}$ of $|X| = |Y|$ disjoint paths in A', connecting the vertices of X to the vertices of Y. Let $z = |X|$. From Menger's theorem, if such a set of paths does not exist, then there is a set J of at most $z-1$ vertices, such that in $A' \setminus J$ there is no path from a vertex of $X \setminus J$ to a vertex of $Y \setminus J$. But from our definition of Q^+, Q', and A, it is clear that no such set of vertices exists. ◀

Let r be the smallest integral power of 2 greater than $4\sqrt{m}+4$, so $r = \Theta(\sqrt{m})$. Our next step is to partition Q into a collection $\mathcal{X}$ of disjoint sub-grids of size $(r \times r)$ each. For $1 \leq p,q \leq m/r$, we let $X_{p,q}$ be the sub-grid of Q, spanned by rows $R_{(p-1)r+1}, \ldots, R_{pr}$ and columns $C_{(q-1)r+1}, \ldots, C_{qr}$ of Q. We then let $\mathcal{X} = \{X_{p,q} \mid 1 \leq p,q \leq m/r\}$. The next theorem is key to finding the final routing.

▶ **Theorem 10.** *There is a subset $\mathcal{M}_1 \subseteq \mathcal{M}$ of $\Omega(F^*/n^{1/4})$ demand pairs, such that every vertex of $Q \cup Q'$ participates in at most one demand pair. Moreover, if S_1 and T_1 denote the sets of all source and all destination vertices of the pairs in $\mathcal{M}_1$, respectively, then:*

- *for every square $X_{p,q} \in \mathcal{X}$, at most one vertex of $X_{p,q}$ belongs to S_1; and*
- *there is a boundary edge L' of Q', with $L' \not\subseteq \Gamma(G)$, and a set $\mathcal{P}_1$ of node-disjoint paths in graph Q', connecting every vertex of T_1 to a distinct vertex of L'.*

Proof. Let U be the union of the boundary edges L' of Q', with $L' \not\subseteq \Gamma(G)$. We build a flow network $\mathcal{N}$, starting with the graph Q'. We add a source vertex a, that connects to every vertex in U with a directed edge. Let $S \subseteq Q$ be the set of all vertices participating in the demand pairs in $\mathcal{M}$ as sources. Observe that each vertex $s \in S$ may participate in several demand pairs in $\mathcal{M}$. We add every vertex $s \in S$ to graph $\mathcal{N}$, and for each demand pair $(s,t) \in \mathcal{M}$, we connect t to s with a directed edge. Next, for each square $X_{p,q} \in X$, we add a vertex $u_{p,q}$, and we connect every vertex $s \in S \cap X_{p,q}$ to $u_{p,q}$ with a directed edge. Finally, we add a destination vertex b, and connect every vertex $u_{p,q}$ for $1 \leq p,q \leq m/r$ to b with a directed edge. We set all vertex-capacities (except for those of a and b) to 1.

We claim that there is a valid flow of value $\Omega(F^*/\sqrt{m})$ from a to b in $\mathcal{N}$. Indeed, consider the multicommodity flow between the demand pairs in $\mathcal{M}$, given by our current LP-solution. For each (s_j, t_j)-pair in $\mathcal{M}$, we send $x_j/4r$ flow units on the edge (t_j, s_j) in $\mathcal{N}$. For each flow-path $P \in \mathcal{P}_j$, notice that P must contain some vertex of U. Let v be the last such vertex on P (where we view P as directed from s_j to t_j), and let P' be the sub-path of P from v to t_j. We send $f(P)/4r$ flow units on every edge in P'. For every vertex $v \in U$, we set the flow on the edge (a, v) to be the total flow leaving the vertex v; for each vertex $s \in S$, with $s \in X_{p,q}$, we set the flow on the edge $(s, u_{p,q})$ to be the total amount of flow entering s. The flow on edge $(u_{p,q}, b)$ is then set to the total amount of flow entering $u_{p,q}$. Notice that for each square $X_{p,q}$, every flow-path originating at a vertex of $S \cap X_{p,q}$ must cross the boundary $\Gamma(X_{p,q})$ of $X_{p,q}$, that contains at most $4r$ vertices. Therefore, the total amount of flow in the original LP-solution leaving the vertices in $S \cap X_{p,q}$ is at most $4r$. It is now easy to see that we have defined a valid a-b flow of value $\tilde{F} = \Omega(F^*/\sqrt{m})$.

From the integrality of flow, there is an integral flow of the same value in $\mathcal{N}$. Let $\mathcal{P}$ be the set of paths carrying one flow unit in the resulting flow. Then there is a boundary edge L' of Q', such that $L' \not\subseteq \Gamma(G)$, with at least $\tilde{F}/4$ of the paths in $\mathcal{P}$ containing a vertex of L'. Let $\mathcal{P}' \subseteq \mathcal{P}$ be this set of paths. We are now ready to define the final set $\mathcal{M}_1$ of the demand pairs, and the corresponding set $\mathcal{P}_1$ of paths. Consider some path $P \in \mathcal{P}'$, and let (t, s) be the unique edge with $(s, t) \in \mathcal{M}$ on this path. We then add (s, t) to $\mathcal{M}_1$. Let P' be the sub-path of P, starting from the last vertex on P that belongs to L', to vertex t. We add P' to $\mathcal{P}_1$. This finishes the definition of the subset $\mathcal{M}_1$ of demand pairs, and the corresponding set $\mathcal{P}_1$ of paths. ◀

If $|\mathcal{M}_1| > \sqrt{m}$, then we discard pairs from $\mathcal{M}_1$, until $|\mathcal{M}_1| \le \sqrt{m}$ holds, and we update the sets S_1, T_1, and $\mathcal{P}_1$ accordingly.

For $w, w' \in \{0, 1\}$, let $S_{w,w'}$ be a subset containing all vertices $s \in S_1$ lying in the squares $X_{p,q}$, where $p = w \mod 2$ and $q = w' \mod 2$. Then there is some choice of $w, w' \in \{0, 1\}$, so that $|S_{w,w'}| \ge |S_1|/4$. We let $S_2 = S_{w,w'}$ for this choice of w, w', and we define $\mathcal{M}_2 = \{(s, t) \in \mathcal{M}_1 \mid s \in S_2\}$, and T_2 as the set of all destination vertices for the pairs in $\mathcal{M}_2$. Let $\mathcal{P}_2 \subseteq \mathcal{P}_1$ be the set of paths originating from the vertices of T_2. Let Y be the set of endpoints of the paths in $\mathcal{P}_2$ that lie on the boundary edge L' of Q'. Finally, from Observation 9, there is a boundary edge L of Q^+, a set Y' of $|Y|$ vertices of L, and a set $\mathcal{P}'_2$ of disjoint paths in A, connecting every vertex in Y to a distinct vertex of Y', so that the paths in $\mathcal{P}'_2$ are internally disjoint from $Q^+ \cup Q'$. By concatenating the paths in $\mathcal{P}_2$ and $\mathcal{P}'_2$, we obtain a new set $\mathcal{P}^*$ of paths, connecting every vertex of T_2 to a distinct vertex of Y'. Denote $\mathcal{M}_2 = \{(s_j, t_j)\}_{j=1}^{|\mathcal{M}_2|}$, and let $u_j \in Y'$ be the vertex where the path $P_j \in \mathcal{P}^*$, originating at vertex t_j, terminates. Notice that all vertices in S_2 are now at the L_∞-distance at least $r > 4\sqrt{m} + 4$ from each other, and at distance at least $4\sqrt{m} + 5$ from the boundaries of Q^+, and $|\mathcal{M}_1| \le \sqrt{m}$. From Theorem 2, we can efficiently find a set $\mathcal{Y}$ of disjoint paths in graph Q^+, connecting every vertex $s_j \in S_2$ to the corresponding vertex $u_j \in Y'$. By concatenating the paths in $\mathcal{P}^*$ and $\mathcal{Y}$, we obtain a set of paths routing all pairs in $\mathcal{M}_2$.

Notice that from the above discussion, $|\mathcal{M}_2| = \min\{\Omega(\sqrt{m}), \Omega(F^*/\sqrt{m})\}$. It is easy to see that $F^* \le 4m$, since every flow-path routing a pair in $\mathcal{M}$ must cross the boundary of Q'. Therefore, $|\mathcal{M}_2| = \Omega(F^*/\sqrt{m})$. Since $m \le N = \sqrt{n}$, our algorithm routes $\Omega(F^*/n^{1/4})$ demand pairs.

4.2 Routing the Bad Pairs

The goal of this section is to prove the following theorem.

▶ **Theorem 11.** *Let $(G, \mathcal{M})$ be an instance of the NDP problem, where G is an $(N \times N)$ grid, and $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$. Assume further that for each demand pair (s_j, t_j), both $d_\infty(s_j, \Gamma(G)), d_\infty(t_j, \Gamma(G)) < d^*$, for some parameter $1 \leq d^* \leq N/4$. Then there is an efficient algorithm that finds an $O(d^*)$-approximate solution to the NDP instance $(G, \mathcal{M})$.*

Notice that by setting $d^* = 4\sqrt{N} + 5$, so that $d^* = \Theta(n^{1/4})$, we obtain an $O(n^{1/4})$-approximate solution for NDP instances on grid graphs, where all demand pairs are bad.

The rest of this section is dedicated to proving Theorem 11. Let T be the set of all vertices participating in the bad demand pairs. We call the vertices in T *terminals*. Let L_1, L_2, L_3, L_4 be the four boundary edges of the grid G. Notice that a terminal $t \in T$ may be within distance d^* from up to two boundary edges. For each terminal $t \in T$, we let $L(t)$ be any boundary edge of G, such that $d_\infty(t, V(L(t))) < d^*$. We now partition all bad demand pairs into 16 subsets: for $1 \leq p, q \leq 4$, set $\mathcal{M}_{p,q}$ contains all pairs (s_j, t_j), where $L(s_j) = L_p$ and $L(t_j) = L_q$. Let OPT be the optimal solution to the NDP instance. For every possible choice of $1 \leq p, q \leq 4$, let $\mathsf{OPT}_{p,q}$ be the optimal solution restricted to the pairs in $\mathcal{M}_{p,q}$. Clearly, there is a choice of p and q, such that at least $|\mathsf{OPT}|/16$ of the demand pairs routed in OPT belong to $\mathcal{M}_{p,q}$, and so $|\mathsf{OPT}_{p,q}| \geq \mathsf{OPT}/16$. For each choice of values $1 \leq p, q \leq 4$, we show an algorithm that routes $\Omega(|\mathsf{OPT}_{p,q}|/d^*)$ demand pairs in $\mathcal{M}_{p,q}$. We then take the best of these solutions, thus obtaining an $O(d^*)$-approximation algorithm.

Fix some $1 \leq p, q \leq 4$. We consider three cases.

The first case happens when L_p and L_q are two distinct opposing boundary edges of G. We assume without loss of generality that L_p is the top, and L_q is the bottom boundary of G. We say that a subset $\mathcal{M}' \subseteq \mathcal{M}_{p,q}$ of demand pairs is a *monotone matching*, if the following holds. Let S' be the set of all source vertices, and T' the set of all destination vertices, participating in the pairs in $\mathcal{M}'$. Then:

- All vertices of S' lie in distinct columns of G;
- All vertices of T' lie in distinct columns of G;
- Every vertex of $S' \cup T'$ participates in exactly one demand pair; and
- For any two distinct pairs $(s_i, t_i), (s_j, t_j) \in \mathcal{M}'$, $\mathrm{col}(s_i) < \mathrm{col}(s_j)$ iff $\mathrm{col}(t_i) < \mathrm{col}(t_j)$.

The following observation is immediate.

▶ **Observation 12.** *Let $\mathcal{M}' \subseteq \mathcal{M}_{p,q}$ be any monotone matching with $|\mathcal{M}'| \leq N/2$. Then there is an efficient algorithm to route all pairs in $\mathcal{M}'$ in graph G.*

Our algorithm then simply computes the largest monotone matching $\mathcal{M}' \subseteq \mathcal{M}_{p,q}$, using standard dynamic programming: We maintain a dynamic programming table Π, that contains, for all $0 \leq x, y \leq N$, an entry $\Pi(x, y)$, whose value is the size of the largest monotone matching $\mathcal{M}(x, y) \subseteq \mathcal{M}_{p,q}$, such that every source vertex s participating in pairs in $\mathcal{M}(x, y)$ has $1 \leq \mathrm{col}(s) \leq x$, and every destination vertex t participating in pairs in $\mathcal{M}(x, y)$ has $1 \leq \mathrm{col}(t) \leq y$. We fill the entries of the table from smaller to larger values of $x + y$, initializing $\Pi(x, 0) = 0$ and $\Pi(0, y) = 0$ for all x and y. Entry $\Pi(x, y)$ is computed as follows. If there is a pair $(s, t) \in \mathcal{M}_{p,q}$, with $\mathrm{col}(s) = x$ and $\mathrm{col}(t) = y$, then we let $\Pi(x, y)$ be the maximum of $\Pi(x-1, y-1) + 1$, $\Pi(x-1, y)$, and $\Pi(x, y-1)$. Otherwise, $\Pi(x, y)$ is the maximum of $\Pi(x-1, y)$, and $\Pi(x, y-1)$. The size of the largest monotone matching $\mathcal{M}' \subseteq \mathcal{M}_{p,q}$ is then stored in $\Pi(N, N)$, and we can use standard techniques to compute the matching itself. Finally, we show that there is a large enough monotone matching $\mathcal{M}' \subseteq \mathcal{M}_{p,q}$.

▶ **Lemma 13.** *There is a monotone matching $\mathcal{M}' \subseteq \mathcal{M}_{p,q}$ of cardinality $\Omega(\mathsf{OPT}_{p,q}/d^*)$.*

Proof. For every source vertex s of a demand pair in $\mathcal{M}_{p,q}$, let $P(s)$ denote the segment of the column in which s lies, from the first row of G to s itself. Similarly, for each destination vertex t of a demand pair in $\mathcal{M}_{p,q}$, let $P(t)$ denote the segment of the column in which t lies, from t to the last row of G.

Consider the solution $\mathsf{OPT}_{p,q}$, and let $\mathcal{M}^* \subseteq \mathcal{M}_{p,q}$ be the set of the demand pairs routed in it. For each pair $(s_i, t_i) \in \mathcal{M}^*$, let $P_i \in \mathsf{OPT}_{p,q}$ be the path routing this demand pair in the solution. We say that two demand pairs (s_i, t_i) and (s_j, t_j) in $\mathcal{M}^*$ *have a conflict* iff either P_i contains a vertex of $P(s_j) \cup P(t_j)$, or P_j contains a vertex of $P(s_i) \cup P(t_i)$.

Let H be a directed graph, that contains a vertex v_i for every pair $(s_i, t_i) \in \mathcal{M}^*$, and a directed edge (v_i, v_j) iff path P_i intersects $P(s_j)$ or $P(t_j)$. Notice that the length of every path $P(s_j)$ or $P(t_j)$ is bounded by d^*, and so every vertex of H has in-degree bounded by $2d^*$. Therefore, any vertex-induced sub-graph H' of H with z vertices has at most $2d^*z$ edges, and contains at least one vertex whose degree (including the incoming and the outgoing edges) is at most $4d^*$.

We now construct the set $\mathcal{M}'$ of demand pairs as follows. Start with $\mathcal{M}' = \emptyset$. While H is non-empty, let v_i be any vertex of degree at most $4d^*$. Delete v_i and all its neighbors from H, and add the pair (s_i, t_i) to $\mathcal{M}'$. When this procedure terminates, it is easy to see that $\mathcal{M}'$ contains at least $|\mathsf{OPT}_{p,q}|/(4d^* + 1) = \Omega(|\mathsf{OPT}_{p,q}|/d^*)$ demand pairs. Moreover, if (s_i, t_i) and (s_j, t_j) are distinct pairs in $\mathcal{M}'$, then there is no conflict between (s_i, t_i) and (s_j, t_j). In particular, this means that $\mathrm{col}(s_i) \neq \mathrm{col}(s_j)$ and $\mathrm{col}(t_i) \neq \mathrm{col}(t_j)$. Moreover, if we assume that $\mathrm{col}(s_i) < \mathrm{col}(s_j)$, then $\mathrm{col}(t_i) < \mathrm{col}(t_j)$ must hold: this is since the union of $P_i, P(s_i)$ and $P(t_i)$ partitions the face defined by $\Gamma(G)$ into a number of sub-faces, and both s_j and t_j must be contained in a single sub-face, as the path P_j cannot intersect the paths $P_i, P(s_i)$ and $P(t_i)$. ◀

This concludes the analysis of the algorithm for the case where L_p and L_q are two distinct opposing boundary edges of G. The case where L_p and L_q are two adjacent boundary edges of G is dealt with very similarly. Finally, we consider the case where $L_p = L_q$. Assume without loss of generality that L_p is the bottom boundary edge of the grid. We say that a subset $\mathcal{M}' \subseteq \mathcal{M}_{p,q}$ is a *nested matching*, if the following holds. Let S' be the set of all source vertices, and T' the set of all destination vertices, participating in the pairs in $\mathcal{M}'$. Then:

- All vertices of S' lie in distinct columns of G;
- All vertices of T' lie in distinct columns of G;
- Every vertex of $S' \cup T'$ participates in exactly one demand pair; and
- For any two distinct pairs $(s_i, t_i), (s_j, t_j) \in \mathcal{M}'$, with $\mathrm{col}(s_i)$ lying to the left of $\mathrm{col}(s_j)$, either both $\mathrm{col}(s_i), \mathrm{col}(t_i)$ lie to the left of both $\mathrm{col}(s_j), \mathrm{col}(t_j)$, or both $\mathrm{col}(s_j), \mathrm{col}(t_j)$ lie between $\mathrm{col}(s_i)$ and $\mathrm{col}(t_i)$, or both $\mathrm{col}(s_i), \mathrm{col}(t_i)$ lie between $\mathrm{col}(t_j)$ and $\mathrm{col}(s_j)$.

It is immediate to see that any nested matching $\mathcal{M}' \subseteq \mathcal{M}_{p,q}$, with $|\mathcal{M}'| \leq N/2$ can be routed efficiently in G. As before, we can find a largest-cardinality nested matching $\mathcal{M}' \subseteq \mathcal{M}_{p,q}$ using standard dynamic programming techniques. The following lemma will then finish the proof.

▶ **Lemma 14.** *There is a nested matching $\mathcal{M}' \subseteq \mathcal{M}_{p,q}$ of cardinality $\Omega(\mathsf{OPT}_{p,q}/d^*)$.*

Proof. We construct the paths $P(s), P(t)$, the graph H', and the matching $\mathcal{M}'$ corresponding to an independent set in H' exactly as in the proof of Lemma 13. As before, $|\mathcal{M}'| = \Omega(\mathsf{OPT}_{p,q}/d^*)$. Moreover, if (s_i, t_i) and (s_j, t_j) are distinct pairs in $\mathcal{M}'$, then there is no conflict between (s_i, t_i) and (s_j, t_j). As before, this means that $\mathrm{col}(s_i) \neq \mathrm{col}(s_j)$ and $\mathrm{col}(t_i) \neq \mathrm{col}(t_j)$. Assume now that $\mathrm{col}(s_i)$ lies to the left of $\mathrm{col}(s_j)$. Then the union of

$P_i, P(s_i)$ and $P(t_i)$ partitions the face defined by $\Gamma(G)$ into a number of sub-faces, and both s_j and t_j must be contained in a single sub-face, as before. In this case, this means that either both $\mathrm{col}(s_i), \mathrm{col}(t_i)$ lie to the left of both $\mathrm{col}(s_j), \mathrm{col}(t_j)$, or both $\mathrm{col}(s_j), \mathrm{col}(t_j)$ lie between $\mathrm{col}(s_i)$ and $\mathrm{col}(t_i)$, or both $\mathrm{col}(s_i), \mathrm{col}(t_i)$ lie between $\mathrm{col}(t_j)$ and $\mathrm{col}(s_j)$. ◀

4.3 Putting Everything Together

Our algorithm for an input NDP instance $(G, \mathcal{M})$, where G is an $(N \times N)$ grid, applies the algorithm from Section 4.1 to the set $\mathcal{M}'$ of the good demand pairs, and the algorithm from Section 4.2 to the set $\mathcal{M}''$ of the bad demand pairs, and returns the better of the two solutions. Since each of the two algorithms achieves an $O(n^{1/4} \log n)$-approximation to the corresponding problem, and since at least half of the demand pairs routed in the optimal solution are either all good pairs, or all bad pairs, we obtain an $O(n^{1/4} \log n)$-approximation overall.

5 APX-Hardness Proof

In this section we prove that NDP does not have a $(1+\delta)$-approximation algorithm on grid graphs, for some fixed $\delta > 0$, unless $\mathsf{P} = \mathsf{NP}$. We perform a reduction from the 3SAT(5) problem. In this problem we are given a 3SAT formula φ on n variables and $5n/3$ clauses. Each clause contains exactly 3 distinct literals and each variable participates in exactly 5 different clauses. We say that φ is a Yes-Instance if it is satisfiable. We say that φ is a No-Instance with respect to some parameter ϵ, if no assignment satisfies more than an ϵ-fraction of clauses. The following well-known theorem follows from the PCP theorem [7, 6].

▶ **Theorem 15.** *There is a constant $\epsilon : 0 < \epsilon < 1$, such that it is NP-hard to distinguish between* Yes-Instances *and* No-Instances *(defined with respect to ϵ) of the 3SAT(5) problem.*

Let φ be the input 3SAT(5) formula, defined over the set $\{x_1, \ldots, x_n\}$ of variables, and a set $C_1, \ldots, C_m$ of clauses, where $m = 5n/3$. Our graph G is the $(N \times N)$ grid, where $N = (m+1)(4m+6)$. The set $\mathcal{M}$ of demand pairs consists of three subsets: set $\mathcal{M}_1$ representing the variables of φ, set $\mathcal{M}_2$ representing the clauses, and set $\mathcal{M}_3$ of additional auxiliary pairs. We now define each set of the demand pairs in turn.

Let $I_1, \ldots, I_n$ be any set of mutually disjoint sub-paths of the top row R_1 of the grid, each containing exactly 13 vertices of R_1. For $1 \le j \le n$, let s_j be the vertex lying exactly in the middle of I_j, so s_j is the 7th vertex of I_j from the left. Let t_j and t'_j be the first and the last vertices of I_j, respectively. We then define:

$$\mathcal{M}_1 = \left\{(s_j, t_j), (s_j, t'_j) \mid 1 \le j \le n\right\}.$$

Let $V(j,T)$ be the set of vertices lying on I_j between t_j and s_j (excluding t_j and s_j), and similarly, let $V(j,F)$ be the set of vertices lying on I_j between s_j and t'_j. The intuition is that, since the paths routing the demand pairs are required to be completely disjoint, for each $1 \le j \le n$, we can only route one of the two pairs: (s_j, t_j) or (s_j, t'_j). The routing of the former pair is interpreted as assigning the value 'F' to variable x_j, and the routing of the latter pair is interpreted as assigning the value 'T' to variable x_j. Intuitively, in the former case, all vertices of $V(j,T)$ will be "blocked" by the path routing (s_j, t_j), while in the latter case all vertices of $V(j,F)$ are "blocked".

We now turn to define the second set, $\mathcal{M}_2$ of the demand pairs. Let $R = R_{N-4m-6}$ be the row lying within distance $4m+6$ from the bottom row of the grid. Let $y_1, \ldots, y_m$ be any set of m vertices on R, ordered from left to right, so that the distance between every consecutive pair is at least $4m+5$; the distance between y_1 and the left boundary of G is at least $4m+5$, and the distance between y_m and the right boundary of G is at least $4m+5$. Since the grid size is $N \times N$, and $N = (m+1)(4m+6)$, we can find such vertices $y_1, \ldots, y_m$. For each $1 \leq h \leq m$, vertex y_h will serve as a source vertex corresponding to the clause C_h. We will associate it with three destination vertices, z_h^1, z_h^2, z_h^3, as follows. Assume that $C_h = \ell_{h_1} \vee \ell_{h_2} \vee \ell_{h_3}$. For $1 \leq i \leq 3$, let x_{h_i} be the variable corresponding to the literal ℓ_{h_i}. If $\ell_{h_i} = x_{h_i}$, then we let z_h^i be some vertex in set $V(h_i, T)$, and otherwise we let z_h^i be some vertex in set $V(h_i, F)$. We select the vertices z_h^i in such a way, that all vertices in set $Z = \left\{z_h^i \mid 1 \leq h \leq m, 1 \leq i \leq 3\right\}$ are distinct. Since each variable participates in exactly 5 clauses, and each set $V(j,T), V(j,F)$ contains 5 vertices, we can ensure that all vertices in Z are distinct. We define:

$$\mathcal{M}_2 = \left\{(y_h, z_h^1), (y_h, z_h^2), (y_h, z_h^3) \mid 1 \leq h \leq m\right\}.$$

Before we define the third set of the demand pairs, we provide some intuition. As mentioned above, we associate each assignment in $\{T, F\}$ to each variable x_j with the routing of either (s_j, t_j) or (s_j, t'_j) along the corresponding segment of the first row. For each clause C_h, if at least one of its literals ℓ_{h_i} is satisfied, we will route the corresponding demand pair (y_h, z_h^i) (we discuss this in more detail later). However, in the No-Instance case, a solution can "cheat" by routing the pairs (s_j, t_j), or (s_j, t'_j) differently: for example, we can route them on a path that goes around some of the sources y_h. In order to avoid this, we create an artificial "bottleneck" by adding a new set of demand pairs. Recall that $v(i,j)$ is a vertex lying in the intersection of row R_i and column C_j of the grid. The last set $\mathcal{M}_3$ of demand pairs contains $8m$ demand pairs $\{a_i, b_i\}_{i=1}^{8m}$, where for $1 \leq i \leq 8m$, we define $a_i = v(m+4+i, m+1)$, and $b_i = v(m+4+i, N)$. In other words, the ith demand pair in set $\mathcal{M}_3$ consists of the $(m+1)$st and the last vertex of the row R_{m+4+i}. The final set of the demand pairs is $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$. This completes the description of the NDP instance. We now analyze its properties.

Completeness

Assume that the 3SAT(5) formula φ is a Yes-Instance. We show that in this case we can route $9m + n = 16n$ demand pairs. Consider the assignment $f : \{x_1, \ldots, x_n\} \to \{T, F\}$ that satisfies φ.

For each $1 \leq i \leq n$, if x_i is assigned the value 'T', then we route the pair (s_i, t'_i) via the segment of the row R_1 between these two vertices; if x_i is assigned value F, then we route the pair (s_i, t_i) via the corresponding segment of R_1. For each pair $(a_i, b_i) \in \mathcal{M}_3$, we route (a_i, b_i) via the segment of row R_{m+4+i} connecting these two vertices. Finally, we define the routing of m demand pairs in $\mathcal{M}_2$. For each clause C_h, let ℓ_h^* be any of the literals of C_h that is satisfied by the assignment f, and let $z_h = z_h^i$ be the destination vertex corresponding to ℓ_h^*, so that $(y_h, z_h) \in \mathcal{M}_2$. We will route the pairs $\{(y_h, z_h)\}_{1 \leq h \leq m}$.

In order to do so, we define three sub-grids of G: B_1 is the sub-grid spanned by rows $R_2, \ldots, R_{m+5}$, and all columns of the grid; B_2 is the sub-grid spanned by rows $R_{m+5}, \ldots, R_{9m+4}$ and columns $C_1, \ldots, C_m$ of the grid; and B_3 spanned by rows $R_{9m+4}, \ldots, R_N$ and all columns of the grid.

For each $1 \leq h \leq m$, let e_h be the unique vertical edge of the grid incident on vertex z_h, and let z'_h be its other endpoint. Let $S_1 = \{z'_h \mid 1 \leq h \leq m\}$, so S_1 contains m distinct vertices on the top row of B_1, and let $E' = \{e_h \mid 1 \leq h \leq m\}$. Let S_2 be the set of m vertices on the top boundary of B_2. Then the vertices of S_2 also lie on the bottom boundary of B_1, and from Observation 1, there is a set $\mathcal{P}_1$ of disjoint paths in B_1, connecting all vertices of S_1 to the vertices of S_2, so that the paths in $\mathcal{P}_1$ are internally disjoint from $V(R_2 \cup R_{m+5})$. Let S_3 be the set of m vertices on the bottom boundary of B_2, and let $\mathcal{P}_2$ be the set of the columns of B_2, so $\mathcal{P}_2$ is a set of m paths, connecting all vertices of S_2 to the vertices of S_3, in graph B_2. Finally, consider the graph B_3, and observe that S_3 is a set of m distinct vertices lying on the top boundary of B_3, while $\{y_h \mid 1 \leq h \leq m\}$ is a set of m vertices lying at L_∞-distance at least $4m+5$ from each other, and from the boundary of B_3. From Theorem 2, we can route any matching between the vertices of S_3 and the vertices of $\{y_h \mid 1 \leq h \leq m\}$ in graph S_3. Let $\mathcal{P}'$ be the set of paths obtained by concatenating $E', \mathcal{P}_1, \mathcal{P}_2$. Then $\mathcal{P}'$ is a set of disjoint paths connecting the vertices of $\{z_h \mid 1 \leq h \leq m\}$ to the vertices of S_3. We denote the vertices of S_3 by $\{z''_1, \ldots, z''_m\}$, where z''_h is the vertex that serves as an endpoint of the path of $\mathcal{P}'$ originating at z_h. We can now construct a set $\mathcal{P}_3$ of disjoint paths in B_3, routing the pairs $\{(y_h, z''_h) \mid 1 \leq h \leq m\}$. By concatenating the paths in $\mathcal{P}'$ and $\mathcal{P}_3$, we obtain the final routing of the pairs in $\{(y_h, z_h) \mid 1 \leq h \leq m\}$. Altogether, we route n demand pairs in $\mathcal{M}_1$, all $8m$ demand pairs in $\mathcal{M}_3$, and m demand pairs in $\mathcal{M}_2$, routing $n + 9m = 16n$ pairs in total.

Soundness

Let $\delta = (1-\epsilon)/200$, where ϵ is the constant from Theorem 15. Assume that φ is a No-Instance, so no assignment can satisfy more than ϵm clauses of φ. We show that the value of the optimal solution of the corresponding NDP problem is at most $(1-\delta)\cdot 16n$. Assume otherwise, and let $\mathcal{P}$ be a set of paths, routing more than $(1-\delta)\cdot 16n$ demand pairs.

Our first observation is that at least $6m$ of the demand pairs in $\mathcal{M}_3$ must be routed by $\mathcal{P}$. Indeed, assume otherwise. Then $\mathcal{P}$ routes at most n pairs in $\mathcal{M}_1$, fewer than $6m$ pairs in $\mathcal{M}_3$, and at most m pairs in $\mathcal{M}_2$. In total, $\mathcal{P}$ routes at most $n + 7m = 38n/3 < (1-\delta)\cdot 16n$ pairs, since $\delta < 1/200$. Therefore, at least $6m$ of the demand pairs in $\mathcal{M}_3$ are routed. Let i be the smallest index, so that (a_i, b_i) is routed in $\mathcal{P}$, and let $P \in \mathcal{P}$ be the path routing (a_i, b_i). Let U be the set of vertices of column C_{m+1} (the column where the sources of the pairs in $\mathcal{M}_3$ lie), that belong to rows $R_1, \ldots, R_{9m+4}$. We use the following observation.

▶ **Observation 16.** *There is a contiguous sub-path P' of P, containing b_i and some vertex of U, such that P' is internally disjoint from U, and it does not contain any vertex of row $R = R_{N-4m-6}$.*

Proof. If P does not contain any vertex of R, then, since it must contain at least one vertex of U (the vertex a_i), such path P' clearly exist. Therefore, we assume that $P \cap R \neq \emptyset$. Let v be the last vertex of P lying on row R, where we view P as directed from a_i to b_i. Let P^* be the segment of P from v to b_i.

We claim that $P^* \cap U \neq \emptyset$. Indeed, assume otherwise. Let C_j be the column in which v lies and let Q be the segment of C_j from v to the bottom vertex of C_j. If C_j is the last column, then path P^* separates all vertices in $\{a_j\}_{j=1}^{8m}$ from all vertices in $\{t_j\}_{j=i+1}^{8m}$, contradicting the fact that at least $6m$ demand pairs in $\mathcal{M}_3$ are routed, and i is the smallest index for which pair (a_i, b_i) is routed. Therefore, C_j is not the last column. The union of Q and P^* partitions the face defined by $\Gamma(G)$ into a number of sub-faces. Let F_2 be the sub-face containing the top left boundary of the grid, and let F_1 be the union of the remaining sub-faces. Since $P^* \cup Q$ is disjoint from U, all vertices $\{a_j\}_{j=1}^{8m}$ belong to F_2, while the vertices $\{t_j\}_{j=i+1}^{8m}$ belong to F_1. Therefore, all paths of $\mathcal{P}$ routing the pairs in $\mathcal{M}_3$ must intersect Q, while Q contains only $4m+7$ vertices, a contradiction. We conclude that

$P^* \cap U \neq \emptyset$. Let u be the last vertex on P^* that belongs to U. We can then let P' be the segment of P^* between u and b_i. ◀

Let v^* be the endpoint of P' lying in U, and let $R' = \text{row}(v^*)$. Let I be the sub-path of R' between v^* and the first vertex of row R' (excluding v^*). Since path P' is disjoint from row R, it is easy to see that every path in $\mathcal{P}$ that routes a demand pair in $\mathcal{M}_2$ has to contain at least one vertex of I.

We partition the set of variables of φ into three subsets. Set X_1 contains all variables x_j, such that none of the pairs (s_j, t_j), (s_j, t'_j) is routed by $\mathcal{P}$; X_2 contains all variables x_j, such that one of the pairs (s_j, t_j), (s_j, t'_j) is routed by some path $Q_j \in \mathcal{P}$, and $|Q_j \cap I| \geq 2$. Set X_3 contains all remaining variables. We need the following three observations.

▶ **Observation 17.** $|X_1| \leq 16\delta n$.

Proof. Assume otherwise. Then $\mathcal{P}$ routes fewer than $n(1-16\delta)$ pairs of $\mathcal{M}_1$, at most $8m$ pairs of $\mathcal{M}_2$ and at most m pairs of $\mathcal{M}_3$. In total, this is fewer than $n(1-16\delta) + 9m = 16n(1-\delta)$ pairs, a contradiction. ◀

▶ **Observation 18.** $|X_2| \leq 8\delta n$.

Proof. Assume otherwise. As observed above, if $(y, z) \in \mathcal{M}_2$ is routed by $\mathcal{P}$ via some path Q, then $Q \cap I \neq \emptyset$. Since $|I| = m$, the number of pairs in $\mathcal{M}_2$ routed by $\mathcal{P}$ is less than $m - 16\delta n$, and the total number of pairs routed is smaller than $n + (m - 16\delta n) + 8m = 16n(1-\delta)$. ◀

▶ **Observation 19.** *Let $x_j \in X_3$ be some variable, and let $Q \in \mathcal{P}$ be the path originating at s_j. If Q terminates at t_j, then no path of $\mathcal{P}$, routing a demand pair in $\mathcal{M}_2$, may contain any vertex of $V(j, T)$, and if Q terminates at t'_j, then no path of $\mathcal{P}$, routing a demand pair in $\mathcal{M}_2$, may contain any vertex of $V(j, F)$.*

Proof. Assume that Q terminates at t_j: the proof for t'_j is symmetric. Since $|I \cap Q| < 2$, the path Q, together with the sub-path of R_1 between t_j and s_j, forms a closed curve L in the natural drawing of the grid, such that all sources of all pairs in $\mathcal{M}_2$ lie outside L. Therefore, the paths of $\mathcal{P}$ originating from the sources of the demand pairs in $\mathcal{M}_2$ cannot contain the vertices of $V(j, T)$. ◀

We now define an assignment to the variables of φ that satisfies more than ϵm clauses of φ, leading to a contradiction. The assignment is defined as follows. For each variable $x_j \in X_3$, let $Q_j \in \mathcal{P}$ be the path originating at s_j. If Q_j terminates at t_j, then we assign the value 'F' to x_j; otherwise we assign the value 'T' to it. All other variables are assigned arbitrary values.

Let $\mathcal{C}$ be the collection of clauses C_h, such that there is a path originating at vertex y_h in $\mathcal{P}$. It is easy to see that $|\mathcal{C}| \geq m - 16\delta n$, since otherwise $\mathcal{P}$ contains fewer than $n + 8m + (m - 16\delta n) = 16n(1-\delta)$ paths. Let $\mathcal{C}' \subseteq \mathcal{C}$ be the subset of clauses containing the variables of $X_1 \cup X_2$. Since each variable participates in at most 5 clauses, from Observations 17 and 18, $|\mathcal{C}'| \leq 5 \cdot 24\delta n = 120\delta n$. Let $\mathcal{C}^* = \mathcal{C} \setminus \mathcal{C}'$. Then $|\mathcal{C}^*| \geq m - 136\delta n \geq \epsilon m$. We claim that every clause $C_h \in \mathcal{C}^*$ is satisfied by our assignment. Indeed, let $P \in \mathcal{P}$ be the path originating at y_h, and let z_h^i be its other endpoint. Assume that the corresponding literal ℓ_{h_i} corresponds to variable x_j. From our definition of $\mathcal{C}^*$, $x_j \in X_3$. Let $P' \in \mathcal{P}$ be the path originating from s_j. If $z_h^i \in V(j, T)$, then $\ell_{h_i} = x_j$. From Observation 19, P' terminates at t'_j, and variable x_j is assigned the value 'T'. If $z_h^i \in V(j, F)$, then $\ell_{h_i} = \overline{x}_j$. From Observation 19, P' terminates at t_j, and variable x_j is assigned the value 'F'. In either case, the assignment to x_j satisfies the clause C_h.

To conclude, we have shown an efficient algorithm, that, given a 3SAT(5) formula φ, constructs an instance $(G, \mathcal{M})$ of the NDP problem, where G is a grid graph, whose size is polynomial in the size of φ. If φ is a Yes-Instance, then there is a solution of value $16n$ to the NDP instance, and if φ is a No-Instance, then no solution routes more than $16n(1-\delta)$ demand pairs in the NDP instance, for some constant δ. Since it is NP-hard to distinguish the Yes- and the No-instances of 3SAT(5), we conclude that no efficient algorithm can obtain a better than $(1-\delta)$-approximation for NDP on grids, unless $\mathsf{P} = \mathsf{NP}$.

6 Integrality Gap of (LP-flow) for Good Pairs

We prove that the integrality gap of (LP-flow) is $\Omega(n^{1/8})$ even when all of the terminals are far from the grid boundary. We note that the family of instances that we construct here was previously used by Cutler and Shiloah [16], to provide a lower bound on the size of permutation layouts. Our analysis also closely follows theirs.

Given any integer $p > 10$, let $k = p^2$ and $N = 6k$. We show that the integrality gap of (LP-flow) on the $(N \times N)$ grid G, where all terminals are within distance at least $N/6$ from $\Gamma(G)$ is $\Omega(k^{1/4}) = \Omega(n^{1/8})$, where $n = N^2$ is the number of vertices in the grid.

In order to define the demand pairs, we let S be any set of k consecutive vertices on row R_{2k} of G, where all vertices are at distance at least $2k$ from both the left and the right boundary of G, and define a set T of k consecutive vertices on row R_{4k} similarly. We partition the set S into p subsets $S_1, \ldots, S_p$ of p consecutive vertices each, where for $1 \leq i, j \leq p$, the jth vertex in set S_i is denoted by $s_{i,j}$. Similarly, we partition T into p subsets $T_1, \ldots, T_p$ of p consecutive vertices each, and for $1 \leq i, j \leq p$, the jth vertex in set T_i is denoted by $t_{i,j}$. The set $\mathcal{M}$ of the demand pairs is then:

$$\mathcal{M} = \{(s_{i,j}, t_{j,i}) \mid 1 \leq i, j \leq p\}.$$

It is easy to see that there is a solution to (LP-flow) of value $k/3$: for each pair $(s_{i,j}, t_{j,i})$, we send $1/3$ flow unit on the path P, lying in the union $\mathrm{col}(s_{i,j}), \mathrm{col}(t_{j,i})$ and R_{ip+j}, that connects $s_{i,j}$ to $t_{j,i}$. We next show that the value of any integral solution is $O(k^{3/4})$, thus establishing the integrality gap of $\Omega(k^{1/4})$.

In our analysis we use the notions of graph drawing and graph crossing number. A drawing of a graph H in the plane is a mapping, in which every vertex of H is mapped into a point in the plane, and every edge into a continuous curve connecting the images of its endpoints, such that no three curves meet at the same point, and no curve contains an image of any vertex other than its endpoints. A *crossing* in such a drawing is a point where the images of two edges intersect, and the *crossing number* of a graph H, denoted by $\mathsf{cr}(H)$, is the smallest number of crossings achievable by any drawing of H in the plane. We use the following well-known theorem [2, 23].

▶ **Theorem 20.** *For any graph $H = (V, E)$ with $|E| > 7|V|$, $\mathsf{cr}(H) \geq \frac{|E|^3}{29|V|^2}$.*

Let OPT denote the optimal integral solution for the instance $(G, \mathcal{M})$, let $\mathcal{M}^* \subseteq \mathcal{M}$ be the set of the demand pairs routed by OPT, and let $x = |\mathsf{OPT}|$. We define two bipartite graphs. The first bipartite graph, $H = (S, T, E^*)$ is defined over the sets S and T of the source and the destination vertices of $\mathcal{M}$, and it contains an edge $e = (s, t)$ for every pair $(s, t) \in \mathcal{M}^*$. The second graph is $H' = (A, B, E')$, where $A = \{v_1, \ldots, v_p\}$, $B = \{u_1, \ldots, u_p\}$, and E' contains all edges (v_i, u_j), where $(s_{i,j}, t_{j,i}) \in \mathcal{M}^*$. The following claim is central to our analysis.

▶ **Claim 21.** *There is a drawing of H' with at most $2px$ crossings.*

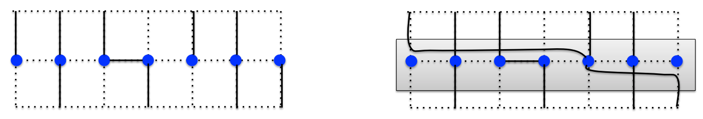

(a) Before **(b)** After

Figure 5 Altering the drawing around S_i.

We prove Claim 21 below, after we complete the analysis of the integrality gap here. If $|E'| \le 14p$, then $|\mathsf{OPT}| = O(\sqrt{k})$ and we are done, so we assume that $|E'| > 14p$. Then from Theorem 20, $\mathsf{cr}(H') \ge \frac{x^3}{116p^2}$, while from Claim 21, $\mathsf{cr}(H') \le 2px$. Therefore, $x = O(p^{3/2}) = O(k^{3/4})$. It now remains to prove Claim 21.

Proof of Claim 21. Notice that the natural drawing of the grid G, together with the solution OPT to the NDP instance gives a planar drawing φ of the graph H in the plane. For each $1 \le i \le p$, let $S'_i \subseteq S_i$ be the set of the sources that have an edge incident to them in E^*, and define $T'_i \subseteq T_i$ similarly. Let $x_i = |S'_i|$ and $y_i = |T'_i|$. For each $1 \le i \le p$, if $x_i = 0$, then the vertex v_i of H', corresponding to S_i is an isolated vertex, and we can draw it anywhere. Otherwise, let $s_{i,j} \in S'_i$ be any vertex. We draw v_i at $\varphi(s_{i,j})$. Let $I(i)$ be the segment of row R_{2k} containing the vertices of S_i, and no other vertices. Let L_i be a very thin strip (of height 1/10) around the segment $I(i)$ (see Figure 5). We alter the drawings of all edges in E^*, originating at the vertices of S'_i, so that they now originate at $\varphi(s_{i,j})$, by re-routing them inside the strip L_i. Since the number of paths in OPT containing the vertices of S_i is bounded by p, it is easy to do so, by introducing at most px_i crossings. We perform the same transformation for the sets T_i of destination vertices, and obtain a drawing of the graph H' with at most $p\sum_{i=1}^{p}(x_i + y_i) \le 2px$ crossings. ◀

7 Approximation Algorithm for EDP on Wall Graphs

In this section we show that the algorithm from Section 4 can be adapted to give an $O(n^{1/4} \cdot \log n)$-approximation for EDP on wall graphs of width and height $N = \Omega(\sqrt{n})$. In order to construct a wall W of height h and width r (or an $(h \times r)$- wall), we start from a grid of height h and width $2r$. Consider some column C_j of the grid, for $1 \le j \le r$, and let $e^j_1, e^j_2, \ldots, e^j_{h-1}$ be the edges of C_j, in the order of their appearance on C_j, where e^j_1 is incident on $v(1, j)$. If j is odd, then we delete from the graph all edges e^j_i where i is even. If j is even, then we delete from the graph all edges e^j_i where i is odd. We process each column C_j of the grid in this manner, and in the end delete all vertices of degree 1. The resulting graph is a wall of height h and width r, that we denote by W (See Figure 1).

Let E_1 be the set of edges of W that correspond to the horizontal edges of the original grid, and let E_2 be the set of the edges of W that correspond to the vertical edges of the original grid. The sub-graph of W induced by E_1 is a collection of h node-disjoint paths, that we refer to as the rows of W. We denote these rows by $R_1, \ldots, R_h$, where for $1 \le i \le h$, R_i is incident on $v(i, 1)$. Let V_1 denote the set of all vertices in the first row of W, and V_h the set of vertices in the last row of W. There is a unique set $\mathcal{C}$ of r node-disjoint paths, where each path $C \in \mathcal{C}$ starts at a vertex of V_1, terminates at a vertex of V_h, and is internally disjoint from $V_1 \cup V_h$. We refer to these paths as the columns of W. We order these columns from left to right, and denote by C_j the jth column in this ordering, for $1 \le j \le r$. The sub-graph $\Gamma(W) = R_1 \cup C_1 \cup R_h \cup C_r$ of W is a simple cycle, that we call the boundary of W.

For every vertex $v \in V(W)$, we let $\mathrm{col}(v)$ and $\mathrm{row}(v)$ denote the column and the row of W to which v belongs. As before, for a pair $u, v \in V(W)$ of vertices, we define:

$$d_\infty(u,v) = \max\left\{|\operatorname{col}(v) - \operatorname{col}(u)|, |\operatorname{row}(v) - \operatorname{row}(u)|\right\},$$

and for a vertex v and a subset $U \subseteq V(W)$ of vertices, we let $d_\infty(v,U) = \min_{u\in U}\{d_\infty(u,v)\}$.

Assume now that we are given an $(N \times N)$-wall graph $G = (V,E)$, so $n = |V| = \Theta(N^2)$, and a collection $\mathcal{M} = \{(s_i,t_i)\}_{i=1}^k$ of demand pairs. As before, we say that a demand pair (s_i,t_i) is bad if both $d_\infty(s_i,\Gamma(G)), d_\infty(t_i,\Gamma(G)) \le 4\sqrt{N}+4$, and we say that it is good otherwise. Let $\mathcal{M}', \mathcal{M}'' \subseteq \mathcal{M}$ denote the sets of the good and the bad demand pairs in $\mathcal{M}$, respectively. We find an approximate solution to each of the two sub-problems, defined by $\mathcal{M}'$ and $\mathcal{M}''$, separately, and take the better of the two solutions.

The algorithm for the bad pairs remains exactly the same as the algorithm from Section 4.2. We now focus on the problem defined by the set $\mathcal{M}'$ of the good pairs. Let G' be the $(N \times N)$-grid obtained from G, by contracting, for each $1 \le i,j \le N$, the unique edge $e \in R_i \cap C_j$, and consider the NDP problem instance $(G', \mathcal{M}')$. Any collection $\mathcal{P}'$ of node-disjoint paths in G', routing a subset $\tilde{\mathcal{M}} \subseteq \mathcal{M}'$ of the demand pairs immediately gives a collection $\mathcal{P}''$ of edge-disjoint paths in G, routing the same subset of the demand pairs. Moreover, it is easy to see that there is an LP-solution to (LP-flow) on instance $(G', \mathcal{M}')$ of value $\mathsf{OPT}'/2$, where OPT' is the optimal solution for the EDP instance $(G, \mathcal{M}')$. Indeed, for every path $P \in \mathsf{OPT}'$, we simply set $f(P') = 1/2$, where P' is the path of G' corresponding to the path P of G, and for every demand pair (s_j,t_j) routed by OPT', we set $x_j = 1/2$. It is immediate to verify that this is a feasible solution to (LP-flow) on NDP instance $(G', \mathcal{M}')$, of value $\mathsf{OPT}'/2$. We then use the algorithm from Section 4.1 to find an $O(n^{1/4} \cdot \log n)$-approximation solution to $(G', \mathcal{M}')$, which in turn gives an $O(n^{1/4} \cdot \log n)$-approximation solution to the EDP instance $(G, \mathcal{M}')$.

References

1 Alok Aggarwal, Jon Kleinberg, and David P. Williamson. Node-disjoint paths on the mesh and a new trade-off in VLSI layout. *SIAM J. Comput.*, 29(4):1321–1333, February 2000.

2 M. Ajtai, V. Chvátal, M.M. Newborn, and E. Szemerédi. Crossing-free subgraphs. In Gert Sabidussi Peter L. Hammer, Alexander Rosa and Jean Turgeon, editors, *Theory and Practice of Combinatorics A collection of articles honoring Anton Kotzig on the occasion of his sixtieth birthday*, volume 60 of *North-Holland Mathematics Studies*, pages 9–12. North-Holland, 1982.

3 Matthew Andrews. Approximation algorithms for the edge-disjoint paths problem via Raecke decompositions. In *Proceedings of IEEE FOCS*, pages 277–286, 2010.

4 Matthew Andrews, Julia Chuzhoy, Venkatesan Guruswami, Sanjeev Khanna, Kunal Talwar, and Lisa Zhang. Inapproximability of edge-disjoint paths and low congestion routing on undirected graphs. *Combinatorica*, 30(5):485–520, 2010.

5 Matthew Andrews and Lisa Zhang. Hardness of the undirected edge-disjoint paths problem. In *STOC*, pages 276–283. ACM, 2005.

6 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45:501–555, May 1998.

7 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of np. *J. ACM*, 45:70–122, January 1998.

8 Yonatan Aumann and Yuval Rabani. Improved bounds for all optical routing. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, SODA'95, pages 567–576, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

9 Chandra Chekuri and Julia Chuzhoy. Half-integral all-or-nothing flow. Unpublished Manuscript.

10 Chandra Chekuri and Alina Ene. Poly-logarithmic approximation for maximum node disjoint paths with constant congestion. In *Proc. of ACM-SIAM SODA*, 2013.

11 Chandra Chekuri, Sanjeev Khanna, and F Bruce Shepherd. Edge-disjoint paths in planar graphs. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 71–80. IEEE, 2004.

12 Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *Proc. of ACM STOC*, pages 183–192, 2005.

13 Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. An $O(\sqrt{n})$ approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of Computing*, 2(1):137–146, 2006.

14 Julia Chuzhoy. Routing in undirected graphs with constant congestion. In *Proc. of ACM STOC*, pages 855–874, 2012.

15 Julia Chuzhoy and Shi Li. A polylogarithimic approximation algorithm for edge-disjoint paths with congestion 2. In *Proc. of IEEE FOCS*, 2012.

16 M. Cutler and Y. Shiloach. Permutation layout. *Networks*, 8:253–278, 1978.

17 R. Karp. On the complexity of combinatorial problems. *Networks*, 5:45–68, 1975.

18 Ken-Ichi Kawarabayashi and Yusuke Kobayashi. An o(log n)-approximation algorithm for the edge-disjoint paths problem in eulerian planar graphs. *ACM Trans. Algorithms*, 9(2):16:1–16:13, March 2013.

19 Jon M. Kleinberg. An approximation algorithm for the disjoint paths problem in even-degree planar graphs. In *FOCS'05*, pages 627–636, 2005.

20 Jon M. Kleinberg and Éva Tardos. Disjoint paths in densely embedded graphs. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 52–61, 1995.

21 Jon M. Kleinberg and Éva Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. *J. Comput. Syst. Sci.*, 57(1):61–73, 1998.

22 Stavros G. Kolliopoulos and Clifford Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99:63–87, 2004.

23 Frank Thomson Leighton. *Complexity Issues in VLSI: Optimal Layouts for the Shuffle-exchange Graph and Other Networks.* MIT Press, Cambridge, MA, USA, 1983.

24 Harald Räcke. Minimizing congestion in general networks. In *Proc. of IEEE FOCS*, pages 43–52, 2002.

25 Prabhakar Raghavan and Clark D. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, December 1987.

26 Satish Rao and Shuheng Zhou. Edge disjoint paths in moderately connected graphs. *SIAM J. Comput.*, 39(5):1856–1887, 2010.

27 N. Robertson and P. D. Seymour. Outline of a disjoint paths algorithm. In *Paths, Flows and VLSI-Layout.* Springer-Verlag, 1990.

28 Neil Robertson and Paul D. Seymour. Graph minors. VII. disjoint paths on a surface. *J. Comb. Theory, Ser. B*, 45(2):212–254, 1988.

29 Neil Robertson and Paul D Seymour. Graph minors. XIII. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

30 Loïc Seguin-Charbonneau and F. Bruce Shepherd. Maximum edge-disjoint paths in planar graphs with congestion 2. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS'11, pages 200–209, Washington, DC, USA, 2011. IEEE Computer Society.

Approximating Upper Degree-Constrained Partial Orientations[*]

Marek Cygan and Tomasz Kociumaka

Institute of Informatics, University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
{cygan,kociumaka}@mimuw.edu.pl

Abstract

In the Upper Degree-Constrained Partial Orientation (UDPO) problem we are given an undirected graph $G = (V, E)$, together with two degree constraint functions $d^-, d^+ : V \to \mathbb{N}$. The goal is to orient as many edges as possible, in such a way that for each vertex $v \in V$ the number of arcs entering v is at most $d^-(v)$, whereas the number of arcs leaving v is at most $d^+(v)$. This problem was introduced by Gabow [SODA'06], who proved it to be MAXSNP-hard (and thus APX-hard). In the same paper Gabow presented an LP-based iterative rounding 4/3-approximation algorithm.

As already observed by Gabow, the problem in question is a special case of the classic 3-Dimensional Matching, which in turn is a special case of the k-Set Packing problem. Back in 2006 the best known polynomial time approximation algorithm for 3-Dimensional Matching was a simple local search by Hurkens and Schrijver [SIDMA'89], the approximation ratio of which is $(3+\varepsilon)/2$; hence the algorithm of Gabow was an improvement over the approach brought from the more general problems.

In this paper we show that the UDPO problem when cast as 3-Dimensional Matching admits a special structure, which is obliviously exploited by the known approximation algorithms for k-Set Packing. In fact, we show that already the local-search routine of Hurkens and Schrijver gives $(4+\varepsilon)/3$-approximation when used for the instances coming from UDPO. Moreover, the recent approximation algorithm for 3-Set Packing [Cygan, FOCS'13] turns out to be a $(5+\varepsilon)/4$-approximation for UDPO. This improves over 4/3 as the best ratio known up to date for UDPO.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases graph orientations, degree-constrained orientations, approximation algorithm, local search

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.212

1 Introduction

During the last decades several graph orientation problems were studied (see Section 8.7 in [2] and Section 61.1 in [13]). One of the most recently introduced is the Upper Degree-Constrained Partial Orientation, abbreviated as UDPO. In the UDPO problem we are given an undirected graph $G = (V, E)$, together with two degree constraint functions $d^-, d^+ : V \to \mathbb{N}$. The goal is to orient as many edges as possible, in such a way that for each vertex $v \in V$ the number of arcs entering v is at most $d^-(v)$, whereas the number of arcs leaving v is at most $d^+(v)$. This problem was introduced by Gabow [9], motivated by a variant of the maximum bipartite matching problem arising when planning a two-day event

[*] This work is partially supported by Foundation for Polish Science grant HOMING PLUS/2012-6/2.

with several parallel sessions and each participant willing to attend one chosen session each day, but without a particular order on the two selected sessions (for the exact definition, see [9]).

Upper Degree-Constrained Partial Orientation (UDPO)
Input: Undirected graph G, degree constraints $d^+, d^- : V(G) \to \mathbb{Z}_{\geq 0}$
Find: A subset $\overline{F} \subseteq E(G)$ which admits an orientation F satisfying $\deg^+_F(v) \leq d^+(v)$ and $\deg^-_F(v) \leq d^-(v)$ for each $v \in V(G)$.
Maximize: $|F|$

Gabow proved the problem to be MAXSNP-hard (thus also APX-hard), and showed an LP-based iterative rounding 4/3-approximation algorithm. As he already observed, UDPO is a special case of the 3-Dimensional Matching problem, which in turn is a special case of the k-set packing problem defined as follows.

k-set packing
Input: A family $\mathcal{F}$ of subsets of a finite universe U, such that $|F| \leq k$ for every $F \in \mathcal{F}$
Find: A subfamily $\mathcal{F}_0 \subseteq \mathcal{F}$ of pairwise-disjoint subsets
Maximize: $|\mathcal{F}_0|$

Note that at the time when these results were published, the best known approximation ratio for 3-set packing was the simple local search of Hurkens and Schrijver [12] with approximation ratio $(3 + \varepsilon)/2$. Therefore the 4/3 ratio achieved by Gabow improved upon the results obtained using the algorithm for the more general problem.

1.1 Our results

In this paper we analyze the behaviour of two known approximation algorithms for the k-set packing problem as solutions for UDPO. We prove that these algorithms obliviously exploit hidden structural properties present in every 3-set packing instance obtained via the natural reduction from UDPO. Consequently, when cast as algorithms for the UDPO problem, these local-search routines attain better approximation ratios than they do for the worst-case instances of the 3-set packing or 3-Dimensional Matching problems.

First, we show that already the simple local-search routine of Hurkens and Schrijver [12] is a $(4 + \varepsilon)/3$-approximation when used for the instances coming from UDPO. Next, we prove that the recent algorithm for 3-set packing [7], again, used as a black box, turns out to be a $(5 + \varepsilon)/4$-approximation for UDPO. This way we derive the best known ratio for UDPO, improving over 4/3 obtained by the algorithm of Gabow. In fact, our approximation ratio matches the 5/4 lower bound on the integrality gap of the underlying natural LP relaxation [9].

Technical contribution of our paper is based on the analysis of *simple* instances, where all the degree bounds are either zero or one, which means that each vertex can have only zero or one incoming and outgoing arcs. Interestingly, for a wide class of local-search routines, simple instances are actually no easier than arbitrary ones. The properties of these instances give rise to a 4-set packing-like structure which can be used in the analysis, though it is not explicitly used by the algorithms.

1.2 Organization of the paper

In the following subsection we discuss related work on the subject. Next, in Section 2.1 we recall the reduction from UDPO to 3-set packing, followed by Section 2.2 where we describe the local-search algorithms from previous work on k-set packing. In Section 3 we state the main properties behind the analyses of the approximation ratios of both algorithms and slightly strengthen both these results.

In the remaining sections we provide an improved analysis of the performance of both algorithms on instances obtained via the reduction from UDPO. In Section 4 we prove that the worst-case approximation ratio is already attained by simple instances (with all degree bounds at most one). The properties of simple instances are applied in Section 5 to obtain a performance guarantee complementary to those in Section 3. Finally, in Section 6 we combine the two to derive our main results.

1.3 Related work on k-set packing

Prior to the the recent improvements for the k-SET PACKING problem [7, 14], quasipolynomial-time algorithms with approximation ratios $(k+2)/3$ [10] and $(k+1+\varepsilon)/3$ [8] were obtained.

There is also a line of research on the weighted variant of k-SET PACKING, where we want to select a maximum-weight family of pairwise-disjoint sets from $\mathcal{F}$. Arkin and Hassin [1] gave a $(k-1+\varepsilon)$-approximation algorithm, later Chandra and Halldórsson [6] improved it to a $(2k+2+\varepsilon)/3$-approximation. Currently, the best-known approximation ratio is $(k+1+\varepsilon)/2$ due to Berman [3]. All the mentioned results are based on local search.

For the standard (unweighted) k-SET PACKING problem, Chan and Lau [5] also presented a strengthened LP relaxation with integrality gap $(k+1)/2$.

On the other hand, Hazan et al. [11] proved that k-SET PACKING is hard to approximate within a factor of $\mathcal{O}(k/\log k)$. Concerning small values of k, Berman and Karpinski [4] obtained a $(98/97-\varepsilon)$-hardness for 3-DIMENSIONAL MATCHING, which implies the same lower bound for 3-SET PACKING.

2 Preliminaries

Let G be an undirected (multi)graph. We sometimes treat G as a directed graph where each *edge* $e \in E(G)$ is represented by a pair of oppositely directed *arcs* in $A(G)$. For an arc $e \in A(G)$ we denote by $\overline{e}$ the corresponding edge in $E(G)$, and by e^R the reverse arc. We also define $\overline{A} = \{\overline{e} : e \in A\}$ and $A^R = \{e^R : e \in A\}$ for an arbitrary subset $A \subseteq A(G)$.

A *partial orientation* of G can be defined as a subset $F \subseteq A(G)$ such that $F^R \cap F = \emptyset$. It is called *feasible* (for degree constraints $d = (d^+, d^-)$), if $\deg_F^+(v) \leq d^+(v)$ and $\deg_F^-(v) \leq d^-(v)$ for each $v \in V(G)$, that is, if the number of arcs leaving v and the number of arcs entering v do not violate the upper bounds. Now, UDPO can be reformulated as the problem of finding a maximum feasible partial orientation F, rather than the corresponding set of undirected edges $\overline{F}$.

For an undirected (multi)graph G and a set $U \subseteq V(G)$ we also define $N_G(U)$ as the set of vertices $v \notin U$ adjacent to some $u \in U$; we also set $N_G[U] = N_G(U) \cup U$. The subgraph induced by a subset $X \subseteq V(G)$ is denoted as $G[X]$. A bipartite graph H with a fixed bipartition $V(H) = A \cup B$ is often represented as a triple $(A, B, E(H))$. The subgraph induced by $A' \cup B'$ with $A' \subseteq A$ and $B' \subseteq B$ is then dented as $H[A', B']$.

2.1 Reduction to 3-set packing

The following reduction to 3-SET PACKING was introduced by Gabow [9]. Let $I = (G, d)$ be an instance of UDPO. We construct an equivalent instance of the 3-SET PACKING problem, i.e., a set family $\mathcal{F}$ over a universe U.

The universe U is a disjoint union of three sets: V^+, V^- and E. The set V^+ contains $d^+(v)$ copies v_i^+ of each $v \in V(G)$, V^- contains $d^-(v)$ copies v_i^- of each $v \in V(G)$, and E is defined as $E(G)$. The family $\mathcal{F}$ consists of sets $\{u_i^+, v_j^-, e\}$ and $\{v_j^+, u_i^-, e\}$ for each edge $e = \{u, v\}$ and all possible indices i, j.

Given a feasible partial orientation F, the degree constraints clearly let us choose for each arc $e = uv$ two copies u_i^+ and v_j^-, so that the choices are distinct across all arcs leaving u and entering v, respectively. Consequently, the sets $\{u_i^+, v_j^-, \bar{e}\}$ form a disjoint subfamily of $\mathcal{F}$. Similarly, given any disjoint set-family $\mathcal{F}_0 \subseteq \mathcal{F}$ it is easy to see that orienting $\bar{e}$ from u to v for any $\{u_i^+, v_j^-, \bar{e}\} \in \mathcal{F}_0$ gives a feasible partial orientation.

2.2 Local search for k-set packing

In this section we recall and reinterpret some of the results behind two local-search approaches to the k-SET PACKING problem: the classic one yielding a $(k+\varepsilon)/2$-approximation [12] and the recent $(k+1+\varepsilon)/3$-approximation by Cygan [7].

For an instance $(U, \mathcal{F})$ of the k-SET PACKING problem, we build an undirected conflict graph $G = G(\mathcal{F})$ with $V(G) = \mathcal{F}$ and vertices F, F' made adjacent if $F \cap F' \neq \emptyset$. Observe that solutions to this instance of k-SET PACKING correspond to independent sets in this graph. The algorithms maintain a solution $\mathcal{F}_0 \subseteq \mathcal{F}$ and try to replace it with a larger, but similar solution. They try to use a disjoint family $X \subseteq \mathcal{F} \setminus \mathcal{F}_0$ and replace $\mathcal{F}_0$ by $\mathcal{F}_0' = (\mathcal{F}_0 \setminus N_G(X)) \cup X$, where $G = G(\mathcal{F})$ is the conflict graph. Note that $N_G(X) \cap \mathcal{F}_0$ consists exactly of those members of $\mathcal{F}_0$ which cannot be present together with X in a single disjoint family. It is reasonable to preform this operation if the resulting family $\mathcal{F}_0'$ is larger than $\mathcal{F}_0$, or equivalently $|N_G(X) \cap \mathcal{F}_0| < |X|$. This leads to a notion of *improving sets*, defined for $\mathcal{F}_0 \subseteq \mathcal{F}$ as disjoint families $X \subseteq \mathcal{F} \setminus \mathcal{F}_0$ such that $|N_G(X) \cap \mathcal{F}_0| < |X|$. The classic approach to the k-SET PACKING problem is to search for improving sets of sufficiently large constant size.

▶ **Fact 1** (Weak rule). *There exists an algorithm that, given a k-SET-PACKING instance $\mathcal{F}$ and a disjoint family $\mathcal{F}_0 \subseteq \mathcal{F}$, in $|\mathcal{F}|^{\mathcal{O}(r)}$ time determines whether there exists an improving set $X \subseteq \mathcal{F} \setminus \mathcal{F}_0$ of size at most r and if so, finds such an improving set.*

The novel idea of [7] was to consider larger improving sets satisfying structural properties, which allow for efficient detection of these sets. This is achieved using a structural parameter of a graph called *pathwidth*. In this paper we only use some results of [7] as a black-box, so we do not need to recall the relatively complex definition of pathwidth. Pathwidth of an undirected graph G, denoted as $\mathrm{pw}(G)$, does not exceed the number of vertices in any connected component of G. Pathwidth of an improving set X is defined as $\mathrm{pw}(G[X \cup \mathcal{F}_0])$ where $G = G(\mathcal{F})$ is the conflict graph. The following theorem uses techniques of fixed-parameter tractability to find improving sets of logarithmic size and constant pathwidth.

▶ **Theorem 2** (Strong rule: [7], Theorem 3.6). *There is an algorithm that, given a k-SET-PACKING instance $\mathcal{F}$ and a disjoint family $\mathcal{F}_0 \subseteq \mathcal{F}$, in $|\mathcal{F}|^{\mathcal{O}(C \cdot k)}$ time determines whether there exists an improving set $X \subseteq \mathcal{F} \setminus \mathcal{F}_0$ of size at most $C \log |\mathcal{F}|$ and pathwidth at most C, and if so, finds such an improving set.*

Note that $\mathrm{pw}(G[X \cup \mathcal{F}_0]) \leq |X \cup (\mathcal{F}_0 \cap N(X))| < 2|X|$ for any improving set. Thus, the strong rule is able to find all improving sets discovered by the weak rule if only we set $C \geq 2r$. Moreover, let us note that both rules are *monotone* in a certain sense.

▶ **Observation 3.** *If no improving set can be found using Theorem 2 for $\mathcal{F}_0 \subseteq \mathcal{F}$, then one still cannot find an improving set if the instance $\mathcal{F}$ is restricted to any $\mathcal{F}'$ such that $\mathcal{F}_0 \subseteq \mathcal{F}' \subseteq \mathcal{F}$. The weak rule of Fact 1 enjoys the same property.*

We say that a partial orientation F is a *local optimum* if the underlying family $\mathcal{F}_0$ cannot be improved using the reduction rule in question. For the weak rule of Fact 1 (with fixed r) we call such orientations *weak local optima* and for the strong rule of Theorem 2 (with fixed C) — *strong local optima*.

The remaining part of this paper is devoted to the analysis how large these local optima can be compared to the global optimum. More precisely, we show that for every $\varepsilon > 0$ there is an appropriate choice of parameters such that $|F| \geq (\frac{3}{4} - \varepsilon)|OPT|$ for any weak local optimum F and global optimum OPT, while for any strong local optimum this can be improved to $|F| \geq (\frac{4}{5} - \varepsilon)|OPT|$. The parameters $r = r_\varepsilon$ and $C = C_\varepsilon$ do not depend on the instance, so the running time of the implementations of both local-search rules is polynomial.

3 Tools from k-set packing

In this section we recall and reinterpret several pieces of the analyses of the local-search algorithms for k-SET PACKING; see [12, 7]. We focus on the subgraph of the conflict graph $G(\mathcal{F})$ induced by two solutions: a local and a global optimum. Sets belonging to both families can be ignored, which leads to a bipartite graph with degrees bounded by k. The following results are stated in the language of abstract bipartite graphs so that we can later use some of them in a slightly different context.

▶ **Definition 4.** Let $H = (A, B, E(H))$ be a bipartite graph. A set $X \subseteq B$ is called *improving*, if $|N_H(X)| < |X|$.

A slightly simpler version of the following lemma is a part of the analysis of the classic $(k+\varepsilon)/2$-approximation local search, which goes back to Hurkens and Schrijver [12]. Here, we observe that the worst-case $(k+\varepsilon)/2$ ratio can be attained only if (almost) all vertices in A are of degree k. If a constant fractions of vertices does not satisfy this property, our variant lets us derive a better bound.

▶ **Lemma 5.** *Fix a positive integer $k \geq 3$. For every $\varepsilon > 0$ there exists a constant c_ε satisfying the following property. Let $H = (A, B, E(H))$ be a bipartite graph with degrees not exceeding k. If there is no improving set $X \subseteq B$ with $|X| \leq c_\varepsilon$, then*

$$|B| \leq \tfrac{k-1+\varepsilon}{2}|A| + \tfrac{1}{2}|\{a \in A : \deg_H(a) = k\}|.$$

The proof below is based on the proof of Lemma 3.11 in [7], where the following auxiliary lemma is (implicitly) proved. For completeness, we provide its proof in the Appendix.

▶ **Lemma 6** ([7]). *Fix a positive integer $k \geq 3$ and a real number $\varepsilon > 0$. Let $H = (A, B, E(H))$ be a bipartite graph with degrees not exceeding k. If there is no improving set $X \subseteq B$ with $|X| \leq 2(k+1)^{\varepsilon^{-1}}$, then there exists an induced subgraph $H' = H[A', B']$ such that:*
(a) $|A \setminus A'| = |B \setminus B'|$,
(b) *there are no vertices $b \in B'$ with $\deg_{H'}(b) = 0$,*
(c) *there are at most $\varepsilon|A|$ vertices $b \in B'$ with $\deg_{H'}(b) = 1$.*

Proof of Lemma 5. Lemma 6 applied to the graph H gives its subgraph $H' = H[A', B']$. For every integer d define $B'_d = \{b \in B' : \deg_{H'}(b) = d\}$ and $B'_{d+} = \{b \in B' : \deg_{H'}(b) \geq d\}$. Let us count edges of H': clearly, $|E(H')| \leq (k-1)|A'| + |\{a \in A' : \deg_{H'}(a) = k\}|$ since the degrees do not exceed k. On the other hand, $|E(H')| \geq |B'_1| + 2|B'_{2+}|$, and consequently $|B'_1| + 2|B'_{2+}| \leq (k-1)|A'| + |\{a \in A' : \deg_{H'}(a) = k\}|$. Combining this inequality with the properties of H' following from Lemma 6, we get

$$\begin{aligned}&2|B| = 2|B \setminus B'| + 2|B'| = 2|A \setminus A'| + 2|B'_1| + 2|B'_{2+}| \leq \\ &2|A \setminus A'| + |B'_1| + (k-1)|A'| + |\{a \in A' : \deg_{H'}(a) = k\}| \leq \\ &\qquad\qquad (k - 1 + \varepsilon)|A| + |\{a \in A : \deg_H(a) = k\}|,\end{aligned}$$

that is, $|B| \leq \frac{k-1+\varepsilon}{2}|A| + \frac{1}{2}|\{a \in A : \deg_H(a) = k\}|$. ◀

The following lemma is a slightly stronger variant of Lemma 3.11 in [7]. Again we provide a better bound whenever a constant fraction of vertices in A does not have full degree.

▶ **Lemma 7.** *Fix a positive integer $k \geq 3$. For every $\varepsilon > 0$ there exists a constant C_ε satisfying the following property. Let $H = (A, B, E(H))$ be a bipartite graph with degrees not exceeding k. If there is no improving set $X \subseteq B$ such that $|X| \leq C_\varepsilon \log |V(H)|$ and $\mathrm{pw}(H[N_G[X]]) \leq C_\varepsilon$, then*

$$|B| \leq (\tfrac{k}{3} + \varepsilon)|A| + \tfrac{1}{3}|\{a \in A : \deg_H(a) \geq k\}|.$$

Proof. The proof is similar to that of Lemma 5. Instead of Lemma 6, we use the following auxiliary result proved in [7]. The construction of the subgraph H' is the same as in the proof of Lemma 6 (see Appendix). However, to derive point (4), formulated as Claim 3.12 in [7], the wider range of possibilities for X is exploited.

▶ **Claim 8** ([7]). *For every $\varepsilon > 0$ the constant C_ε can be chosen so that there exists an induced subgraph $H' = H[A', B']$ such that:*
(a) $|A \setminus A'| = |B \setminus B'|$,
(b) *there are no vertices $b \in B'$ with $\deg_{H'}(b) = 0$,*
(c) *there are at most $\varepsilon|A|$ vertices $b \in B'$ with $\deg_{H'}(b) = 1$,*
(d) *there are at most $(1+\varepsilon)|A'|$ vertices $b \in B'$ with $\deg_{H'}(b) = 2$.*

Again, for every integer d define $B'_d = \{b \in B' : \deg_{H'}(b) = d\}$ and $B'_{d+} = \{b \in B' : \deg_{H'}(b) \geq d\}$. As before, we count edges $E(H')$. We have $|E(H')| \geq |B'_1| + 2|B'_2| + 3|B'_{3+}|$ and $|E(H')| \leq (k-1)|A'| + |\{a \in A' : \deg_{H'}(a) = k\}|$. Summing up, we obtain

$$\begin{aligned}&3|B| = 3|B \setminus B'| + 3|B'_1| + 3|B'_2| + 3|B'_{3+}| = 3|A \setminus A'| + 2|B'_1| + |B'_2| + |E(H')| \leq \\ &3|A \setminus A'| + 2\varepsilon|A| + (1+\varepsilon)|A'| + (k-1)|A'| + |\{a \in A' : \deg_{H'}(a) = k\}| \leq \\ &3|A \setminus A'| + k|A'| + 3\varepsilon|A| + |\{a \in A' : \deg_H(a) = k\}| \leq \\ &\qquad\qquad (k + 3\varepsilon)|A| + |\{a \in A : \deg_H(a) = k\}|,\end{aligned}$$

that is, $|B| \leq (\frac{k}{3} + \varepsilon)|A| + \frac{1}{3}|\{a \in A : \deg_H(a) = k\}|$. ◀

4 Reduction to simple instances

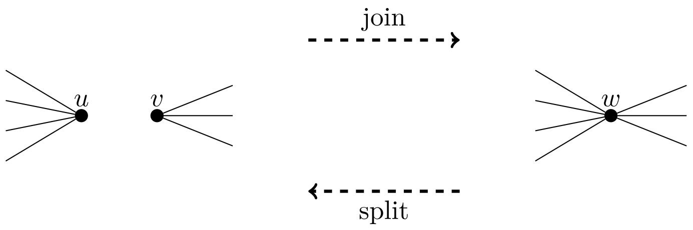
An instance $I = (G, d)$ of UDPO is called *simple* if $d^+(v), d^-(v) \in \{0, 1\}$ for every $v \in V(G)$ and *proper* if $\deg_G(v) \geq \max(d^+(v), d^-(v)) > 0$ for every $v \in V$. Clearly, any instance can be easily reduced to an equivalent proper instance by decreasing the degree constraints. In this section we show that it suffices to analyze the local-search algorithms for simple instances. More precisely, we prove that the worst-case ratio between the sizes of a local and a global optima is attained already for simple instances. Although this is stated below as an existential result, our reduction is constructive and it could be efficiently implemented.

▶ **Theorem 9.** *Fix a monotone local-search rule for* 3-SET PACKING. *Suppose that there exists an instance I of* UDPO *with a locally-optimum partial orientation F such that $|F| = \alpha|OPT_I|$. Then there exists a simple instance I' of* UDPO *with a locally-optimum partial orientation F' satisfying $|F'| = \alpha|OPT_{I'}|$.*

Let $I = (G, d)$ be an arbitrary instance. For a pair of distinct non-adjacent vertices $u, v \in V(G)$ we define the operation of *joining* u and v as follows: u and v are *identified* in G into a single vertex w and their degree constraints for w are obtain by summing the respective constraints for u and v.

Let us analyze how joining can be interpreted in terms of 3-SET PACKING instances obtained through the reduction of Section 2.1. Let $\mathcal{F}$ and $\mathcal{F}'$ families of 3-sets produced from instances I and I', respectively before and after joining. The universes $U = V^+ \cup V^- \cup E$ and $U' = V'^+ \cup V'^- \cup E'$ of both families can regarded as equal. This is because identifying non-adjacent vertices preserves the edge-set of the graph and because $d^+(u)$ copies of u and $d^+(v)$ copies of v in V^+ can be identified with $d^+(w) = d^+(u) + d^+(v)$ copies of w in V'^+ (similarly for V^- and V'^-). In this setting all 3-sets in $\mathcal{F}$ also belong to $\mathcal{F}'$ (though $\mathcal{F}'$ might be a strict superset of $\mathcal{F}$). Consequently, if a partial orientation is feasible in I, it is also feasible in the resulting instance I', but the converse does not necessarily hold.

If I' is obtained from I by joining u and v into w, we say that I can be obtained from I' by *splitting* w. Splitting is said to *preserve* a partial orientation A, if A is feasible in I' and remains feasible in I.



▶ **Lemma 10.** *Let $I = (G, d)$ be a proper instance with two feasible partial orientations A, B. If $\max(d^+(v), d^-(v)) \geq 2$ for some $v \in V(G)$, then one can split v so that both A and B are preserved and the resulting instance I' is proper.*

Proof. First, let us introduce an auxiliary vertex v' connected to v by $d^+(v) + d^-(v)$ parallel edges. We extend d to v' setting the constraints in v' large enough to accommodate all edges incident to v'. Note that this operation does not alter the original edges of G and the constraints at their endpoints. Hence, it has no effect on feasibility of A or B, in particular on whether one can split v.

Now, let us modify A to obtain A' by orienting $d^+(v) - \deg^+_A(v)$ edges from v to v' and $d^-(v) - \deg^-_A(v)$ edges from v' to v. Note that A' is feasible in the extended graph and the degree constraints for v are tight. Analogously, we extend B to B'. A larger partial orientation may only be harder to preserve, so it suffices to prove that one can split v preserving A' and B'. Equivalently, the construction in this paragraph lets us assume that $\deg^+_A(v) = \deg^+_B(v) = d^+(v)$ and $\deg^-_A(v) = \deg^-_B(v) = d^-(v)$.

Both for A and B we classify edges of G incident to v into three types: oriented towards v $(-)$, oriented towards the other endpoint $(+)$ and not included in the orientation (0). In total, we get a partition of the set $\delta(v)$, consisting of edges incident to v, into nine sets E_{ab}

with $a, b \in \{+, -, 0\}$; here a corresponds to the orientation in A and b to the orientation in B.

In some situations, one can clearly take a few edges incident to v, and split v into two vertices, one *new* vertex $\tilde{v}$ incident to the selected edges, and the other, still denoted as v, incident to the remaining edges. We refer to this operation as *splitting out* some edges. Note that in order to preserve both A and B, we need to split out edges so that for $\tilde{v}$ the number incoming edges is the same in both orientations, similarly for the outgoing arcs. We shall make sure that this number is always 0 or 1, i.e., $(d^+(\tilde{v}), d^-(\tilde{v})) \in \{(0,1), (1,0), (1,1)\}$. The constraints at v are decreased accordingly.

1. If $E_{++} \neq \emptyset$, one can split out a single edge $e \in E_{++}$ setting constraints $(1, 0)$; symmetrically if $E_{--} \neq \emptyset$ one sets $(0, 1)$.
2. If $E_{+-} \neq \emptyset$ and $E_{-+} \neq \emptyset$, one can split out two edges – one of each type, setting constraints $(1, 1)$.
3. If $E_{0+} \neq \emptyset$ and $E_{+0} \neq \emptyset$, one can split out two edges – one of each type, setting constraints $(1, 0)$; symmetrically if $E_{0-} \neq \emptyset$ and $E_{-0} \neq \emptyset$ one sets $(0, 1)$.
4. If $E_{+-} \neq \emptyset$, $E_{0+} \neq \emptyset$, and $E_{-0} \neq \emptyset$, one can split out three edges – one of each type, setting constraints $(1, 1)$; symmetrically, if $E_{-+} \neq \emptyset$, $E_{+0} \neq \emptyset$, and $E_{0-} \neq \emptyset$, one also sets $(1, 1)$.

We shall prove that one of these rules is always applicable. Note that the resulting instance is guaranteed to be proper as we have $\max(d^+(v), d^-(v)) \geq 2$, so it is impossible to leave v with both constraints equal to 0, which is forbidden in proper instances.

We proceed by contradiction, showing that if no rule is applicable, then $d^+(v) = d^-(v) = 0$, which is impossible because I is proper. Let $n_{ab} = |E_{ab}|$. Recall that we have made an assumption that $\deg^+_A(v) = \deg^+_B(v) = d^+(v)$ and $\deg^-_A(v) = \deg^-_B(v) = d^-(v)$, which implies the following equalities:

$$n_{0+} + n_{++} + n_{-+} = d^+(v) = n_{+0} + n_{++} + n_{+-},$$
$$n_{0-} + n_{+-} + n_{--} = d^-(v) = n_{-0} + n_{-+} + n_{--}.$$

If $n_{++} > 0$ or $n_{--} > 0$ we could apply rule 1. Therefore

$$n_{0+} + n_{-+} = d^+(v) = n_{+0} + n_{+-},$$
$$n_{0-} + n_{+-} = d^-(v) = n_{-0} + n_{-+}.$$

If $n_{+-} > 0$ and $n_{-+} > 0$ we could apply rule 2; without loss of generality we assume $n_{+-} = 0$ and thus

$$n_{0+} + n_{-+} = d^+(v) = n_{+0},$$
$$n_{0-} = d^-(v) = n_{-0} + n_{-+}.$$

Consequently, we have $n_{+0} \geq n_{0+}$ and $n_{0-} \geq n_{-0}$. Therefore, if $n_{0+} > 0$ or $n_{-0} > 0$, we could apply rule 3, which means that both these values are equal to 0 and

$$n_{0-} = n_{+0} = n_{-+} = d^+(v) = d^-(v).$$

However, if the common value of these variables was not equal to 0, we could apply rule 4. This way we get the announced contradiction. ◀

▶ **Corollary 11.** *If I is a proper instance with feasible partial orientations A and B, then with a finite sequence of vertex splitting preserving both A and B, one can obtain a simple proper instance I'.*

Proof. It suffices to exhaustively apply Lemma 10. Observe that this process must terminate, as vertex splitting increases the number of vertices and changes neither $D^+ = \sum_{v\in V(G)} d^+(v)$ nor $D^- = \sum_{v\in V(G)} d^-(v)$, while $|V(G)| \le D^+ + D^-$ for every proper instance. ◀

For a proof of Theorem 9, it suffices to apply Corollary 11 for $A = F$ and $B = OPT_I$. Vertex splitting may only reduce the family of feasible partial orientations, so OPT_I is still a global optimum. Also, this operation preserves F as a local optimum with respect any fixed monotone local-search rule. This follows from the fact that vertex splitting can be seen as removing sets in the underlying instance of 3-SET PACKING (without changing the size of the universe), and monotonicity means that removing sets from the universe does not make finding an improving set easier.

Therefore, Corollary 11 gives a simple instance I' for which F and OPT_I are still a local and a global optimum, respectively.

5 Another conflict graph for simple instances

We start the analysis of the local-search algorithms with a different construction of a conflict graph for a pair of feasible partial orientations A and B in a simple instance I of UDPO. The construction exploits the properties of simple instances and does not naturally generalize to arbitrary ones.

Let us consider an undirected graph $G' = (V(G), \overline{A} \cap \overline{B})$ and let $\mathcal{C}$ be the family of connected components of G'. For a connected component $C \in \mathcal{C}$ we define $\delta_A[C]$ as the set of arcs $e \in A$ incident to *exactly* one vertex of C; analogously we define $\delta_B[C]$.

▶ **Fact 12.** *For every component $C \in \mathcal{C}$ we have $|\delta_A[C]| \le 2$ and $|\delta_B[C]| \le 2$.*

Proof. As I is a simple instance, all the vertices in G' are of degree at most two, which means that C is a path or a cycle. Consequently, in either case, again by the assumption that I is simple, we have $|\delta_A[C]| \le 2$ and $|\delta_B[C]| \le 2$, because, both in A and in B, at most $2|C|$ arc endpoints can be incident to C and at least $2(|C| - 1)$ of these are endpoints of arcs induced by C. ◀

Let $A' = \{e \in A : \bar{e} \in \overline{A} \setminus \overline{B}\}$ and $B' = \{e \in B : \bar{e} \in \overline{B} \setminus \overline{A}\}$. We construct a bipartite graph $H = (A', B', E_H)$ so that $a \in A'$ is adjacent to $b \in B'$ whenever there is a component $C \in \mathcal{C}$ such that simultaneously $a \in \delta_A[C]$ and $b \in \delta_B[C]$. Since every arc in A' or B' is incident to exactly two components, Fact 12 lets us easily bound the degrees in H.

▶ **Corollary 13.** *The degree of every vertex in H is at most 4.*

The following lemma lets us interpret H as a conflict graph between A and B.

▶ **Lemma 14.** *For any $X \subseteq B'$ the following three-step procedure modifies A into another feasible partial orientation A_X:*
(a) *remove all arcs in $N_H(X)$,*
(b) *add all arcs in X,*
(c) *reverse all arcs in components C such that $X \cap \delta_B[C] \ne \emptyset$.*

Proof. We shall prove that that resulting orientation A_X satisfies the degree constraints for every vertex $v \in V(G)$. Let C be the component of v in G' (possibly $C = \{v\}$).

If $X \cap \delta_B[C] \ne \emptyset$, we shall prove that arcs incident to v in A_X form a subset of arcs incident to v in B. Indeed, by construction of H, all arcs $e \in A'$ incident to C were removed in step (1). Moreover, all arcs induced by C were reoriented in step (3) (from the orientation consistent with A to the orientation consistent with B). We might have added some arcs $e \in X$ incident to v in step (2) but these arcs are also present in B. Similarly, if $X \cap \delta_B[C] = \emptyset$, we shall prove that arcs incident to v in A_X form a subset of arcs incident to v in A. Indeed, no arcs incident to C could have been added in step (1) and the arcs induced by C were not reoriented in step (3). The only possible changes were removals in step (1).

In both cases we have shown that the arcs incident to v in A_X form a subset of arcs incident to v in another feasible partial orientation. Hence, A_X is feasible at any vertex v. ◀

Unfortunately, improvements through Lemma 14 in general might yield reorientation of many edges, which is unfeasible for our local-search rules. Thus, we slightly restrict the graph H to make sure that small improving sets in H yield small improving sets in the underlying 3-SET PACKING instance.

▶ **Lemma 15.** *Let I be a simple instance of* UDPO *and let A, B be a pair of feasible partial orientations. For any $\varepsilon > 0$ there exists a bipartite graph $H_\varepsilon = (A', B_\varepsilon, E_{H_\varepsilon})$ such that:*

(a) *$B_\varepsilon \subseteq B'$ and $|B_\varepsilon| \geq |B'| - \varepsilon|A|$,*
(b) *degrees in H_ε do not exceed 4,*
(c) *for any $X \subseteq B_\varepsilon$ there is a feasible partial orientation A_X with $|A_X| = |A| + |X| - |N_{H_\varepsilon}(X)|$ and $|A_X \setminus A| \leq (1 + 4\varepsilon^{-1})|X|$.*

Proof. Let $\mathcal{C}_\varepsilon \subseteq \mathcal{C}$ consist of components inducing at least $2\varepsilon^{-1}$ edges in G'. Note that the total size of components $C \in \mathcal{C}$ is $|\overline{A} \cap \overline{B}| \leq |A|$, so $|\mathcal{C}_\varepsilon| \leq \frac{\varepsilon}{2}|A|$. We set B_ε as the set of arcs $e \in B'$ which are not incident to any component $C \in \mathcal{C}_\varepsilon$. Fact 12 implies $|B' \setminus B_\varepsilon| \leq 2|\mathcal{C}_\varepsilon| \leq \varepsilon|A|$ as claimed in (1). We take H_ε as the induced subgraph $H[A, B_\varepsilon]$, so (2) immediately follows from Corollary 13.

Note that $N_{H_\varepsilon}(X) = N_H(X)$ for any $X \subseteq B_\varepsilon$. Thus, we can apply Lemma 14 to obtain the orientation A_X of the desired size. For every arc $b \in X$ step (3) yields reorientation of arcs induced by at most two components $C \in \mathcal{C} \setminus \mathcal{C}_\varepsilon$. These components consist of up to $2\varepsilon^{-1}$ edges each, so in total we reorient no more than $4\varepsilon^{-1}|X|$ arcs. Together with X itself, this gives at most $(1 + 4\varepsilon^{-1})|X|$ arcs in $A_X \setminus A$. ◀

Next, we analyze this conflict graph using tools originally developed for the classic local-search $(2 + \varepsilon)$-approximation of 4-SET PACKING.

▶ **Lemma 16.** *For any $\delta > 0$ there exists a constant r_δ such that for any simple instance I of* UDPO *the following condition holds. Let F be a weak local optimum (with $r = r_\delta$) and let OPT be an optimum partial orientation. Then $|\overline{OPT} \setminus \overline{F}| \leq 2|\overline{F} \setminus \overline{OPT}| + \delta|F|$.*

Proof. We proceed with a proof by contradiction for r_δ to be specified later. We apply Lemma 15 to $A = F$, $B = OPT$ and $\varepsilon = \frac{1}{2}\delta$ to obtain a bipartite graph H_ε. Note that $|B_\varepsilon| \geq |\overline{OPT} \setminus \overline{F}| - \varepsilon|F| \geq 2|\overline{F} \setminus \overline{OPT}| + \varepsilon|F| \geq (2 + \varepsilon)|\overline{F} \setminus \overline{OPT}| = (2 + \varepsilon)|A'|$.

We plug H_ε to Lemma 5 for $k = 4$ to conclude that there is an improving set $X \subseteq B_\varepsilon$ of size at most c_ε. Lemma 15(3) implies that there exists a feasible orientation F_X such that $|F_X| > |F|$ and $|F_X \setminus F| \leq (1 + 4\varepsilon^{-1})c_\varepsilon$. Thus, setting $r_\delta = (1 + 8\delta^{-1})c_{\delta/2}$ we can make sure that the weak rule of Fact 1 is able to perform the underlying improvement. This contradicts the assumption that F is a weak local optimum. ◀

6 Analysis

Finally, we combine Lemma 16 with generic properties of 3-SET PACKING local optima.

▶ **Theorem 17.** *For every $\varepsilon > 0$ there exists a constant r_ε such that for any instance of* UDPO *and any feasible partial orientation F which is a weak local optimum (with $r = r_\varepsilon$), we have $|OPT| \leq (\frac{4}{3} + \varepsilon)|F|$, where OPT is a maximum feasible partial orientation.*

Proof. By Theorem 9, it suffices to prove the claim for simple instances only. Let $C = OPT \cap F$. Note that $F \setminus C$ and $OPT \setminus C$ induce a bipartite subgraph $H = (F \setminus C, OPT \setminus C, E(H))$ of the conflict graph in the underlying instance of 3-SET PACKING.

Suppose $\deg_H(e) = 3$ for some arc $e \in F$. Note that $e = uv$ is represented by a 3-set $\{u_i^+, v_j^-, \bar{e}\}$ for some indices $i \leq d^+(u)$ and $j \leq d^-(v)$. The three neighbours of e in H are represented by disjoint 3-sets intersecting $\{u_i^+, v_j^-, \bar{e}\}$. One of them must contain $\bar{e}$ and since $e \notin OPT$, we conclude that $e^R \in OPT$. Consequently, $|\{e \in F : \deg_H(e) = 3\}| \leq |\overline{OPT} \cap \overline{F}|$.

We set r_ε at least as large as in Lemma 16 and as c_ε in Lemma 5 for $k = 3$. The latter result yields

$$|OPT| = |C| + |OPT \setminus C| \leq |C| + (1+\varepsilon)|F \setminus C| + \tfrac{1}{2}|\{e \in F \setminus C : \deg_H(e) = 3\}| \leq (1+\varepsilon)|F| + \tfrac{1}{2}|\overline{OPT} \cap \overline{F}|.$$

If $|\overline{OPT} \cap \overline{F}| \leq \frac{2}{3}|F|$, this already concludes the proof. Otherwise $|\overline{F} \setminus \overline{OPT}| \leq \frac{1}{3}|F|$ and we apply Lemma 16 to get $|\overline{OPT} \setminus \overline{F}| \leq 2|\overline{F} \setminus \overline{OPT}| + \varepsilon|F|$, and consequently obtain

$$\begin{aligned}|OPT| = |\overline{OPT} \setminus \overline{F}| + |\overline{OPT} \cap \overline{F}| &\leq 2|\overline{F} \setminus \overline{OPT}| + |\overline{OPT} \cap \overline{F}| + \varepsilon|F| \\ &= |\overline{F} \setminus \overline{OPT}| + (1+\varepsilon)|F| \leq (\tfrac{4}{3} + \varepsilon)|F|,\end{aligned}$$

which concludes the proof. ◀

▶ **Theorem 18.** *For every $\varepsilon > 0$ there exists a constant C_ε such that for any instance of* UDPO *and any feasible partial orientation F which is a strong local optimum (with $C = C_\varepsilon$), we have $|OPT| \leq (\frac{5}{4} + \varepsilon)|F|$, where OPT is a maximum feasible partial orientation.*

Proof. We apply the same argument except that we use Lemma 7 instead of Lemma 5. We take C_ε at least as large as in Lemma 7 for $k = 3$ and so that $C_\varepsilon \geq 2r_\varepsilon$ from Lemma 16. Then

$$|OPT| \leq |C| + (1+\varepsilon)|F \setminus C| + \tfrac{1}{3}|\{e \in F \setminus C : \deg_H(e) = 3\}| \leq (1+\varepsilon)|F| + \tfrac{1}{3}|\overline{OPT} \cap \overline{F}|.$$

If $|\overline{OPT} \cap \overline{F}| \leq \frac{3}{4}|F|$, this already concludes the proof. Otherwise $|\overline{F} \setminus \overline{OPT}| \leq \frac{1}{4}|F|$ and we apply Lemma 16 to obtain $|OPT| \leq |\overline{F} \setminus \overline{OPT}| + (1+\varepsilon)|F| \leq (\frac{5}{4} + \varepsilon)|F|$. ◀

Acknowledgement. We thank Harold N. Gabow for pointing us to the UDPO problem.

References

1 Esther M. Arkin and Refael Hassin. On local search for weighted k-set packing. In Rainer Burkard and Gerhard Woeginger, editors, *Algorithms – ESA 1997*, volume 1284 of *LNCS*, pages 13–22. Springer Berlin Heidelberg, 1997.

2 Jørgen Bang-Jensen and Gregory Z. Gutin. *Digraphs: theory, algorithms and applications.* Springer Monographs in Mathematics. Springer, second edition, 2009.

3 Piotr Berman. A $d/2$ approximation for maximum weight independent set in d-claw free graphs. In Magnús M. Halldórsson, editor, *Algorithm Theory – SWAT 2000*, volume 1851 of *LNCS*, pages 214–219. Springer Berlin Heidelberg, 2000.

4 Piotr Berman and Marek Karpinski. Improved approximation lower bounds on small occurrence optimization. *Electronic Colloquium on Computational Complexity (ECCC)*, 10(008), 2003.
5 Yuk Hei Chan and Lap Chi Lau. On linear and semidefinite programming relaxations for hypergraph matching. *Mathematical Programming*, 135(1-2):123–148, 2012.
6 Barun Chandra and Magnús M. Halldórsson. Greedy local improvement and weighted set packing approximation. *Journal of Algorithms*, 39(2):223–240, 2001.
7 Marek Cygan. Improved approximation for 3-dimensional matching via bounded pathwidth local search. In *54th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 509–518. IEEE Computer Society, 2013.
8 Marek Cygan, Fabrizio Grandoni, and Monaldo Mastrolilli. How to sell hyperedges: The hypermatching assignment problem. In *24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 342–351. SIAM, 2013.
9 Harold N. Gabow. Upper degree-constrained partial orientations. In *17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 554–563. SIAM, 2006.
10 Magnús M. Halldórsson. Approximating discrete collections via local improvements. In *6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 160–169. SIAM, 1995.
11 Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k-dimensional matching. In Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai, editors, *Approximation, Randomization, and Combinatorial Optimization – APPROX-RANDOM 2003*, volume 2764 of *LNCS*, pages 83–97. Springer Berlin Heidelberg, 2003.
12 Cor A. J. Hurkens and Alexander Schrijver. On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics*, 2(1):68–72, 1989.
13 Alexander Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.
14 Maxim Sviridenko and Justin Ward. Large neighborhood local search for the maximum set packing problem. In Fedor V. Fomin, Rūsiņš Freivalds, Marta Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming – ICALP 2013*, volume 7965 of *LNCS*, pages 792–803. Springer Berlin Heidelberg, 2013.

A Proof of Lemma 6

▶ **Lemma 6.** *Fix a positive integer $k \geq 3$ and a real number $\varepsilon > 0$. Let $H = (A, B, E(H))$ be a bipartite graph with degrees not exceeding k. If there is no improving set $X \subseteq B$ with $|X| \leq 2(k+1)^{\varepsilon^{-1}}$, then there exists an induced subgraph $H' = H[A', B']$ such that:*
(a) $|A \setminus A'| = |B \setminus B'|$,
(b) *there are no vertices $b \in B'$ with $\deg_{H'}(b) = 0$,*
(c) *there are at most $\varepsilon|A|$ vertices $b \in B'$ with $\deg_{H'}(b) = 1$.*

Proof. We inductively construct a sequence of induced subgraphs $(H_i)_{i=0}^{\ell}$ with $H_0 = H$ and $H_\ell = H'$ such that each $H_i = H[A_i, B_i]$ satisfies the following properties:
1. $|A \setminus A_i| = |B \setminus B_i| \geq \varepsilon i|A|$.
2. in H_i there is no subset $X \subseteq B_i$ such that $|X| \leq 2(k+1)^{\varepsilon^{-1}-i}$ and $|N_{H_i}(X)| < |X|$,

Note that $H_0 = H$ trivially satisfies both these properties. For the inductive step, consider the graph H_i. Let us classify vertices of B_i based on their degree in H_i: we define B_i^d as the set of vertices of degree d, and B_i^{d+} as the set of vertices of degree at least d. Note that the property 1. implies $i \leq \frac{1}{\varepsilon}$, and thus $2(k+1)^{\frac{1}{\varepsilon}-i} \geq 2$. Consequently, by property 2., $B_i^0 = \emptyset$ and the vertices of B_i^1 have distinct neighbours (otherwise we would have an improving set of size one or two, respectively).

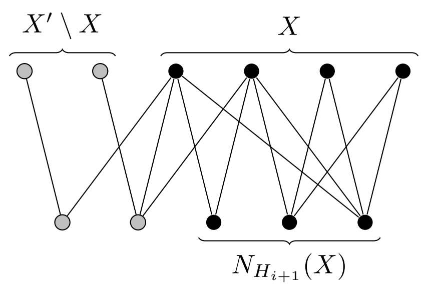


Figure 1 Lifting an improving set X in H_{i+1} to an improving set X' in H_i. Gray vertices belong to H_i but not to H_{i+1}.

We consider two cases, depending on whether $|B_i^1| \leq \varepsilon|A|$. If this inequality is satisfied, we shall prove that we can terminate at $i = \ell$ and return $H' = H_i$. The inequality directly corresponds to (3) in the statement of the lemma. Moreover, $B_i^0 = \emptyset$ is equivalent to (2) and the property 1. gives (1).

Otherwise, if $|B_i^1| > \varepsilon|A|$, we perform a further step of the construction. We build H_{i+1} setting $B_{i+1} = B_i^{2+}$ and $A_{i+1} = A_i \setminus N_{H_i}[B_i^1]$. As we have noted, vertices in B_i^1 do not share neighbours, so $|A_i \setminus A_{i+1}| = |B_i^1| = |B_i \setminus B_{i+1}|$, and consequently $|B \setminus B_{i+1}| = |A \setminus A_{i+1}|$. Also, we clearly have $|B \setminus B_{i+1}| \geq \varepsilon i|A| + |B_i^1| \geq \varepsilon(i+1)|A|$.

Hence, it suffices to show that H_{i+1} satisfies property 2. Take $X \subseteq B_{i+1}$ such that $|N_{H_{i+1}}(X)| < |X|$. We construct $X' \subseteq B_i$ with $|N_{H_i}(X')| < |X'|$ such that $|X'| \leq (k+1)|X|$. Clearly, if X then contradicts 2. for H_{i+1}, so does X' for H_i. Recall that $H_i[B_i \setminus B_{i+1}, A_i \setminus A_{i+1}]$ is a perfect matching. We denote the unique neighbour of a vertex v in this graph by $m(v)$. We simply define $X' = X \cup \{m(a) : a \in (A_i \setminus A_{i+1}) \cap N_{H_i}(X)\}$ (see also Figure 1). Then $N_{H_i}(X') = N_{H_i}(X) = N_{H_{i+1}}(X) \cup \{m(b) : b \in X' \setminus X\}$. Consequently, $|N_{H_i}(X')| = |N_{H_{i+1}}(X)| + |X' \setminus X| < |X| + |X' \setminus X| = |X'|$. Moreover, by the degree restriction in H, we have $|N_{H_i}(X)| \leq k|X|$, and thus $|X'| \leq |X| + |N_{H_i}(X)| \leq (k+1)|X|$, as claimed.

Finally, note that the property 1. implies $i \leq \varepsilon^{-1}$ so the construction terminates. ◀

Approximating Hit Rate Curves using Streaming Algorithms

Zachary Drudi, Nicholas J. A. Harvey, Stephen Ingram, Andrew Warfield, and Jake Wires

Coho Data Inc.
`{zach,nick,stephen,andy,jake}@cohodata.com`

Abstract

A hit rate curve is a function that maps cache size to the proportion of requests that can be served from the cache. (The caching policy and sequence of requests are assumed to be fixed.) Hit rate curves have been studied for decades in the operating system, database and computer architecture communities. They are useful tools for designing appropriate cache sizes, dynamically allocating memory between competing caches, and for summarizing locality properties of the request sequence. In this paper we focus on the widely-used LRU caching policy.

Computing hit rate curves is very efficient from a runtime standpoint, but existing algorithms are not efficient in their *space usage*. For a stream of m requests for n cacheable objects, all existing algorithms that provably compute the hit rate curve use space linear in n. In the context of modern storage systems, n can easily be in the billions or trillions, so the space usage of these algorithms makes them impractical.

We present the first algorithm for provably approximating hit rate curves for the LRU policy with sublinear space. Our algorithm uses $O\big(p^2 \log(n) \log^2(m)/\epsilon^2\big)$ bits of space and approximates the hit rate curve at p uniformly-spaced points to within additive error ϵ. This is not far from optimal. Any single-pass algorithm with the same guarantees must use $\Omega(p^2 + \epsilon^{-2} + \log n)$ bits of space. Furthermore, our use of additive error is necessary. Any single-pass algorithm achieving multiplicative error requires $\Omega(n)$ bits of space.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Cache analysis, hit rate curves, miss rate curves, streaming algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.225

1 Introduction

Caches are a fundamental concept in the design of computer systems. They are a mechanism for addressing the tradeoff between capacity and performance of different memory technologies. Even the early computers of the 1950s involved several memory technologies, and the interplay between these memories led to the foundational research on caching mechanisms in the 1960s.

The importance of caching is even more acute today. The memory hierarchy of a typical modern computer involves three levels of CPU caches, main memory, a mixture of solid-state and spinning disks for secondary storage, not to mention network file storage and web caches. The performance of nearly every modern computer depends crucially on caching mechanisms that aim to store the most appropriate data in the fastest memory.

A hit rate curve is a function that shows the performance benefit of caching as a function of the cache size. The study of hit rate curves began in the 1960s as computer designers aimed to optimize the price-performance ratio of their systems. One of the earliest discussions of hit rate curves appears in Belady's seminal 1966 paper [4] on caching and paging. Naturally, the question of how to efficiently compute a hit rate curve arose soon thereafter, and several algorithms were developed in the 1970s [19, 5, 25].

Although hit rate curves are not well-known within the theoretical computer science community, they feature prominently in the computer architecture and systems communities. Hit rate curves have been used in memory partitioning [27, 32, 26], garbage collection [30], program analysis [10, 31], workload phase detection [24], and cloud computing [16, 6]. CloudPhysics Inc., a datacenter performance analytics company, has recently developed a cloud-based workload analysis system using hit rate curves [28].

There are two key reasons for the resurgence of interest in hit rate curves. The first is the increasing computational penalty of cache misses. Over the past forty years, CPU speeds have improved roughly a thousandfold, whereas the latency of spinning disks has barely improved tenfold. Consequently, cache misses in storage systems are increasingly costly from a computational standpoint: there has been a substantial increase in the number of computational steps that are wasted while waiting to retrieve the data from disk. This phenomenon motivates the study of more computationally intensive cache analysis to avoid costly cache misses.

The second reason for the interest in hit rate curves is the increased amount of cache sharing, particularly due to the use of virtualization. CPU caches are shared among cores and threads; memory and solid-state storage devices are shared among processes and virtual machines. The marginal utility of increasing a process' cache allocation depends on the resulting performance improvement, which itself is determined by that process' hit rate curve. Consequently, hit rate curves are a useful ingredient for effective cache allocation.

Improved algorithms for computing hit rate curves have been developed over the years, but primarily by applied researchers rather than by theoretical algorithms researchers [5, 22, 1, 10, 32, 13, 21, 28]. There are two main goals of these improvements. The first is improved runtime, e.g. [13]: CPU cache analysis typically involves a fairly small data set but very high speeds, so performing this analysis online requires very fast runtime. The second is improved space usage, e.g. [28]: storage cache analysis typically involves extremely large data sets but at much lower speeds, so space usage is the primary concern. Typically a storage system cannot afford even a single bit of main memory for each block that appears in the secondary storage. Accordingly, most of the algorithms for computing hit rate curves in a storage context involve some sort of sampling or compression, but without rigorous guarantees.

This paper is the first to rigorously study space-efficient algorithms for computing hit rate curves. We focus on the LRU (Least Recently Used) caching policy. LRU and its variants are by far the most widely used caching policy for storage systems: the Linux, MacOS and Windows virtual memory systems all use approximations to LRU. In contrast, CPU caches tend to use simpler policies as speed is the primary concern. One policy that could potentially supplant LRU in a storage context is ARC [20], but its use has been hampered by intellectual property concerns.

1.1 Our Results

Our main result is an algorithm that computes an approximate hit rate curve for the LRU caching policy with a workload of length m involving n cacheable objects. The approximate hit rate curve achieves additive error ϵ at p uniformly-spaced points. The algorithm performs a single pass and uses $O\big(p^2 \log(n) \log^2(m)/\epsilon^2\big)$ bits of space.

We also prove lower bounds on the space. Any single-pass algorithm that achieves additive error ϵ at p uniformly-spaced points must use $\Omega(\min\big\{p^2 + \epsilon^{-2} + \log n, n\big\})$ bits of space.

Moreover, our use of additive error is necessary: any single-pass algorithm that achieves *multiplicative* error requires $\Omega(n)$ bits of space.

1.1.1 Practical Impact

It is often the case that an improved theoretical understanding of a problem leads to improved results in practice as well. That is indeed the case with the present work. Our algorithm has been incorporated into the enterprise storage system built by Coho Data, Inc. In these multi-terabyte storage systems, n and m are on the order of billions, whereas the hit rate curve as displayed to the user only requires p on the order of dozens, so the space used by our algorithm is dramatically less than previous methods. At the time of this writing, the Coho Data system is reporting live workload statistics for over a thousand virtual machines deployed in 30 customer environments, representing a level of workload characterization and insight that has not been possible in the past. A discussion of the practical aspects of this system was published in OSDI 2014 [29].

1.2 Preliminaries

In a caching system, there are n objects which can potentially be stored in the cache. Using the terminology of disk caches, we will refer to each item as a *block*, and we will identify the blocks with the integers $[n] = \{1, \ldots, n\}$. Blocks are requested at discrete points in time, which for simplicity we will always assume to be indexed by the set $[m] = \{1, \ldots, m\}$. There is a sequence of blocks $B = (b_1, \ldots, b_m)$ where $b_t \in [n]$ for each $t \in [m]$. We will refer to B as a sequence of *requests*, and say that the block b_t is requested at time t. A caching policy is a scheme for maintaining a set, called the *cache*, of at most k blocks. At each time step t, the block b_t is added to the cache, and some block may need to be removed to ensure that the cache size is at most k. If block b_t was already in the cache at time t then this request is called a *hit*, otherwise it is called a *miss*.

For a fixed sequence of requests B, a fixed size k, and a fixed caching policy, the *hit rate* is the fraction of requests that are a hit. The typical goal of a caching policy is to maximize the hit rate. If a caching policy is parameterized by the size k, then the *hit rate curve* is the function mapping k to the hit rate under this policy with cache size k.

The LRU caching policy can be defined concisely as: the cache consists of the k most recently requested distinct blocks. Whenever a miss occurs, the newly requested block must be inserted into the cache, and the least recently requested block must be removed from the cache (assuming that the cache was already full). LRU caches satisfy the *inclusion property*: for any fixed sequence of requests, the LRU cache of size k is, at each time step, a subset of the LRU cache of any size $K \geq k$. This follows directly from the definition of the LRU caching policy. Therefore a system with a cache of size K can easily simulate, or determine the hit rate, for any smaller cache. In the extreme case of $K = n$, the hit rate for *every* cache size (i.e., the entire hit rate curve) can be determined. This is the key observation that was exploited in previous algorithms [19].

1.3 The Main Idea

To understand this observation in more detail, consider a request b_t at time t, let $r < t$ be the time of the previous request for block $b_t = b_r$, and let $d = |\{b_r, \ldots, b_{t-1}\}|$ be the number of distinct blocks that were requested since time r. The request at time t would be a hit if the cache size k were at least d, and a miss for any cache size $k < d$ or if r does not exist. Thus, to compute the hit rate curve, it suffices to determine, for every request b_t, the number of distinct blocks that were requested since the previous request for b_t. All previous hit rate curve algorithms either compute this number using a dictionary data structure or estimate it using statistical arguments [5, 22, 1, 10, 32, 13, 21, 28]. The statistical arguments unfortunately do not give worst-case guarantees.

Our main idea is very natural. We estimate the number of distinct blocks that were requested since the previous request for b_t using an F_0-estimator (distinct element estimator) from the streaming algorithms literature. That is not the end of the story, of course. A single F_0-estimator would only give the number of distinct elements for a single sequence of requests, whereas we require such an estimate for *all suffixes* of B. This leads to the fruitful idea of using a *sliding window* F_0-estimator [9, 7] which, suitably modified, can provide the required estimate for all suffixes. Unfortunately this does not solve the problem either: the algorithm cannot easily determine which suffix to use because it cannot afford to store the previous request time of all blocks – that would also require $\Omega(n)$ bits of space. Ultimately, our algorithm avoids this issue by continuously measuring the contribution from *all* suffixes, and using those contributions to update the histogram in a somewhat intricate way.

1.4 Notation

To state our results precisely, let us fix some notation. Recall that we are only interested in requests that occur at discrete points in time indexed by $t \in \{1, \dots, m\}$. The set of requested blocks between time t' and strictly before time t is:

$$B(t', t) \;=\; \{\, b_i \,:\, i \in [m] \text{ and } t' \le i < t \,\}\,.$$

At time t, the most recent request for block b_t occurred at time

$$R(t) \;=\; \max\{\, x \,:\, x < t \text{ and } b_x = b_t \,\}\,.$$

We define $R(t) = -\infty$ if b_t was not requested before time t. At time t, the number of distinct blocks requested since the most recent request for block b is

$$D(t) \;=\; \begin{cases} |B(R(t), t)| & \text{(if } R(t) > -\infty) \\ \infty & \text{(otherwise).} \end{cases}$$

As observed above, an LRU cache of size k has a hit at time t if and only $D(t) \le k$. The *hit rate curve* is the function $C : [n] \to [0, 1]$ for which $C(k)$ is the hit rate for an LRU cache of size k. Thus

$$C(k) \;=\; |\{\, t \in [m] \,:\, D(t) \le k \,\}|/m.$$

In this paper we are concerned with computing the hit rate curve at p uniformly-spaced points, where p is a parameter. For simplicity, assume that $n = pw$, where w is an integer that denotes the "width" between the points. The *histogram* of D (with width w) is the function $H : [p] \to \mathbb{N}$ where

$$H(i) \;=\; |\{\, t \in [m] \,:\, (i-1)w < D(t) \le iw \,\}|. \qquad (1)$$

The fraction of requests that are hits with a cache of size xw is $\sum_{i=1}^{x} H(i)/m$. The hit rate curve at the desired p uniformly-spaced points is

$$C(xw) \;=\; \sum_{i=1}^{x} H(i)/m \qquad \forall x \in [p].$$

1.5 Statement of results

Our main algorithmic result is:

▶ **Theorem 1.** *There is an algorithm, parameterized by p and ϵ, that performs a single pass over the input $B \in [n]^m$ and produces an approximate hit rate curve $\hat{C}$ satisfying*

$$C\big((x-1)w\big) - \epsilon \;\leq\; \hat{C}(xw) \;\leq\; C(xw) + \epsilon \qquad \forall x \in [p+1], \tag{2}$$

where C is the true hit rate curve for B. The algorithm uses $O\big(p^2 \log(n) \log^2(m)/\epsilon^2\big)$ bits of space.

The accuracy guarantee of (2) is unusual in that it involves approximation in the domain (horizontal error) and in the range (vertical error) of the function. The following theorem shows that both horizontal and vertical error are necessary: as $\epsilon \to n^{-1/2}$ or $w \to n^{1/2}$, we have $s \to \Omega(n)$.

▶ **Theorem 2.** *Suppose there is an algorithm A that uses s bits of space and outputs a function $\hat{C}$ satisfying*

$$C\big((x-1)w\big) - \epsilon \;\leq\; \hat{C}(xw) \;\leq\; C(xw) + \epsilon \qquad \forall x \in [p+1] \tag{3}$$

where $n = pw$, $p \geq 3$ and $\epsilon \leq 1/5$. Then $s \geq \Omega(\min\big\{p^2 + 1/\epsilon^2 + \log(n), n\big\})$.

In particular, this result shows that other authors' claims of using "constant space" [28] cannot hold in a worst-case sense. As mentioned earlier, our use of additive vertical error in (2) is necessary. We show in Appendix C that any algorithm with multiplicative vertical error must use linear space.

2 Deterministic Algorithms for Hit Rate Curves

It is obvious from the definitions that the hit rate curve C can be computed exactly in polynomial time. It is not hard to see that it can be computed in $O(m \log n)$ time and $O(n)$ space using a balanced tree [5, 22, 1]. In this paper we are interested in approximations to C; in particular, we are only concerned with its value at p uniformly-spaced points. We begin with Algorithm 1 which computes those values. This algorithm can also be implemented in $O(m \log n)$ time and $O(n)$ space.

Algorithm 1: Algorithm for computing the hit rate curve at $p = n/w$ uniformly-spaced points.

```
1 Input: A sequence of requests $(b_1, \ldots, b_m) \in [n]^m$
2 Initialize the vector $H \in \mathbb{N}^p$ with zeros
3 for $t = 1, \ldots, m$ do
4     If $D(t)$ is finite then increment $H[\lceil D(t)/w \rceil]$ by 1
5 ▷ $H[i]$ satisfies condition (1).
6 Output the hit rate curve values $C(xw) = \sum_{i=1}^{x} H[i]/m$ for $x \in [p]$.
```

This paper considers streaming algorithms that access the request sequence B in a single pass. Such algorithms will not be able to compute the function D from scratch, and must update H using a compact data structure that represents D. We define an abstract data type called a *suffix-structure* to encapsulate that compact representation. A suffix-structure supports two operations, Register(t, b), which records that block b was requested at time t, and GetSuffixF0(t), which estimates the number of distinct blocks requested since time t. A trivial and inefficient implementation of a suffix-structure is shown in Algorithm 2.

Algorithm 2: A trivial suffix-structure.

```
1 $c \leftarrow 0$
2 Function Register($t, b_t$):
3     ▷ Assert $t = c + 1$
4     $A[t] \leftarrow b_t$
5     $c \leftarrow t$
6 Function GetSuffixF0($t$):
7     ▷ Assert $t \leq c$
8     Return $|\{A[t], A[t+1], \ldots, A[c]\}|$
```

Next we present Algorithm 3, our algorithm that uses a suffix-structure to approximate hit rate curves. All algorithms in this paper for computing hit rate curves are simply instantiations of Algorithm 3 that use different suffix-structures.

Algorithm 3: An algorithm for approximating the hit rate curve at p uniformly-spaced points, given an implementation $\mathcal{S}$ of a suffix-structure.

```
1  Input: A sequence of requests $(b_1, \ldots, b_m) \in [n]^m$
2  Initialize the vector $H \in \mathbb{N}^p$ with zeros
3  ▷ For convenience, let $\tau_i$ denote $(i-1)w + 1$
4  for $t = 1, \ldots, m$ do
5      ▷ Receive request $b_t$
6      $\mathcal{S}$.Register($t, b_t$)
7      Let $c \leftarrow \lceil t/w \rceil$
8      for $i = 1, \ldots, c$ do
9          Let $X_i(t+1) \leftarrow \mathcal{S}$.GetSuffixF0($\tau_i$)
10     for $i = 1, \ldots, c-1$ do
11         Increment $H[\lceil X_i(t)/w \rceil]$ by $\big(X_{i+1}(t+1) - X_{i+1}(t)\big) - \big(X_i(t+1) - X_i(t)\big)$
12     Increment $H[\lceil X_c(t)/w \rceil]$ by $1 - \big(X_c(t+1) - X_c(t)\big)$
13 Output the hit rate curve approximation given by $C(xw) = \sum_{i=1}^{x} H[i]/m$ for $x \in [p]$.
```

Consider executing Algorithm 3 using a trivial suffix-structure. We now show that its output differs from that of Algorithm 1 only by the presence of horizontal error.

▶ **Lemma 3.** *Let C be the hit rate curve computed by Algorithm 1. Let $\hat{C}$ be the hit rate curve computed by Algorithm 3 using a trivial suffix-structure. Then*

$$C\big((x-1)w\big) \;\leq\; \hat{C}(xw) \;\leq\; C(xw) \qquad \forall x \in [p].$$

Proof Sketch. First of all, note that line 9 in Algorithm 3 satisfies

$$X_i(t) \;=\; |B(\tau_i, t)| \quad \forall i, t. \tag{4}$$

So $X_i(t+1) - X_i(t)$ is 1 if $b_t \notin B(\tau_i, t)$ and otherwise zero. From this one can infer that the increment of line 11 is 1 precisely when the previous request for b_t occured at a time in

$[\tau_i, \tau_{i+1})$; otherwise the increment is zero. It follows that the update to H in Algorithm 3 is very similar to the update in Algorithm 1. ◀

A formal proof appears in Appendix A.

3 Suffix-structures using black-box F_0-estimators

In this section we design an improved suffix-structure using ideas from streaming algorithms. The main idea is to use F_0-estimators, which are probabilistic data structures supporting two operations. The INSERT operation takes a value $x \in [n]$ and the QUERY operation reports a value v satisfying

$$|S| \;\leq\; v \;\leq\; (1+\alpha)|S|, \tag{5}$$

where S is the set of elements that were inserted so far.

The improved suffix-structure, shown in Algorithm 4, periodically creates F_0-estimators, and inserts each new block into all existing F_0-estimators. Note that it only creates a new F_0-estimator at the times $\tau_i = (i-1)w + 1$ for $i \geq 1$, because Algorithm 3 only ever calls GETSUFFIXF0(τ_i) for some i.

Algorithm 4: An approximate suffix-structure, implemented using F_0-estimators.

```
1 $c \leftarrow 1$
2 Function Register($t, b_t$):
3     $c \leftarrow \lceil t/w \rceil$
4     if $t \equiv 1 \pmod w$ then
5         Create the new $F_0$-estimator $\mathcal{K}[c]$
6     for $i = 1, \ldots, c$ do
7         $\mathcal{K}[i]$.INSERT$(b_t)$
8 Function GetSuffixF0($t$):
9     Return $\mathcal{K}[\lceil t/w \rceil]$.QUERY()
```

We consider only F_0-estimators which satisfy the following simple properties.

- **Consistency:** Two consecutive calls to QUERY (without any intervening insertions) return the same value.
- **Idempotency:** Reinserting an item that was previous inserted does not change the value of QUERY.
- **Monotonicity:** The values returned by QUERY do not decrease as more elements are inserted.

There exist F_0-estimators, e.g. [17], that satisfy these properties, for which (5) holds with high probability for poly(m) queries, and which use $s :=$ poly$(1/\alpha, \log(nm))$ bits of space. We do not discuss the exact space usage here as our algorithm of Section 4 will achieve even better space usage.

We now analyze Algorithm 3 when executed with the approximate suffix-structure of Algorithm 4. Our aim is to show that its output is a good approximation to the true hit rate curve. We will use F_0-estimators with accuracy parameter $\alpha = \epsilon w/2n = \epsilon/2p$.

3.1 Accuracy

The following theorem compares the outputs of Algorithm 3 when executed with a trivial suffix-structure or an approximate suffix-structure.

▶ **Theorem 4.** *Let C be the hit rate curve produced by Algorithm 3 using a trivial suffix-structure. Let $\hat{C}$ be the hit rate curve produced using an approximate suffix-structure. Then*

$$C\big((x-1)w\big) - \epsilon \ \leq\ \hat{C}(xw) \ \leq\ C(xw) + \epsilon \qquad \forall x \in [p+1]. \tag{6}$$

▶ **Corollary 5.** *Let C^* denote the true hit rate curve. Let $\hat{C}$ refer to the hit rate curve produced from Algorithm 3 using an approximate suffix-structure, with $2p$ points instead of p. (That is, with $w' = w/2$ instead of w). Then C^* and $\hat{C}$ satisfy*

$$C^*\big((x-1)w\big) - \epsilon \ \leq\ \hat{C}(xw) \ \leq\ C^*(xw) + \epsilon \qquad \forall x \in [p+1].$$

This is the condition guaranteed by Theorem 1.

Proof. Combining (6), Lemma 3 and the definition of α yields

$$C^*\big((x-2)w'\big) - \epsilon \ \leq\ \hat{C}(xw') \ \leq\ C^*(xw') + \epsilon \qquad \forall x \in [2p+1].$$

Substituting $w/2$ for w' completes the proof. ◀

3.2 Space usage

After calling $\mathcal{S}.\text{REGISTER}(t, b_t)$ m times, the approximate suffix-structure will have created $\lceil m/w \rceil$ F_0-estimators, each of which uses s bits of space, so the total space usage is $O(ms/w)$ bits. This does not quite meet our goal of $\text{poly}(p, 1/\epsilon, \log(nm))$ bits. The algorithm can be improved to use only $O(ps/\epsilon)$ bits (while still using the F_0-estimators as a black box) but we do not discuss that improvement here, as the algorithm of Section 4 achieves even better space usage. Details of this improvement may be found in [11].

Proof of Theorem 4. Let H and X_i refer to the quantities using the trivial suffix-structure, and let $\hat{H}$ and $\hat{X}_i$ refer to the corresponding quantities using the approximate suffix-structure. We require the following proposition, which is proven in Appendix B.

▶ **Proposition 6.** *For any times $a \leq b$ and any index i, we have $X_i(a) - X_{i+1}(a) \geq X_i(b) - X_{i+1}(b)$.*

The histogram H and the hit rate curve C are obtained by summing contributions from each consecutive pair of cardinality values X_i and X_{i+1}. The same is true of $\hat{H}$ and $\hat{C}$, using instead the pair $\hat{X}_i$ and $\hat{X}_{i+1}$. So, to prove (6), we will show that the contribution from the pair $\hat{X}_i$ and $\hat{X}_{i+1}$ to $\hat{C}$ approximately equals the contribution from the pair X_i and X_{i+1} to C.

3.3 Contribution to C

Fix any $x \in [p]$ and recall that $C(xw) = \sum_{j=1}^{x} H[j]/m$. By considering lines 11 and 12 of Algorithm 3 we see that the pair X_i and X_{i+1} can only contribute to $C(xw)$ while $\lceil X_i(t)/w \rceil \leq x$. So, let T_i be the first time t at which $\lceil X_i(t)/w \rceil > x$, i.e.,

$$T_i \ =\ \min\{\, t \,:\, X_i(t) > xw \,\}\,.$$

At all times $t \geq T_i$, the pair X_i and X_{i+1} do not contribute to $C(x)$. The contribution to $m \cdot C(xw)$ from the pair X_i and X_{i+1} is

$$\begin{cases} X_{i+1}(t+1) - X_{i+1}(t) - X_i(t+1) + X_i(t) & (\text{for } t \in \{\tau_{i+1}, \ldots, T_i - 1\}) \\ 1 - X_i(t+1) + X_i(t) & (\text{for } t \in \{\tau_i, \ldots, \tau_{i+1} - 1\}). \end{cases}$$

Summing up, the total contribution is

$$\begin{aligned} &\sum_{\tau_i \leq t < \tau_{i+1}-1} \big(1 - X_i(t+1) + X_i(t)\big) \; + \\ &\sum_{\tau_{i+1} \leq t < T_i} \big(X_{i+1}(t{+}1) - X_{i+1}(t) - X_i(t{+}1) + X_i(t)\big) \\ &\quad = \; w - X_i(\tau_{i+1}) + X_i(\tau_i) + X_i(\tau_{i+1}) - X_{i+1}(\tau_{i+1}) + X_{i+1}(T_i) - X_i(T_i) \\ &\quad = \; w + X_{i+1}(T_i) - X_i(T_i). \qquad (7) \end{aligned}$$

3.4 Contribution to $\hat{C}$

Similarly, let $\hat{T}_i = \min\{\, t \,:\, \hat{X}_i(t) > xw \,\}$. Then at all times $t \geq \hat{T}_i$, the pair $\hat{X}_i$ and $\hat{X}_{i+1}$ do not contribute to $\hat{C}(x)$. (This assertion uses the Monotonicity property.) Summing up as before, the total contribution of the pair $\hat{X}_i$ and $\hat{X}_{i+1}$ to $m \cdot \hat{C}(xw)$ is

$$w + \hat{X}_{i+1}(\hat{T}_i) - \hat{X}_i(\hat{T}_i). \qquad (8)$$

3.4.1 Upper bound on contribution to $\hat{C}(xw)$

The difference between the contribution of $\hat{X}_i$ and $\hat{X}_{i+1}$ to $m \cdot \hat{C}(xw)$ and the contribution of X_i and X_{i+1} to $m \cdot C(xw)$ is the difference between (8) and (7), namely

$$\hat{X}_{i+1}(\hat{T}_i) - \hat{X}_i(\hat{T}_i) - X_{i+1}(T_i) + X_i(T_i). \qquad (9)$$

We now upper bound this quantity. First note that $\hat{T}_i \leq T_i$, by (5). Then Proposition 6 shows that (9) is at most

$$\begin{aligned} &\hat{X}_{i+1}(\hat{T}_i) - \hat{X}_i(\hat{T}_i) - X_{i+1}(\hat{T}_i) + X_i(\hat{T}_i) \\ &\leq \; \alpha X_{i+1}(\hat{T}_i) \quad (\text{by } (5)) \\ &\leq \; \alpha X_i(\hat{T}_i) \quad (\text{by definition of } X_i \text{ and } X_{i+1}) \\ &\leq \; \alpha(xw+1) \quad (\text{since } \hat{T}_i \leq T_i \text{ and by definition of } T_i). \qquad (10) \end{aligned}$$

3.4.2 Lower bound on contribution to $\hat{C}(xw)$

For the lower bound, we must consider the contribution of X_i and X_{i+1} to $C((x{-}1)w)$. Define $T'_i = \min\{\, t \,:\, X_i(t) > (x-1)w \,\}$. Arguing as before, we get that the total contribution of this pair to $m \cdot C((x-1)w)$ is

$$w + X_{i+1}(T'_i) - X_i(T'_i). \qquad (11)$$

The difference between (8) and (11) is

$$\hat{X}_{i+1}(\hat{T}_i) - \hat{X}_i(\hat{T}_i) - X_{i+1}(T'_i) + X_i(T'_i). \qquad (12)$$

We now claim that $T'_i < \hat{T}_i$. By definition of T'_i, we have $X_i(T'_i) = (x-1)w+1$. By definition of α, we have $\alpha n = \epsilon w < w$. So, by (5),

$$\begin{aligned}\hat{X}_i(T'_i) \;&\leq\; (1+\alpha)X_i(T'_i) \;\leq\; (1+\alpha)\big((x-1)w+1\big)\\ &\leq\; (x-1)w+1+\alpha n \;<\; xw+1 \;\leq\; \hat{X}_i(\hat{T}_i).\end{aligned}$$

By the Monotonicity property, the claim is proven. So we may use Proposition 6 to show that (12) is at least

$$\begin{aligned}&\hat{X}_{i+1}(\hat{T}_i) - \hat{X}_i(\hat{T}_i) - X_{i+1}(\hat{T}_i) + X_i(\hat{T}_i)\\ &\quad\geq\; -\alpha X_i(\hat{T}_i) \quad \text{(by (5))}\\ &\quad\geq\; -\alpha(xw+1) \quad (\text{since } \hat{T}_i \leq T_i \text{ and by definition of } T_i). \qquad (13)\end{aligned}$$

3.5 Proof of (6)

We now combine our previous observations to establish (6). Recall that $c = \lceil m/w \rceil$ is the total number of F_0-estimators. Summing (10) over all i, we obtain that

$$m\hat{C}(xw) \;\leq\; mC(xw) + \alpha c(xw+1) \;\leq\; mC(xw) + 2\alpha m x.$$

This proves the second inequality of (6). The first inequality of (6) follows analogously from (13). ◀

4 Suffix-structures using a timestamped F_0-estimator

As mentioned in Section 1, the idea of *sliding window estimators* [9] is very relevant for estimating properties of the suffixes of a stream. One of the main ideas is to modify known streaming estimators by incorporating *timestamps*. By restricting the estimator's data structure to entries with sufficiently large timestamps, one can construct an estimator for the desired suffix of the stream. Let us now discuss that idea for the special case of F_0-estimators [9, §7.5].

Many F_0-estimators rely on a $\{0,1\}$-matrix M that is continously updated while processing the stream [2, 14, 3, 17]. Each item b in the stream is hashed to a binary string σ, and then M is updated based on $\text{lsb}(\sigma)$, the number of trailing zeros in σ. We will call such a matrix M a *bitmatrix*.

The simplest F_0-estimator [2] uses a single hash function h, and the matrix M has a single column. At any point during stream processing, M_j is set to 1 if a block b was observed with $\text{lsb}(h(b)) \geq j$. After the stream is processed, the algorithm outputs 2^{j^*}, where j^* is the greatest index of a non-zero row. This gives only a $O(1)$ approximation. Other algorithms refine this estimate by using additional columns and another hash function g, which determines which column to update. The estimate could be, for example, a function of the average of the lowest non-zero value in each column [12, 14], or the number of non-zero elements below a certain row (Algorithm 3 in [3]). All of these algorithms can boost their success probability by taking medians of independent, parallel instantiations.

Suppose that Algorithm 4 uses an F_0-estimator of this type, and that all instantiations of that estimator use the same hash functions h and g. Let M^j denote the bitmatrix corresponding to the jth F_0-estimator. For any $j \leq j'$, M^j has undergone all of the updates that $M^{j'}$ has, so it follows that $M^j \geq M^{j'}$ in an entrywise sense. This observation leads to following idea: instead of storing the bitmatrices for each estimator separately, we can store a single unified matrix from which all bitmatrices can be computed.

Algorithm 5: An implementation of a suffix-structure based on a timestamped F_0-estimator. We consider a F_0-estimator that is based on a bitmatrix as described above. The algorithm $\mathcal{A}$ specifies how the bitmatrix is updated on an INSERT operation, and how to produce the estimate for the QUERY operation. The set $\mathcal{Q}$ has many independent copies of the hash functions and the resulting table. We need only $|\mathcal{Q}| = O(\log(1/\delta))$ with $\delta = m^{-3}$.

```
   Data: A collection $\mathcal{Q}$ of pairs $(Q, \mathcal{H})$, where $Q$ is a matrix, $\mathcal{H}$ is a set of hash functions
         An algorithm $\mathcal{A}$ for estimating $F_0$ from a bitmatrix
1  $c \leftarrow 1$
2  Function Register($t, b_t$):
3      $c \leftarrow \lceil t/w \rceil$
4      for $(Q, \mathcal{H}) \in \mathcal{Q}$ do
5          Update $Q$ using $b_t$ according to $\mathcal{A}$
6  Function GetSuffixF0($t'$):
7      for $Q \in \mathcal{Q}$ do
8          Let $r = \lceil t'/w \rceil$
9          Define the bitmatrix $M^r$ by $M^r_{i,j} = \begin{cases} 1 & \text{if } Q_{i,j} - r \geq 0 \\ 0 & \text{otherwise} \end{cases}$
10         Feed $M^r$ into algorithm $\mathcal{A}$ to obtain estimate $R_Q$
11     Return the median of estimates $R_Q$
```

4.1 Analysis

In order to analyze Algorithm 5, we must specify $\mathcal{A}$, a concrete F_0-estimator. We will use Algorithm 2 from the paper of Bar-Yossef et al. [3].[1] In this algorithm, each matrix Q has $\log(n)$ rows, and $k = O(1/\alpha^2)$ columns. Each collection $\mathcal{H}$ consists of k t-wise independent hash functions, where t is $O\big(\log(1/\alpha)\big)$. To update Q, for $j \in [k]$, we set $Q_{i,j} = c$ if $\text{lsb}(h_j(b_t)) \geq i$. Given the bitmatrix M, the F_0-estimator can produce its estimate.

▶ **Proposition 7.** *The space used by Algorithm 5 is $O\big(p^2 \log(n) \log(m) \log(1/\delta)/\epsilon^2\big)$ bits.*

Proof. Each Q has $O(1/\alpha^2)$ columns, $\log n$ rows, and each cell requires $\log m$ bits space. Thus Q requires $O\big(\log(n)\log(m)/\alpha^2\big)$ bits of space. Each collection $\mathcal{H}$ requires only $O\big(\log^2(1/\alpha)\log n\big)$ bits of space, which is negligible. We have $\log(1/\delta)$ such pairs $(Q, \mathcal{H})$. Thus the total space requirement is $O\big(\alpha^{-2}\log(n)\log(m)\log(1/\delta)\big)$. Substituting $\alpha = \epsilon/p$ completes the proof. ◀

▶ **Theorem 8.** *Algorithm 3 using Algorithm 5 as its suffix-structure satisfies* (2).

Proof. Let $X_i(t)$ be the result of GETSUFFIXF0(τ_i) at time t, which is an estimate of $|B(\tau_i, t)|$. It suffices to show that $X_i(t) \in [1, 1+\alpha] \cdot |B(\tau_i, t)|$ with high probability, in which case the argument of Theorem 1 applies.

[1] It is natural to wonder why we do not use the optimal algorithm of Kane et al. [17]. The reason is that the Kane et al. algorithm is an enhancement of the Bar-Yossef et al. algorithm that manages to save some additional space. In contrast, our algorithm will consume extra space by adding timestamps, which ruins the space-saving enhancements of Kane et al.

For any i, t, by taking the median of $\log(1/\delta)$ estimates of $\mathcal{A}$, we have $\Pr[|X_i(t) - |B(\tau_i, t)|| > \alpha|B(\tau_i, t)|] \leq \delta$. At time t, the algorithm computes estimates for t/w F_0-estimators, and thus $O(m^2)$ estimates are taken in total. By a union bound, the probability that any estimate is poor is $\leq \delta m^2$. ◀

Combining Proposition 7 and Theorem 8 and taking $\delta = 1/m^3$ proves Theorem 1.

5 Discussion

In this paper we have presented the first single-pass, sublinear space algorithm with provable guarantees for estimating hit rate curves for the LRU caching policy. The space usage is $O(p^2 \log(n) \log^2(m)/\epsilon^2)$ bits. This space usage is not far from optimal, due to our $\Omega(p^2 + \epsilon^{-2} + \log(n))$ lower bound. A practical implementation of this algorithm has been deployed in an enterprise storage system [29].

As this is the first theoretical paper on hit rate curve computation, it suggests several directions for further algorithmic research.

- It would be nice to improve either our upper bound or lower bound on the space usage. Our suspicion is that the lower bounds can be improved.
- In practice the runtime of our algorithm is very good [29], but we have not studied the runtime from a theoretical perspective. In particular, optimizing the runtime of Algorithm 5 seems quite interesting.
- Instead of approximating the hit rate curve at p *uniformly*-space points, it is natural to wonder whether the p points can be adaptively chosen during the algorithm.
- It would be interesting to extend our techniques beyond just the LRU caching policy. The algorithm of Mattson et al. [19] works for all policies that satisfy the inclusion property – is there a single-pass streaming algorithm for all such policies?
- A key operation in our algorithm is to take the difference of F_0-estimators. (In fact, we estimate $|A \setminus B|$ where $B \subseteq A$.) There are F_0-estimators that have been explicitly designed for this purpose, e.g., [15]. It would be interesting to study whether such specialized F_0-estimators can improve our algorithm, either theoretically or practically.

Acknowledgements. We thank David Woodruff for a helpful discussion about distinct element estimators. We thank Andrew McGregor and the anonymous reviewers for helpful comments.

References

1 George S. Almási, Călin Caşcaval, and David A. Padua. Calculating stack distances efficiently. In *Proceedings of the 2002 workshop on memory system performance (MSP'02)*, pages 37–43, 2002.

2 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29. ACM, 1996.

3 Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Randomization and Approximation Techniques in Computer Science*, pages 1–10. Springer, 2002.

4 L. A. Belady. A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal*, 5(2):78–101, 1966.

5 Brian T Bennett and Vincent J. Kruskal. LRU stack processing. *IBM Journal of Research and Development*, 19(4):353–357, 1975.

6 Hjortur Bjornsson, Gregory Chockler, Trausti Saemundsson, and Ymir Vigfusson. Dynamic performance profiling of cloud caches. In *Proceedings of the 4th annual Symposium on Cloud Computing (SoCC)*. ACM, 2013.

7 Vladimir Braverman and Rafail Ostrovsky. Smooth histograms for sliding windows. In *Foundations of Computer Science, 2007. Proceedings. 48th Annual IEEE Symposium on*, pages 283–293. IEEE, 2007.

8 Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. *SIAM Journal on Computing*, 41(5):1299–1317, 2012.

9 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002.

10 Chen Ding and Yutao Zhong. Predicting whole-program locality through reuse distance analysis. In *PLDI*, pages 245–257. ACM, 2003.

11 Zachary Drudi. A streaming algorithms approach to approximating hit rate curves. Master's thesis, University of British Columbia, 2014.

12 Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities. In *Algorithms-ESA 2003*, pages 605–617. Springer, 2003.

13 David Eklov and Erik Hagersten. StatStack: Efficient modeling of LRU caches. In *Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on*, pages 55–65. IEEE, 2010.

14 Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. *DMTCS Proceedings*, 0(1), 2008.

15 Sumit Ganguly, Minos Garofalakis, and Rajeev Rastogi. Tracking set-expression cardinalities over continuous update streams. *The VLDB Journal*, 13(4):354–369, 2004.

16 Stephen T. Jones, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Geiger: Monitoring the buffer cache in a virtual machine environment. In *ASPLOS*, pages 14–24. ACM, 2006.

17 Daniel M Kane, Jelani Nelson, and David P Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 41–52. ACM, 2010.

18 E Kushilevitz and N Nisan. Communication complexity, 1997.

19 Richard L. Mattson, Jan Gecsei, Donald R. Slutz, and Irving L. Traiger. Evaluation techniques for storage hierarchies. *IBM Systems Journal*, 9(2):78–117, 1970.

20 Nimrod Megiddo and Dharmendra S Modha. ARC: A self-tuning, low overhead replacement cache. In *FAST*, volume 3, pages 115–130, 2003.

21 Qingpeng Niu, James Dinan, Qingda Lu, and P Sadayappan. Parda: A fast parallel reuse distance analysis algorithm. In *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 1284–1294. IEEE, 2012.

22 Frank Olken. Efficient methods for calculating the success function of fixed space replacement policies. Master's thesis, University of California, Berkeley, 1981.

23 Alexander A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.

24 Xipeng Shen, Yutao Zhong, and Chen Ding. Locality phase prediction. In *ASPLOS*, pages 165–176. ACM, 2004.

25 Alan Jay Smith. Two methods for the efficient analysis of memory address trace data. *Software Engineering, IEEE Transactions on*, 3(1):94–101, 1977.

26 Gokul Soundararajan, Daniel Lupei, Saeed Ghanbari, Adrian Daniel Popescu, Jin Chen, and Cristiana Amza. Dynamic resource allocation for database servers running on virtual storage. In *FAST*. USENIX, 2009.

27 Harold S Stone, John Turek, and Joel L. Wolf. Optimal partitioning of cache memory. *Computers, IEEE Transactions on*, 41(9):1054–1068, 1992.

28 Carl A. Waldspurger, Nohhyun Park, Alexander Garthwaite, and Irfan Ahmad. Efficient MRC construction with SHARDS. In *Proceedings of the 13th USENIX Conference on File and Storage Technologies (FAST'15)*, pages 95–110. USENIX, 2015.

29 Jake Wires, Stephen Ingram, Zachary Drudi, Nicholas J. A. Harvey, and Andrew Warfield. Characterizing storage workloads with counter stacks. In *OSDI*, 2014.

30 Ting Yang, Emery D. Berger, Scott F. Kaplan, and J. Eliot B. Moss. CRAMM: Virtual memory support for garbage-collected applications. In *OSDI*, pages 103–116. ACM, 2006.

31 Yutao Zhong, Maksim Orlovich, Xipeng Shen, and Chen Ding. Array regrouping and structure splitting using whole-program reference affinity. In *PLDI*, pages 255–266. ACM, 2004.

32 Pin Zhou, Vivek Pandey, Jagadeesan Sundaresan, Anand Raghuraman, Yuanyuan Zhou, and Sanjeev Kumar. Dynamic tracking of page miss ratio curve for memory management. In *ASPLOS*, pages 177–188. ACM, 2004.

A Proofs from Section 2

Proof.]Proof of Lemma 3] Let H and $\hat{H}$ respectively denote the histograms computed by Algorithms 1 and 3. Note that $b_t \notin B(t',t)$ for $t' > R(t)$ but $b_t \in B(t',t)$ for $t' \leq R(t)$. Because of (4), we have

$$X_i(t{+}1) - X_i(t) \;=\; \begin{cases} 1 & (\text{if } R(t) < \tau_i \leq t) \\ 0 & (\text{if } 1 \leq \tau_i \leq R(t)) \end{cases}.$$

It follows that the increment in line 11 equals 1 if $\tau_i \leq R(t) < \tau_{i+1}$ and otherwise it equals zero. Similarly, the increment in line 12 equals 1 if $\tau_c \leq R(t)$. At most one of these conditions can hold, so for each value of t, Algorithm 3 increments at most one entry of $\hat{H}$. Specifically, if $R(t)$ is finite then the algorithm increments $\hat{H}[\lceil X_{i^*}(t)/w \rceil]$ where $i^* = \lceil R(t)/w \rceil$.

When $R(t)$ is finite, we have $\tau_{i^*} < R(t) \leq \tau_{i^*+1}$. Since $X_{i^*}(t) = |B(\tau_{i^*}, t)|$ and $D(t) = |B(R(t), t)|$, we derive $X_{i^*+1}(t) \leq D(t) \leq X_{i^*}(t)$. However,

$$\begin{aligned} X_{i^*}(t) - X_{i^*+1}(t) \;&=\; |B(\tau_{i^*}, t)| - |B(\tau_{i^*+1}, t)| \;=\; |B(\tau_{i^*}, t) \setminus B(\tau_{i^*+1}, t)| \\ &\leq\; |B(\tau_{i^*}, \tau_{i^*+1})| \;\leq\; w. \end{aligned}$$

So

$$\left\lceil \frac{X_{i^*}(t)}{w} \right\rceil - 1 \;\leq\; \left\lceil \frac{D(t)}{w} \right\rceil \;\leq\; \left\lceil \frac{X_{i^*}(t)}{w} \right\rceil.$$

Algorithm 1 increments $H[\lceil D(t)/w \rceil]$, whereas Algorithm 3 increments $\hat{H}[\lceil X_{i^*}(t)/w \rceil]$. So

$$\underbrace{\sum_{i=1}^{x} \hat{H}[i]}_{\hat{C}(xw)} \;\leq\; \underbrace{\sum_{i=1}^{x} H[i]}_{C(xw)} \;\leq\; \underbrace{\sum_{i=1}^{x+1} \hat{H}[i]}_{\hat{C}\left((x+1)w\right)} .$$

Rearranging this yields the desired inequality. ◀

B Proofs from Section 3

Proposition 6. *For any times $a \leq b$ and any index i, we have $X_i(a) - X_{i+1}(a) \geq X_i(b) - X_{i+1}(b)$.*

Proof. Recall that $X_i(t) = |B(\tau_i, t)|$. As $\tau_i < \tau_{i+1}$, we get $X_i(t) - X_{i+1}(t) = |B(\tau_i, \tau_{i+1}) \setminus B(\tau_{i+1}, t)|$. Thus

$$\begin{aligned} & X_i(a) - X_{i+1}(a) - (X_i(b) - X_{i+1}(b)) \\ & \quad = |B(\tau_i, \tau_{i+1}) \setminus B(\tau_{i+1}, a)| - |B(\tau_i, \tau_{i+1}) \setminus B(\tau_{i+1}, b)| \\ & \quad \geq 0, \end{aligned}$$

as $B(\tau_{i+1}, a) \subset B(\tau_{i+1}, b)$. ◀

C Lower Bounds

In this section we prove lower bounds on the space needed by one-pass algorithms to compute approximate hit rate curves. As is typical with streaming algorithms, our lower bounds are based on communication complexity [18].

C.1 Lower Bounds for Hit Rate Curve Estimation

Our lower bounds are based on reductions from the Gap Hamming Distance (GHD) problem. In $\mathrm{GHD}_{q,t,g}$, Alice and Bob are respectively given vectors $x, y \in \{0,1\}^q$. They are required to determine whether the Hamming distance between x and y, denoted $d(x, y)$, is less than $t - g$ or greater than $t + g$, outputting 0 or 1, respectively. The following optimal lower bound for GHD is known.

▶ **Theorem 9** (Chakrabarti-Regev [8]). *Any protocol that solves $GHD_{q,q/2,g}$ with probability $\geq 2/3$ communicates $\Omega(\min\{q, q^2/g^2\})$ bits.*

Proof of Theorem 2. Let $c = 10$ and $q = n/(c+2)$. Consider an instance of $\mathrm{GHD}_{q,q/2,g}$, where g will be specified later. Alice has $x \in \{0,1\}^q$ and Bob has $y \in \{0,1\}^q$. Let us say that $\mathrm{GHD}(x, y) = 0$ if $d(x, y) < q/2 - g$, and $\mathrm{GHD}(x, y) = 1$ if $d(x, y) > q/2 + g$.

Alice and Bob produce an input stream to the algorithm A as follows. Each stream element is a member of $[n]$. The elements in $[cq] \subset [n]$ are called "type-1", and those in $[n] \setminus [cq]$ are called "type-2". Alice first provides to A the type-1 elements, then certain type-2 elements. Specifically, she provides the sequence $(1, 2, \ldots, cq)$, then provides $cq + j$ if $x_j = 1$ and $(c+1)q + j$ if $x_j = 0$, for $j \in [q]$. She then communicates A's state to Bob, which requires s bits of communication. Bob first provides to A certain type-2 elements, then the type-1 elements. Specifically, he provides $cq + j$ if $y_j = 0$ and $(c+1)q + j$ if $y_j = 1$, then provides the sequence $(1, 2, \ldots, cq)$. The total length of the stream provided to A is exactly $m = 2(c+1)q$.

Observation 1: The number of type-2 elements that occur in the stream is exactly $2q$. The number that occur twice is exactly $d(x, y)$. Hence, the number of *distinct* type-2 elements that occur in the stream is exactly $2q - d(x, y)$.

Observation 2: Every type-1 element appears exactly twice in the stream. The number of distinct elements that occur between those two occurrences is exactly $cq + 2q - d(x, y)$. Depending on whether $\mathrm{GHD}(x, y)$ is 0 or 1, we have

$$\begin{aligned} \text{If } \mathrm{GHD}(x,y) = 0: &\quad cq + 2q - d(x,y) > (c + 3/2)q + g =: H \\ \text{If } \mathrm{GHD}(x,y) = 1: &\quad cq + 2q - d(x,y) < (c + 3/2)q - g =: L \end{aligned}$$

The only other requests that possibly contribute to C are Bob's type-2 elements, of which there are only q. (Alice's inputs contribute nothing to C.)

Case 1: $w \le \epsilon n$. Equivalently, $p \ge 1/\epsilon$. In this case we take $g = w$. Let $\beta = (c+3/2)q/w$. Then

$$L = (\beta - 1)w \qquad \text{and} \qquad \beta w < H. \tag{14}$$

Suppose that $\hat{C}$ satisfies (3). If $\mathrm{GHD}(x,y) = 1$ then

$$\begin{aligned}
\hat{C}(\beta w) &\ge C\big((\beta-1)w\big) - \epsilon \qquad \text{(by (3))}\\
&= C(L) - \epsilon \qquad \text{(by (14))}\\
&\ge \frac{cq}{m} - \epsilon \qquad \text{(by Observation 2)}\\
&\ge \frac{c}{2(c+1)} - \frac{1}{5} > \frac{1}{4} \qquad (c \ge 10 \text{ and } \epsilon \le 1/5).
\end{aligned}$$

On the other hand, if $\mathrm{GHD}(x,y) = 0$ then

$$\begin{aligned}
\hat{C}(\beta w) &\le C(\beta w) + \epsilon \qquad \text{(by (3))}\\
&\le C(H) + \epsilon \qquad \text{(by (14))}\\
&\le \frac{q}{m} + \epsilon \qquad \text{(by Observation 2)}\\
&\le \frac{1}{2(c+1)} + \frac{1}{5} < \frac{1}{4} \qquad (c \ge 10 \text{ and } \epsilon \le 1/5).
\end{aligned}$$

Therefore Alice and Bob can distinguish whether $\mathrm{GHD}(x,y)$ is 0 or 1. The number of bits of space necessary is $\Omega(\min\{q, q^2/g^2\}) = \Omega(\min\{n, p^2\})$.

Case 2: $w > \epsilon n$. Equivalently, $1/\epsilon > p$. In this case we take $g = 22\epsilon q$. Set $\beta = 1 + \lceil p/(c+2) \rceil$. Then

$$\begin{aligned}
(\beta-1)w &\ge \frac{p}{c+2} w = q\\
\beta w &< (2 + \tfrac{p}{c+2})w = (\tfrac{2}{p} + \tfrac{1}{c+2})n \le (\tfrac{2}{3} + \tfrac{1}{c+2})n < \tfrac{c}{c+2} n = cq.
\end{aligned}$$

By observation 2, the type-1 elements do not contribute to $C(cq)$. So consider any type-2 element. If it appears twice, then the number of distinct elements between those two appearances is at most q. By observation 1, the number of type-2 elements that appear twice is exactly $d(x,y)$. It follows that $C(cq) = C(q) = d(x,y)/m$. So, if $\hat{C}$ satisfies (3), we have

$$\frac{d(x,y)}{m} - \epsilon \le C(q) - \epsilon \le \hat{C}(\beta w) \le C(cq) + \epsilon = \frac{d(x,y)}{m} + \epsilon.$$

Thus

$$|m \cdot \hat{C}(\beta w) - d(x,y)| \le m\epsilon = 2(c+1)q\epsilon < g.$$

It follows that Alice and Bob can distinguish whether $\mathrm{GHD}(x,y)$ is 0 or 1. The number of bits of space necessary is $\Omega(\min\{q, q^2/g^2\}) = \Omega(\min\{n, 1/\epsilon^2\})$.

The lower bound of $\Omega(\log n)$ is left as an easy exercise. ◀

C.2 Impossibility of Multiplicative Error

Lastly, we show that any algorithm with multiplicative vertical error must use linear space, even if it also has additive horizontal error.

▶ **Theorem 10.** *Let $n = pw$ where $w \geq 1$ is arbitrary. Let $\epsilon \in [0, 1)$ be arbitrary. Suppose there is a single-pass algorithm A that uses s bits of space and outputs a function $\hat{C}$ satisfying*

$$(1-\epsilon) \cdot C\big((i-1)w\big) \;\leq\; \hat{C}(iw) \;\leq\; (1+\epsilon) \cdot C(iw) \qquad \forall i \in [p+1]. \tag{15}$$

Then $s = \Omega(n)$.

Proof. The disjointness problem DISJ : $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is defined to be $\mathrm{DISJ}(x,y) = \prod_i (1 - x_i y_i)$. Razborov's lower bound [23, 18] implies that, even under the promise that $\sum_i x_i = \sum_i y_i = (n+1)/4$ and $\sum_i x_i y_i \in \{0,1\}$, any randomized communication protocol that can decide DISJ must use $\Omega(n)$ bits of communication.

C.2.1 Reduction

Alice and Bob produce an input stream to the algorithm A as follows. Each stream element is a member of $[n]$. Alice provides to A the set $\{\, i \in [n] : x_i = 1 \,\}$ in any order. She then communicates A's state to Bob, which requires s bits. Bob provides to A the set $\{\, i \in [n] : y_i = 1 \,\}$ in any order. The total length of the stream provided to A is exactly $m = (n+1)/2$.

If $\mathrm{DISJ}(x,y) = 1$ then every stream element is distinct, so $C(n) = 0$. On the other hand, if $\mathrm{DISJ}(x,y) = 0$ then the promise ensures that $C(n) = 1/m$. Let us apply (15) with $i = p+1$, and recall from the definition of C that $C(pw) = C\big((p+1)w\big)$. It follows that Alice and Bob can decide $\mathrm{DISJ}(x,y)$, so Razborov's result implies that $s = \Omega(n)$. ◀

Terminal Embeddings

Michael Elkin, Arnold Filtser, and Ofer Neiman

Department of Computer Science, Ben-Gurion University of the Negev
Beer-Sheva, Israel
{elkinm,arnoldf,neimano}@cs.bgu.ac.il

Abstract

In this paper we study *terminal embeddings*, in which one is given a finite metric (X, d_X) (or a graph $G = (V, E)$) and a subset $K \subseteq X$ of its points are designated as *terminals*. The objective is to embed the metric into a normed space, while approximately preserving all distances among pairs that contain a terminal. We devise such embeddings in various settings, and conclude that even though we have to preserve $\approx |K| \cdot |X|$ pairs, the distortion depends only on $|K|$, rather than on $|X|$.

We also strengthen this notion, and consider embeddings that approximately preserve the distances between *all* pairs, but provide improved distortion for pairs containing a terminal. Surprisingly, we show that such embeddings exist in many settings, and have optimal distortion bounds both with respect to $X \times X$ and with respect to $K \times X$.

Moreover, our embeddings have implications to the areas of Approximation and Online Algorithms. In particular, [7] devised an $\tilde{O}(\sqrt{\log r})$-approximation algorithm for sparsest-cut instances with r demands. Building on their framework, we provide an $\tilde{O}(\sqrt{\log |K|})$-approximation for sparsest-cut instances in which each demand is incident on one of the vertices of K (aka, terminals). Since $|K| \leq r$, our bound generalizes that of [7].

1998 ACM Subject Classification G.2.2 Graph algorithms

Keywords and phrases embedding, distortion, terminals

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.242

1 Introduction

Embedding of finite metric spaces is a very successful area of research, due to both its algorithmic applications and its natural geometric appeal. Given two metric space (X, d_X), (Y, d_Y), we say that X embeds into Y with distortion α if there is a map $f : X \to Y$ and a constant $c > 0$, such that for all $u, v \in X$,

$$d_X(u, v) \leq c \cdot d_Y(f(u), f(v)) \leq \alpha \cdot d_X(u, v) \ .$$

Some of the basic results in the field of metric embedding are: a theorem of [15], asserting that any metric space on n points embeds with distortion $O(\log n)$ into Euclidean space (which was shown to be tight by [35]), and probabilistic embedding into a distribution of ultrametrics (or trees) with expected distortion $O(\log n)$ [23], or expected congestion $O(\log n)$ [40] (which are also tight [12]).

In this paper we study a natural variant of embedding, in which the input consists of a finite metric space or a graph, and in addition, a subset of the points are designated as *terminals*. The objective is to embed the metric into a simpler metric (e.g., Euclidean metric), or into a simpler graph (e.g., a tree), while approximately preserving the distances between the terminals to *all other points*. We show that such embeddings, which we call *terminal embeddings*, can have improved parameters compared to embeddings that must preserve all pairwise distances. In particular, the distortion (and the dimension in embedding to normed spaces) depends only on the number of terminals, regardless of the cardinality of the metric space.

We also consider a strengthening of this notion, which we call *strong terminal embedding*. Here we want a distortion bound on *all pairs*, and in addition an improved distortion bound on pairs that contain a terminal. Such strong terminal embeddings enhance the classical embedding results, essentially saying that one can obtain the same distortion for all pairs, with the option to select some of the points, and obtain improved approximation of the distances between any selected point to any other point.

As a possible motivation for studying such embeddings, consider a scenario in which a certain network of clients and servers is given as a weighted graph (where edges correspond to links, weights to communication/travel time). It is conceivable that one only cares about distances between clients and servers, and that there are few servers. We would like to have a simple structure, such as a tree spanning the network, so that the client-server distances in the tree are approximately preserved.

We show that there exists a general phenomenon; essentially any known metric embedding into an ℓ_p space or a graph family can be transformed via a general transformation into a terminal embedding, while paying only a constant blow-up in the distortion. In particular, we obtain a terminal embedding of any finite metric into any ℓ_p space with terminal distortion $O(\log k)$, using only $O(\log k)$ dimensions. We also show that many of the embeddings into normed spaces, probabilistic embedding into ultrametrics (including capacity preserving ones), and into spanning trees, have their *strong* terminal embedding counterparts. Our results are tight in most settings.[1]

It is well known that embedding a graph into a single tree may cause (worst-case) distortion $\Omega(n)$ [38]. However, if one only cares about client-server distances, we show that it is possible to obtain distortion $2k-1$, where k is the number of servers, and that this is tight. Furthermore, we study possible tradeoffs between the distortion and the total weight of the obtained tree. This generalizes the notion of *shallow light trees* [32, 11, 21], which provides a tradeoff between the distortion with respect to a single designated server and the weight of the tree.

We then address probabilistic approximation of metric spaces and graphs by ultrametrics and spanning trees. This line of work started with the results of [4, 12], and culminated in the $O(\log n)$ expected distortion for ultrametrics by [23], and $\tilde{O}(\log n)$ for spanning trees by [3]. These embeddings found numerous algorithmic applications, in various settings, see [23, 19, 3] and the references therein for details. In their work on Ramsey partitions, [36] implicitly showed that there exists a probabilistic embedding into ultrametrics with expected terminal distortion $O(\log k)$ (see Section 2 for the formal definitions). Here we generalize this result by obtaining a *strong* terminal embedding with the same expected $O(\log k)$ distortion guarantee for all pairs containing a terminal, and $O(\log n)$ for all other pairs. We also show a similar result that extends the embedding of [3] into spanning trees, with $\tilde{O}(\log k)$ expected distortion for pairs containing a terminal, and $\tilde{O}(\log n)$ for all pairs. A slightly different notion, introduced by [39], is that of trees which approximate the congestion (rather than the distortion), and [40] showed a distribution over trees with expected congestion $O(\log n)$. We provide a strong terminal version of this result, and show expected congestion of $O(\log k)$ for all edges incident on a terminal, and $O(\log n)$ for the rest. In [1], it was shown that the

[1] All our terminal embeddings are tight, except for the probabilistic spanning trees, where they match the state-of-the-art [3], and except for our terminal spanners.

average distortion (taken over all pairs) in an embedding into a single tree can be bounded by $O(1)$ (in contrast to the $\Omega(\log n)$ lower bound for the average stretch over edges). Here we extend and simplify their result, and obtain $O(1)$ average terminal distortion, that is, the average is over pairs containing a terminal. We do this both in the ultrametric and in the spanning tree settings.

We also consider spanners, with a stretch requirement only for pairs containing a terminal. Our general transformation produces for any $t \geq 1$ a $(4t-1)$-terminal stretch spanner with $O(k^{1+1/t}+n)$ edges. The drawback is that this is a metric spanner, not a subgraph of the input graph. We alleviate this issue by constructing a graph spanner with the same stretch and $O(\sqrt{n} \cdot k^{1+1/t}+n)$ edges.[2] A result of [41] implicitly provides a terminal graph spanner with $(2t-1)$ stretch and $O(t \cdot n \cdot k^{1/t})$ edges. Our graph terminal spanner is sparser than that of [41] as long as $k \leq t \cdot n^{1/2(1+1/t)}$.

1.1 Algorithmic Applications

We overview a few of the applications of our results to approximation and online algorithms. Some of the most striking applications of metric embeddings are to various cut problems, such as the sparsest-cut, min-bisection, and also to several online problems. Our method provides improved guarantees when the input graph has a small set of "important" vertices. Specifically, these vertices can be considered as terminals, and we obtain approximation factors that depend on the cardinality of the terminal set, rather than on the input size. The exact meaning of importance is problem specific; e.g. in the cut problems, we require that the set of important vertices touches every demand pair, or every edge (that is, forms a vertex cover).

For instance, consider the (general) *sparsest-cut* problem [34, 10, 35]. We are given a graph $G=(V,E)$ with capacities on the edges $c: E \to \mathbb{R}_+$, and a collection of pairs $(s_1,t_1),\ldots,(s_r,t_r)$ along with their demands $D_1,\ldots,D_r$. The goal is to find a cut $S \subseteq V$ that minimizes the ratio between capacity and demand across the cut:

$$\phi(S) = \frac{\sum_{\{u,v\}\in E} c(u,v)|\mathbf{1}_S(u)-\mathbf{1}_S(v)|}{\sum_{i=1}^{r} D_i|\mathbf{1}_S(s_i)-\mathbf{1}_S(t_i)|} ,$$

where $\mathbf{1}_S(\cdot)$ is the indicator for membership in S. Following the breakthrough result of [9], which showed $O(\sqrt{\log n})$ approximation for the uniform demand case, [7] devised an $\tilde{O}(\sqrt{\log r})$ approximation for the general case. If there is a set of k important vertices, such that every demand pair contains an important vertex, we obtain an $\tilde{O}(\sqrt{\log k})$ approximation using the terminal embedding of negative-type metrics to ℓ_1. Observe that $k \leq r$, and so our result subsumes the result of [7]. Our bound is particularly useful for instances with many demand pairs but few distinct sources s_i (or few targets t_i).

We also consider other cut problems, and show a similar phenomenon: the $O(\log n)$ approximation for the *min-bisection* problem can be improved to an approximation of only $O(\log k)$, where k is the size of the minimum vertex cover of the input graph. For this result we employ our terminal variant of Räcke's result [40] on capacity-preserving probabilistic embedding into trees.

We then focus on one application of probabilistic embedding into ultrametrics [12, 23], and illustrate the usefulness of our terminal embedding result by the (online) *constrained file migration* problem [13]. Given a graph $G=(V,E)$ representing a network, each node

[2] Note that the number of edges is linear whenever $k \leq n^{1/(2(1+1/t))}$.

$v \in V$ has a memory capacity m_v, and there is a set of files that reside at the nodes, at most m_v files may be stored at node v at any given time. The cost of accessing a file is the distance in the graph to the node storing it (no copies are allowed). Files can also be migrated from one node to another. This costs D times the distance, for a given parameter $D \geq 1$. When a sequence of file requests from nodes arrives online, the goal is to minimize the cost of serving all requests. [12] showed a algorithm with $O(\log m \cdot \log n)$ competitive ratio for graphs on n nodes, where $m = \sum_{v \in V} m_v$ is the total memory available.[3] A setting which seems particularly natural is one where there is a small set of nodes who can store files (servers), and the rest of the nodes can only access files but not store them (clients). We employ our probabilistic terminal embedding into ultrametrics to provide a $O(\log m \cdot \log k)$ competitive ratio, for the case where there are k servers. (Note that this ratio is independent of n.)

1.2 Overview of Techniques

The weak variant of our terminal embedding into ℓ_2 maps every terminal x into its image $f(x)$ under an original black-box (e.g., Bourgain's) embedding of K into ℓ_2. This embedding is then appended with one additional coordinate. Terminals are assigned 0 value in this coordinate, while each non-terminal point y is mapped to $(f(x), d(x, y))$, where x is the closest terminal to y. It is not hard to see that this embedding guarantees terminal distortion $O(\gamma(k))$, where $\gamma(k)$ is the distortion of the original black-box embedding, i.e., $O(\log k)$ in the case of Bourgain's embedding. On the other hand, the new embedding employs only $\beta(k) + 1$ dimensions, where $\beta(k)$ is the dimension of the original blackbox embedding (i.e., $O(\log^2 k)$ in the case of Bourgain's embedding). [4] This idea easily generalizes to a number of quite general scenarios, and under mild assumptions (see Theorem 3) it can be modified to produce strong terminal embeddings.

This framework, however, does not apply in many important settings, such as embedding into subgraphs, and does not provide strong terminal guarantees in others. Therefore we devise embeddings tailored to each particular setting in a non-black-box manner. For instance, our probabilistic embedding into trees with strong terminal congestion requires an adaptation of a theorem of [6], about the equivalence of distance-preserving and capacity-preserving random tree embeddings, to the terminal setting. Perhaps the most technically involved is our probabilistic embedding into spanning trees with strong terminal distortion. This result requires a set of modifications to the recent algorithm of [3], which is based on a certain hierarchical decomposition of graphs. We adapt this algorithm by giving preference to the terminals in the decomposition (they are the first to be chosen as cluster centers), and the crux is assuring that the distortion of any pair containing a terminal is essentially not affected by choices made for non-terminals. Furthermore, one has to guarantee that each such pair can be separated in at most $O(\log k)$ levels of the hierarchy.

The basic technical idea that we use for constructing $(4t-1)$-terminal subgraph spanners with $O(\sqrt{n}k^{1+1/t} + n)$ edges is the following one. As was mentioned above, our general transformation constructs metric (i.e., non-subgraph) $(4t-1)$-terminal spanners with $O(n + k^{1+1/t})$ edges. The latter spanners employ some edges which do not belong to the original graph. We provide these edges as an input to a pairwise preserver. A pairwise preserver

[3] The original paper shows $O(\log m \cdot \log^2 n)$, the improved factor is obtained by using the embedding of [23].

[4] We can also get dimension $O(\log k)$ for terminal embeddings into ℓ_2 by replacing Bourgain's embedding with that of [2].

[18] is a sparse subgraph that preserves exactly all distances between a designated set of vertex pairs. We use these preservers to fill in the gaps in the non-subgraph terminal spanner constructed via our general transformation. As a result we obtain a subgraph terminal spanner which outperforms previously existing terminal spanners of [41] in a wide range of parameters.

1.3 Related Work

Already in the pioneering work of [35], an embedding that has to provide a distortion guarantee for a subset of the pairs is presented. Specifically, in the context of the sparsest-cut problem, [35] devised a non-expansive embedding of an arbitrary metric into ℓ_1, with distortion at most $O(\log r)$ for a set of r specified demand pairs.

Terminal distance oracles were studied by [41], who called them *source restricted* distance oracles. In their paper, [41] show $(2t-1)$-terminal stretch using $O(t \cdot n \cdot k^{1/t})$ space. Implicit in our companion paper [20] is a distance oracle with $(4t-1)$-terminal stretch, $O(t \cdot k^{1/t} + n)$ space and $O(1)$ query time. Terminal spanners with additive stretch for unweighted graphs were recently constructed in [31]. Specifically, they showed a spanner with $\tilde{O}(n^{5/4} \cdot k^{1/4})$ edges and additive stretch 2 for pairs containing a terminal. Another line of work introduced *distance preservers* [18]; these are spanners which preserve exactly distances for a given collection of pairs.

In the context of preserving distances just between the terminals, [26, 16, 22, 30] studied embeddings of a graph into a minor over the terminals, while approximately preserving distances. In their work on the requirement cut problem, among other results, [27] obtain for any metric with k specified terminals, a distribution over trees with expected expansion $O(\log k)$ for all pairs, and which is non-contractive for terminal pairs. (Note that this is different from our setting, as the extra guarantee holds for terminals only, not for pairs containing a terminal.)

Another line of research [37, 17, 22] studied cut and vertex sparsifiers. A *cut sparisifier* of a graph $G = (V, E)$ with respect to a subset K of terminals is a graph $H = (K, E_H)$ on just the set of terminals, so that for any subset $A \subset K$, the minimum value of a cut in G that separates A from $K \setminus A$ is approximately equal to the value of the cut $(A, K \setminus A)$ in H. Note that this notion is substantially different from the notion of terminal congestion-preserving embedding, which we study in the current paper.

In a companion paper [20], we study prioritized metric structures and embeddings. In that setting, along with the input metric (X, d), a priority ranking of the points of X is given, and the goal is to obtain a data structure (distance oracle, routing scheme) or an embedding with stretch/distortion that depends on the ranking of the points. This has some implications to the terminal setting, since the k terminals can be given as the first k points in the priority ranking. More concretely, implicit in [20] is an embedding into a single (non-subgraph) tree with strong terminal distortion $O(k)$, a probabilistic embedding into ultrametrics with expected strong terminal distortion $O(\log k)$, and embedding into ℓ_p space with strong terminal distortion $\tilde{O}(\log k)$. In the current paper we provide stronger and more general results: our single tree embedding has tight $2k-1$ stretch, the tree is a subgraph, and it can have low weight as well (at the expense of slightly increased stretch); we obtain probabilistic embedding into *spanning* trees. and in congestion-preserving trees; and our terminal embedding to ℓ_p space has a tight strong terminal distortion $(O(\log k), O(\log n))$ and low dimension. Furthermore, the results of this paper apply to numerous other settings (e.g., embeddings tailored for graphs excluding a fixed minor, negative-type metrics, spanners, etc.).

1.4 Organization

The general transformations are presented in Section 3, and the results on graph spanners appear in Section 4. The tradeoff between terminal distortion and lightness in a single tree embedding is shown in Section 5 (corresponding lower bounds in several settings are deferred to the full version). The probabilistic congestion preserving embedding into trees appears in Section 6 (see [20] for the distortion version). Finally, in Section 7 we describe some algorithmic applications of terminal embeddings.

In the full version of the paper we present our probabilistic embedding into spanning trees with strong terminal distortion, and an embedding into a single tree (ultrametric or a spanning tree) with constant average terminal distortion.

2 Preliminaries

Here we provide formal definitions for the notions of terminal distortion. Let (X, d_X) be a finite metric space, with $K \subseteq X$ a set of terminals. Throughout the paper we assume $|K| \le |X|/2$.

▶ **Definition 1.** Let (X, d_X) be a metric space, and let $K \subseteq X$ be a subset of terminals. For a target metric (Y, d_Y), an embedding $f : X \to Y$ has *terminal distortion* α if there exists $c > 0$, such that for all $v \in K$ and $u \in X$,[5]

$$d_X(v,u) \le c \cdot d_Y(f(v), f(u)) \le \alpha \cdot d_X(v,u) \ .$$

We say that the embedding has *strong terminal distortion* (α, β) if it has terminal distortion α, and in addition there exists $c' > 0$, such that for all $u, w \in X$,

$$d_X(u,w) \le c' \cdot d_Y(f(u), f(w)) \le \beta \cdot d_X(u,w) \ .$$

For a graph $G = (V, E)$ with a terminal set $K \subseteq V$, an α*-terminal (metric) spanner* is a graph H on V such that for all $v \in K$ and $u \in V$,

$$d_G(u,v) \le d_H(u,v) \le \alpha \cdot d_G(u,v) \ . \tag{1}$$

H is a *graph* spanner if it is a subgraph of G.

Denote by $\mathrm{diam}(X) = \max_{y,z \in X}\{d_X(y,z)\}$. For any $x \in X$ and $r \ge 0$ let $B_X(x,r) = \{y \in X \mid d_X(x,y) \le r\}$ (we often omit the subscript when the metric is clear from context). For a point $x \in X$ and a subset $A \subseteq X$, $d_X(x,A) = \min_{a \in A}\{d_X(x,a)\}$. For $K \subseteq X$ we denote by (K, d_K) the metric space where d_K is the induced metric.

3 A General Transformation

In this section we present general transformation theorems that create terminal embeddings into normed spaces and graph families from standard ones. We say that a family of graphs $\mathcal{G}$ is *leaf-closed*, if it is closed under adding leaves. That is, for any $G \in \mathcal{G}$ and $v \in V(G)$, the graph G' obtained by adding a new vertex u and connecting u to v by an edge, belongs to $\mathcal{G}$. Note that many natural families of graphs are leaf-closed, e.g. trees, planar graphs, minor-free graphs, bounded tree-width graphs, bipartite graphs, general graphs, and many others.

[5] In most of our results the embedding has a one-sided guarantee (that is, non-contractive or non-expansive) for all pairs.

▶ **Theorem 2.** *Let $\mathcal{X}$ be a family of metric spaces. Fix some $(X, d_X) \in \mathcal{X}$, and let $K \subseteq X$ be a set of terminals of size $|K| = k$, such that $(K, d_K) \in \mathcal{X}$. Then the following assertions hold:*

- *If there are functions $\alpha, \gamma : \mathbb{N} \to \mathbb{R}$, such that every $(Z, d_Z) \in \mathcal{X}$ of size $|Z| = m$ embeds into $\ell_p^{\gamma(m)}$ with distortion $\alpha(m)$, then there is an embedding of X into $\ell_p^{\gamma(k)+1}$ with* terminal *distortion $2^{(p-1)/p} \cdot ((2\alpha(k))^p + 1)^{1/p}$.*[6]
- *If $\mathcal{G}$ is a leaf-closed family of graphs, and any $(Z, d_Z) \in \mathcal{X}$ of size $|Z| = m$ embeds into $\mathcal{G}$ with distortion $\alpha(m)$ such that the target graph has at most $\gamma(m)$ edges, then there is an embedding of X into $\mathcal{G}$ with* terminal *distortion $2\alpha(k) + 1$ and the target graph has at most $\gamma(k) + n - k$ edges.*

▶ Remark. The second assertion holds under probabilistic embeddings as well.

Proof. We start by proving the first assertion. By the assumption there exists an embedding $f : K \to \mathbb{R}^{\gamma(k)}$ with distortion $\alpha(k)$ under the ℓ_p norm. We assume w.l.o.g that f is non-contractive. For each $x \in X$, let $k_x \in K$ be the nearest point to x in K (that is, $d(x, K) = d(x, k_x)$). Extend f to an embedding $\hat{f} : X \to \mathbb{R}^{\gamma(k)+1}$ by defining for $x \in X$, $\hat{f}(x) = (f(k_x), d(x, k_x))$. Observe that this is indeed an extension. Fix any $t \in K$ and $x \in X$. Note that by definition of k_x, $d(x, k_x) \leq d(x, t)$, and by the triangle inequality, $d(t, k_x) \leq d(t, x) + d(x, k_x) \leq 2d(t, x)$, so that,

$$\begin{aligned}
\|\hat{f}(t) - \hat{f}(x)\|_p^p &= \|f(t) - f(k_x)\|_p^p + d(x, k_x)^p \\
&\leq (\alpha(k) \cdot d(t, k_x))^p + d(x, k_x)^p \\
&\leq (2\alpha(k) \cdot d(t, x))^p + d(t, x)^p \\
&= d(t, x)^p \cdot ((2\alpha(k))^p + 1) \ .
\end{aligned}$$

On the other hand, since f does not contract distances,

$$\begin{aligned}
\|\hat{f}(t) - \hat{f}(x)\|_p^p &= \|f(t) - f(k_x)\|_p^p + d(x, k_x)^p \\
&\geq d(t, k_x)^p + d(x, k_x)^p \\
&\geq (d(t, k_x) + d(x, k_x))^p / 2^{p-1} \\
&\geq d(x, t)^p / 2^{p-1} \ ,
\end{aligned}$$

where the second inequality is by the power mean inequality. We conclude that the terminal distortion is at most $2^{(p-1)/p} \cdot ((2\alpha(k))^p + 1)^{1/p}$.

For the second assertion, there is a non-contractive embedding f of K into $G \in \mathcal{G}$ with distortion at most $\alpha(k)$. As above, for each $x \in X \setminus K$ define k_x as the nearest point in K to x. And for each $x \in X$, add to G a new vertex $f(x)$ that is connected by an edge of length $d_G(x, k_x)$ to $f(k_x)$. The resulting graph $G' \in \mathcal{G}$, because it is a leaf-closed family. Fix any $x \in X$ and $t \in K$, then as above $d(t, k_x) \leq 2d(t, x)$, and so

$$\begin{aligned}
d_{G'}(f(t), f(x)) &= d_G(f(t), f(k_x)) + d_{G'}(f(x), f(k_x)) \\
&\leq \alpha(k) \cdot d(t, k_x) + d(x, k_x) \\
&\leq d(t, x) \cdot (2\alpha(k) + 1) \ .
\end{aligned}$$

Also note that

$$d_{G'}(f(t), f(x)) = d_G(f(t), f(k_x)) + d(x, k_x) \geq d(t, k_x) + d(x, k_x) \geq d(t, x) \ ,$$

[6] Note that for any $p, \alpha \geq 1$ we have that $2^{(p-1)/p} \cdot ((2\alpha)^p + 1)^{1/p} \leq 4\alpha$.

so the terminal distortion is indeed $2\alpha(k)+1$. Since f embeds into a graph with $\gamma(k)$ edges, and we added $n-k$ new edges, the total number of edges is bounded accordingly, which concludes the proof. ◀

Next, we study strong terminal embeddings into normed spaces. Fix any metric (X,d), a set of terminals $K \subseteq X$ and $1 \le p \le \infty$. Let $f : K \to \ell_p$ be a non-expansive embedding. We say that f is *Lipschitz extendable*, if there exists a non-expansive $\hat{f} : X \to \ell_p$ which is an extension of f (that is, the restriction of $\hat{f}$ to K is exactly f). It is not hard to verify that any Fréchet embedding[7] is Lipschitz extendable. For example, the embeddings of [15, 33, 8] are Fréchet.

▶ **Theorem 3.** *Let $\mathcal{X}$ be a family of metric spaces. Fix some $(X,d_X) \in \mathcal{X}$, and let $K \subseteq X$ be a set of terminals of size $|K| = k$, such that $(K,d_K) \in \mathcal{X}$. If any $(Z,d_Z) \in \mathcal{X}$ of size $|Z| = m$ embeds into $\ell_p^{\gamma(m)}$ with distortion $\alpha(m)$ by a* Lipschitz extendable map, *then there is a (non-expansive) embedding of X into $\ell_p^{\gamma(n)+\gamma(k)+1}$ with* strong *terminal distortion $O(\alpha(k),\alpha(n))$.*

Proof. By the assumptions there is a non-expansive embedding $g : X \to \ell_p^{\gamma(n)}$ with distortion at most $\alpha(n)$, and there exists a Lipschitz extendable embedding $f : K \to \ell_p^{\gamma(k)}$, which is non-expansive and has distortion $\alpha(k)$. Let $\hat{f}$ be the extension of f to X, note that by definition of Lipschitz extendability, $\hat{f}$ is also non-expansive. Finally, let $h : X \to \mathbb{R}$ be defined by $h(x) = d(x,K)$. The embedding $F : X \to \ell_p^{\gamma(n)+\gamma(k)+1}$ is defined by the concatenation of these maps $F = g \oplus \hat{f} \oplus h$.

Since all the three maps $g, \hat{f}, h$ are non-expansive, it follows that for any $x, y \in X$,

$$\|F(x)-F(y)\|_p^p \le \|g(x)-g(y)\|_p^p + \|\hat{f}(x)-\hat{f}(y)\|_p^p + |h(x)-h(y)|^p \le 3d(x,y)^p \ ,$$

so F has expansion at most $3^{1/p}$ for all pairs (which can easily be made 1 without affecting the distortion by more than a constant factor). Also note that

$$\|F(x)-F(y)\|_p \ge \|g(x)-g(y)\|_p \ge \frac{d(x,y)}{\alpha(n)} \ ,$$

which implies the distortion bound for all pairs is satisfied. It remains to bound the contraction for all pairs containing a terminal. Let $t \in K$ and $x \in X$, and let $k_x \in K$ be such that $d(x,K) = d(x,k_x)$ (it could be that $k_x = x$). If it is the case that $d(x,t) \le 3\alpha(k)\cdot d(x,k_x)$ then by the single coordinate of h we get sufficient contribution for this pair:

$$\|F(t)-F(x)\|_p \ge |h(t)-h(x)| = h(x) = d(x,k_x) \ge \frac{d(x,t)}{3\alpha(k)} \ .$$

The other case is that $d(x,t) > 3\alpha(k)\cdot d(x,k_x)$, here we will get the contribution from $\hat{f}$. First observe that by the triangle inequality,

$$d(t,k_x) \ge d(t,x) - d(x,k_x) \ge d(t,x)(1-1/(3\alpha(k))) \ge 2d(t,x)/3 \ . \tag{2}$$

[7] In our context, it will be convenient to call an embedding $f : K \to \ell_p^t$ *Fréchet*, if there are sets $A_1,\ldots,A_t \subseteq X$ such that for all $i \in [t]$, $f_i(x) = \frac{d(x,A_i)}{t^{1/p}}$.

By another application of the triangle inequality, using that $\hat{f}$ is non-expansive, and that f has distortion $\alpha(k)$ on the terminals, we get the required bound on the contraction:

$$\begin{aligned}
\|F(t)-F(x)\|_p &\geq \|\hat{f}(t)-\hat{f}(x)\|_p \\
&\geq \|\hat{f}(t)-\hat{f}(k_x)\|_p - \|\hat{f}(k_x)-\hat{f}(x)\|_p \\
&\geq \|f(t)-f(k_x)\|_p - d(x,k_x) \\
&\geq \frac{d(t,k_x)}{\alpha(k)} - \frac{d(t,x)}{3\alpha(k)} \\
&\overset{(2)}{\geq} \frac{2d(t,x)}{3\alpha(k)} - \frac{d(t,x)}{3\alpha(k)} \\
&= \frac{d(t,x)}{3\alpha(k)} \,.
\end{aligned}$$

◀

▶ Remark. The results of Theorems 2 and 3 hold also if $\mathcal{X}$ is a family of graphs, rather than of metrics, provided that the embedding for this family has the promised guarantees even for graphs with Steiner nodes. (E.g., if $\hat{Z} \in \mathcal{X}$ is a graph and Z is a set of vertices of size m, then there exists a (Lipschitz extendable) embedding of (Z, d_Z) to $\ell_p^{\gamma(m)}$ with distortion $\alpha(m)$, where d_Z is the shortest path metric on $\hat{Z}$ induced on Z.) We note that many embeddings of graph families satisfy this condition, e.g. the embedding of [33] to planar and minor-free graphs.[8]

Here are some of the implications of Theorems 2 and 3.

▶ **Corollary 4.** *Let (X,d) be a metric space on n points, and $K \subseteq X$ a set of terminals of size $|K| = k$. Then for any $1 \leq p \leq \infty$,*

1. *(X,d) can be embedded to $\ell_p^{O(\log k)}$ with terminal distortion $O(\log k)$.*
2. *If (X,d) is an ℓ_2 metric, it can be embedded to $\ell_2^{O(\log k)}$ with terminal distortion $O(1)$.*
3. *For any $t \geq 1$ there exists a $(4t-1)$-terminal (metric) spanner of X with at most $O(k^{1+1/t}) + n$ edges.*
4. *If (X,d) is an ℓ_2 metric, for any $t \geq 1$ there exists a $O(t)$-terminal spanner of X with at most $O(k^{1+1/t^2}) + n$ edges.*
5. *(X,d) can be embedded to $\ell_p^{O(\log n + \log^2 k)}$ with strong terminal distortion $(O(\log k), O(\log n))$.*
6. *If (X,d) is a shortest-paths metric of a graph that excludes a fixed minor (e.g., a planar metric), it can be embedded to ℓ_p with strong terminal distortion $(O((\log k)^{\min\{1/2,1/p\}}), O((\log n)^{\min\{1/2,1/p\}}))$.*
7. *If (X,d) is a negative type metric, it can be embedded to ℓ_2 with strong terminal distortion $(\tilde{O}(\sqrt{\log k}), \tilde{O}(\sqrt{\log n}))$.*

The first two items use the first assertion of Theorem 2, the next two use its second assertion, and the last three apply Theorem 3. The corollary follows from known embedding results: (1) and (5) are from [15], with improved dimension due to [2], (2) is from [29], (3) is from [5] and (4) from [28], (6) from [33], and (7) from [8, 7].

[8] We remark that this requirement is needed for those graph families for which the following question is open: given a graph Z in the family with terminals K, is there another graph in the family over the vertex set K, that preserves the shortest-path distances with respect to Z (up to some constant). This question is open, e.g., for planar metrics.

4 Graph Terminal Spanners

While Theorem 2 provides a general approach to obtain terminal spanners, it cannot provide spanners which are subgraphs of the input graph. We devise a construction of such terminal spanners in this section, while somewhat increasing the number of edges. Specifically, we show the following.

▶ **Theorem 5.** *For any parameter $t \geq 1$, a graph $G = (V, E)$ on n vertices, and a set of terminals $K \subseteq V$ of size k, there exists a $(4t-1)$-terminal* graph *spanner with at most $O(n + \sqrt{n} \cdot k^{1+1/t})$ edges.*

▶ Remark. Note that the number of edges is linear in n whenever $k \leq n^{1/(2(1+1/t))}$.

We shall use the following result:

▶ **Theorem 6** ([18]). *Given a weighted graph $G = (V, E)$ on n vertices and a set $P \subseteq \binom{V}{2}$ of size p, then there exists a subgraph G' with $O(n + \sqrt{n} \cdot p)$ edges, such that for all $\{u, v\} \in P$, $d_G(u, v) = d_{G'}(u, v)$.*

Proof of Theorem 5. The construction of the subgraph spanner with terminal stretch will be as follows. Consider the metric induced on the terminals K by the shortest path metric on G. Create a $(2t-1)$ (metric) spanner H' of this metric, using [5], and let $P \subseteq \binom{K}{2}$ be the set of edges of H'. Note that $p = |P| \leq O(k^{1+1/t})$. Now, apply Theorem 6 on the graph G with the set of pairs P, and obtain a graph G' that for every $\{u, v\} \in P$, has $d_{G'}(u, v) = d_G(u, v)$. This implies that G' is a $(2t-1)$-spanner for each pair of vertices $u, w \in K$. Moreover, G' has at most $O(n + \sqrt{n} \cdot p)$ edges. Finally, create H out of G' by adding a shortest path tree in G with the set K as its root. This will guarantee that the spanner H will have for each non-terminal, a shortest path to its closest terminal in G. This concludes the construction of H, and now we turn to bounding the stretch. Since H is a subgraph clearly it is non-contracting. Fix any $v \in K$ and $u \in V$, let k_u be the closest terminal to u, then $d_G(k_u, v) \leq d_G(k_u, u) + d_G(u, v) \leq 2d_G(u, v)$, and thus

$$d_H(u, v) \leq d_H(u, k_u) + d_{G'}(k_u, v) \leq d_G(u, v) + (2t-1)d_G(k_u, v) \leq (4t-1)d_G(u, v) .$$

Finally observe that the total number of edges in H is at most $O(n + \sqrt{n} \cdot p) = O(n + \sqrt{n} \cdot k^{1+1/t})$. ◀

5 Light Terminal Trees for General Graphs

In this section we find a single spanning tree of a given graph, that has both light weight, and approximately preserves distances from a set of specified terminals. Theorem 2 can provide a tree with terminal distortion $2k-1$ (using that any graph has a tree with distortion $n-1$), but that tree may not be a subgraph and may have large weight.

For a weighted graph $G = (V, E, w)$ where $w : E \to \mathbb{R}_+$, given a subgraph H of G, let $w(H) = \sum_{e \in E(H)} w(e)$, and define the *lightness* of H to be $\Psi(H) = \frac{w(H)}{w(MST(G))}$, where $w(MST)$ is the weight of a minimum spanning tree of G. The result of this section is summarized as follows.

▶ **Theorem 7.** *For any parameter $\alpha \geq 1$, given a weighted graph $G = (V, E, w)$, and a subset of terminals $K \subseteq V$ of size k, there exists a spanning tree T of G with terminal distortion $k \cdot \alpha + (k-1)\alpha^2$ and lightness $2\alpha + 1 + \frac{2}{\alpha - 1}$.*

When substituting $\alpha = 1$ in Theorem 7 we obtain a single tree with terminal distortion exactly $2k-1$, which is optimal. More specifically, for small $\epsilon > 0$, we get terminal distortion $2k - 1 + \epsilon$ and lightness $3 + \frac{6k}{\epsilon}$. Also, note that the bound $2\alpha + 1 + \frac{2}{\alpha-1}$ is minimized by setting $\alpha = 2$, so there is no point in using the theorem with $\alpha > 2$.

Next we describe the algorithm for constructing a spanning tree that satisfies the assertion of Theorem 7.

A spanning tree T is an (α, β)-SLT with respect to a root $v \in V$, if for all $u \in V$, $d_T(v, u) \leq \alpha \cdot d_G(v, u)$, and T has lightness β. A small modification of an SLT-constructing algorithm produces for any subset $K \subset V$, a forest F, such that every component of F contains exactly one vertex of K.[9] The forest F has distortion α with respect to K, and *lightness* $1 + \frac{2}{\alpha-1}$. (Such a forest F is said to have distortion α with respect to K, if for every vertex $u \in V$, $d_F(K, u) \leq \alpha \cdot d_G(K, u)$.)

The algorithm starts by building the aforementioned SLT-forest F from the terminal set K. No two terminals belong to the same connected component of F. Denote $K = \{v_1, \ldots, v_k\}$, let V_i be the unique connected component of F containing v_i, and let $T_i \subseteq F$ be the edges of the forest F induced by V_i. It follows that for every $u \in V_i$, $d_F(K, u) = d_{T_i}(v_i, u) \leq \alpha \cdot d_G(K, u)$. Let $G' = (K, E', w')$ be the super-graph in which two terminals share an edge between them if and only if there is an edge between the components V_i to V_j in G. Formally, $E' = \{\{v_i, v_j\} : \exists u_i \in V_i, u_j \in V_j \text{ such that } \{u_i, u_j\} \in E\}$. The weight $w'(v_i, v_j)$ is defined to be the length of the shortest path between v_i and v_j which uses *exactly one edge* that does not belong to F. (In other words, among all the paths between v_i and v_j in G which use exactly one edge that does not belong to F, let P be the shortest one. Then $w'(v_i, v_j) = w(P)$.) Note also that $w'(v_i, v_j)$ is given by $w'(v_i, v_j) = \min_{e \in E}\left\{d_{F \cup \{e\}}(v_i, v_j)\right\}$. We call the edge $e_{i,j} = \{u_i, u_j\}$ that implements this minimum ($w'(v_i, v_j) = d_{F \cup \{e_{i,j}\}}(v_i, v_j)$) the *representative edge* of $\{v_i, v_j\}$. (W.l.o.g the shortest paths, and thus the representative edges, are unique.) Observe that $\{v_i, v_j\} \in E'$ implies that $w'(v_i, v_j) < \infty$. Let T' be the MST of G'. Define $R = \left\{e_{i,j} \mid e_{i,j} \text{ is the representative edge of } e'_{i,j} = (v_i, v_j) \in T'\right\}$. Finally, set $T = F \cup R = \bigcup_{i=1}^{k} T_i \cup R$. Obviously, T is a spanning tree of G.

5.1 Proof of Theorem 7

As an embedding of a graph into its spanning tree is non-contractive, the tree T will have terminal distortion α if for all $v \in K$, $u \in V$, $d_T(v, u) \leq \alpha \cdot d_G(v, u)$. We shall assume w.l.o.g that all edge weights are different, and every two different paths have different lengths. If it is not the case, then one can break ties in an arbitrary (but consistent) way.

The next lemma shows that for every pair of terminals v_i, v_j, there is a path between them in G' in which all edges have weight (with respect to w') at most $\alpha \cdot d_G(v_i, v_j)$.

▶ **Lemma 8.** *[The bottleneck lemma:] For every $v_i, v_j \in K$, there exists a path $P : v_i = z_0, z_1, ..., z_r = v_j$ in G' such that for every $s = 0, 1, \ldots, r-1$, it holds that $\{z_s, z_{s+1}\} \in E'$ and $w'(z_s, z_{s+1}) \leq \alpha \cdot d_G(v_i, v_j)$.*

Proof. Let $P_{i,j} : v_i = u_0, u_1, ..., u_s = v_j$ be the shortest path from v_i to v_j in G, i.e., $w(P_{i,j}) = d_G(v_i, v_j)$. For each $0 \leq a \leq s$, denote by $V^{(a)}$ the connected component of F that contains u_a, and let $v^{(a)}$ be the unique terminal in that component. Consider the path

[9] To obtain such a forest F, one should add a new vertex v_K to the graph and connect it to each of the vertices of K with edges of weight zero. Then we compute an (α, β)-SLT with respect to v_K in the modified graph. Finally, we remove v_K from the SLT. The resulting forest is F.

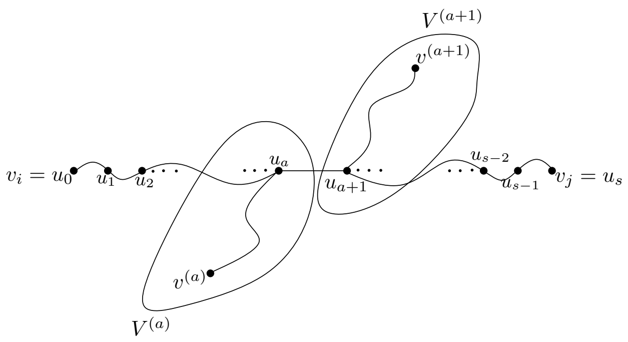


Figure 1 An illustration for the bottleneck lemma: v_i and v_j are terminals. The edge $\{u_a, u_{a+1}\}$ belongs to the shortest path from v_i to v_j in G. We conclude that for terminals $v^{(a)}, v^{(a+1)}$ such that $u_a \in V^{(a)}$ and $u_{a+1} \in V^{(a+1)}$ it holds that $w'\left(v^{(a)}, v^{(a+1)}\right) \le \alpha \cdot d_G(v_i, v_j)$.

$P = v^{(0)}, v^{(1)}, ..., v^{(s)}$. (This path is not necessarily simple. In particular, it might contain self-loops.) For every index $a < s$, (see Figure 1 for illustration)

$$\begin{aligned}
w'\left(v^{(a)}, v^{(a+1)}\right) &\underset{(1)}{\le} d_{F\cup\{\{u_a,u_{a+1}\}\}}\left(v^{(a)}, v^{(a+1)}\right) \\
&= d_F\left(v^{(a)}, u_a\right) + d_G(u_a, u_{a+1}) + d_F\left(u_{a+1}, v^{(a+1)}\right) \\
&\underset{(2)}{\le} \alpha \cdot d_G(v_i, u_a) + d_G(u_a, u_{a+1}) + \alpha \cdot d_G(v_j, u_{a+1}) \\
&< \alpha \cdot (d_G(v_i, u_a) + d_G(u_a, u_{a+1}) + d_G(v_j, u_{a+1})) \\
&\underset{(3)}{=} \alpha \cdot d_G(v_i, v_j) .
\end{aligned}$$

Note that if for some index a it holds that $v^{(a)} = v^{(a+1)}$ then $w'\left(v^{(a)}, v^{(a+1)}\right) = 0$, and the inequality above holds trivially. Otherwise, if $v^{(a)} \ne v^{(a+1)}$, then inequality (1) follows from the assumptions that $\{u_a, u_{a+1}\} \in E$, $u_a \in V^{(a)}$, $u_{a+1} \in V^{(a+1)}$. Inequality (2) follows from the properties of the SLT tree T (as $d_F\left(v^{(a)}, u_a\right) = d_F(K, u_a) \le \alpha \cdot d_G(K, u_a) \le \alpha \cdot d_G(v_i, u_a)$) . Equality (3) follows because the edge $\{u_a, u_{a+1}\}$ is on the shortest path from v_i to v_j in G.

In particular, one can remove cycles from P and obtain a simple path with the desired properties. We get a simple path P' such that for every edge v, v' on this path, we have $w'(v, v') \le \alpha \cdot d_G(v_i, v_j)$, as required. ◀

The following is a simple corollary.

▶ **Corollary 9.** *For $\{v_i, v_j\} \in T'$, we have $w'(v_i, v_j) = d_T(v_i, v_j) \le \alpha \cdot d_G(v_i, v_j)$.*

Proof. By Lemma 8, $w'(v_i, v_j) \le \alpha \cdot d_G(v_i, v_j)$. (Indeed, otherwise the edge $\{v_i, v_j\}$ is strictly the heaviest edge in a cycle in G', contradiction to the assumption that it belongs to the MST of G'.) Since $\{v_i, v_j\} \in E'$ and the representative edge of $\{v_i, v_j\}$ was taken into T, it follows that $w'(v_i, v_j) = d_T(v_i, v_j)$. ◀

We conclude the following lemma, which bounds the stretch of terminal pairs.

▶ **Lemma 10.** *For $v_i, v_j \in K$, we have $d_T(v_i, v_j) \le d_{T'}(v_i, v_j) \le \alpha \cdot (k-1) \cdot d_G(v_i, v_j)$.*

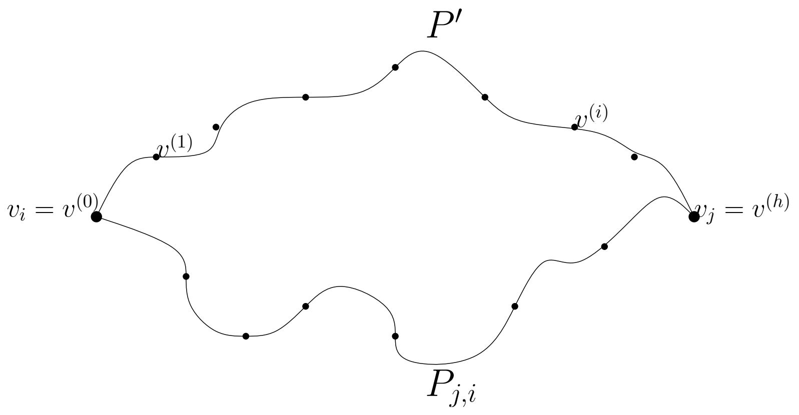


Figure 2 The two paths P' and $P_{j,i}$ considered in the proof of Lemma 10. The path P' is contained in T', while all edges of $P_{j,i}$ have weight at most $\alpha \cdot d_G(v_i, v_j)$.

Proof. Let $P' : v_i = v^{(0)}v^{(1)} \ldots v^{(h)} = v_j$ be the (unique) path in T' between v_i and v_j. Since T' is a spanning tree of the k-vertex graph G', it follows that $h \le k - 1$. Observe also that for every index $a \in [h-1]$, by Corollary 9 the edge $w'\left(v^{(a)}, v^{(a+1)}\right) = d_T\left(v^{(a)}, v^{(a+1)}\right)$. Also, we next argue that $w'\left(v^{(a)}, v^{(a+1)}\right) \le \alpha \cdot d_G(v_i, v_j)$. Indeed, suppose for contradiction that $w'\left(v^{(a)}, v^{(a+1)}\right) > \alpha \cdot d_G(v_i, v_j)$. Let $P_{j,i}$ be a path between v_j and v_i in G' such that all its edges have weight at most $\alpha \cdot d_G(v_i, v_j)$. The existence of this path is guaranteed by Lemma 8. In particular, since $w'\left(v^{(a)}, v^{(a+1)}\right) > \alpha \cdot d_G(v_i, v_j)$, it follows that $\left\{v^{(a)}, v^{(a+1)}\right\} \notin P_{j,i}$. Consider the cycle $P' \circ P_{j,i}$ in G'. It is not necessarily a simple cycle, but since $\left\{v^{(a)}, v^{(a+1)}\right\} \notin P_{j,i}$, the edge $\left\{v^{(a)}, v^{(a+1)}\right\}$ belongs to a simple cycle C contained in $P' \circ P_{j,i}$. The heaviest edge of C clearly does not belong to $P_{j,i}$, because the edge $\left\{v^{(a)}, v^{(a+1)}\right\}$ is heavier than each of them. Hence the heaviest edge belongs to P', but $P' \subseteq T'$. This is a contradiction to the assumption that T' is an MST of G'. (See Figure 2 for an illustration). Hence $d_T\left(v^{(a)}, v^{(a+1)}\right) = w'\left(v^{(a)}, v^{(a+1)}\right) \le \alpha \cdot d_G(v_i, v_j)$. Finally,

$$\begin{aligned} d_T(v_i, v_j) &\le \sum_{a=0}^{h-1} d_T\left(v^{(a)}, v^{(a+1)}\right) = \sum_{a=0}^{h-1} w'\left(v^{(a)}, v^{(a+1)}\right) \\ &\le \sum_{a=0}^{h-1} \alpha \cdot d_G(v_i, v_j) \le h \cdot \alpha \cdot d_G(v_i, v_j) \le \alpha \cdot (k-1) \cdot d_G(v_i, v_j). \end{aligned}$$

◀

Next, we analyze the terminal distortion of T.

▶ **Lemma 11.** *The terminal distortion of T is at most $k \cdot \alpha + (k-1)\alpha^2$.*

Proof. For each terminal $v_i \in K$ and any vertex $u \in V_j$, it holds that

$$\begin{aligned} d_T(v_i, u) &\le d_T(v_i, v_j) + d_T(v_j, u) \le \alpha \cdot (k-1) \cdot d_G(v_i, v_j) + \alpha \cdot d_G(v_i, u) \\ &\le \alpha \cdot (k-1) \cdot (d_G(v_i, u) + d_G(u, v_j)) + \alpha \cdot d_G(v_i, u) \\ &\le \alpha \cdot (k-1) \cdot (d_G(v_i, u) + \alpha \cdot d_G(v_i, u)) + \alpha \cdot d_G(v_i, u) \\ &= \left(k \cdot \alpha + (k-1)\alpha^2\right) \cdot d_G(v_i, u). \end{aligned}$$

The last inequality is because $d_G(v_j, u) \le d_F(v_j, u) = d_F(K, u) \le \alpha \cdot d_G(K, u) \le \alpha \cdot d_G(v_i, u)$. ◀

We now turn to analyze the lightness of T. A tree $T=(K,E',w')$ is called a *Steiner tree* for a graph $G=(V,E,w)$ if (1) $V\subseteq K$, (2) for any edge $e\in E\cap E'$, the edge has the same weight in both G and T, i.e. $w(e)=w'(e)$, and (3) for any pair of vertices $u,v\in V$ it holds that $d_T(u,v)\geq d_G(u,v)$. The *minimum Steiner tree* T of G, denoted $SMT(G)$, is a Steiner tree of G with minimum weight. It is well-known that for any graph G, $w(SMT(G))\geq\frac{1}{2}MST(G)$. (See, e.g., [25], Section 10.) The next lemma bounds the lightness of the tree T.

▶ **Lemma 12.** *The lightness of T is bounded by $\Psi(T)\leq 2\alpha+1+\frac{2}{\alpha-1}$.*

Proof. The main challenge is to bound $w(R)$. (Recall that R is the set of the *representative edges* of T'.) Consider an edge $\{v_i,v_j\}\in T'$, and let $\{u_i,u_j\}$ be its representative edge. Then $d_G(u_i,u_j)\leq w'(v_i,v_j)$. Also, since $\{v_i,v_j\}\in T'\subseteq E'$, it follows that $w'(v_i,v_j)=d_{G'}(v_i,v_j)$. Hence $d_G(u_i,u_j)\leq d_{G'}(v_i,v_j)$. Therefore, $w(R)\leq w'(T')$. Next we provide an upper bound for $w'(T')$. Define the graph $\widetilde{G}$ as the complete graph on the vertex set K, with weights $\tilde{w}$ induced by d_G (the shortest path distances in G). Let $\widetilde{T}$ be the MST of $\widetilde{G}$. We build a new tree $\hat{T}$ by the following process:

1. Let $\hat{T}\leftarrow\widetilde{T}$;
2. For each $\{v_i,v_j\}=\tilde{e}\in\widetilde{T}$:
   - **a.** Let $P_{\tilde{e}}$ be a path from v_i to v_j which consists of edges in E', such that for each edge e in $P_{\tilde{e}}$, $w'(e)\leq\alpha\cdot d_G(v_i,v_j)=\alpha\cdot\tilde{w}(\tilde{e})$; (By Lemma 8, such a path exists);
   - **b.** Let $e'\in P_{\tilde{e}}$ be an edge such that $(\hat{T}\setminus\{\tilde{e}\})\cup\{e'\}$ is connected;
   - **c.** Set $\hat{T}\leftarrow(\hat{T}\setminus\{\tilde{e}\})\cup\{e'\}$;

In each step in the loop we replace an edge $\tilde{e}=\{v_i,v_j\}$ from $\widetilde{T}$ by an edge e' from G' of weight $w'(e)\leq\alpha\cdot\tilde{w}(\tilde{e})$. Hence, the resulting tree $\hat{T}$ is a spanning tree of G', and $w'\left(\hat{T}\right)\leq\alpha\cdot\tilde{w}\left(\widetilde{T}\right)$. Since T' is the MST of G', it follows that $w'(T')\leq w'\left(\hat{T}\right)$. The MST of G is a Steiner tree for $\tilde{G}$, so that $\tilde{w}(SMT(\tilde{G}))\leq w(MST(G))$. Also, $\tilde{w}(MST(\tilde{G}))=\tilde{w}(\tilde{T})\leq 2\cdot\tilde{w}(SMT(\tilde{G}))\leq 2\cdot w(MST(G))$. We obtain that

$$w(R)\quad\leq\quad w'(T')\leq w'\left(\hat{T}\right)\leq\alpha\cdot\tilde{w}\left(\widetilde{T}\right)\leq 2\cdot\alpha\cdot w(MST(G)).$$

Since $w(F)\leq\left(1+\frac{2}{\alpha-1}\right)\cdot w(MST(G))$, we conclude that

$$w(T)\quad=\quad w(R\cup F)=w(R)+w(F)\leq\left(2\alpha+1+\frac{2}{\alpha-1}\right)\cdot w(MST(G)).$$

◀

6 Probabilistic Embedding into Trees with Terminal Congestion

In this section we focus on embeddings into trees that approximate capacities of cuts, rather than distances between vertices. This framework was introduced by Räcke [39] (for a single tree), and in [40] he showed how to obtain capacity preserving probabilistic embedding from a distance preserving one, such as the ones given by [23]. Later, [6] showed a complete equivalence between these notions in random tree embeddings. Here we show our terminal variant of these results. Informally, we construct a distribution over *capacity-dominating* trees (each cut in each tree is at least as large as the corresponding cut in the original graph), and for each edge, its expected congestion is bounded accordingly, with an improved bound for edges *containing a terminal.*

Recall that an ultrametric (U, d) is a metric space satisfying a strong form of the triangle inequality, that is, for all $x, y, z \in U$, $d(x, z) \leq \max\{d(x, y), d(y, z)\}$. The following definition is known to be an equivalent one (see [14]).

▶ **Definition 13.** An ultrametric U is a metric space (U, d) whose elements are the leaves of a rooted labeled tree T. Each $z \in T$ is associated with a label $\Phi(z) \geq 0$ such that if $q \in T$ is a descendant of z then $\Phi(q) \leq \Phi(z)$ and $\Phi(q) = 0$ iff q is a leaf. The distance between leaves $z, q \in U$ is defined as $d_T(z, q) = \Phi(\text{lca}(z, q))$ where $\text{lca}(z, q)$ is the least common ancestor of z and q in T.

Next, we define probabilistic embeddings with terminal distortion. For a class of metrics $\mathcal{Y}$, a distribution $\mathcal{D}$ over embeddings $f_Y : X \to Y$ with $Y \in \mathcal{Y}$ has *expected terminal distortion* α if each f_Y is non-contractive (that is, for all $u, w \in X$ and $Y \in \text{supp}(\mathcal{D})$, it holds that $d_X(u, w) \leq d_Y(f_Y(u), f_Y(w))$), and for all $v \in K$ and $u \in X$,

$$\mathbb{E}_{Y \sim \mathcal{D}}[d_Y(f_Y(v), f_Y(u))] \leq \alpha \cdot d_X(v, u) \ .$$

The notion of strong terminal distortion translates to this setting in the obvious manner.

We will need the following theorem, implicit in our companion paper [20].

▶ **Theorem 14.** *[20] Given a metric space (X, d) of size $|X| = n$ and a subset of terminals $K \subseteq X$ of size $|K| = k$, there exists a distribution over embeddings of X into ultrametrics with strong terminal distortion $(O(\log k), O(\log n))$.*

We next elaborate on the notions of capacity and congestion, and their relation to distance and distortion, following the notation of [6]. Given a graph $G = (V, E)$, let $\mathcal{P}$ be a collection of multisets of edges in G. A map $M : E \to \mathcal{P}$, where $M(e)$ is a path between the endpoints of e, is called a *path mapping* (the path is not necessarily simple). Denote by $M_{e'}(e)$ the number of appearances of e' in $M(e)$.

The path mapping relevant to the rest of this section is constructed as follows: given a tree $T = (V, E_T)$ (not necessarily a subgraph), for each edge $e' \in E_T$ let $P_G(e')$ be a shortest path between the endpoints of e' in G (breaking ties arbitrarily), and similarly for $e \in E$, let $P_T(e)$ be the unique path between the endpoints of e in T. Then for an edge $e \in E$, where $P_T(e) = e'_1 e'_2 \ldots e'_r$, the path $M(e)$ is defined as $M(e) = P_G(e'_1) \circ P_G(e'_2) \circ \cdots \circ P_G(e'_r)$ (where $\circ$ denotes concatenation). In what follows fix a tree T, and let M be the path mapping of T.

Fix a weight function $w : E \to \mathbb{R}_+$, and a capacity function $c : E \to \mathbb{R}_+$. For an edge $e \in E$, $\text{dist}_T(e) = \sum_{e' \in E} M_{e'}(e) \cdot w(e')$ is the weight of the path $M(e)$, and $\text{load}_T(e) = \sum_{e' \in E} M_e(e') \cdot c(e')$ is the sum (with multiplicities) of the capacities of all the edges whose path is using e. Define $\text{distortion}_T(e) = \frac{\text{dist}_T(e)}{w(e)}$ to be the distortion of e in T, and $\text{cong}_T(e) = \frac{\text{load}_T(e)}{c(e)}$ is the congestion of e. Note that if T is a subgraph of G, then $\text{dist}_T(e)$ is the length of the unique path between the endpoints of e, while $\text{load}_T(e)$ is the total capacity of all the edges of E that are in the cut obtained by deleting e from T (for $e \notin T$, $\text{load}_T(e) = 0$).

▶ **Definition 15.** Let $K \subseteq V$ be a set of terminals of size k, and let $E_K \subseteq E$ be the set of edges that contain a terminal. We say that a distribution $\mathcal{D}$ over trees has strong terminal congestion (α, β) if for every $e \in E_K$.

$$\text{cong}_{\mathcal{D}}(e) := \mathbb{E}_{T \sim \mathcal{D}}[\text{cong}_T(e)] \leq \alpha \ ,$$

and for any $e \in E$, $\text{cong}_{\mathcal{D}}(e) \leq \beta$.

A tight connection between distance preserving and capacity preserving mappings was shown in [6]. We generalize their theorem to the terminal setting in the following manner.

▶ **Theorem 16.** *The following statements are equivalent for a graph G:*

- *For every possible* weight *assignment G admits a probabilistic embedding into trees with strong terminal* distortion (α, β).
- *For every possible* capacity *assignment G admits a probabilistic embedding into trees with strong terminal* congestion (α, β).

An immediate corollary of Theorem 16, achieved by applying Theorem 14, is:[10]

▶ **Corollary 17.** *For any graph $G = (V, E)$ on n vertices, a set $K \subseteq V$ of k terminals, and any capacity function, there exists a distribution over trees with strong terminal congestion $(O(\log k), O(\log n))$.*

Proof of Theorem 16. Assuming the first item holds we prove the second. Let $\kappa(e) = \begin{cases} 1/\alpha & e \in E_K \\ 1/\beta & \text{otherwise} \end{cases}$. Given any capacity function $c : E \to \mathbb{R}_+$, we would like to show that there exists a distribution $\mathcal{D}'$ such that for any $e \in E$, $\mathbb{E}_{T\sim\mathcal{D}'}[\kappa(e) \cdot \text{cong}_T(e)] \le 1$. By applying the minimax principle (as in [4]), it suffices to show that for any coefficients $\{\lambda_e\}_{e\in E}$ with $\lambda_e \ge 0$ and $\sum_{e\in E} \lambda_e = 1$, there exists a single tree T such that

$$\sum_{e\in E} \lambda_e \cdot \kappa(e) \cdot \text{cong}_T(e) \le 1 \ . \tag{3}$$

To this end, define the weights $w(e) = \kappa(e) \cdot \frac{\lambda_e}{c(e)}$, and by the first assertion there exists a distribution $\mathcal{D}$ over trees such that for any $e \in E$,

$$\mathbb{E}_{T\sim\mathcal{D}}[\kappa(e) \cdot \text{distortion}_T(e)] \le 1 \ .$$

By applying the minimax again, there exists a single tree T such that

$$\sum_{e\in E} \lambda_e \cdot \kappa(e) \cdot \text{distortion}_T(e) \le 1 \ .$$

Now,

$$\begin{aligned} 1 &\ge \sum_{e\in E} \lambda_e \cdot \kappa(e) \cdot \text{distortion}_T(e) \\ &= \sum_{e\in E} \lambda_e \cdot \kappa(e) \cdot \frac{\sum_{e'\in E} M_{e'}(e) \cdot w(e')}{w(e)} \\ &= \sum_{e\in E} \lambda_e \cdot \kappa(e) \cdot \frac{\sum_{e'\in E} M_{e'}(e) \cdot \kappa(e') \cdot \lambda_{e'}/c(e')}{\kappa(e) \cdot \lambda_e/c(e)} \\ &= \sum_{e'\in E} \lambda_{e'} \cdot \kappa(e') \cdot \frac{\sum_{e\in E} M_{e'}(e) \cdot c(e)}{c(e')} \\ &= \sum_{e'\in E} \lambda_{e'} \cdot \kappa(e') \cdot \text{cong}_T(e') \ , \end{aligned}$$

which concludes the proof of (3). The second direction is symmetric. ◀

[10] Even though the embedding of Theorem 14 is into ultrametrics, which contain Steiner vertices, these can be removed while increasing the distortion of each pair by at most a factor of 8 [26].

Capacity Domination Property. As [40, 6] showed, under the natural capacity assignment, any tree T supported by the distribution of Theorem 16 has the following property: Any multi-commodity flow in G can be routed in T with no larger congestion. We would like to show this explicitly, using the language of cuts, as this will be useful for the algorithmic applications.

Fix some tree $T = (V, E_T)$, and for any edge $e' \in E_T$ let $S_{T,e'} \subseteq V$ be the cut obtained by deleting e' from T. Define the capacities $C_T : E_T \to \mathbb{R}_+$ by

$$C_T(e') = \sum_{e \in E(S_{T,e'}, \bar{S}_{T,e'})} c(e) \ ,$$

where $E(S, \bar{S})$ denotes the set of edges in the graph crossing the cut S. (Observe that for spanning trees, $C_T(e) = \text{load}_T(e)$.)

▶ **Lemma 18.** *For any graph $G = (V, E)$ and tree $T = (V, E_T)$ with capacities as defined above, for any set $S \subseteq V$ it holds that*

$$\sum_{e \in E\left(S, \bar{S}\right)} c(e) \le \sum_{e' \in E_T\left(S, \bar{S}\right)} C_T(e') \le \sum_{e \in E\left(S, \bar{S}\right)} \text{load}_T(e) \ . \tag{4}$$

Proof. We begin with the left inequality. For any graph edge $e \in E(S, \bar{S})$, there exists a tree edge $e' \in E_T(S, \bar{S})$ such that $e' \in P_T(e)$, because the path $P_T(e)$ must cross the cut. Since removing e' from T separates the endpoints of e, $C_T(e')$ will contain the term $c(e)$. We conclude that

$$\sum_{e \in E\left(S, \bar{S}\right)} c(e) \le \sum_{e' \in E_T\left(S, \bar{S}\right)} C_T(e') \ .$$

For the right hand side, consider an edge $e \in E$, and note that for any tree edge $e' \in E_T$ such that $e \in P_G(e')$, every edge $e'' \in E(S_{T,e'}, \bar{S}_{T,e'})$ will have $e \in M(e'')$ and thus contribute to $\text{load}_T(e)$ (perhaps multiple times, due to different e'). This implies that

$$\text{load}_T(e) = \sum_{e' \in E_T \ : \ e \in P_G(e')} C_T(e') \ . \tag{5}$$

Next, observe that any tree edge $e' \in E_T(S, \bar{S})$ must have at least one graph edge $e \in E(S, \bar{S})$ such that $e \in P_G(e')$. This suggests that

$$\begin{aligned}
\sum_{e \in E(S, \bar{S})} \text{load}_T(e) &\overset{(5)}{=} \sum_{e \in E(S, \bar{S})} \sum_{e' \in E_T : e \in P_G(e')} C_T(e') \\
&= \sum_{e' \in E_T} |E(S, \bar{S}) \cap P_G(e')| \cdot C_T(e') \\
&\ge \sum_{e' \in E_T(S, \bar{S})} C_T(e') \ .
\end{aligned}$$

◀

7 Applications

In this section we illustrate several algorithmic applications of our techniques. Some of our applications are suitable for graphs with a small vertex cover. Recall that for a graph $G = (V, E)$, a set $A \subseteq V$ is a vertex cover of G, if for any edge $e \in E$, at least one of its endpoints is in A. A polynomial time 2-approximation algorithm to this problem is folklore.

7.1 Sparsest-Cut

In the sparsest-cut problem we are given a graph $G = (V, E)$ with capacities on the edges $c : E \to \mathbb{R}_+$, and a collection of pairs $(s_1, t_1), \ldots, (s_r, t_r)$ along with their demands $D_1, \ldots, D_r$. The goal is to find a cut $S \subseteq V$ that minimizes the ratio between capacity and demand across the cut:

$$\phi(S) = \frac{\sum_{\{u,v\}\in E} c(u,v)|\mathbf{1}_S(u) - \mathbf{1}_S(v)|}{\sum_{i=1}^{r} D_i|\mathbf{1}_S(s_i) - \mathbf{1}_S(t_i)|} ,$$

where $\mathbf{1}_S(\cdot)$ is the indicator for membership in S. Arora et. al. [7] present an $\tilde{O}\left(\sqrt{\log r}\right)$ approximation algorithm to this problem. Our contribution is the following.

▶ **Theorem 19.** *If there exists a set $K \subseteq V$ of size k such that any demand pair contains a vertex of K, then there exists a $\tilde{O}\left(\sqrt{\log k}\right)$ approximation algorithm for the sparsest-cut problem.*

The key ingredient of the algorithm of [7] is a non-expansive embedding from ℓ_2^2 (negative-type metrics) into ℓ_1, which has $\tilde{O}\left(\sqrt{\log r}\right)$ contraction for all demand pairs. We will use the strong terminal embedding for negative type metrics given in item (7) of Corollary 4 to improve the distortion to $\tilde{O}\left(\sqrt{\log k}\right)$.

We now elaborate on how to use the embedding of ℓ_2^2 into ℓ_1 to obtain an approximation algorithm for the sparsest-cut, all the details can be found in [35, 9, 7], and we provide them just for completeness. First, write down the following SDP relaxation with triangle inequalities:

Algorithm 1 Sparsest Cut SDP Relaxation

```
min $\sum_{\{u,v\}\in E} c(u,v) \cdot \|\bar{u} - \bar{v}\|_2^2$
s.t. $\sum_{i=1}^{r} D_i \cdot \|\bar{s_i} - \bar{t_i}\|_2^2 = 1$
    For all $u, v, w \in V$, $\|\bar{u} - \bar{v}\|_2^2 + \|\bar{v} - \bar{w}\|_2^2 \ge \|\bar{u} - \bar{w}\|_2^2$
    For all $u \in V$, $\bar{u} \in \mathbb{R}^n$
```

Note that this is indeed a relaxation: if S is the optimal cut, set $\rho = \sum_{i=1}^{r} D_i \cdot |\mathbf{1}_S(s_i) - \mathbf{1}_S(t_i)|$; for $u \in S$ set $\bar{u} = (\frac{1}{\sqrt{\rho}}, 0, ..., 0)$, and for $u \notin S$, set $\bar{u} = (0, ..., 0)$. It can be checked to be a feasible solution of value equal to that of the cut S.

Let $K \subseteq V$ be a vertex cover of the demand graph $(V, \{\{s_i, t_i\}_{i=1}^{r}\})$ of size at most $2k$ (recall that we can find such a cover in polynomial time). Let $X = \{\bar{v} \in \mathbb{R}^n \mid v \in V\}$ be an optimal solution to the SDP (it can be computed in polynomial time), which is in particular an ℓ_2^2 (pseudo) metric. By Corollary 4 there exists a non-expansive embedding $f : X \to \ell_1$ with terminal distortion $\tilde{O}\left(\sqrt{\log k}\right)$ (where K is the terminal set).[11] This implies that for any $u, v \in V$ and any $1 \le i \le r$,

$$\begin{aligned} &\|\bar{u} - \bar{v}\|_2^2 \ge \|f(\bar{v}) - f(\bar{u})\|_1 \\ &\|\bar{s_i} - \bar{t_i}\|_2^2 \le \tilde{O}(\sqrt{\log k}) \cdot \|f(\bar{s_i}) - f(\bar{t_i})\|_1 . \end{aligned} \tag{6}$$

Let $\|f(\bar{v}) - f(\bar{u})\|_1 = \sum_{S\subseteq V} \alpha_S |\mathbf{1}_S(v) - \mathbf{1}_S(u)|$ be a representation of the ℓ_1 metric as a nonnegative linear combination of cut metrics (it is well known that there is such a

[11] The embedding of Corollary 4 is in fact into ℓ_2, but there is an efficient randomized algorithm to embed ℓ_2 into ℓ_1 with constant distortion [24].

representation with polynomially many cuts S having $\alpha_S > 0$). We conclude

$$\begin{aligned}
\text{opt(SDP)} &= \sum_{\{u,v\}\in E} c(u,v)\cdot\|\bar{u}-\bar{v}\|_2^2 \\
&= \frac{\sum_{\{u,v\}\in E} c(u,v)\cdot\|\bar{u}-\bar{v}\|_2^2}{\sum_{i=1}^{r} D_i\cdot\|\bar{s_i}-\bar{t_i}\|_2^2} \\
&\overset{(6)}{\geq} \frac{\sum_{\{u,v\}\in E} c(u,v)\cdot\|f(\bar{v})-f(\bar{u})\|_1}{\sum_{i=1}^{r} D_i\cdot\tilde{O}\left(\sqrt{\log k}\right)\cdot\|f(\bar{s_i})-f\left(\bar{t_i}\right)\|_1} \\
&= \frac{1}{\tilde{O}\left(\sqrt{\log k}\right)}\cdot\frac{\sum_{\{u,v\}\in E} c(u,v)\cdot\sum_{S\subsetneq V}\alpha_S\left|\mathbf{1}_S(v)-\mathbf{1}_S(u)\right|}{\sum_{i=1}^{r} D_i\cdot\sum_{S\subsetneq V}\alpha_S\left|\mathbf{1}_S(s_i)-\mathbf{1}_S(t_i)\right|} \\
&\geq \frac{1}{\tilde{O}\left(\sqrt{\log k}\right)}\min_{S:\alpha_S>0}\frac{\sum_{\{u,v\}\in E} c(u,v)\cdot\left|\mathbf{1}_S(v)-\mathbf{1}_S(u)\right|}{\sum_{i=1}^{r} D_i\cdot\left|\mathbf{1}_S(s_i)-\mathbf{1}_S(t_i)\right|} \\
&= \min_{S:\alpha_S>0}\frac{\phi(S)}{\tilde{O}\left(\sqrt{\log k}\right)}.
\end{aligned}$$

In particular, among the polynomially many sets $S\subseteq V$ with $\alpha_S>0$, there exists one which has sparsity at most $\tilde{O}(\sqrt{\log k})$ times larger than the optimal one.

7.2 Min Bisection

In the min-bisection problem, we are given a graph $G=(V,E)$ on an even number n of vertices, with capacities $c:E\to\mathbb{R}_+$. The purpose is to find a partition of V into two equal parts $S\subseteq V$ and $\bar{S}=V\setminus S$, that minimizes $\sum_{e\in E(S,\bar{S})} c(e)$. This problem is NP-hard, and the best known approximation is $O(\log n)$ by [40]. We obtain the following generalization.

▶ **Theorem 20.** *There exists a $O(\log k)$ approximation algorithm for min-bisection, where k is the size of a minimal vertex cover of the input graph.*

Proof. Our algorithm follows closely the algorithm of [40], the major difference is that we use our embedding into trees with terminal congestion. Let $K\subseteq V$ be the set of terminals, which is a vertex cover of size at most $2k$, and $\mathcal{D}$ a distribution over trees with strong terminal congestion $(O(\log k), O(\log n))$ given by Corollary 17. The algorithm will sample a tree $T=(V,E_T)$ from $\mathcal{D}$, find an optimal bisection in T and return it. We refer the reader to Section 6 for details on notation and on the definition of capacities $C_T:E_T\to\mathbb{R}_+$ for T. We note that there is polynomial time algorithm (by dynamic programming) to find a min-bisection in trees.

It remains to analyze the algorithm. Let $S\subseteq V$ be the optimal solution in G, and S_T be the optimal bisection for the tree T. The expected cost of using S_T in G can be bounded using Lemma 18 as follows

$$\begin{aligned}
\sum_{T\in\mathrm{supp}(\mathcal{D})}\Pr[T]\sum_{e\in E\left(S_T,\bar{S}_T\right)}c(e) &\overset{(4)}{\leq} \sum_{T}\Pr[T]\sum_{e'\in E_T\left(S_T,\bar{S}_T\right)}C_T(e') \\
&\leq \sum_{T}\Pr[T]\sum_{e'\in E_T\left(S,\bar{S}\right)}\mathcal{C}_T(e') \\
&\overset{(4)}{\leq} \sum_{T}\Pr[T]\sum_{e\in E\left(S,\bar{S}\right)}\mathrm{load}_T(e) \\
&= \sum_{e\in E\left(S,\bar{S}\right)}\mathbb{E}_{T\sim\mathcal{D}}\left[\mathrm{load}_T(e)\right] \\
&\leq \sum_{e\in E\left(S,\bar{S}\right)}O(\log k)\cdot c(e) \\
&= O(\log k)\cdot\mathrm{opt}(G) \quad ,
\end{aligned}$$

where the last inequality uses that every edge touches a terminal, so its expected congestion is $O(\log k)$. The algorithm can be derandomized using standard methods, see e.g. [6]. ◀

7.3 Online Algorithms: Constrained File Migration

We illustrate the usefulness of our probabilistic terminal embedding into ultrametric via the constrained file migration problem. This is an online problem, in which we are given a graph $G=(V,E)$ representing a network, each node $v\in V$ has a memory capacity m_v, and a parameter $D\geq 1$. There is some set of files that reside at the nodes, at most m_v files may be stored at node v in any given time. The cost of accessing a file that currently lies at v from node u is $d_G(u,v)$ (no copies of files are allowed). Files can also be migrated from one node to another, this costs D times the distance. When a sequence of file requests arrives online, the goal is to minimize the cost of serving all requests. The *competitive ratio* of an online algorithm is the maximal ratio between its cost to the cost of an optimal (offline) solution. For randomized algorithms the expected cost is used.

We consider the case where there exists a small set of vertices which are allowed to store files (i.e. $m_v>0$). One may think about these vertices as servers who store files, while allowing file requests from all end users. Let $K\subseteq V$ be the set of terminal vertices that are allowed to store files, with $|K|=k$. Our result is captured by the following theorem.

▶ **Theorem 21.** *There is a randomized algorithm for the constrained file migration problem with competitive ratio $O(\log m\cdot\log k)$, where k vertices can store files and m is the total memory available.*

This theorem generalizes a result of [12], who showed an algorithm with competitive ratio $O(\log m\cdot\log n)$ for arbitrary graphs on n nodes. Both results are based on the following theorem. (Recall that a 2-HST is an ultrametric (see Definition 13) such that the ratio between the label of a node to any of its children's label is at least 2.)

▶ **Theorem 22** ([12]). *For any 2-HST, there is a randomized algorithm with competitive ratio $O(\log m)$ for constrained file migration with total memory m.*

By Theorem 14 there is a distribution $\mathcal{D}$ over embeddings of G into ultrametrics with expected terminal distortion $O(\log k)$, but in fact every tree in that distribution is a 2-HST.

Assume that in the optimal (offline) solution there are s_{uv} times a file residing on v was accessed by u, and t_{uv} files were migrated from v to u. Let $c_{uv} = s_{uv} + D \cdot t_{uv}$ be the total cost of file traffic from v to u. Note that as $m_v = 0$ for any $v \notin K$, then for any $u \in V$ we have $c_{uv} = 0$. Using the fact that the terminal distortion guarantee of $\mathcal{D}$ applies to all of the relevant distances, we obtain that

$$\begin{aligned} \mathrm{opt}_G &= \sum_{u \in V, v \in K} c_{uv} \cdot d_G(u,v) \qquad (7)\\ &\geq \frac{1}{O(\log k)} \cdot \sum_{u \in V, v \in K} c_{uv} \cdot \mathbb{E}_{T \sim \mathcal{D}}[d_T(u,v)] \\ &= \frac{1}{O(\log k)} \cdot \mathbb{E}_{T \sim \mathcal{D}}\Big[\sum_{u \in V, v \in K} c_{uv} \cdot d_T(u,v)\Big] \ . \end{aligned}$$

Observe that for any tree $T \in \mathrm{supp}\,(\mathcal{D})$ we could have served the request sequence in the same manner as the optimal algorithm, which would have the cost $\sum_{u \in V, v \in K} c_{uv} \cdot d_T(u,v)$. In particular, the optimal solution opt_T for the same requests with the input graph T cannot be larger than that, i.e.

$$\sum_{u \in V, v \in K} c_{uv} \cdot d_T(u,v) \geq \mathrm{opt}_T \ . \qquad (8)$$

Our algorithm will operate as follows: Pick a random tree according to the distribution $\mathcal{D}$, pick a random strategy S for transmitting files in T according to the distribution $\mathcal{S}(T)$ guaranteed to exists by Theorem 22, and serve the requests according to S. Denote by $\mathrm{cost}_H(S)$ the cost of applying strategy S with distances taken in the graph H. For any possible $T \in \mathrm{supp}\,(\mathcal{D})$ it holds that

$$\mathrm{opt}_T \geq \frac{\mathbb{E}_{S \sim \mathcal{S}(T)}[\mathrm{cost}_T\,(S)]}{O(\log m)} \geq \frac{\mathbb{E}_{S \sim \mathcal{S}(T)}[\mathrm{cost}_G\,(S)]}{O(\log m)}, \qquad (9)$$

where the last inequality holds since T dominates G (i.e. $d_T(u,v) \geq d_G(u,v)$ for all $u, v \in V$). Combining equations (7), (8) and (9) we get that

$$\mathrm{opt}_G \geq \frac{\mathbb{E}_{T \sim \mathcal{D}} \mathbb{E}_{S \sim \mathcal{S}(T)}[\mathrm{cost}_G\,(S)]}{O(\log k \log m)} \ .$$

Hence our randomized algorithm has $O\left(\log m \log k\right)$ competitive ratio, as promised.

Acknowledgements. We would like to thank Robert Krauthgamer, Yair Bartal and Manor Mendel for fruitful discussions, and to an anonymous referee for an idea leading to Theorem 3.

References

1 Ittai Abraham, Yair Bartal, and Ofer Neiman. Embedding metrics into ultrametrics and graphs into spanning trees with constant average distortion. In *SODA*, pages 502–511, 2007.

2 Ittai Abraham, Yair Bartal, and Ofer Neiman. Advances in metric embedding theory. *Advances in Mathematics*, 228(6):3026–3126, 2011.

3 Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. In *STOC*, pages 395–406, 2012.

4 Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, 1995.

5 I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993.

6 Reid Andersen and Uriel Feige. Interchanging distance and capacity in probabilistic mappings, 2009.

7 S. Arora, J. R. Lee, and A. Naor. Euclidean distortion and the sparsest cut. *Journal of the American Mathematical Society 21*, 1:1–21, 2008.

8 Sanjeev Arora, James R. Lee, and Assaf Naor. Fréchet embeddings of negative type metrics. *Discrete & Computational Geometry*, 38(4):726–739, 2007.

9 Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), 2009.

10 Yonatan Aumann and Yuval Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing*, 27(1):291–301, 1998.

11 B. Awerbuch, A. Baratz, and D. Peleg. Efficient broadcast and light-weight spanners. Technical Report CS92-22, The Weizmann Institute of Science, Rehovot, Israel., 1992.

12 Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *FOCS*, pages 184–193, 1996.

13 Yair Bartal, Amos Fiat, and Yuval Rabani. Competitive algorithms for distributed data management. *J. Comput. Syst. Sci.*, 51(3):341–358, 1995.

14 Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. Some low distortion metric ramsey problems. *Discrete & Computational Geometry*, 33(1):27–41, 2005.

15 J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985.

16 T.-H. Chan, Donglin Xia, Goran Konjevod, and Andrea Richa. A tight lower bound for the steiner point removal problem on trees. In *Proceedings of the 9th International Conference on Approximation Algorithms for Combinatorial Optimization Problems, and 10th International Conference on Randomization and Computation*, APPROX'06/RANDOM'06, pages 70–81, Berlin, Heidelberg, 2006. Springer-Verlag.

17 Moses Charikar, Tom Leighton, Shi Li, and Ankur Moitra. Vertex sparsifiers and abstract rounding algorithms. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 265–274, 2010.

18 Don Coppersmith and Michael Elkin. Sparse source-wise and pair-wise distance preservers. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'05, pages 660–669, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

19 Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. *SIAM Journal on Computing*, 38(2):608–628, 2008.

20 Michael Elkin, Arnold Filtser, and Ofer Neiman. Prioritized metric structures and embedding. In *Proceedings of the 47th ACM Symposium on Theory of Computing*, STOC'15, 2015.

21 Michael Elkin and Shay Solomon. Steiner shallow-light trees are exponentially lighter than spanning ones. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 373–382, 2011.

22 Matthias Englert, Anupam Gupta, Robert Krauthgamer, Harald Räcke, Inbal Talgam-Cohen, and Kunal Talwar. Vertex sparsifiers: New results from old techniques. *SIAM J. Comput.*, 43(4):1239–1262, 2014.

23 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

24 T. Figiel, J. Lindenstrauss, and V.D. Milman. The dimension of almost spherical sections of convex bodies. *Acta Mathematica*, 139(1):53–94, 1977.

25 E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, Jan 1968.

26 Anupam Gupta. Steiner points in tree metrics don't (really) help. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'01, pages 220–227, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

27 Anupam Gupta, Viswanath Nagarajan, and R. Ravi. An improved approximation algorithm for requirement cut. *Oper. Res. Lett.*, 38(4):322–325, 2010.

28 Sariel Har-Peled, Piotr Indyk, and Anastasios Sidiropoulos. Euclidean spanners in high dimensions. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'13, pages 804–809. SIAM, 2013.

29 William Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

30 Lior Kamma, Robert Krauthgamer, and Huy L. Nguyen. Cutting corners cheaply, or how to remove steiner points. In *SODA*, pages 1029–1040, 2014.

31 Telikepalli Kavitha and Nithin M. Varma. Small stretch pairwise spanners. In *Proceedings of the 40th International Conference on Automata, Languages, and Programming – Volume Part I*, ICALP'13, pages 601–612, Berlin, Heidelberg, 2013. Springer-Verlag.

32 Samir Khuller, Balaji Raghavachari, and Neal E. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14(4):305–321, 1995.

33 Robert Krauthgamer, James R. Lee, Manor Mendel, and Assaf Naor. Measured descent: a new embedding method for finite metrics. *Geometric and Functional Analysis*, 15(4):839–858, 2005.

34 Frank Thomson Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

35 N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

36 Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. In *FOCS*, pages 109–118, 2006.

37 Ankur Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *FOCS*, pages 3–12, 2009.

38 Yuri Rabinovich and Ran Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete & Computational Geometry*, 19(1):79–94, 1998.

39 Harald Räcke. Minimizing congestion in general networks. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS'02, pages 43–52, Washington, DC, USA, 2002. IEEE Computer Society.

40 Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *STOC*, pages 255–264, 2008.

41 Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In *Proceedings of the 32Nd International Conference on Automata, Languages and Programming*, ICALP'05, pages 261–272, Berlin, Heidelberg, 2005. Springer-Verlag.

On Linear Programming Relaxations for Unsplittable Flow in Trees

Zachary Friggstad[1] **and Zhihan Gao**[2]

1 **Department of Computing Science, University of Alberta**
Edmonton, AB, Canada, T6G 2E8
`zacharyf@cs.ualberta.ca`

2 **Department of Combinatorics and Optimization, University of Waterloo**
Waterloo, ON, Canada, N2L 3G1
`z9gao@uwaterloo.ca`

Abstract

We study some linear programming relaxations for the Unsplittable Flow problem on trees (UFP-TREE). Inspired by results obtained by Chekuri, Ene, and Korula for Unsplittable Flow on paths (UFP-PATH), we present a relaxation with polynomially many constraints that has an integrality gap bound of $O(\log n \cdot \min\{\log m, \log n\})$ where n denotes the number of tasks and m denotes the number of edges in the tree. This matches the approximation guarantee of their combinatorial algorithm and is the first demonstration of an efficiently-solvable relaxation for UFP-TREE with a sub-linear integrality gap.

The new constraints in our LP relaxation are just a few of the (exponentially many) *rank constraints* that can be added to strengthen the natural relaxation. A side effect of how we prove our upper bound is an efficient $O(1)$-approximation for solving the rank LP. We also show that our techniques can be used to prove integrality gap bounds for similar LP relaxations for packing demand-weighted subtrees of an edge-capacitated tree.

On the other hand, we show that the inclusion of all rank constraints does not reduce the integrality gap for UFP-TREE to a constant. Specifically, we show the integrality gap is $\Omega(\sqrt{\log n})$ even in cases where all tasks share a common endpoint. In contrast, intersecting instances of UFP-PATH are known to have an integrality gap of $O(1)$ even if just a few of the rank 1 constraints are included.

We also observe that applying two rounds of the Lovász-Schrijver SDP procedure to the natural LP for UFP-TREE derives an SDP whose integrality gap is also $O(\log n \cdot \min\{\log m, \log n\})$.

1998 ACM Subject Classification G.1.6 Optimization, G.2.2 Graph Theory, I.1.2 Algorithms

Keywords and phrases Unsplittable flow, Linear programming relaxation, Approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.265

1 Preliminaries

In the Unsplittable Flow problem on trees (UFP-TREE), we are given a tree $T = (V, E)$ with a nonnegative capacity $c_e \geq 0$ specified for each edge $e \in E$. Throughout, we will let m denote the number of edges in T. Additionally, we are given n tasks where each task $1 \leq i \leq n$ is specified by endpoints $s_i, t_i \in V$, a demand $d_i \geq 0$, and a weight $w_i \geq 0$.

For a task i we let $\text{span}(i)$ denote all edges $e \in E$ lying between the unique $s_i - t_i$ path in T. A set of tasks S is said to be *feasible* if

$$\sum_{i \in S : e \in \text{span}(i)} d_i \leq c_e \text{ for each edge } e \in E.$$

The goal is to find a feasible set of tasks S with maximum possible weight.

The seemingly unusual "Unsplittable Flow" name is inherited from a generalization to arbitrary graphs G where the problem is to select a maximum weight set tasks and select a single $s_i - t_i$ path for each chosen task upon which to route all of the task's demand. This generalization captures the well-studied Edge-Disjoint Paths problem in undirected graphs for which the best true approximation is $O(\sqrt{n})$ [10]. We will not pursue a discussion of this generalization as all of our results pertain only to trees.

In UFP-TREE, there is only one possible path for each task to follow so the difficulty is only in selecting which tasks to route. Still, UFP-TREE is NP-hard even if the tree consists of only a single edge as this case is just a reformulation of the classic KNAPSACK problem. A more interesting specialization of UFP-TREE when the tree is a path (UFP-PATH). Currently, the best approximation for UFP-PATH is $2 + \epsilon$ for any constant $\epsilon > 0$ [2] and the best lower bound is only strong NP-hardness [6]. It may still be possible to obtain a PTAS for UFP-PATH; indeed a $(1 + \epsilon)$-approximation with running time $n^{O_\epsilon(\log n)}$ was recently developed by Batra et al. [4], improving over a previous $(1 + \epsilon)$-approximation with running time $n^{O_\epsilon(\log n \cdot \log(nD))}$ where D is the ratio of the maximum to minimum density (i.e. w_i/d_i) [3].

The current best approximation for UFP-TREE is considerably worse than the $(2 + \epsilon)$-approximation for UFP-PATH, with a ratio of $O(\log n \cdot \min\{\log m, \log n\})$ [9]. It is known that UFP-TREE is APX-hard [14], which rules out a PTAS unless P = NP.

A major barrier to developing better approximations for UFP-TREE is that the following natural LP relaxation has an $\Omega(n)$ integrality gap even in UFP-PATH instances [8].

$$\begin{aligned} \text{maximize} \quad & \sum_i w_i \cdot x_i & & \textbf{(Nat-LP)} \\ \text{s.t.} \quad & \sum_{i:e\in \text{span}(i)} d_i \cdot x_i \le c_e & \forall\, e \in E & \\ & 0 \le x_i \le 1 & \forall\, 1 \le i \le n & \end{aligned}$$

This bad gap is witnessed by a simple "staircase" example: suppose the nodes in the underlying path are indexed by $\{0, 1, \ldots, n\}$. Then for every $1 \le i \le n$ we set the capacity of edge $(i-1, i)$ to 2^{-i} and create a task with endpoints $0, i$, demand 2^{-i} and weight 1. The all-$\frac{1}{2}$ solution is feasible for **Nat-LP** with value $n/2$, but the optimum integer solution selects only one task (if $i < j$ then $d_i + d_j > 2^{-i} = c_{(i-1,i)}$).

Nat-LP can still be used to obtain reasonable approximations in two important special cases of UFP-TREE. First, if there is some value B such that $d_i \le B$ for all tasks i and $B \le c_e$ for all edges e (the *no-bottleneck* property) then the integrality gap is at most 48 [11]. Second, if $\delta > 0$ is such that $d_i \le (1 - \delta) \cdot c_e$ for each task i and each $e \in \text{span}(i)$, then the integrality gap is $O(\delta^{-3} \cdot \log(1/\delta))$ [9].

Strengthenings of the natural LP relaxation for UFP-PATH have been considered in [9, 1]. In [9], a polynomial-size LP relaxation was presented for UFP-PATH and its integrality gap was proven to be $O(\min\{\log m, \log n\})$. In particular, they show the gap is $O(1)$ if all tasks span a common vertex and used a simple reduction from the general case to this intersecting case that loses an $O(\min\{\log m, \log n\})$-factor. The constraints they added are of the form $\sum_{i\in S} x_i \le 1$ for a certain collection of subsets S such that $\{i, j\}$ is infeasible for any distinct $i, j \in S$.

These are just some of the so-called *rank constraints* one can add to strengthen a packing LP. In their full generality, the rank constraint for a set of tasks S is the constraint $\sum_{i\in S} x_i \le \text{rank}(S)$ where $\text{rank}(S)$ is the size of the largest feasible subset of S. The authors of [9] also consider the more powerful LP that includes adding the rank constraints for every set of tasks that span a common node (**Rank-LP** in our paper, formally defined in Section 1.2). They show how to solve **Rank-LP** in UFP-PATH instances within an $O(1)$-factor and leave approximating **Rank-LP** in UFP-TREE instances as an open problem.

Additionally, two other polynomial-size LP relaxation for UFP-PATH were introduced in [1]. The constraints in one of these relaxations are motivated by a geometric view of UFP-PATH that was initially identified in [6]. They showed its integrality gap was $O(1)$ in unit-weight, but not necessarily intersecting, instances (i.e. $w_i = 1$ for all i). Their LP also approximates **Rank-LP** for UFP-PATH within $O(1)$. The other relaxation essentially "embeds" a dynamic programming algorithm introduced by Bonsma et al [6] for instances that have a bad integrality gap and is shown to have a constant integrality gap. We remark that no such dynamic programming procedure is known for UFP-TREE, so these techniques do not seem to apply to this more general setting.

To date, there has not been any demonstration of a polynomial-time solvable (or even $O(1)$-approximable) LP relaxation for UFP-TREE that has a $o(n)$ integrality gap. Our results settle this open problem affirmatively by presenting a LP relaxation for UFP-TREE with polynomially many constraints that has an integrality gap of $O(\log n \cdot \min\{\log m, \log n\})$. Meanwhile, we show how to solve **Rank-LP** in UFP-TREE instances within a constant factor.

Another potential avenue to strengthen the natural LP relaxation would be to use *lift-and-project* techniques (a.k.a *hierarchies*). Such techniques start with a linear or semidefinite programming relaxation of a $\{0,1\}$ integer program and strengthen the relaxation through a number of rounds. Typically, one can solve the ℓ'th round of the resulting relaxation with $n^{O(\ell)}$ overhead over solving the original formulation. We omit an introduction to such techniques (Lovász-Schrijver, Sherali-Adams, Lasserre hierarchies, etc.) from this extended abstract since our lift-and-project observations are secondary to our main results. A good introduction can be found in [12].

While some positive lift-and-project results are known for the restricted case of a single edge (i.e. KNAPSACK) [16, 13], the only known result for more general UFP-TREE instances is a negative one. Namely, LP-based hierarchies seem ineffective even for UFP-PATH: the integrality gap of **Nat-LP** strengthened with ℓ rounds of the Sherali-Adams hierarchy is $\Omega(n/\ell)$ [9]. In this paper, we show that applying two rounds of the Lovász-Schrijver SDP procedure (a SDP version of the Lovász-Schrijver hierarchy) to the natural LP for UFP-TREE derives a SDP relaxation for UFP-TREE with an integrality gap of $O(\log n \cdot \min\{\log m, \log n\})$.

1.1 A Generalization to Packing Trees

We will also (briefly) consider the following generalization of UFP-TREE to the setting where each task is now a subtree of T, rather than just a path in T. Here a task i is specified by a subtree T_i of T, a demand d_i, and a weight w_i. The goal is still to find a maximum-weight subset of tasks S so that $\sum_{i \in S : e \in \text{span}(i)} d_i \leq c_e$ for each edge e where, naturally, span(i) denotes the edges lying on T_i. We let k-TREEPACKING denote this problem where each input tree T_i is further restricted to contain at most k leaves. In this way, UFP-TREE is the same as k-TREEPACKING with $k = 2$.

While some special cases of k-TREEPACKING have been studied (e.g. UFP-PATH, UFP-TREE, and, as discussed below, the Maximum Independent Set Problem), it seems that the general problem has not been considered before. There is a simple reduction from the Maximum Independent Set Problem in graphs with degree at most k to k-TREEPACKING instances where T is just a star, and all demands, capacities, and weights are 1. Namely, if $G = (V, E)$ is a Maximum Independent Set instance then we let T be a star with leaves indexed by E. For each $v \in V$, we create a subtree T_v whose leaves in T are the edges in G incident to v. Thus, an independent set in G is the same as a feasible collection of subtrees in the k-TreePacking instance and we get the following as a corollary of Maximum Independent Set hardness results for bounded degree graphs in [7, 5].

▶ **Corollary 1** (of [7, 5]). *The following hardness results hold for k-TreePacking even if all demands, capacities, and weights are 1 and T is a star:*
1. *There is no $\frac{k}{O(\log^4 k)}$-approximation unless $P = NP$.*
2. *There is no $\frac{k}{O(\log^2 k)}$-approximation unless the Unique Games conjecture is false.*

While k-TreePacking has not been explicitly studied before, it is easy to generalize the $O(\log n \cdot \min\{\log m, \log n\})$-approximation for UFP-tree in [9] to get a combinatorial $O(k \cdot \log n \cdot \min\{\log m, \log(kn)\})$ approximation. However, as with UFP-tree, no compact LP relaxation was known for k-TreePacking that has a $o(n)$ integrality gap. In this paper, we first present a LP relaxation for k-TreePacking with an integrality gap of $O(k \cdot \log n \cdot \min\{\log m, \log(kn)\})$, which is $o(n)$ when considering k as a fixed constant. In particular, this LP relaxation has an integrality gap at most $4k + 1$ for the instances with unit weight subtrees sharing a common node. Note that both ratios in Corollary 1 are asymptotically larger than k^{1-c} for any constant $c > 0$. Thus, in this case the integrality gap of our LP relaxation is close to matching the hardness lower bounds stated in Corollary 1.

1.2 Results and Techniques

In this subsection, we present all our main results and techniques. The proofs are deferred to later sections and the appendix. Our main results pertain to UFP-tree. Our techniques extend to obtain LP relaxations with bounded integrality gaps for k-TreePacking, but those are secondary to our main result and will be discussed later. We assume that the singleton set $\{i\}$ is feasible for each task i. Otherwise, we can discard any task that does not fit by itself [1].

We establish some notation to describe our strengthening of **Nat-LP**. For any two vertices u, v we let $P(u, v)$ be the set of edges lying between u and v in T. Similarly, for an edge e and vertex v we let $P(e, v)$ be the set of edges lying between e and v in T, including e itself.

▶ **Definition 2.** For every task i, every vertex v spanned by task i, and every endpoint $a \in \{s_i, t_i\}$ we form a *blocking set* $C(i, v, a)$ of tasks as follows. $C(i, v, a)$ includes i and every other task j that satisfies the following conditions.
1. v is also spanned by j
2. $d_j \geq d_i$
3. $d_i + d_j > c_e$ for some $e \in P(a, v) \cap \text{span}(j)$

This is a natural generalization of the RightBlock and LeftBlock sets used in the relaxation **Compact UFP-LP** for UFP-path from [9].

For every collection of tasks S such that $\{i, j\}$ is infeasible for any distinct $i, j \in S$, we say S is a *pairwise infeasible clique.* The following lemma, whose proof is found at the start of Section 2, shows that a blocking set is a pairwise infeasible clique.

▶ **Lemma 3.** *For any distinct j, j' in some blocking set $C(i, v, a)$, the set $\{j, j'\}$ is not feasible.*

[1] This preprocessing step is not necessary when using lift-and-project techniques as a single level of even the Lovász-Schrijver LP hierarchy will enforce $x_i = 0$ for such tasks.

From this, we formulate our stronger LP relaxation for UFP-TREE.

$$\begin{aligned} \text{maximize} \quad & \sum_i w_i \cdot x_i && \textbf{(Compact-LP)} \\ \text{s.t.} \quad & \sum_{i: e \in \text{span}(i)} d_i \cdot x_i \le c_e \quad \forall\, e \in E && (1) \\ & \sum_{i \in C(j,v,a)} x_i \le 1 \quad \forall \text{ blocking sets } C(j,v,a) && (2) \\ & 0 \le x_i \le 1 \quad \forall\, 1 \le i \le n \end{aligned}$$

Note that there are $O(n \cdot m)$ constraints in this relaxation. We could omit the $x_i \le 1$ constraints because they are enforced by the blocking constraints (2), but we will keep them for ease of notation because the dual variables for $x_i \le 1$ serve a slightly different purpose than dual variables for blocking sets in our analysis.

We say that a set of tasks S is *intersecting* if there is some vertex v that lies on the $s_i - t_i$ path for every $i \in S$. A UFP-TREE instance is said to be *intersecting* if the set of all tasks is intersecting. Finally, say that an instance is a *unit-weight* instance if $w_i = 1$ for all tasks i.

▶ **Theorem 4.** *The integrality gap of* **Compact-LP** *is* $O(\log n \cdot \min\{\log m, \log n\})$ *and is at most* 9 *in unit-weight, intersecting instances of* UFP-TREE.

More specifically, we show that the greedy combinatorial algorithm in [9] for unit-weight, intersecting instances of UFP-TREE finds a feasible solution S such that $|S|$ is within a factor of 9 from the LP optimum. In our analysis, we construct a feasible dual solution and then verify that a relaxation of the complementary slackness conditions holds, in some appropriate sense, on average.

Chekuri, Ene, and Korula also introduce a larger family of constraints. For every collection of tasks S we say rank(S) is the size of the largest subset of S that is feasible (paying no attention to the weights w_i). They consider the following even stronger LP which, in our language, is presented as follows.

$$\begin{aligned} \text{maximize} \quad & \sum_i w_i \cdot x_i && \textbf{(Rank-LP)} \\ \text{s.t.} \quad & \sum_{i: e \in \text{span}(i)} d_i \cdot x_i \le c_e \qquad \forall\, e \in E \\ & \sum_{i \in S} x_i \le \text{rank}(S) \quad \forall \text{ intersecting sets of tasks } S && (3) \\ & 0 \le x_i \le 1 \qquad \forall\, 1 \le i \le n \end{aligned}$$

In the full version of [9], the integrality gap of **Rank-LP** is shown to be $O(1)$ on intersecting instances of UFP-PATH with arbitrary weights w_i. They also show a connection to their version of the blocking sets for UFP-PATH which, in our notation, means that if x is a feasible solution to **Compact-LP** for a UFP-PATH instance, then $x/18$ is feasible for **Rank-LP**. However, they do not identify any such connection for UFP-TREE nor do they provide a way to even approximate **Rank-LP**; this is left as an open problem.

We resolve this open problem affirmatively by showing that a consequence of Theorem 4 is that we can solve **Rank-LP** within constant factors in UFP-TREE. Specifically, we prove the following as a special case of a slightly more general statement about packing problems.

▶ **Theorem 5.** *If x is a feasible solution to* **Compact-LP** *for* UFP-TREE*, then $x/9$ is a feasible solution to* **Rank-LP**.

It is known that there is a reduction from the (general) UFP-TREE instances to the intersecting UFP-TREE instances losing an approximation factor at most $O(\min\{\log m, \log n\})$. An intriguing possibility for obtaining an $O(\log n)$-approximation for UFP-TREE would be to show the integrality gap of **Rank-LP** is $O(1)$ in intersecting cases. Indeed, this is the case for UFP-PATH [9]. Unfortunately, we have examples showing that the integrality gap can be super-constant in intersecting cases of UFP-TREE.

▶ **Theorem 6.** *The integrality gap of* **Rank-LP** *for* UFP-TREE *is* $\Omega(\sqrt{\log n})$ *even in instances with a common end node* r *(i.e.* $t_i = r$ *for all tasks* i*).*

We would like to briefly reflect on this result. Essentially by definition, the integrality gap of **Rank-LP** on intersecting, unit-weight instances of UFP-TREE is 1: consider the rank constraint for S being the set of all tasks. In fact, for *any* subset of tasks S in an intersecting instance, the LP is requiring that $\sum_{i \in S} x_i \leq \text{rank}(S)$ so this might seem like a very strong formulation. However, it is not the case that the set of feasible solutions to **Rank-LP** is equal to the convex hull of integer solutions. Theorem 6 basically says that one can choose the weight vector to make the integrality gap very large. We are not aware of any other packing problems for which the integrality gaps of the unweighted and a weighted versions have been observed to differ by a super-constant factor (though, it is easy to argue the difference is never worse than $O(\log n)$, see Appendix A.1). This observation may be of general interest.

Our techniques extend easily to k-TREEPACKING. The notion of a blocking set naturally generalizes to k-TREEPACKING and one can consider an analogous relaxation of **Compact-LP** (details of this generalization are in Appendix B).

▶ **Theorem 7.** *The integrality gap of* **Compact-LP** *for* k-TREEPACKING *is* $O(k \cdot \log n \cdot \min\{\log m, \log(kn)\})$ *and is at most* $4k+1$ *in intersecting, unit weight instances.*

Note that the latter bound is close to the hardness lower bounds stated in Corollary 1. Also, our integrality gap analysis is tight within constant factors for intersecting, unit weight instances.

▶ **Lemma 8.** *For any* $k \geq 2$*, there are intersecting, unit weight instances instances of* k-TREEPACKING *with integrality gap at least* $k/2$ *in* **Compact-LP**.

Finally, we observe that applying two rounds of Lovász-Schrijver SDP operatator (see [12] for a definition) to **Nat-LP** derives the constraint $\sum_{i \in S} x_i \leq 1$ for any pairwise infeasible clique S. Since a blocking set is a pairwise infeasible clique, the integrality gap bounds for **Compact-LP** stated in Theorem 4 also holds for the two-round Lovász-Schrijver SDP for UFP-TREE and, more generally, for k-TREEPACKING.

▶ **Lemma 9.** *Let* LS_+^t *denote the* t *rounds of the Lovász-Schrijver SDP operator, and let* $\mathcal{P}$ *be the polytope defined by the constraints of* **Nat-LP** *for* k-TREEPACKING*. Then the integrality gap of* $\max\{w^T \cdot x : x \in \text{LS}_+^2(\mathcal{P})\}$ *is* $O(k \cdot \log n \cdot \min\{\log m, \log(kn)\})$*. In particular, for* UFP-TREE*, the integrality gap is* $O(\log n \cdot \min\{\log m, \log n\})$*.*

Thus, SDP hierarchies are much more effective than LP hierarchies for k-TREEPACKING since, as mentioned earlier, the integrality gap using t rounds of the Sherali-Adams operator is $\Omega(n/t)$ even in UFP-PATH instances [9].

The paper is organized as follows. Section 2 contains the proof of Theorem 4, Section 3 contains the proof of Theorem 5, and Section 4 presents the lower bound in Theorem 6. Concluding remarks are made in Section 5. For the sake of space, the results for k-TREEPACKING mentioned in Theorem 7 and Lemma 8 are discussed in Appendix B and the proof of Lemma 9 is deferred to Appendix C.

2 A Stronger, Compact LP For UFP-tree

We begin by proving Lemma 3 from Section 1.2, referring to the three conditions in Definition 2 for the blocking sets.

Proof of Lemma 3. Condition 3 implies $\{i, j\}$ itself is not feasible for any $j \in C(i, v, a) \setminus \{i\}$. Now consider any two distinct $j, j' \in C(i, v, a) \setminus \{i\}$. Let $e, e' \in P(a, v)$ be any edges that are violated by $\{i, j\}$ and $\{i, j'\}$, respectively, as in condition 3. Suppose, without loss of generality, that $e' \in P(e, v)$ so condition 1 implies $e' \in \text{span}(j)$ as well. By conditions 2 and 3, we have $d_j + d_{j'} \geq d_i + d_{j'} > c_{e'}$ so $\{j, j'\}$ violates the capacity of edge e'. ◀

The following summarizes some of the reductions performed in the combinatorial UFP-TREE-approximation [9] that remain valid for our LP-based arguments. For convenience, we have sketched these reductions in Appendix A.

▶ **Lemma 10.** *If the integrality gap of* **Compact-LP** *for intersecting, unit-weight* UFP-TREE *instances is* $O(1)$*, then the integrality gap of of* **Compact-LP** *in general* UFP-TREE *instances is* $O(\log n \cdot \min\{\log m, \log n\})$.

Thus, to prove Theorem 4 it suffices to prove that the integrality gap of **Compact-LP** is 9. The rest Section 2 is devoted to proving this statement.

2.1 Duality and Complementary Slackness

From now on, we will assume that there is a *root node* r such that every task spans r. We will also assume, for simplicity, that there are precisely $2n$ leaves of the tree and each leaf of T is an endpoint of precisely one task. This is without any loss of generality since we can append a new node ℓ to every endpoint of every task i, move that endpoint of i to the new node ℓ, and set the capacity of the parent edge of ℓ to d_i. This does not change the set of feasible LP solutions for **Compact-LP**.

It is important to remember that we are considering **Compact-LP** in unit-weight instances in this analysis, which is why we state the dual of **Compact-LP** only for unit-weight instances. To avoid clutter, we will let C refer to a blocking set of the form $C(j, v, a)$. For example, a sum of the form $\sum_{C:i\in C}$ sums over all blocking sets of the form $C(j, v, a)$ that contain i. We let y_e be the dual variables for constraints (1), z_C be the dual variables for constraints (2) and z'_i be the dual variables for constraints $x_i \leq 1$.

$$\text{minimize} \quad \sum_e c_e \cdot y_e + \sum_C z_C + \sum_i z'_i \qquad \textbf{(Dual-LP)}$$

$$\text{s.t.} \quad \sum_{e\in\text{span}(i)} d_i \cdot y_e + \sum_{C:i\in C} z_C + z'_i \geq 1 \quad \forall \text{ tasks } i \qquad (4)$$

$$y, z, z' \geq 0$$

Relaxed Complementary Slackness. We construct feasible primal x and dual (y, z, z') solutions satisfying the following conditions.

1. $x_i \in \{0, 1\}$ for each task i
2. $x_i = 1 \Longrightarrow \sum_{e\in\text{span}(i)} d_i \cdot y_e \leq 2$
3. $y_e > 0 \Longrightarrow \sum_{i:e\in\text{span}(i)} d_i \cdot x_i \geq \frac{c_e}{2}$
4. $\sum_C z_C + \sum_i z'_i \leq 5 \sum_i x_i$

Let OPT_f denote the optimal fractional solution to **Compact-LP** for the intersecting, unit-weight instance we are considering.

▶ **Lemma 11.** *Suppose x and (y, z, z') are feasible primal and dual solutions that satisfy conditions $1 - 4$ above. Then x is an integer solution with value $\geq OPT_f/9$.*

Proof. Let $\alpha, \beta > 0$ be quantities we will set later that satisfy $\alpha + \beta = 1$. Then

$$\begin{aligned}
\sum_i x_i &= \alpha \sum_i x_i + \beta \sum_i x_i \\
&\geq \frac{\alpha}{2} \sum_i \sum_{e \in \text{span}(i)} x_i \cdot d_i \cdot y_e + \frac{\beta}{5}\left(\sum_C z_C + \sum_i z'_i\right) \\
&= \frac{\alpha}{2} \sum_e y_e \sum_{i : e \in \text{span}(i)} x_i \cdot d_i + \frac{\beta}{5}\left(\sum_C z_C + \sum_i z'_i\right) \\
&\geq \frac{\alpha}{4} \sum_e c_e \cdot y_e + \frac{\beta}{5}\left(\sum_C z_C + \sum_i z'_i\right).
\end{aligned}$$

The first inequality uses conditions 2 and 4 and the second inequality uses condition 3. Setting $\alpha = \frac{4}{9}$ and $\beta = \frac{5}{9}$ shows

$$\sum_i x_i \geq \frac{1}{9}\left(\sum_e c_e \cdot y_e + \sum_C z_C + \sum_i z'_i\right).$$

Finally, by weak duality and since (y, z, z') is feasible for **Dual-LP** with cost $\sum_e c_e \cdot y_e + \sum_C z_C + \sum_i z'_i$, then $\sum_i x_i \geq \frac{1}{9} OPT_f$. ◀

We will prove there indeed exists such x and (y, z, z'). The x-values will be obtained by a simple greedy algorithm and the corresponding (y, z, z')-values will be carefully constructed to witness the near-optimality of x as a solution to **Compact-LP**.

2.2 The Greedy Algorithm

Algorithm 1 is essentially the greedy algorithm of [9] for unit-weight, intersecting instances of UFP-TREE. We have augmented it with some bookkeeping for use in our analysis. In the algorithm, we say that e is *undersaturated by S* if $\sum_{i \in S : e \in \text{span}(i)} d_i < c_e/2$.

Algorithm 1 Greedy algorithm for unit-weight, intersecting instances.

```
1: Initialize $S, D^u$ and $D^s$ to $\emptyset$.
2: for each task $i$ in increasing order of demand $d_i$ do
3:     if $S \cup \{i\}$ is feasible then add $i$ to $S$
4:     else
5:         Let $B(i)$ be the edges whose capacities are violated by $S \cup \{i\}$.
6:         if some $e \in B(i)$ is undersaturated by $S$ then add $i$ to $D^u$
7:         else add $i$ to $D^s$
8: return $S$
```

Intuitively, for $i \notin S$ we have that $B(i)$ consists of the edges that "blocked" i; those edges that would have their capacity constraint violated by $S' \cup \{i\}$ (for the set $S' \subseteq S$ of tasks that were chosen at the time i was considered). Then D^u consists of tasks that were blocked

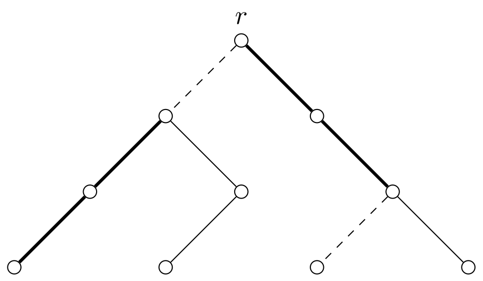


Figure 1 The tree T' is partitioned into six paths. Two are drawn with bold edges, two with thin edges, and two with dashed edges. Note that the root has degree 2, but the path it lies on is broken into two paths.

by at least one undersaturated edge and D^s consists of tasks that were blocked only by "mostly saturated" edges. Note that once an edge blocks some task, no more tasks spanning that edge will be added to S.

Let x be defined by $x_i = 1$ if $i \in S$ and $x_i = 0$ if $i \notin S$. By construction, x is a feasible integer solution to **Compact-LP**. We must show $\sum_i x_i = |S|$ is within a factor of 9 from the optimum LP solution.

2.3 Constructing the Dual Solution

We will show that D^u can be partitioned into at most $4 \cdot |S|$ sets such that each is a subset of some blocking set of the form $C(j, a, s)$. Given this, we set $z_C = 1$ for each of the at most $4 \cdot |S|$ blocking sets C that contain one of the partitions of D^u and set $z_{C'} = 0$ for the remaining blocking sets C'. Finally, we set $z'_i = 1$ for each $i \in S$ and $z'_i = 0$ for $i \notin S$. This will satisfy the 4th complementary slackness condition.

Note that the dual constraints (4) in **Dual-LP** for $i \in D^u$ will be satisfied by the z-variables alone and that the dual constraints for $i \in S$ will be satisfied by z' alone. Finally, we will set the y_e values to satisfy the dual constraints for $i \in D^s$. The rest of the analysis breaks into two parts: 1) finding the appropriate partition of D^u into subsets of blocking sets and 2) setting the y_e variables to satisfy the dual constraints for $i \in D^s$. The second part must be done carefully to ensure the dual constraints for $i \in S$ are not too slack to satisfy condition 2 while maintaining $y_e = 0$ for undersaturated edges to satisfy condition 3.

2.4 Finding the Blocking Sets

Consider the subtree T' of T consisting only of the nodes and edges of T that are spanned by some task $i \in S$. Since we are assuming the leaves of T are in one-to-one correspondence with the $2 \cdot n$ endpoints of the tasks, then T' has precisely $2 \cdot |S|$ leaves. Let $\mathcal{P}$ be the collection of paths in T' such that for every $P \in \mathcal{P}$, the endpoints of P have degree $\neq 2$ in T' and the internal nodes of P have degree 2. If it so happens that r has degree 2 in T', then we also break the path P containing r into two paths, both containing r as one endpoint. The paths in $\mathcal{P}$ form a partition of the set of edges of T'. Figure 1 illustrates the partitioning of a tree into paths $\mathcal{P}$ in this manner.

Since the number of leaves of T' is $2 \cdot |S|$ and we only split at most one of the degree-2 paths in T' into two paths, then there are at most $4 \cdot |S|$ paths in $\mathcal{P}$. We will partition D^u into at most $4 \cdot |S|$ subsets that we denote by $C(P), P \in \mathcal{P}$. Partitioning D^u is straightforward. For each $i \in D^u$ we have that some $e \in B(i)$ was undersaturated when i was considered in the algorithm and remains undersaturated throughout the rest of the algorithm. Pick any such edge and call it $e(i)$. Add i to $C(P)$ where $P \in \mathcal{P}$ is such that $e(i) \in P$.

For each $P \in \mathcal{P}$ with $C(P) \neq \emptyset$, we will identify a blocking set C containing $C(P)$. Let i_P denote the task with least demand in $C(P)$, let v_P denote the node on P nearest to r (which must be spanned by i since $i \in C(P)$), and let a_P be the endpoint of i_P such that $e(i) \in P(a, v_P)$.

▶ **Lemma 12.** *For this choice of i_P, v_P, a, we have $C(P) \subseteq C(i_P, v_P, a_P)$.*

Proof. Consider any $j \in C(P)$. We clearly have $d_{i_P} \leq d_j$ by our choice of i_P. Furthermore, since P lies below v_P and since $e(j) \in P$ then $v_P \in \text{span}(j)$.

Note that every $i' \in S$ that spans some edge of P must, in fact, span all of P by how we decomposed T' into degree-2 subpaths. Let $\Delta = \sum_{i' \in S: P \subseteq \text{span}(i')} d_{i'}$ be the total demand of tasks in S routed across P. Since task j is blocked by $e(j)$, then $d_j + \Delta > c_{e(j)}$. Since $e(j)$ was undersaturated when j was blocked, then $\Delta < c_{e(j)}/2$ so $d_j > c_{e(j)}/2 > \Delta$. Note that this argument also works for i_P: $d_{i_P} > \Delta$.

To finish the proof, we have to show that if $j \neq i_P$ then j conflicts with i_P somewhere on $P(a_P, v_P)$. Now, since both $e(j)$ and $e(i_P)$ lie in P, then either $e(j) \in \text{span}(i_P)$ or $e(i_P) \in \text{span}(j)$. Suppose $e(j) \in \text{span}(i_P)$ (the other case is similar). Then $d_j + d_{i_P} > d_j + \Delta > c_{e(j)}$ meaning $\{j, i_P\}$ conflicts across $e(j) \in P$. Thus, $C(P) \subseteq C(i_P, v_P, a_P)$. ◀

2.5 Setting y_e

Recall that for an edge e, the set $P(e, r)$ consists of all edges on the path between e and r including e itself. Also, for a vertex v we let $P(v, r)$ denote the set of all edges lying on the unique $v - r$ path in the tree T.

Let F' be the set of all edges $e \in E$ such that $\sum_{i \in S: e \in \text{span}(i)} d_i \geq c_e/2$. Fix any subset $F \subseteq F'$ that is minimal with respect to the property that for every task $i \in D^s$ and every endpoint $a \in \{s_i, t_i\}$, if $F' \cap B(i) \cap P(a, r) \neq \emptyset$ then $F \cap B(i) \cap P(a, r) \neq \emptyset$. In other words, we are looking at each $a - r$ subpath for each endpoint a of a task $i \in D^s$. If i was blocked by some edge on this subpath, then F should still contain some edge on this subpath that blocked i.

Say that an edge $e \in F$ is *critical* for $i \in D^s$ if $F \cap B(i) \cap P(a, r) = \{e\}$ for some endpoint a of i. Note that up to (but no more than) 2 edges may be critical for a single task i, one per endpoint of i. By minimality of F, every $e \in F$ is critical for at least one task in D^s. So, for any $e \in F$ we define $i(e) := \arg\min\{d_i : i \in D^s \text{ and } e \text{ is critical for } i\}$ (breaking ties arbitrarily).

We now set values to the dual variables $y_e, e \in E$.

▶ **Lemma 13.** *There is a $y \geq 0$ such that $y_e = 0$ for $e \notin F$ and $\sum_{e' \in P(e,r)} d_{i(e)} y_{e'} = 1$ for $e \in F$.*

Proof. We set the values $y_e, e \in F$ inductively in increasing size of $|P(e, r) \cap F|$. If $P(e, r) \cap F = \{e\}$ then we simply set $y_e = \frac{1}{d_{i(e)}}$.

If $|P(e, r) \cap F| \geq 2$ then let e' be the deepest edge on $(P(e, r) \cap F) \setminus \{e\}$. That is, $F \cap (P(e, r) \setminus \{e\}) = F \cap P(e', r)$. Set $y_e = \frac{1}{d_{i(e)}} - \frac{1}{d_{i(e')}}$; it must be that $y_e \geq 0$. Otherwise, $i(e')$ is considered before $i(e)$ in Algorithm 1. But then $e' \in B(i(e))$, contradicting the fact that $e \in F$ is critical for $i(e)$.

Finally, by our setting of y_e and because $\sum_{e'' \in P(e',r)} d_{i(e')} y_{e''} = 1$, we have

$$\sum_{e'' \in P(e,r)} d_{i(e)} y_{e''} = 1.$$

◀

The following Lemma shows the dual constraints are now satisfied for each $i \in D^s$ even if the blocking set variables z_C are ignored.

▶ **Lemma 14.** *For $i \in D^s$ we have $\sum_{e \in \mathrm{span}(i)} d_i y_e \geq 1$.*

Proof. The statement holds for each i of the form $i(e)$ for some $e \in F$ by Lemma 13 (and noting $P(e,r) \subseteq \mathrm{span}(i)$). So, we suppose that $i \in D^s$ is such that $i \neq i(e)$ for all $e \in F$.

Since $i \in D^s$, there is some endpoint a of i such that $P(a,r) \cap B(i) \neq \emptyset$. By how we selected F, then $P(a,r) \cap B(i) \cap F \neq \emptyset$ as well. Let e be an edge in $P(a,r) \cap B(i) \cap F$ that is furthest from the root. The claim is that $d_i \geq d_{i(e)}$. If so, then $\sum_{e' \in \mathrm{span}(i)} d_i y_{e'} \geq \sum_{e' \in P(e,r)} d_i y_{e'} \geq \sum_{e' \in P(e,r)} d_{i(e)} y_{e'} = 1$ by Lemma 13.

There are two cases.

1. $P(s_i, r) \cap B(i) \cap F = \{e\}$. Then e is critical for i. But since $i(e) \neq i$, it must be that by our choice of $i(e)$ (being the least demand task for which e is critical) that $d_{i(e)} \leq d_i$.
2. $|P(s_i, r) \cap B(i) \cap F| \geq 2$. Since e is furthest from the root, then there is some $e' \neq e$ with $e' \in P(e,r) \cap B(i) \cap F$. If $d_i < d_{i(e)}$, then i was considered before $i(e)$ in Algorithm 1. But since $e' \in F$ blocks i, it would have also blocked $i(e)$ contradicting the fact that e is critical for $i(e)$.

◀

2.6 Putting It All Together

▶ **Lemma 15.** *x and (y, z, z') are feasible for* **Compact-LP** *and its dual and satisfy the relaxed complementary slackness conditions.*

Proof. Clearly x is a feasible solution since it is the indicator vector of the set S selected by the greedy algorithm. Now, y, z and z' are nonnegative by construction. We had set $z'_i = 1$ for each $i \in S$, so the dual constraints for $i \in S$ are satisfied. We also partitioned D^u into subsets of blocking sets and set the z-value for each such blocking set to 1, so the dual constraints for $i \in D^u$ are also satisfied. Finally, the the dual constraints for $i \in D^s$ are satisfied by Lemma 14.

Next we verify the relaxed complementary slackness conditions. By construction, all x_i are $\{0,1\}$-valued. The third condition holds because $y_e > 0$ only for edges that are not undersaturated by S (c.f. Lemma 13).

We set $z_C = 1$ for at most $4 \cdot |S|$ blocking sets, and $z_C = 0$ for the rest. Similarly, $z'_i = 1$ for $i \in S$ and $z'_i = 0$ for $i \notin S$. Thus, the fourth relaxed complementary slackness conditions hold.

The only thing left to prove is that the second relaxed complementary slackness conditions hold. So, consider some $i \in S$. We will show $\sum_{e \in P(a,r)} d_i y_e \leq 1$ for each endpoint a of i. Since each $e \in \mathrm{span}(i)$ lies on some $P(a,r)$ path for some endpoint a of i, then $\sum_{e \in \mathrm{span}(i)} d_i y_e \leq 2$.

Recall the definitions of F and $i(e)$ from Section 2.5. If $P(a,r) \cap F = \emptyset$, then we have $\sum_{e \in P(a,r)} d_i y_e = 0$. Otherwise, let e be the deepest edge $P(a,r) \cap F$. That is, $e \in F$ and $F \cap P(a,r) = F \cap P(e,r)$. It must be that $d_i \leq d_{i(e)}$, otherwise $i(e)$ would have been considered before i in Algorithm 1. This is impossible because e would then have blocked $i \in S$. Thus, $\sum_{e' \in P(a,r)} d_i y_{e'} = \sum_{e' \in P(e,r)} d_i y_{e'} \leq \sum_{e' \in P(e,r)} d_{i(e)} y_{e'} = 1$ where the last equality is by Lemma 13. ◀

3 Approximating Rank-LP

We prove Theorem 5 as a special case of the following more general statement about packing problems. Suppose $Ax \leq b, x \in \{0,1\}^n$ defines the set of feasible solutions to an integer program over n variables where all entries of A are nonnegative. For a nonempty subset of indices $S \subseteq \{1, \ldots, n\}$, let A^S and x^S denote the restriction of A to the columns indexed by S and x to the entries indexed by S. Also let $\text{rank}(S)$ be the largest subset of S that can be packed feasibly. Finally, let $\mathbb{1}$ denote the all-1 vector.

▶ **Lemma 16.** *Let $\mathcal{S}$ be a collection of nonempty subsets of $\{1, \ldots, n\}$. Suppose $\overline{x} \in \mathbb{R}^n$ satisfies $A\overline{x} \leq b$ and $\overline{x} \in [0,1]^n$. Finally, suppose α is an upper bound on the integrality gaps of all (unit-weight) linear programs $\max\{\mathbb{1}^T \cdot x_i^S : A^S x^S \leq b, x^S \in [0,1]^{|S|}\}$ for $S \in \mathcal{S}$. Then for every $S \in \mathcal{S}$ we have $\sum_{i \in S} \overline{x}_i \leq \alpha \cdot \text{rank}(S)$.*

Proof. For any $S \in \mathcal{S}$, $\overline{x}^S$ is feasible for the unit-weight LP $\max\{\mathbb{1}^T \cdot x_i^S : A^S x^s \leq b, x^S \in [0,1]^{|S|}\}$ because A is nonnegative. By the integrality gap assumption, there is a feasible packing of at least $\sum_{i \in S} \overline{x}_i^S / \alpha$ items in S. That is, $\sum_{i \in S} \overline{x}_i^S / \alpha \leq \text{rank}(S)$. ◀

To prove Theorem 5, apply the integrality gap bound from Theorem 4 to Lemma 16, with $\mathcal{S}$ being the collection of all intersecting collections of tasks.

4 Lower Bound

We give an example showing that the integrality gap of **Rank-LP** in weighted, intersecting cases of UFP-TREE can be as bad as $\Omega(\sqrt{\log n})$. Note that an upper bound of $O(\log n)$ for weighted intersecting cases follows from the $O(1)$ upper bound for unit-weight, intersecting cases demonstrated in Section 2 and the reduction in Appendix A.1. This also shows that the our averaging argument using relaxed complementary slackness cannot be adapted to prove a constant gap for weighted intersecting instances.

For any integer $h \geq 2$, we define a tree T^h. Initially, consider a complete tree with height $h-1$ and branching factor 2^{h-1} and say $level_i, 1 \leq i \leq h$, are the vertices in level i of this tree. Finally, we add one additional node r and connect r to the single vertex in $level_1$ to obtain our tree T^h. We say that r is the root of T and that $level_0 = \{r\}$. The number of nodes in T^h is $n = 1 + \frac{2^{h(h-1)}-1}{2^{h-1}-1} \leq 2^{h^2}$. Hence, $h \geq \sqrt{\log_2 n}$.

For each edge uv with $u \in level_{k-1}$ and $v \in level_k$, we set $c_e = 2^{h(h-k+1)}$. Finally, for every $v \in level_k, 1 \leq k \leq h$ we create a single task $i(v)$ with start node v and end node r. We give $i(v)$ demand $d_{i(v)} = 2^{h(h-k+1)} - 2^{h(h-k)}$ and weight $w_{i(v)} = \frac{1}{2^{(k-1)(h-1)}} = \frac{1}{|level_k|}$. That is, for each $1 \leq k \leq h$ we have distributed exactly one unit of weight evenly among the tasks $\{i(v) : v \in level_k\}$.

Figure 2 illustrates the construction of T^h for $h = 3$. For convenience, we define $S(v)$ for a vertex v of T^h to be the set of tasks i where s_i lies in the subtree rooted at v. We begin by establishing a lower bound on the integrality gap of **Compact-LP**.

▶ **Lemma 17.** *The solution $x_i = \frac{1}{2}$ for all tasks i is feasible for* **Compact-LP** *on instance T^h with value $h/2$.*

Proof. Consider the solution $x_i = \frac{1}{2}$ for each task i, which has objective function value $h/2$. We first prove by reverse induction on k that the constraint for an edge $e = uv$ with $u \in level_{k-1}, v \in level_k$ is satisfied by x. This is clearly true for $k = h$.

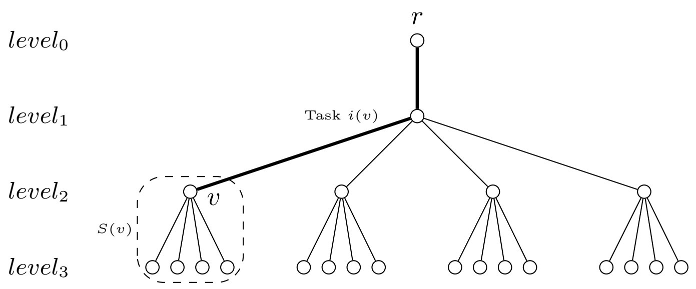


Figure 2 Bad instance T^3. The span of task $i(v)$ is drawn with bold lines. The outlined group of nodes are the starting points of tasks in $S(v)$.

Inductively, consider $k < h$ and suppose no child edge of v has its corresponding constraint violated by x. Recall that there are 2^{h-1} children of v, each with capacity $2^{h(h-k)}$. By induction, the total fractional demand from tasks in $S(v)$ in the solution x is at most

$$\frac{d_{i(v)}}{2} + \sum_{u \text{ child of } v} c_{vu} \leq \frac{c(e)}{2} + 2^{h(h-k)} \cdot 2^{h-1} = c(e)$$

so the Constraint (1) for edge e is satisfied.

Note that $\{i, j\}$ is feasible for *any* tasks i, j. That is, suppose $\{i, j\}$ violated the capacity of some edge c_{uv}. Then $i, j \in S(v)$ and if $\{i, j\}$ violates the capacity of uv, then so to does $i(v), i(w)$ for some child w of v. But a simple calculation shows $d_{i(v)} + d_{i(w)} \leq c_{uv}$, which is a contradiction. This means Constraints (2) of **Compact-LP** are vacuous, thus trivially satisfied. ◀

▶ **Lemma 18.** *Every* UFP-TREE *solution in T^h has value at most 2.*

Proof. Consider any edge $e = (u, v)$ where $u \in level_{k-1}$ and $v \in level_k$. Note that $S(v_1)$ is the set of all tasks where $level_1 = \{v_1\}$. Thus, it suffices to show the following.

Claim: If $v \in level_k$, the maximum weight of a feasible subset $I \subseteq S(v)$ is at most $\frac{2}{2^{(h-1)(k-1)}}$.

We prove this claim by induction on k from h to 1. Clearly, for $k = h$ it is true since the weight of each task from the lowest level is $\frac{1}{2^{(h-1)(h-1)}}$. Inductively, consider $k < h$ and suppose the statement is true for all $v' \in level_{k+1}$.

Case a: $i(v) \notin I$.
In this case, I is a union of feasible solutions $I_w \subseteq S(w)$ for each child w of v. By the induction hypothesis, the weight of each I_w is at most $\frac{2}{2^{(h-1)k}}$. Since there are 2^{h-1} children of v, then the weight of I is bounded by $2^{h-1}\frac{2}{2^{(h-1)k}} = \frac{2}{2^{(h-1)(k-1)}}$.

Case b: $i(v) \in I$.
The weight of $i(v)$ is $\frac{1}{2^{(h-1)(k-1)}}$ and the remaining capacity of e is $2^{h(h-k)}$. By how we set the demands and weights, it is not hard to see that the task from the lowest level have the largest density (i.e. w_i/d_i), which is $\frac{1}{2^h-1} \cdot \frac{1}{2^{(h-1)\cdot(h-1)}} \leq \frac{1}{2^{h(h-1)}}$. Hence, the weight of $I - i(v)$ is at most $\frac{2^{h(h-k)}}{2^{h(h-1)}} \leq \frac{1}{2^{(h-1)(k-1)}}$. Adding this to $w_{i(v)}$ completes the proof. ◀

By Lemmas 17 and 18, the integrality gap of **Compact-LP** is at least $h/4 = \Omega(\sqrt{\log n})$. To complete the proof of Theorem 6, simply note that since the all-1/2 solution is feasible for **Compact-LP** then the solution x with $x_i = 1/18$ for all tasks i is feasible for **Rank-LP** by Theorem 5. Thus, the integrality gap of **Rank-LP** is also $\Omega(\sqrt{\log n})$.

5 Conclusion

We saw how adding only $O(n \cdot m)$ constraints to the natural LP relaxation for UFP-TREE reduces the integrality gap from $\Omega(n)$ to $O(\log n \cdot \min\{\log m, \log n\})$. Unfortunately, we also know that including all rank constraints does not reduce the gap to a constant. The bad gap example we demonstrated has all tasks sharing a common endpoint. Interestingly, such instances admit an FPTAS.

Our analysis of the upper bound of **Rank-LP** may not be tight. It may also be possible to further strengthen the LP. Closing the gap between the upper and lower bound is an important problem, especially since UFP-TREE has been a testbed for more general column-restricted packing LP ideas (e.g. [9, 11]).

It would also be interesting to determine the integrality gap of **Rank-LP** on unit-weight instances of UFP-TREE that are not necessarily intersecting. In UFP-PATH, it is known to be $O(1)$ [1]. If it is also constant in UFP-TREE, then this immediately leads to an $O(\log n)$-approximation in general. On the other hand, if this gap is super-constant then this may indicate that UFP-TREE has no constant-factor approximation.

For the more general problem k-TREEPACKING, we gave an $O(k)$ upper bound on the integrality gap of **Compact-LP** (in Appendix B), matching the guarantee of the combinatorial approximation implicit in [9]. Corollary 1 means that we cannot find significantly better approximations, but it may still be possible to get a $o(k)$-approximation. In particular, there is an $\tilde{O}\left(\frac{k}{\log^2 k}\right)$-approximation for the Maximum Independent Set problem in degree $\leq k$ graphs [15] (the tilde is supressing $\log \log k$ terms). Our integrality gap analysis for **Compact-LP** was asymptotically tight, so other techniques must be considered to get a slightly better approximation.

Acknowledgements. The authors thank Joseph Cheriyan and Chaitanya Swamy for many helpful discussions.

References

1 A. Anagnostopoulos, F. Grandoni, S. Leonardi, and A. Wiese. Constant integrality gap LP formulations of unsplittable flow on a path. In proceedings of IPCO, 2013.
2 A. Anagnostopoulos, F. Grandoni, S. Leonardi, and A. Wiese. A mazing $(2+\epsilon)$-approximation for unsplittable flow on a path. In proceedings of SODA, 2014.
3 N. Bansal, A. Chakrabarti, A. Epstein, and B. Scheiber. A quasi-PTAS for unsplittable flow on line graphs. In proceedings of STOC, 2006.
4 J. Batra, N. Garg, A. Kumar, T. Mömke, and A. Wiese, New approximation schemes for unsplittable flow on a path. In proceedings of SODA, 2015.
5 P. Austrin, S. Khot, and M. Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. Theory of Computing, 7:27–43, 2007.
6 P. Bonsma, J. Schulz, and A. Wiese. A constant-factor approximation for unsplittable flow on paths. In proceedings of FOCS, 2011.
7 S. O. Chan, Approximation resistance from pairwise independent subgroups. In proceedings of STOC, 2013.

8 A. Chakrabarti, C. Chekuri, A. Kumar, and A. Gupta. Approximation algorithms for the unsplittable flow problem. Algorithmica, 47(1):53–78, 2007.

9 C. Chekuri, A. Ene, and N. Korula. Unsplittable flow on paths, trees, and column-restricted packing integer programs. In proceedings of APPROX, 2009. Full version with additional results available at `http://www.cs.princeton.edu/~aene/research.html`.

10 C. Chekuri, S. Khanna, and B. Shepherd. An $O(\sqrt{n})$-approximation and integrality gap for disjoint paths and unsplittable flow. Theory of Computing, 2, 137–146, 2006.

11 C. Chekuri, M. Mydlarz, and B. Shepherd. Multicommodity demand flow in a tree and packing integer programs. ACM Trans. on Algorithms 3(3), 2007.

12 E. Chlamtáč and M. Tulsiani. Convex relaxations and integrality gaps. Handbook on Semidefinite, Conic and Polynomial Optimization, Springer, 2012.

13 E. Chlamtáč, Z. Friggstad, and K. Georgiou. Lift-and-project methods for set cover and knapsack. In proceedings of WADS, 2013.

14 N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximations for integral flow and multicut in trees. Algorithmica, 18(1):3–20, 1997.

15 N. Bansal, A. Gupta, and G. Guruganesh. On the Lovász theta function for independent sets. In proceedings of STOC, 2015.

16 A. Karlin, C. Mathieu, and C. Nguyen. Integrality gaps of linear and semi-definite programming relaxations for knapsack. In proceedings of IPCO, 2011.

17 A. Schrijver. Combinatorial Optimization - Polyhedra and Efficiency. Springer, 2003.

A Reduction to Unit-Weight, Intersecting Cases

The proof of Lemma 10 uses essentially the same arguments as in [9]. We start with a general instance of UFP-TREE.

A.1 Reduction to Unit-Weight Instances

The idea is to bucket the tasks by their weight and round each bucket separately. The general idea works for every packing problem, not just UFP-TREE.

Consider a feasible LP solution x to **Compact-LP**. Let W be the maximum weight of the tasks. We know the value of x is at least W because we are assuming each task is feasible by itself. Discard all tasks i with $w_i \leq \frac{W}{2n}$ and let x' denote the restriction of x to the remaining tasks. Since we discarded at most n tasks and since $x_i \leq 1$ for each tasks i, then $w'^T \cdot x' \geq w^T \cdot x/2$ (where w' is the restriction of w to the remaining tasks).

For $a \in \{1, \ldots, \lceil \log_2 2n \rceil\}$, form the "bucket" $B_a = \{i : 2^{-a}W < w_i \leq 2^{-a+1}W\}$. Notice that there are $O(\log n)$ different buckets B_a and they partition the remaining tasks. For each such a, let x^a denote the restriction of x to tasks in B_a.

Now, x^a is a feasible LP solution **Compact-LP**. If we know the integrality gap is β for unit-weight instances, then we can find a feasible set of at least $\sum_{i \in B_a} x_i/\beta$ tasks in B_a. The weight of these tasks is at least $(w^a)^T \cdot x^a/2\beta$ (where w^a denotes the restriction of w to B_a) and the best solution found among all buckets B_a has weight at least $w'^T \cdot x'/2\beta\lceil \log_2 2n \rceil$. Thus, the integrality gap is $O(\log n \cdot \beta)$.

Note that β will depend on the input size in the coming arguments, namely $\beta = O(\min\{\log m, \log n\})$. However, since it is non-decreasing with the input size then the above arguments remain valid.

A.2 Reduction to Intersecting Instances

Recall that we are assuming the integrality gap of unit-weight, intersecting cases is bounded by $\alpha = O(1)$. We describe a recursive algorithm, with the base case being when the instance itself is already intersecting. In this case, there is a feasible set of tasks S of size at least $\sum_i x_i/\alpha$ by assumption

Otherwise, recall that every tree T has a "centre" node v such that the number of edges in each component of $T - v$ is at most half the number of edges m of T. Fix such a centre node v and let $\mathcal{I}_v$ be the set of tasks spanning v. The integrality gap assumption means there is a feasible subset S of $\mathcal{I}_v$ with size at least $\sum_{i\in\mathcal{I}_v} x_i/\alpha$.

Let $T^{(1)}, \ldots, T^{(b)}$ denote the connected subtrees of T that remain after v is deleted. For each $i \notin \mathcal{I}_v$, the entire $s_i - t_i$ path is entirely contained in some $T^{(j)}$ so we consider the b different UFP-TREE instances defined by each $T^{(j)}$ and the tasks contained entirely within $T^{(j)}$, say $\mathcal{I}_j$. Furthermore, the restriction of x to each of these subinstances is feasible for that instance. Finally, each $T^{(j)}$ has at most $m/2$ edges by our choice of v.

Recursively, we find a feasible set of tasks S_j of size at least $\sum_{i\in\mathcal{I}_j} x_i/(\alpha \cdot \log_2(m/2))$ from each subinstance I_j. The set $\cup_j S_j$ is feasible because no two tasks contained in different subinstances span a common edge. This gives us a feasible solution $\cup_j S_j$ of size $\sum_{i\notin\mathcal{I}_v} x_i/(\alpha \cdot \log_2(m/2))$.

Keep the largest of S or $\cup_j S_j$ as our solution for the instance on the tree T. A quick calculation shows that $\max\{|S|, |\cup_j S_j|\}$ has size at least $\sum_i x_i/(\alpha \cdot \log_2 m\cdot)$. That is, the integrality gap of **Compact-LP** in unit-weight instances is at most $\alpha \cdot \log_2 m = O(\log m)$.

A.3 Reducing the Number of Edges

Combining the previous two reductions shows the integrality gap is at most $O(\log n \cdot \log m)$. We can also bound the integrality gap by $O(\log^2 n)$ by performing the following preprocessing step before applying the previous two reductions.

Consider a node v of T that has degree at most 2 and is not an endpoint of any task. If v has degree 2 with incident edges uv and vw, we remove v from T and add the edge uw with capacity $\min\{c_{uv}, c_{vw}\}$. If v is a leaf of T, then we just discard v and its incident edge from T. In either case, the set of feasible solutions x to **Compact-LP** does not change.

Let m' be the number of edges in the resulting tree, the claim is that $m' \leq 4n$. To see this, recall that the number of edges in a tree with ℓ leaves and b degree 2 nodes is at most $2\ell + b$. The only leaves and degree 2 nodes in the resulting tree are endpoints of one of the n tasks, so there are at most $2n$ leaves and degree 2 nodes. Thus, $m' \leq 4n$.

Applying the previous two reductions to this tree that we obtained, we see the integrality gap can also be bounded by $O(\log n \log m')$, which is bounded by $O(\log^2 n)$.

B Extensions to k-TreePacking

Here we briefly discuss how to modify the algorithm and analysis from Section 2 to get integrality gap bounds for k-TREEPACKING. Recall that each subtree T_i in the input has at most k leaves.

We use similar notation, for a subset S of input tasks/subtrees we let $\text{rank}(S)$ denote the largest subset of S that is feasible. A subset S is called intersecting if there is a vertex r that lies on all subtrees T_i for tasks in S. In this way, **Rank-LP** can also be regarded as a relaxation for k-TREEPACKING.

Because the integrality gap of **Rank-LP** is just 1 in intersecting, unit-weight instances of k-TreePacking and because such instances are hard to approximate within factors close to k (c.f. Corollary 1), we cannot hope to solve this LP within a factor that is much better than k. We will sketch how to solve it within a factor of $4k+1$ by adapting our approach for UFP-tree.

First, we generalize the notion of a blocking set. For any vertex v, any i such that the subtree T_i contains v, and any leaf node a of T_i we let $C(i,v,a)$ denote the set containing i and all j such 1) $d_j \geq d_i$, 2) T_j spans v, and 3) $d_i + d_j > c_e$ for some edge $e \in P(a,v) \cap T_j$. Lemma 12 and its proof generalize without effort to k-TreePacking.

▶ **Lemma 19.** *For any such i, v, a,* $\text{rank}(C(i,v,a)) \leq 1$.

Thus, we may also consider the generalization of **Compact-LP** to k-TreePacking. It has $O(n \cdot m \cdot k)$ constraints. Before discussing the integrality gap upper bound, we begin by providing a lower bound.

Proof of Lemma 8. Let T be a star with $\binom{k+1}{2}$ leaves. Index the leaves by subsets of $\{1,\ldots,k+1\}$ of size 2. For each $1 \leq i \leq k$, create a subtree T_i with leaves being the k leaves of T that correspond to pairs containing i. Set all demands, capacities, and weights to 1.

The solution $x_i = \frac{1}{2}$ is feasible for **Compact-LP** since each blocking set has size at most 2. However, the optimum k-TreePacking solution picks only a single subtree, as selecting any pair of subtrees T_i, T_j would violate the capacity of the edge incident to of leaf $\{i,j\}$. ◀

Finally, our upper bound for k-TreePacking is the following.

▶ **Theorem 20.** *The integrality gap of* **Compact-LP** *for k-TreePacking is at most $4k+1$ in intersecting, unit-weight instances and is $O(k \cdot \log n \cdot \min\{\log m, \log(kn)\})$ in general instances.*

Rather than presenting the whole proof from scratch, we just mention how to generalize the proof for UFP-tree to this setting.

The algorithm for unit-weight, intersecting instances is the same as Algorithm 1 for UFP-tree: greedily try to add subtrees in increasing order of demand and form the sets D^u, D^s for the tasks i that are not included in the final solution S. For a subtree T_i, let span(i) naturally denote the set of edges lying on T_i. The relaxed complementary slackness conditions we consider are:

1. $x_i \in \{0,1\}$ for each subtree T_i
2. $x_i = 1 \Longrightarrow \sum_{e \in \text{span}(i)} d_i \cdot y_e \leq k$
3. $y_e > 0 \Longrightarrow \sum_{i: e \in \text{span}(i)} d_i \cdot x_i \geq \frac{c_e}{2}$
4. $\sum_C z_C + \sum_i z'_i \leq (2k+1)\sum_i x_i$

The proof of why this suffices is the same as the proof of Lemma 11, except we choose $\alpha = \frac{2k}{4k+1}, \beta = \frac{2k+1}{4k+1}$.

We still set $z'_i = 0$ for $i \notin S$ and $z'_i = 1$ for $i \in S$. The set D^u is partitioned into at most $2k \cdot |S|$ sets, each of which can be shown to be contained in some blocking set $C(i,v,a)$. More specifically, the steps in Section 2.4 are adapted to this setting in the following way. Construct the subtree T' of T consisting of edges used by tasks in S and note that D^u will have at most $k \cdot |S|$ leaves. Partitioning T' into maximal paths (again, perhaps also splitting the path that goes through the root) produces at most $2k \cdot |S|$ paths, and the tasks $C(P)$ that were blocked by an undersaturated edge on P can be shown to be contained in some in the same way as in the proof of Lemma 12. This shows the last relaxed complementary slackness condition holds.

The setting of the dual variables y_e is essentially the same and the second complementary slackness condition holds because this construction ensures $\sum_{e \in P(a,r)} d_i \cdot y_e \leq 1$ for each $i \in S$ and each of the k leaves a of T_i. This also satisfies the third condition because positive dual is assigned only to y_e variables that are mostly saturated.

Finally, to get the $O(k \cdot \log n \cdot \min\{\log m, \log(kn)\})$ bound in the general case we reduce to the unit-weight case and lost an $O(\log n)$ as in Appendix A.1 and the reduction to intersecting instances is the same as Appendix A.2 and loses an additional $O(\log m)$-factor.

Finally, an easy adaptation of the preprocessing in Appendix A.3 reduces the number of edges in the tree to at most $2nk$. That is, we can merge the edges incident to a degree-2 vertex that is not an endpoint of some task and remove leaf nodes do not lie on any subtree. This reduction produces a tree where the number of leaf nodes plus the number of internal degree-2 nodes is at most nk, meaning it has $O(nk)$ edges overall.

C Lift-and-Project Bounds

The definitions of the hierarchies discussed here can be found in [12], for example. Let $\mathcal{P}$ denote the polytope defined by the constraints of **Nat-LP**. For an integer $t \geq 0$, let Las^t and LS_+^t denote t rounds of the Lasserre and Lovász-Schrijver SDP operators, respectively.

The following lemma is stated for k-TreePacking, but it generalizes immediately to any relaxation of a packing integer program. It is easy to see it holds if LS_+^2 is replaced by Las^2 by invoking the decomposition theorem of Karlin, Mathieu, and Nguyen [16]. However, it is interesting to note that the result still holds in the weaker Lovász-Schrijver SDP hierarchy.

▶ **Lemma 21.** *Suppose $x \in \text{LS}_+^2(\mathcal{P})$. For any pairwise infeasible clique S of subtrees, $\sum_{i \in S} x_i \leq 1$.*

Proof. Suppose $x \in \text{LS}_+^2$ and let $Y \succeq 0$ be a protection matrix for x. Y is indexed by the subtrees $1 \leq i \leq n$ and one additional index which we denote by 0. Then Y is symmetric and $Y_0 = \text{diag}(Y) = (1, x)$ where Y_0 is the first row of Y. We also claim that $Y_{i,j} = 0$ whenever $\{i, j\}$ is an infeasible pair of subtrees.

Consider any distinct pair of subtrees i, j with $Y_{i,j} > 0$. We can condition on $x_i = 1$ to get a point $x' \in \text{LS}_+^1(\mathcal{P})$ with $x'_{i'} = \frac{Y_{i,i'}}{Y_{i,i}}$ for any subtree i'. In particular, $x'_i = 1$ and $x'_j > 0$. We can further condition on $x'_j = 1$ to get a point $x'' \in \mathcal{P}$ that has both $x''_i = x''_j = 1$. Thus, $\{i, j\}$ is a feasible pair of subtrees.

Finally, we verify $\sum_{i \in S} x_i \leq 1$ for any pairwise infeasible clique of subtrees S, meaning x is a feasible solution to **Compact-LP**. This follows by standard theta body theory for graphs (e.g. Chapter 67 of [17]) since Y witnesses the inclusion of x in the theta body of the graph H whose vertices correspond to subtrees and whose edges correspond to infeasible pairs of subtrees. However, the argument is simple so we include it for completeness.

Consider the vector z with $z_0 = 1, z_i = -1$ for $i \in S$ and $z_i = 0$ for $v \notin S$. Because $Y \succeq 0$ we have the following bound. Note, the indices in the sums on the first line below range over all subtrees i but not index 0.

$$\begin{aligned}0 \le z^T Y z &= z_0 Y_{0,0} z_0 + 2\sum_i z_0 z_i Y_{0,i} + \sum_{i,j} z_i z_j Y_{i,j} \\ &= 1 - 2\sum_{i\in S} x_i + \sum_{i,j\in S} Y_{i,j} \\ &= 1 - 2\sum_{i\in S} x_i + \sum_{i\in S} x_i \\ &= 1 - \sum_{i\in S} x_i.\end{aligned}$$

The first and second equalities follow simply by definition of z and the fact that Y is symmetric with $Y_0 = (1, x)$. The third equality uses $Y_{i,j} = 0$ for distinct $i, j \in S$ and $\mathrm{diag}(Y) = (1, x)$. ◀

In fact, this proof does not require the "level 1" protection matrices for $x \in \mathrm{LS}^2_+(\mathcal{P})$ to be positive semidefinite.

Lemma 9 immediately follows Lemma 21 and Theorem 7. In fact, the integrality gaps are reduced even further in special cases of UFP-PATH that were studied in [1, 9]. By how we proved Theorem 5, if x is feasible for **Compact-LP** in a UFP-PATH instance then $x/9$ is feasible for **Rank-LP**. All integrality gaps mentioned in the following corollary are known to hold in **Rank-LP**, so they also hold (within a factor of 9) in the mentioned SDPs. Again, recall that $\mathcal{P}$ is the polytope defined by the constraints of **Nat-LP**.

▶ **Corollary 22.** *The integrality gap of the SDP* $\max\{w^T \cdot x : x \in \mathrm{LS}^2_+(\mathcal{P})\}$ *is* $O(1)$ *in intersecting, unit-weight instances of* UFP-TREE, $O(\min\{\log m, \log n\})$ *for* UFP-PATH *instances, and* $O(1)$ *in intersecting or unit-weight instances of* UFP-PATH.

Inapproximability of H-Transversal/Packing

Venkatesan Guruswami[*] **and Euiwoong Lee**[†]

Carnegie Mellon University
Pittsburgh, PA 15213, USA
guruswami@cmu.edu, euiwoonl@cs.cmu.edu

Abstract

Given an undirected graph $G = (V_G, E_G)$ and a fixed pattern graph $H = (V_H, E_H)$ with k vertices, we consider the H-Transversal and H-Packing problems. The former asks to find the smallest $S \subseteq V_G$ such that the subgraph induced by $V_G \setminus S$ does not have H as a subgraph, and the latter asks to find the maximum number of pairwise disjoint k-subsets $S_1, ..., S_m \subseteq V_G$ such that the subgraph induced by each S_i has H as a subgraph.

We prove that if H is 2-connected, H-Transversal and H-Packing are almost as hard to approximate as general k-Hypergraph Vertex Cover and k-Set Packing, so it is NP-hard to approximate them within a factor of $\Omega(k)$ and $\widetilde{\Omega}(k)$ respectively. We also show that there is a 1-connected H where H-Transversal admits an $O(\log k)$-approximation algorithm, so that the connectivity requirement cannot be relaxed from 2 to 1. For a special case of H-Transversal where H is a (family of) cycles, we mention the implication of our result to the related Feedback Vertex Set problem, and give a different hardness proof for directed graphs.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Constraint Satisfaction Problems, Approximation resistance

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.284

1 Introduction

Given a collection of subsets $S_1, ..., S_m$ of the underlying set U, the *Set Transversal* problem asks to find the smallest subset of U that intersects every S_i, and the *Set Packing* problem asks to find the largest subcollection $S_{i_1}, ..., S_{i_{m'}}$ which are pairwise disjoint.[1] It is clear that optimum of the former is always at least that of the latter (i.e. weak duality holds). Studying the (approximate) reverse direction of the inequality (i.e. strong duality) as well as the complexity of both problems for many interesting classes of set systems is arguably the most studied paradigm in combinatorial optimization.

This work focuses on set systems where the size of each set is bounded by a constant k. With this restriction, Set Transversal and Set Packing are known as k-Hypergraph Vertex cover (k-HVC) and k-Set Packing (k-SP), respectively. This assumption significantly simplifies the problem since there are at most n^k sets. While there is a simple factor k-approximation algorithm for both problems, it is NP-hard to approximate k-HVC and k-SP within a factor less than $k-1$ [21] and $O(\frac{k}{\log k})$ [34] respectively.

[*] Supported in part by NSF grant CCF-1115525.
[†] Supported by a Samsung Fellowship and NSF CCF-1115525.
[1] These problems are called many different names in the literature. Set Transversal is also called Hypergraph Vertex Cover, Set Cover (of the dual set system), and Hitting Set. Set Packing is also called Hypergraph Matching. We try to use Transversal / Packing unless another name is established in the literature (e.g. k-Hypergraph Vertex Cover).

Given a large graph $G = (V_G, E_G)$ and a fixed graph $H = (V_H, E_H)$ with k vertices, one of the natural attempts to further restrict set systems is to set $U = V_G$, and take the collection of subsets to be all *copies* of H in G (formally defined in the next subsection). This natural representation in graphs often results in a deeper understanding of the underlying structure and better algorithms, with Maximum Matching ($H = K_2$) being the most well-known example. Kirkpatrick and Hell [39] proved that Maximum Matching is essentially the only case where H-Packing can be solved exactly in polynomial time – unless H is the union of isolated vertices and edges, it is NP-hard to decide whether V_G can be partitioned into k-subsets each inducing a subgraph containing H. A similar characterization for the edge version (i.e. $U = E_G$) was obtained much later by Dor and Tarsi [22].

We extend these results by studying the approximability of H-Transversal and H-Packing. We use the term *strong inapproximability* to denote NP-hardness of approximation within a factor $\Omega(k/polylog(k))$. We give a simple sufficient condition that implies strong inapproximability – if H is 2-vertex connected, H-Transversal and H-Packing are almost as hard to approximate as k-HVC and k-SP. We also show that there is a 1-connected H where H-Transversal admits an $O(\log k)$-approximation algorithm, so 1-connectivity is not sufficient for strong inapproximability for H-Transversal. It is an interesting open problem whether 1-connectivity is enough to imply strong inapproximability of H-Packing, or there is a class of connected graphs where H-Packing admits a significantly nontrivial approximation algorithm (e.g. factor k^ϵ for some $\epsilon < 1$).

Our results give an unified answer to questions left open in many independent works studying a special case where H is a cycle or clique, and raises some new open questions. In the subsequent subsections, we state our main results, review related work, and state potential future directions.

1.1 Problems and Our Results

Given an undirected graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ with $|V_H| = k$, we define the following problems.

- H-Transversal asks to find the smallest $F \subseteq V_G$ such that the subgraph of G induced by $V_G \setminus F$ does not have H as a subgraph.
- H-Packing asks to find the maximum number of pairwise disjoint k-subsets of $S_1, ..., S_m$ of V_G such that the subgraph induced by each S_i has H as a subgraph.

Our main result states that 2-connectivity of H is sufficient to make H-Transversal and H-Packing hard to approximate.

▶ **Theorem 1.** *If H is a 2-vertex connected with k vertices, unless* NP $\subseteq$ BPP, *no polynomial time algorithm approximates H-Transversal within a factor better than $k - 1$, and H-Packing within a factor better than $\Omega(\frac{k}{\log^7 k})$.*

Let k-Star denote $K_{1,k-1}$, the complete bipartite graph with 1 and $k - 1$ vertices on each side. The following theorem shows that k-Star Transversal admits a good approximation algorithm, so the assumption of 2-connectedness in Theorem 1 is required for strong inapproximability of H-Transversal.

▶ **Theorem 2.** *k-Star Transversal can be approximated within a factor of $O(\log k)$ in polynomial time.*

This algorithmic result matches $\Omega(\log k)$-hardness of k-Star Transversal via a simple reduction from Minimum Dominating Set on degree-k graphs [16]. This problem has the following equivalent but more natural interpretation: given a graph $G = (V_G, E_G)$, find the smallest $F \subseteq V_G$ such that the subgraph induced by $V_G \setminus F$ has maximum degree at most $k-2$. Our algorithm, which uses iterative roundings of 2-rounds of Sherali-Adams hierarchy of linear programming (LP) followed by a simple greedy algorithm for *Constrained Set Cover*, is also interesting in its own right, but we defer the details to Appendix A.

Our hardness results for transversal problems rely on hardness of k-HVC which is NP-hard to approximate within a factor better than $k-1$ [21]. Our hardness results for packing problems rely on hardness of Maximum Independent Set on graphs with maximum degree k and girth strictly *greater than* g (MIS-k-g). Almost tight inapproximability of MIS on graphs with maximum degree k (MIS-k) is recently proved in Chan [11], which rules out an approximation algorithm with ratio better than $\Omega(\frac{k}{\log^4 k})$. We are able to extend his result to MIS-k-g with losing only a polylogarithmic factor. All applications in this work require $g = \Theta(k)$.

▶ **Theorem 3.** *For any constants k and g, unless* NP $\subseteq$ BPP*, no polynomial time algorithm approximates MIS-k-g within a factor of* $\Omega(\frac{k}{\log^7 k})$.

We remark that assuming the Unique Games Conjecture (UGC) slightly improves our hardness ratios through better hardness of k-HVC [38] and MIS-k [3], and even simplifies the proof for some problems (e.g. k-Clique Transversal) through structured hardness of k-HVC [5]. Indeed, an earlier (unpublished) version of this work [30] relied on the UGC to prove that MIS-k-k is hard to approximate within a factor of $\Omega(\frac{k}{\log^4 k})$, while only giving $\widetilde{\Omega}(\sqrt{k})$-factor hardness without it. Now that we obtain almost matching hardness, we focus on proving hardness results without the UGC.

1.2 Related Work and Special Cases

After the aforementioned work characterizing those pattern graphs H admitting the existence of a polynomial-time exact algorithm for H-Packing [39, 22], Lund and Yannakakis [45] studied the maximization version of H-Transversal (i.e. find the largest $V' \subseteq V_G$ such that the subgraph induced by V' does not have H as a subgraph), and showed it is hard to approximate within factor $2^{\log^{1/2-\epsilon} n}$ for any $\epsilon > 0$. They also mentioned the minimization version of two extensions of H-Transversal. The most general node-deletion problem is APX-hard for every nontrivial hereditary (i.e. closed under node deletion) property, and the special case where the property is characterized by a finite number of forbidden subgraphs (i.e. $\{H_1, ..., H_l\}$-Transversal in our terminology) can be approximated with a constant ratio. They did not provide explicit constants (one trivial approximation ratio for $\{H_1, ..., H_l\}$-Transversal is $\max(|V_{H_1}|, ..., |V_{H_l}|)$), and our result can be viewed as a quantitative extension of their inapproximability results for the special case of H-Transversal.

H-Transversal / Packing has been also studied outside the approximation algorithms community. The duality between our H-Transversal and H-Packing is closely related to the famous Erdős-Pósa property actively studied in combinatorics. The recent work of Jansen and Marx [36] considered problems similar to our H-Packing with respect to fixed-parameter tractability (FPT).

Many other works on H-Transversal / Packing focus on a special case where H is a cycle or clique. We define k-Cycle (resp. k-Clique) to be the cycle (resp. clique) on k vertices.

1.2.1 Cycles

The initial motivation for our work was to prove a super-constant factor inapproximability for the Feedback Vertex Set (FVS) problem without relying on the Unique Games Conjecture. Given a (directed) graph G, the FVS problem asks to find a subset F of vertices with the minimum cardinality that intersects every cycle in the graph (equivalently, the induced subgraph $G \setminus F$ is acyclic). One of Karp's 21 NP-complete problems, FVS has been a subject of active research for many years in terms of approximation algorithms and fixed-parameter tractability (FPT). For FPT results, see [8, 14, 18, 15] and references therein.

FVS on undirected graphs has a 2-approximation algorithm [4, 7, 17], but the same problem is not well-understood in directed graphs. The best approximation algorithm [48, 26, 25] achieves an approximation factor of $O(\log n \log\log n)$. The best hardness result follows from a simple approximation preserving reduction from Vertex Cover, which implies that it is NP-hard to approximate FVS within a factor of 1.36 [20]. Assuming UGC [37], it is NP-hard to approximate FVS in directed graphs within any constant factor [29, 50] (we give a simpler proof in [30]). The main challenge is to *bypass* the UGC and to show a super-constant inapproximability result for FVS assuming only $\mathsf{P} \neq \mathsf{NP}$ or $\mathsf{NP} \not\subseteq \mathsf{BPP}$.

By Theorem 1, we prove that k-Cycle Transversal is hard to approximate within factor $\Omega(k)$. In the full version of this work [31], we prove the following theorem that improves the result of Theorem 1 in the sense that in the completeness case, a small number of vertices not only intersect cycles of length exactly k, but intersect every cycle of length $3, 4, ..., O(\frac{\log n}{\log\log n})$.

▶ **Theorem 4.** *Fix an integer $k \geq 3$ and $\epsilon \in (0, 1)$. Given a graph $G = (V_G, E_G)$ (directed or undirected), unless $\mathsf{NP} \subseteq \mathsf{BPP}$, there is no polynomial time algorithm to tell apart the following two cases.*

- *Completeness: There exists $F \subseteq V_G$ with $\frac{1}{k-1} + \epsilon$ fraction of vertices that intersects every cycle of at most length $O(\frac{\log n}{\log\log n})$ (hidden constant in O depends on k and ϵ).*
- *Soundness: Every subset F with less than $1 - \epsilon$ fraction of vertices does not intersect at least one cycle of length k. Equivalently, any subset with more than ϵ fraction of vertices has a cycle of length exactly k in the induced subgraph.*

This can be viewed as some (modest) progress towards showing inapproximability of FVS in the following sense. Consider the following standard linear programming (LP) relaxation for FVS.

$$\min \sum_{v \in V_G} x_v \quad \text{subject to} \quad \sum_{v \in C} x_v \geq 1 \quad \forall \text{ cycle } C\,, \quad \text{and} \quad 0 \leq x_v \leq 1 \;\; \forall v \in V_G$$

The integrality gap of the above LP is upper bounded by $O(\log n)$ for undirected graphs [6] and $O(\log n \log\log n)$ for directed graphs [26]. Suppose in the completeness case, there exists a set of measure c that intersects every cycle of length at most $\log^{1.1} n$ (or any number bigger than the known integrality gaps). If we remove these vertices and consider the above LP on the remaining subgraphs, since every cycle is of length at least $\log^{1.1} n$, setting $x_v = 1/\log^{1.1} n$ is a feasible solution, implying that the optimal solution to the LP is at most $n/\log^{1.1} n$. Since the integrality gap is at most $O(\log n \log\log n)$, we can conclude that the remaining cycles can be hit by at most $O(n \log\log n/\log^{0.1} n) = o(n)$ vertices, extending the completeness result to every cycle. Thus, improving our result to hit cycles of length $\omega(\log n \log\log n)$ in the completeness case will prove a factor-$\omega(1)$ inapproximability of FVS.

Another interesting aspect about Theorem 4 is that it also holds for undirected graphs. This should be contrasted with the fact that undirected graphs admit a 2-approximation algorithm for FVS, suggesting that to overcome $\log n$-cycle barrier mentioned above, some properties of directed graphs must be exploited. Towards developing a directed graph specific approach, we also present a different reduction technique called *labeling gadget*. It has an additional advantage of being derandomized and assumes only $\mathsf{P} \neq \mathsf{NP}$.

For cycles of bounded length, Kortsarz et al. [41] studied k-Cycle Edge Transversal, and suggested a $(k-1)$-approximation algorithm as well as proved that improving the ratio 2 for K_3 will have the same impact on Vertex Cover, refuting the Unique Games Conjecture [38].

For the dual problem of packing cycles of any length, called Vertex-Disjoint Cycle Packing (VDCP), the results of [42, 28] imply that the best approximation factor by any polynomial time algorithm lies between $\Omega(\sqrt{\log n})$ and $O(\log n)$. In a closely related problem Edge-Disjoint Cycle Packing (EDCP), the same papers showed that $\Theta(\log n)$ is the best possible. In directed graphs the vertex and edge version have the same approximability, the best known algorithms achieves $O(\sqrt{n})$-approximation while the best hardness result remains $\Omega(\log n)$.

Variants of k-Cycle Packing have also been considered in the literature. Rautenbach and Regen [47] studied k-Cycle Edge Packing on graphs with girth k and small degree. Chalermsook et al. [10] studied a variant of k-Cycle Packing on directed graphs for $k \geq n^{1/2}$ where we want to pack as many disjoint cycles of length at most k as possible, and proved that it is NP-hard to approximate within a factor of $n^{1/2-\epsilon}$. This matches the algorithm implied by [42].

1.2.2 Cliques

Minimum Maximal (resp. Maximum) Clique Transversal asks to find the smallest subset of vertices that intersects every maximal (resp. maximum) clique in the graph. In mathematics, Tuza [51] and Erdős et al. [24] started to estimate the size of the smallest such set depending on structure of graphs. See the recent work of Shan et al. [49] and references therein. In computer science, exactly computing the smallest set on special classes of graphs appears in many works [32, 44, 12, 23, 43].

Both the edge and vertex version of k-Clique Packing also have been studied actively both in mathematics and computer science. In mathematics, the main focus of research is lower bounding the maximum number of edge or vertex-disjoint copies of K_k in very dense graphs (note that even K_3 does not exist in $K_{n,n}$ which has $2n$ vertices and n^2 edges). See the recent paper [52] or the survey [53] of Yuster. The latter survey also mentions approximation algorithms, including APX-hardness and the general approximation algorithm for k-Set Packing which now achieves $\frac{k+1+\epsilon}{3}$ for the vertex version and $\frac{\binom{k}{2}+1+\epsilon}{3}$ for the edge version [19]. Feder and Subi [27] considered H-Edge Packing and showed APX-hardness when H is k-cycle or k-clique. Chataigner et al. [13] considered an interesting variant where we want to pack vertex-disjoint cliques of any size to maximize the total number of edges of the packed cliques, and proved APX-hardness and a 2-approximation algorithm. Exact algorithms for special classes of graphs have been considered in [9, 33, 35, 40].

1.3 Open Problems

For H-Transversal, 1-connectivity is not sufficient for strong hardness, because k-Star Transversal admits an $O(\log k)$-approximation algorithm by Theorem 2. It is open whether 1-connectivity is sufficient or not for such strong hardness for H-Packing. k-Star Packing is at least as hard as MIS-k by a trivial reduction, but the approximability of k-Path Packing appears to be still unknown. Whether k-Path Transversal admits a factor $o(k)$ approximation algorithm is also an intriguing question. For directed acyclic graphs, Svensson [50] proved that it is Unique Games-hard to approximate k-Path Transversal within a factor better than k.

The approximability of H-Edge Transversal and H-Edge Packing is less understood than the vertex versions. Proving tight characterizations for the edge versions similar to Theorem 1 is an interesting open problem.

1.4 Organization

The rest of the main body is devoted to proving Theorem 1 for H-Transversal / Packing and Theorem 3 for MIS-k-g. Section 2 recalls and extends previous hardness results for the problems we reduce from; Sections 3 and 4 prove hardness of H-Transversal and H-Packing respectively. Appendix A gives an $O(\log k)$-approximation algorithm for k-Star Transversal, proving Theorem 2. Theorem 4 is proved in the full version of this work [31].

2 Preliminary

2.1 Notation

A k-uniform hypergraph is denoted by $P = (V_P, E_P)$ such that each $e \in E_P$ is a k-subset of V_P. We denote e as an *ordered* k-tuple $e = (v^1, \dots, v^k)$. The ordering can be chosen arbitrarily given P, but should be fixed throughout. If v indicates a vertex of some graph, we use a superscript v^i to denote another vertex of the same graph, and e^i to denote the ith (hyper)edge. For an integer m, let $[m] = \{1, 2, \dots, m\}$. Unless otherwise stated, the *measure* of $F \subseteq V$ is obtained under the uniform measure on V, which is simply $\frac{|F|}{|V|}$.

2.2 k-HVC

An instance of k-HVC consists of a k-uniform hypergraph P, where the goal is to find a set $C \subseteq V_P$ with the minimum cardinality such that it intersects every hyperedge. The result of Dinur, Guruswami, Khot and Regev [21] states that

▶ **Theorem 5** ([21]). *Given a k-uniform hypergraph ($k \geq 3$) and $\epsilon > 0$, it is NP-hard to tell apart the following cases:*

- *Completeness: There exists a vertex cover of measure $\frac{1+\epsilon}{k-1}$.*
- *Soundness: Every vertex cover has measure at least $1 - \epsilon$.*

Therefore, it is NP-hard to approximate k-HVC within a factor $k - 1 + 2\epsilon$.

Moreover, the above result holds even when the degree of a hypergraph is bounded by d depending on k and ϵ.

2.3 MIS-k

Given a graph $G = (V_G, E_G)$, a subset $S \subseteq V_G$ is *independent* if the subgraph induced by S does not contain any edge. The Maximum Independent Set (MIS) problem asks to find the largest independent set, and MIS-k indicates the same problem where G is promised to have maximum degree at most k. The recent result of Chan [11] implies

▶ **Theorem 6** ([11]). *Given a graph G with maximum degree at most k, it is NP-hard to tell apart the following cases:*

- *Completeness: There exists an independent set of measure $\Omega(1/(\log k))$.*
- *Soundness: Every subset of vertices of measure $O(\frac{\log^3 k}{k})$ contains an edge.*

Therefore, it is NP-hard to approximate MIS-k within a factor $\Omega(\frac{k}{\log^4 k})$.

3 H-Transversal

In this section, given a 2-connected graph $H = (V_H, E_H)$ with k vertices, we give a reduction from k-HVC to H-Transversal. The simplest try will be, given a hypergraph $P = (V_P, E_P)$ (let $n = |V_P|, m = |E_P|$), to produce a graph $G = (V_G, E_G)$ where $V_G = V_P$, and for each hyperedge $e = (v^1, \dots, v^k)$ add $|E_H|$ edges that form a *canonical copy* of H to E_G. While the soundness follows directly (if $F \subseteq V_P$ contains a hyperedge, the subgraph induced by F contains H), the completeness property does not hold since edges that belong to different canonical copies may form an unintended non-canonical copy. To prevent this, a natural strategy is to replace each vertex by a set of many vertices (call it a *cloud*), and for each hyperedge $(v^1, \dots, v^k)$, add many canonical copies on the k clouds (each copy consists of one vertex from each cloud). If we have too many canonical copies, soundness works easily but completeness is hard to show due to the risk posed by non-canonical copies, and in the other extreme, having too few canonical copies could result in the violation of the soundness property. Therefore, it is important to control the structure (number) of canonical copies that ensure both completeness and soundness at the same time.

Our technique, which we call *random matching*, proceeds by creating a carefully chosen number of *random* copies of H for each hyperedge to ensure both completeness and soundness. We remark that properties of random matchings are also used to bound the number of short non-canonical paths in inapproximability results for edge-disjoint paths on undirected graphs [2, 1]. The details in our case are different as we create *many* copies of H based on a *hypergraph*.

Fix $\epsilon > 0$, apply Theorem 5, let $c := \frac{1+\epsilon}{k-1}, s := 1 - \epsilon$ be the measure of the minimum vertex cover in the completeness and soundness case respectively, and $d := d(k, \epsilon)$ be the maximum degree of hard instances. Let a and B be integer constants greater than 1, which will be determined later. Lemma 7 and 9 with these parameters imply the first half of Theorem 1.

3.1 Reduction

Without loss of generality, assume that $V_H = [k]$. Given a hypergraph $P = (V_P, E_P)$, construct an undirected graph $G = (V_G, E_G)$ such that

- $V_G = V_P \times [B]$. Let $n = |V_P|$ and $N = |V_G| = nB$. For $v \in V_P$, let $\mathsf{cloud}(v) := \{v\} \times [B]$ be the copy of $[B]$ associated with v.
- For each hyperedge $e = (v^1, \dots, v^k)$, for aB times, take $l^1, \dots, l^k$ independently and uniformly from $[B]$. For each edge $(i, j) \in H$ $(1 \le i < j \le k)$, add $((v^i, l^i), (v^j, l^j))$ to E_G. Each time we add $|E_H|$ edges isomorphic to H, and we have aB of such copies of H per each hyperedge. Call such copies *canonical*.

3.2 Completeness

The next lemma shows that if P has a small vertex cover, G also has a small H-Transversal.

▶ **Lemma 7.** *Suppose P has a vertex cover C of measure c. For any $\epsilon > 0$, with probability at least $3/4$, there exists a subset $F \subseteq V_G$ of measure at most $c + \epsilon$ such that the subgraph induced by $V_G \setminus F$ has no copy of H.*

Proof. Let $F = C \times [B]$. We consider the expected number of copies of H that avoid F and argue that a small fraction of additional vertices intersect all of these copies. Choose k vertices $(v^1, l^1), \dots, (v^k, l^k)$ which satisfy

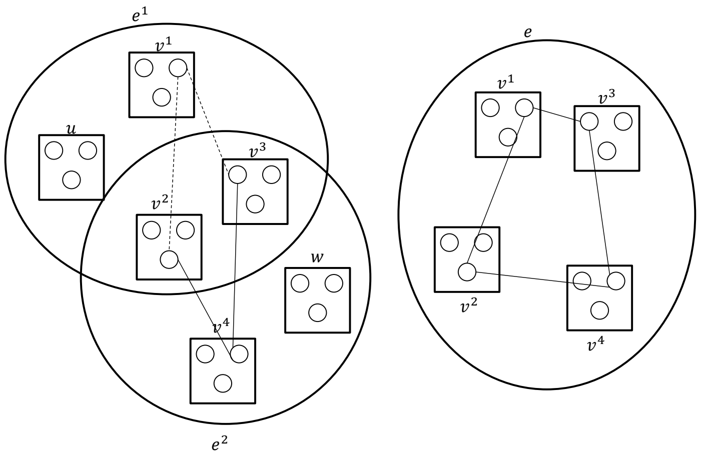


Figure 1 Two examples where $k = 4$ and H is a 4-cycle. On the left, purported edges are divided into two groups (dashed and solid edges). Each copy of canonical cycle should match the labels of three vertices to ensure it covers 2 designated edges (6 labels total). On the right, one canonical copy can cover all the edges, and it only needs to match the labels of four vertices (4 labels total).

- $v^1 \in V_P$ can be any vertex.
- $l^1, \dots, l^k \in B$ can be arbitrary labels.
- For each $(i, j) \in E_H$, there must be a hyperedge of P containing both i and j.

There are n possible choices for v^1, B choices for each l^i, and at most kd choices for each v^i $(i > 1)$. The number of possibilities to choose such $(v^1, l^1), \dots, (v^k, l^k)$ is bounded by $n(dk)^k B^k$. Note that no other k-tuple of vertices induce a connected graph and contain a copy of H. Further discard the tuple when two vertices are the same.

We calculate the probability that the subgraph induced by $((v^1, l^1), \dots, (v^k, l^k))$ contains a copy *in this order* – formally, for all $(i, j) \in E_H$, $((v^i, l^i), (v^j, l^j)) \in E_G$. For each $(i, j) \in E_H$, we call a pair $((v^i, l^i), (v^j, l^j)) \in \binom{V_G}{2}$ a *purported edge*. For a set of purported edges, we say that this set can be *covered by a single canonical copy* if one copy of canonical copy of H can contain all purported edges with nonzero probability. Suppose that all $|E_H|$ purported edges can be covered by a single canonical copy of H. It is only possible when there is a hyperedge whose k vertices are exactly $\{v^1, \dots, v^k\}$. In this case, $((v^1, l^1), \dots, (v^k, l^k))$ intersects F. (right case of Figure 1). When $|E_H|$ purported edges have to be covered by more than one canonical copy, some vertices must be covered by more than one canonical copy, and each canonical copy covering the same vertex should give the same label to that vertex. This redundancy makes it unlikely to have all k edges exist at the same time. (left case of Figure 1). The below claim formalizes this intuition.

▶ **Claim 1.** *Suppose that $((v^1, l^1), \dots, (v^k, l^k))$ cannot be covered by a single canonical copy. Then the probability that it forms a copy of H is at most $\frac{(adk)^{k^2}}{B^k}$.*

Proof. Fix $2 \le p \le |E_H|$. Partition $|E_H|$ purported edges into p nonempty *groups* $I_1, \dots, I_p$ such that each group can be covered by a single canonical copy of H. There are at most $p^{|E_H|}$ possibilities to partition. For each $v \in V_P$, there are at most d hyperedges containing v and at most aBd canonical copies intersecting $\mathsf{cloud}(v)$. Therefore, all edges in one group can be covered simultaneously by at most aBd copies of canonical copies. There are at most $(aBd)^p$ possibilities to assign a canonical copy to each group. Assume that one canonical copy is responsible for exactly one group. This is without loss of generality since if one canonical copy is responsible for many groups, we can merge them and this case can be dealt with smaller p.

Focus on one group I of purported edges, and one canonical copy $L = (V_L, E_L)$ which is supposed to cover them. Let $I' \subseteq V_G$ be the set of vertices which are incident on the edges in I. Suppose $V_L = \{(u^1, l'^1), \dots, (u^k, l'^k)\}$, which is created by a hyperedge $f = (u^1, \dots, u^k) \in E_P$. We calculate the probability that L contains all edges in I over the choice of labels $l'^1, \dots, l'^k$ for L. One necessary condition is that $\{v | (v, l) \in I' \text{ for some } l \in [B]\}$ (i.e. the set I' projected to V_P) is contained in f. Otherwise, some vertices of I' cannot be covered by L. Another necessary condition is $v^i \neq v^j$ for any $(v^i, l^i) \neq (v^j, l^j) \in I'$. Otherwise (i.e. $(v, l^i), (v, l^j) \in I'$ for $l^i \neq l^j$), since L gives only one label to each vertex in $f \subseteq V_P$, (v, l^i) and (v, l^j) cannot be contained in L simultaneously. Therefore, we have a nice characterization of I': It consists of at most one vertex from the cloud of each vertex in f.

The probability that L contains I is at most the probability that for each $(v^i, l^i) \in I'$, l^i is equal to the label L assigns to v^i, which is $B^{-|I'|}$. Now we need the following lemma saying that the sum of $|I'|$ is large, which relies on 2-connectivity of H.

▶ **Lemma 8.** *Fix $p \geq 2$. For any partition $I_1, ..., I_p$ of purported edges into p non-empty groups, $\sum_{i=1}^{p} |I'_i| \geq k + p$.*

Proof. Let t be the number of vertices contained in at least two I'_is. Call them *boundary vertices*. Note that exactly $k - t$ vertices belongs to exactly one I'_i. For $i = 1, ..., p$, let b_i be the number of boundary vertices in $|I'_i|$. Since (I'_i, I_i) is a proper subgraph of H and H is 2-vertex connected, $b_i \geq 2$ for each i. Therefore,

$$\sum_{i=1}^{p} |I'_i| = (k - t) + \max(2p, 2t) \geq k + p. \qquad ◀$$

We conclude that for each partition, the probability of having all the edges is at most

$$(aBd)^p \prod_{q=1}^{p} B^{-|I'_q|} = \frac{(aBd)^p}{B^{k+p}} = \frac{(ad)^p}{B^k} .$$

The probability that $((v^1, l^1), \dots, (v^k, l^k))$ forms a copy is therefore bounded by

$$\sum_{p=2}^{|E_H|} p^{|E_H|} \frac{(ad)^p}{B^k} \leq \frac{(adk)^{k^2}}{B^k} . \qquad ◀$$

Therefore, the expected number of copies that avoid F is bounded by $n(kd)^k B^k \cdot \frac{(adk)^{k^2}}{B^k}$. With probability at least $3/4$, the number of such copies is at most $4n(adk)^{2k^2}$. Let $B \geq \frac{4(adk)^{2k^2}}{\epsilon}$. Then these copies of H can be covered by at most $\epsilon n B = \epsilon N$ vertices. ◀

3.3 Soundness

The soundness claim above is easier to establish. By an averaging argument, a subset I of V_G of measure 2ϵ must contain ϵB vertices from the clouds corresponding to a subset S of measure ϵ in V_P. There must be a hyperedge e contained within S, and the chosen parameters ensure that one of the canonical copies corresponding to e is likely to lie within I.

▶ **Lemma 9.** *For $a = a(k, \epsilon)$ and $B = \Omega(\log |E_P|)$, if every subset of V_P of measure at least ϵ contains a hyperedge in the induced subgraph, with probability at least $3/4$, every subset of V_G with measure 2ϵ contains a canonical copy of H.*

Proof. We want to show that the following property holds for every hyperedge $e = (v^1, \dots, v^k)$: if a subset of vertices $I \subseteq V_G$ has at least ϵ fraction of vertices from each $\mathsf{cloud}(v^i)$, then I will contain a canonical copy. Fix $A^1 \subseteq \mathsf{cloud}(v^1), \dots, A^k \subseteq \mathsf{cloud}(v^k)$ be such that for each i, $|A^i| \geq \epsilon B$. There are at most 2^{kB} ways to choose such A's. The probability that one canonical copy associated with e is not contained in $(v^1, A^1) \times \cdots \times (v^k, A^k)$ is at most $1 - \epsilon^k$. The probability that none of canonical copy associated with e is contained in $(v^1, A^1) \times \cdots \times (v^k, A^k)$ is $(1 - \epsilon^k)^{aB} \leq \exp(-aB\epsilon^k)$.

By union bound over all $A^1, \dots, A^k$, the probability that there exists $A^1, \dots, A^k$ containing no canonical copy is at most $\exp(kB - aB\epsilon^k) = \exp(-B) \leq \frac{1}{4|E_P|}$ by taking a a large enough constant depending on k and ϵ, and $B = \Omega(\log |E_P|)$. Therefore, with probability at least $3/4$, the desired property holds for all hyperedges.

Let I be a subset of V_G of measure at least 2ϵ. By an averaging argument, at least ϵ fraction of *good* vertices $v \in V_P$ satisfy that $|\mathsf{cloud}(v^i) \cap I| \geq \epsilon B$. By the soundness property of P, there is a hyperedge e contained in the subgraph induced by the good vertices, and the above property for e ensures that I contains a canonical copy. ◀

4 H-Packing and MIS-k-g

Given a 2-connected graph H, the reduction from MIS-k-k to H-Packing is relatively straightforward. Here we assume that hard instances of MIS-k-k are indeed k-regular for simplicity. Given an instance $M = (V_M, E_M)$ of MIS-k-k, we take $G = (V_G, E_G)$ to be its *line graph* – $V_G = E_M$, and $e, f \in V_G$ are adjacent if and only if they share an endpoint as edges of M.

For each vertex $v \in V_M$, let $\mathsf{star}(v) := \{e \in V_G : v \in e\}$. $\mathsf{star}(v)$ induces a k-clique, and for $v, u \in V_M$, $\mathsf{star}(v)$ and $\mathsf{star}(u)$ share one vertex if u and v are adjacent, and share no vertex otherwise. Given an independent set S of M, we can find $|S|$ pairwise disjoint stars in G, which gives $|S|$ vertex-disjoint copies of H. On the other hand, 2-connectivity of H and large girth of M implies that any copy of H must be entirely contained in one star, proving that many disjoint copies of H in G also give a large independent set of M with the same cardinality, completing the reduction from MIS-k-k to H-Packing. The following theorem formalizes the above intuition.

▶ **Lemma 10.** *For a 2-connected graph H with k vertices, there is an approximation-preserving reduction from MIS-k-k to H-Packing.*

Proof. Let $M = (V_M, E_M)$ be an instance of MIS-k-k M with maximum degree k and girth greater than k. First, let $G = (V_G = E_M, E_G)$ be the line graph of M. For each vertex $v \in V_M$ with degree strictly less than k, we add $k - \mathsf{deg}(v)$ new vertices to V_G. Let $\mathsf{star}(v) \subseteq V_G$ be the union of the edges of M incident on v and the newly added vertices for v. Note that $|\mathsf{star}(v)| = k$ for all $v \in V_M$. Add edges to G to ensure that every $\mathsf{star}(v)$ induces a k-clique. For two vertices u and v of M, $\mathsf{star}(u)$ and $\mathsf{star}(v)$ share exactly one vertex if u and v are adjacent in M, and share no vertex otherwise.

Let S be an independent set of M. The $|S|$ stars $\{\mathsf{star}(v)\}_{v \in S}$ are pairwise disjoint and each induces a k-clique, so G contains at least $|S|$ disjoint copies of H.

We claim that any k-subset of V_G that induces a 2-connected subgraph must be $\mathsf{star}(v)$ for some v. Assume towards contradiction, let T be a k-subset inducing a 2-connected subgraph of G that cannot be contained in a single star. We first show T must contain two disjoint edges of M. Take any $(u,v) \in T$. Since $T \notin \mathsf{star}(u)$, T contains an edge of M not incident on u. If it is not incident on v either, we are done. Otherwise, let (w,v) be this edge. The same argument from $T \notin \mathsf{star}(v)$ gives another edge (w',u) in T. If $w \neq w'$, (w,v) and (w',u) are disjoint. Otherwise, w,u,v form a triangle in M, contradicting a large girth. Let (u,v), (w,x) be two disjoint edges of M in contained in T.

Since the subgraph of G vertex-induced by T is 2-connected, there are two internally vertex-disjoint paths P_1, P_2 in G from (u,v) to (w,x). The sum of the two lengths is at most k, where the length of a path is defined to be the number of edges. By considering the internal vertices of P_i (edges of M) and deleting unnecessary portions, we have two edge-disjoint paths P_1', P_2' in M where each P_i' connects $\{u,v\}$ and $\{w,x\}$, with length at most the length of P_i minus one. There is a cycle in M consists only of the edges of P_1', P_2' together with $(u,v),(w,x)$. Since $|P_1'|+|P_2'|+2 \leq k$, it contradicts that M has girth strictly greater than k. ◀

We prove that MIS-k-g is also hard to approximate by a reduction from MIS-d ($d = \widetilde{\Omega}(k)$), using a slightly different *random matching* idea. Given a degree-d graph with possibly small girth, we replace each vertex by a cloud of B vertices, and replace each edge by a copies of random matching between the two clouds. While maintaining the soundness guarantee, we show that there are only a few small cycles, and by deleting a vertex from each of them and *sparsifying* the graph we obtain a hard instance for MIS-k-g. Note that g does not affect the inapproximability factor but only the runtime of the reduction.

▶ **Theorem 11** (Restatement of Theorem 3). *For any constants k and g, unless* NP $\subseteq$ BPP*, no polynomial time algorithm approximates MIS-k-g within a factor of $\Omega(\frac{k}{\log^7 k})$.*

Proof. We reduce from MIS-d to MIS-k-g where $k = O(d \log^2 d)$. Given an instance $G_0 = (V_{G_0}, E_{G_0})$ of MIS-d, we construct $G = (V_G, E_G)$ and $G' = (V_{G'}, E_{G'})$ by the following procedure:

- $V_G = V_{G_0} \times [B]$. As usual, let $\mathsf{cloud}(v) = \{v\} \times [B]$.
- For each edge $(u,v) \in E_{G_0}$, for a times, add a *random matching* as follows.
  - Take a random permutation $\pi : [B] \to [B]$.
  - Add an edge $((u,i),(v,\pi(i))$ for all $i \in [B]$.
- Call the resulting graph G. To get the final graph G',
  - For any cycle of length at most g, delete an arbitrary vertex from the cycle. Repeat until there is no cycle of length at most g.

Note that the step of eliminating the small cycles can be implemented trivially in time $O(n^g)$. Let $n = |V_{G_0}|, m = |E_{G_0}|, N = nB = |V_G| \geq |V_{G'}|, M = m \cdot aB = |E_G| \geq |E_{G'}|$. The maximum degree of G and G' is at most ad. By construction, girth of G' is at least $g+1$.

Girth Control. We calculate the expected number of small cycles in G, and argue that the number of these cycles is much smaller than the total number of vertices, so that $|V_G|$ and $|V_{G'}|$ are almost the same. Let k' be the length of a purported cycle. Choose k' vertices $(v^1,l^1),\ldots,(v^{k'},l^{k'})$ which satisfy

- $v^1 \in V_{G_0}$ can be any vertex.
- For each $1 \leq i < k'$, $(v^i, v^{i+1}) \in E_{G_0}$.
- $l^1,\ldots,l^{k'} \in B$ can be arbitrary labels.

There are n possible choices for v^1, B choices for each l^i, and d choices for each v^i $(i > 1)$. The number of possibilities to choose such $(v^1, l^1), \ldots, (v^{k'}, l^{k'})$ is bounded by $nd^{k'-1}B^{k'}$. Without loss of generality, assume that no vertices appear more than once.

For each edge $e = (u, w) \in G_0$, consider the intersection of the purported cycle $((v^1, l^1), ..., (v^{k'}, l^{k'}))$ and the subgraph induced by $\mathsf{cloud}(u) \cup \mathsf{cloud}(w)$. It is a bipartite graph with the maximum degree 2. Suppose there are q purported edges $e^1, \ldots, e^q$ (ordered arbitrarily) in this bipartite graph. By slightly abusing notation, let e^i also denote the event that e^i exists in G. The following claim upper bounds $\Pr[e^i | e^1, \ldots, e^{i-1}]$ for each e^i.

▶ **Claim 2.** $\Pr[e^i | e^1, \ldots, e^{i-1}] \le \frac{a}{B-i}$.

Proof. There are a random matchings between $\mathsf{cloud}(u)$ and $\mathsf{cloud}(w)$, and for each $j < i$, there is at least one random matching including e^j. We fix one random matching and calculate the probability that the random matching contains e^i, conditioned on the fact that it already contains some of $e^1, \ldots, e^{i-1}$.

If there is e^j $(j < i)$ that shares a vertex with e^i, e^i cannot be covered by the same random matching with e^j. If a random matching covers p of $e^1, \ldots, e^{i-1}$ which are disjoint from e^i, the probability that e^i is covered by that random matching is $\frac{1}{B-p}$, and this is maximized when $p = i - 1$.

By a union bound over the a random matchings, $\Pr[e^i | e^1, \ldots, e^{i-1}] \le \frac{a}{B-i}$. ◀

The probability that all of $e^1, \ldots, e^q$ exist is at most

$$\prod_{i=1}^{q} \frac{a}{B-i} \le \left(\frac{a}{B-q}\right)^q \le \left(\frac{a}{B-k'}\right)^q .$$

Since edges of G_0 are processed independently, the probability of success for one fixed purported cycle is $(\frac{a}{B-k'})^{k'}$. The expected number of cycles of length k' is

$$nd^{k'-1}B^{k'} \cdot \left(\frac{a}{B-k'}\right)^{k'} = nd^{k'-1}a^{k'}\left(1 + \frac{k'}{B-k'}\right)^{k'}$$
$$\le nd^{k'-1}a^{k'} \exp\left(\frac{k'^2}{B-k'}\right) \le en(ad)^{k'}$$

by taking $B - k' \ge k'^2$. Summing over $k' = 1, \ldots, g$, the expected number of cycles of length up to g, is bounded by $eg(ad)^g n$. Take $B \ge 4d^2 \cdot eg(ad)^g$. Then with probability at least $3/4$, the number of cycles of length at most g is at most $\frac{Bn}{d^2}$. By taking $1/d^2$ fraction of vertices away (one for each short cycle), we have a girth at least $g + 1$, which implies $\left(1 - \frac{1}{d^2}\right)|V_G| \le |V_{G'}| \le |V_G|$.

Hardness of MIS-d states that it is NP-hard to distinguish the case G_0 has an independent set of measure $c := \Omega(\frac{1}{\log d})$ and the case where the maximum independent set has measure at most $s := O(\frac{\log^3 d}{d})$.

Completeness. Let I_0 be an independent set of G_0 of measure c. Then $I = I_0 \times [B]$ is also an independent set of G of measure c. Let $I' = I \cap V_{G'}$. I' is independent in both G and G', and the measure of I' in G' is at least the measure of I' in G, which is at least $c - 1/d^2 = \Omega(\frac{1}{\log d})$.

Soundness. Suppose that every subset of V_{G_0} of measure at least s contains an edge. Say a graph is (β, α)-*dense* if we take β fraction of vertices, at least α fraction of edges lie within the induced subgraph. We also say a bipartite graph is (β, α)-*bipartite dense* if we take β fraction of vertices from each side, at least α fraction of edges lie within the induced subgraph.

▶ **Claim 3.** *For $a = O(\frac{\log(1/s)}{s})$ and $B = O(\frac{\log m}{s})$ the following holds with probability at least $3/4$: For every $(u, w) \in E_{G_0}$, the bipartite graph between* $\mathsf{cloud}(u)$ *and* $\mathsf{cloud}(w)$ *is $(\epsilon, \epsilon^2/8)$-bipartite dense for all $\epsilon \geq s$.*

Proof. Fix (u, w), and $\epsilon \in [s, 1]$, and $X \subseteq \mathsf{cloud}(u)$ and $Y \subseteq \mathsf{cloud}(w)$ be such that $|X| = |Y| = \epsilon B$. The possibilities of choosing X and Y is

$$\binom{B}{\epsilon B}^2 \leq \exp(O(\epsilon \log(1/\epsilon) B))$$

Without loss of generality, let $X = Y = [\epsilon B]$. In one random matching, let X_i ($i \in [\epsilon B]$) be the random variable indicating whether vertex $(u, i) \in X$ is matched with a vertex in Y or not. $\Pr[X_1 = 1] = \epsilon$, and $\Pr[X_i = 1 | X_1, \ldots, X_{i-1}] \geq \epsilon/2$ for $i \in [\epsilon B/2]$ and any $X_1, \ldots, X_{i-1}$. Therefore, the expected number of edges between X and Y is at least $\epsilon^2 B/4$. With a random matchings, the expected number is at least $a\epsilon^2 B/4$. By Chernoff bound, the probability that it is less than $a\epsilon^2 B/8$ is at most $\exp(\frac{a\epsilon^2 B}{32})$. By union bound over all possibilities of choosing X and Y, the probability that the bipartite graph is not $(\epsilon, \epsilon^2/8)$-bipartite dense is

$$\exp(\epsilon \log(1/\epsilon) B) \cdot \exp\Big(-\frac{a\epsilon^2 B}{32}\Big) \leq \frac{1}{4mB}$$

by taking $a = O(\frac{\log(1/s)}{s})$ and $B = O\big(\frac{\log m}{s}\big)$. A union bound over all possible choices of ϵ (B possibilities) and m edges of E_0 implies the claim. ◀

▶ **Claim 4.** *With the parameters a and B above, G is $(4s \log(1/s), \Omega(\frac{s}{d}))$-dense.*

Proof. Fix a subset S of measure $4s \log(1/s)$. For a vertex v of G_0, let $\mu(v) := \frac{|\mathsf{cloud}(v) \cap S|}{B}$. Note that $\mathbb{E}_v[\mu(v)] = 4s \log(1/s)$. Partition V_{G_0} into $t + 1$ buckets $B_0, \ldots, B_t$ ($t := \lceil \log_2(1/s) \rceil$), such that B_0 contains v such that $\mu(v) \leq s$, and for $i \geq 1$, B_i contains v such that $\mu(v) \in (2^{i-1}s, 2^i s]$. Denote

$$\mu(B_i) := \frac{\sum_{v \in B_i} \mu(v)}{|V_{G_0}|} .$$

Clearly $\mu(B_0) \leq s$. Pick $i \in \{1, \ldots, t\}$ with the largest $\mu(B_i)$. We have $\mu(B_i) \geq 2s$ since $\mathbb{E}_v[\mu(v)] \geq 4s \log(1/s)$. Let $\gamma = 2^{i-1}s$. All vertices of B_i has $\mu(v) \in [\gamma, 2\gamma]$, so $|B_i| \geq (s/\gamma)n$.

Since G_0 has no independent set with more than ns vertices, Turán's Theorem says that the subgraph of G_0 induced by B_i has at least $\frac{|B_i|}{2}(\frac{|B_i|}{ns} - 1) = \Omega(\frac{s}{\gamma^2}n)$ edges. This is at least $\Omega(\frac{s}{d\gamma^2})$ fraction of the total number of edges.

For each of these edges, by Claim 3, at least $\gamma^2/8$ fraction of the edges from the bipartite graph connecting the clouds of its two endpoints, lie in the subgraph induced by S (since $\gamma \geq s$). Overall, we conclude that there are at least $\Omega(\frac{s}{d\gamma^2}) \cdot \frac{\gamma^2}{8} = \Omega(\frac{s}{d})$ fraction of edges inside the subgraph induced by S. ◀

Sparsification. Recall that G' is obtained from G by deleting at most $\frac{1}{d^2}$ fraction of vertices to have girth greater than g. In the completeness case, G' has an independent set of measure at least $c - 1/d^2 = \Omega(\frac{1}{\log d})$. In the soundness case, G is $(4s\log(1/s), \Omega(\frac{s}{d}))$-dense, so G' is (β, α)-dense where $\beta := \Omega(\frac{\log^4 d}{d}), \alpha := \Omega(\frac{\log^3 d}{d^2})$. Using density of G', we sparsify G' again – keep each edge of G' by probability $\frac{kn}{|E_{G'}|}$ so that the expected total number of edges is kn.

Fix a subset $S \subseteq V_{G'}$ of measure β. Since there are at least α fraction of edges in the subgraph induced by S, the expected number of picked edges in this subgraph is at least αkn. By Chernoff bound, the probability that it is less than $\frac{\alpha kn}{8}$ is at most $\exp(-\frac{\alpha kn}{32})$. By union bound over all sets of measure exactly β (there are at most $\binom{n}{n\beta} \leq \exp(2\beta\log(1/\beta)n)$ of them), and over all possible values of β (there are at most n possible sizes), the desired property fails with probability at most

$$n \cdot \max_{\beta\in[\beta_0,1]} \left\{\exp(-\alpha kn/32) \cdot \exp(2\beta\log(1/\beta)n)\right\} \leq n \cdot e^{-n}$$

when $k = O(\frac{\beta\log(1/\beta)}{\alpha}) = O(d\log^2 d)$. In the last step we remove all the vertices of degree more than $10k$. Since the expected degree of each vertex is at most $2k$, the expected fraction of deleted vertices is $\exp(-\Omega(k)) \ll \beta$.

Combining all these results, we have a graph with small degree $10k = O(d\log^2 d)$ and girth strictly greater than g, where it is NP-hard to approximate MIS within a factor of $\frac{c-\frac{1}{d^2}}{\beta} = \Omega(\frac{d}{\log^5 d}) = \Omega(\frac{k}{\log^7 k})$. Therefore, it is NP-hard to approximate MIS-k-g within a factor of $\Omega(\frac{k}{\log^7 k})$. ◀

References

1 Matthew Andrews, Julia Chuzhoy, Venkatesan Guruswami, Sanjeev Khanna, Kunal Talwar, and Lisa Zhang. Inapproximability of edge-disjoint paths and low congestion routing on undirected graphs. *Combinatorica*, 30(5):485–520, 2010.

2 Matthew Andrews and Lisa Zhang. Logarithmic hardness of the undirected edge-disjoint paths problem. *Journal of the ACM*, 53(5):745–761, 2006.

3 P. Austrin, S. Khot, and M. Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. In *Proceedings of the 24th annual IEEE Conference on Computational Complexity*, CCC'09, pages 74–80, 2009.

4 V. Bafna, P. Berman, and T. Fujito. Constant ratio approximations of the weighted feedback vertex set problem for undirected graphs. In *Proceedings of the 6th International Symposium on Algorithms and Computation*, ISAAC'95, pages 142–151, 1995.

5 N. Bansal and S. Khot. Inapproximability of hypergraph vertex cover and applications to scheduling problems. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming*, ICALP'10, pages 250–261, 2010.

6 R. Bar-Yehuda, D. Geiger, J. Naor, and R. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and bayesian inference. *SIAM Journal on Computing*, 27(4):942–959, 1998.

7 A. Becker and D. Geiger. Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83(1):167–188, 1996.

8 H. L. Bodlaender. On disjoint cycles. *International Journal of Foundations of Computer Science*, 5(01):59–68, 1994.

9 A. Brandstädt, V. Chepoi, and F. Dragan. Clique r-domination and clique r-packing problems on dually chordal graphs. *SIAM J. on Discrete Mathematics*, 10(1):109–127, 1997.

10 Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. Pre-reduction graph products: Hardnesses of properly learning DFAs and approximating EDP on DAGs. In *Proc. of the 55rd Annual Symp. on Foundations of Computer Science (FOCS'14)*. IEEE, 2014.

11 S. Chan. Approximation resistance from pairwise independent subgroups. In *Proc. of the 45th Annual ACM Symp. on Symposium on Theory of Computing*, STOC'13, pages 447–456, 2013.

12 Maw-Shang Chang, Ton Kloks, and Chuan-Min Lee, editors. *Graph-Theoretic Concepts in Computer Science*, volume 2204 of *Lecture Notes in Computer Science*. Springer, 2001.

13 F. Chataigner, G. Manić, Y. Wakabayashi, and R. Yuster. Approximation algorithms and hardness results for the clique packing problem. *Discrete Applied Mathematics*, 157(7):1396–1406, 2009.

14 J. Chen, Y. Liu, S. Lu, B. O'sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM*, 55(5):21:1–21:19, 2008.

15 R. Chitnis, M. Cygan, M. Hajiaghayi, and D. Marx. Directed subset feedback vertex set is fixed-parameter tractable. In *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming*, ICALP'12, pages 230–241, 2012.

16 Miroslav Chlebík and Janka Chlebíková. Approximation hardness of dominating set problems in bounded degree graphs. *Information and Computation*, 206(11):1264–1275, 2008.

17 F. A. Chudak, M. X. Goemans, D. S. Hochbaum, and D. P. Williamson. A primal-dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs. *Operations Research Letters*, 22(4–5):111–118, 1998.

18 M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. Subset feedback vertex set is fixed-parameter tractable. In *Proceedings of the 38th International Colloquim on Automata, Languages and Programming*, ICALP'11, pages 449–461, 2011.

19 Marek Cygan. Improved approximation for 3-dimensional matching via bounded pathwidth local search. In *Proceedings of the 54th annual IEEE symposium on Foundations of Computer Science*, FOCS'13, pages 509–518, 2013.

20 I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, pages 439–485, 2005.

21 Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered PCP and the hardness of hypergraph vertex cover. *SIAM Journal on Computing*, 34(5):1129–1146, 2005.

22 D. Dor and M. Tarsi. Graph decomposition is NP-complete: A complete proof of Holyer's conjecture. *SIAM Journal on Computing*, 26(4):1166–1187, 1997.

23 Guillermo Durán, Min Chih Lin, Sergio Mera, and Jayme L Szwarcfiter. Algorithms for finding clique-transversals of graphs. *Annals of Operations Research*, 157(1):37–45, 2008.

24 Paul Erdős, Tibor Gallai, and Zsolt Tuza. Covering the cliques of a graph with vertices. *Discrete Mathematics*, 108(1):279–289, 1992.

25 G. Even, J. Naor, S. Rao, and B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *Journal of the ACM*, 47(4):585–616, 2000.

26 G. Even, J. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20:151–174, 1998.

27 Tomas Feder and Carlos Subi. Packing edge-disjoint triangles in given graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 2012. TR12-013.

28 Z. Friggstad and M. R. Salavatipour. Approximability of packing disjoint cycles. In *Proc. of the 18th Int'l Conf. on Algorithms and Computation*, ISAAC'07, pages 304–315, 2007.

29 V. Guruswami, R. Manokaran, and P. Raghavendra. Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. In *Proceedings of the 49th annual IEEE symposium on Foundations of Computer Science*, FOCS'08, pages 573–582, 2008.

30 Venkatesan Guruswami and Euiwoong Lee. Inapproximability of feedback vertex set for bounded length cycles. *Electronic Colloquium on Computational Complexity (ECCC)*, TR 14-006, 2014.

31 Venkatesan Guruswami and Euiwoong Lee. Inapproximability of H-transversal/matching. *arXiv preprint arXiv:1506.06302*, 2015.

32 Venkatesan Guruswami and C. Pandu Rangan. Algorithmic aspects of clique-transversal and clique-independent sets. *Discrete Applied Mathematics*, 100(3):183–202, 2000.

33 Venkatesan Guruswami, C. Pandu Rangan, M. S. Chang, G. J. Chang, and C. K. Wong. The K_r-packing problem. *Computing*, 66(1):79–89, 2001.

34 Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k-set packing. *Computational Complexity*, 15(1):20–39, 2006.

35 P. Hell, S. Klein, L.T. Nogueira, and F. Protti. Packing r-cliques in weighted chordal graphs. *Annals of Operations Research*, 138(1):179–187, 2005.

36 Bart M. P. Jansen and Dániel Marx. Characterizing the easy-to-find subgraphs from the viewpoint of polynomial-time algorithms, kernels, and turing kernels. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 616–629, 2015.

37 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th annual ACM Symposium on Theory of Computing*, STOC'02, pages 767–775, 2002.

38 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\epsilon$. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

39 D. Kirkpatrick and P. Hell. On the complexity of general graph factor problems. *SIAM Journal on Computing*, 12(3):601–609, 1983.

40 Ton Kloks. Packing interval graphs with vertex-disjoint triangles. *arXiv preprint arXiv:1202.1041*, 2012.

41 Guy Kortsarz, Michael Langberg, and Zeev Nutov. Approximating maximum subgraphs without short cycles. *SIAM Journal on Discrete Mathematics*, 24(1):255–269, 2010.

42 M. Krivelevich, Z. Nutov, M. R. Salavatipour, J. V. Yuster, and R. Yuster. Approximation algorithms and hardness results for cycle packing problems. *ACM Transactions on Algorithms*, 3(4), November 2007.

43 Chuan-Min Lee. Weighted maximum-clique transversal sets of graphs. *ISRN Discrete Mathematics*, 2011, 2012.

44 CM Lee, MS Chang, and SC Sheu. The clique transversal and clique independence of distance hereditary graphs. In *Proceedings of the 19th Workshop on Combinatorial Mathematics and Computation Theory, Taiwan*, pages 64–69, 2002.

45 Carsten Lund and Mihalis Yannakakis. The approximation of maximum subgraph problems. In *Automata, Languages and Programming*, volume 700 of *Lecture Notes in Computer Science*, pages 40–51. Springer, 1993.

46 Sridhar Rajagopalan and Vijay V. Vazirani. Primal-dual rnc approximation algorithms for set cover and covering integer programs. *SIAM J. on Computing*, 28(2):525–540, 1998.

47 Dieter Rautenbach and Friedrich Regen. On packing shortest cycles in graphs. *Information Processing Letters*, 109(14):816–821, 2009.

48 P. D. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15(2):281–288, 1995.

49 Erfang Shan, Zuosong Liang, and Liying Kang. Clique-transversal sets and clique-coloring in planar graphs. *European Journal of Combinatorics*, 36:367–376, 2014.

50 Ola Svensson. Hardness of vertex deletion and project scheduling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 7408 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2012.

51 Zsolt Tuza. Covering all cliques of a graph. In S.T. Hedetniemi, editor, *Topics on Domination*, volume 48 of *Annals of Discrete Mathematics*, pages 117–126. Elsevier, 1991.

52 R. Yuster. Edge-disjoint cliques in graphs with high minimum degree. *SIAM Journal on Discrete Mathematics*, 28(2):893–910, 2014.

53 Raphael Yuster. Combinatorial and computational aspects of graph packing and graph decomposition. *Computer Science Review*, 1(1):12–26, 2007.

A Approximation Algorithm for k-Star Transversal

In this section, we show that k-Star Transversal admits an $O(\log k)$-approximation algorithm, matching the $\Omega(\log k)$-hardness obtained via a simple reduction from Minimum Dominating Set on degree-$(k-1)$ graphs [16], and proving Theorem 2. Let $G = (V_G, E_G)$ be the instance of k-Star Transversal. This problem has a natural interpretation that it is equivalent to finding the smallest $F \subseteq V_G$ such that the subgraph induced by $V_G \setminus F$ has maximum degree at most $k-2$. Our algorithm consists of two phases.

1. Iteratively solve 2-rounds of Sherali-Adams linear programming (LP) hierarchy and put vertices with a large fractional value in the transversal. If this phase terminates with a partial transversal F, the remaining subgraph induced by $V_G \setminus F$ has small degree (at most $2k$) and the LP solution to the last iteration is *highly fractional.*
2. We reduce the remaining problem to *Constrained Set Multicover* and use the standard greedy algorithm. While the analysis of the greedy algorithm for Constrained Set Multicover is used as a black-box, low degree of the remaining graph and high fractionality of the LP solution imply that the analysis is almost tight for our problem as well.

A.1 Iterative Sherali-Adams

Given G, 2-rounds of Sherali-Adams hierarchy of LP relaxation has variables $\{x_v\}_{v \in V_G} \cup \{x_{u,v}\}_{u,v \in V_G}$. An integral solution $y : V_G \mapsto \{0,1\}$, where $y(v) = 1$ indicates that v is picked in the transvesal, naturally gives a feasible solution to the hierarchy by $x_v = y_v$, $x_{u,v} = y_u y_v$. Consider the following relaxation for k-Star Transversal.

$$\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{v \in V_G} x_v & \\
\text{subject to} & 0 \le x_{u,v}, x_v \le 1 & \forall u, v \in V_G \\
& x_{u,v} \le x_u & \forall u, v \in V_G \\
& x_u + x_v - x_{u,v} \le 1 & \forall u, v \in V_G \\
& \displaystyle\sum_{v:(u,v) \in E_G} (x_v - x_{u,v}) \ge (\mathsf{deg}(u) - k + 2)(1 - x_u) & \forall u \in V_G
\end{array}$$

The first three constraints are common to any 2-rounds of Sherali-Adams hierarchy, and ensure that for any $u, v \in V_G$, the *local distribution* on four assignments $\alpha : \{u,v\} \mapsto \{0,1\}$ forms a valid distribution. In other words, the following four numbers are nonnegative and sum to 1: $\Pr[\alpha(u) = \alpha(v) = 1] := x_{u,v}$, $\Pr[\alpha(u) = 0, \alpha(v) = 1] := x_v - x_{u,v}$, $\Pr[\alpha(u) = 1, \alpha(v) = 0] := x_u - x_{u,v}$, $\Pr[\alpha(u) = \alpha(v) = 0] := 1 - x_u - x_v + x_{u,v}$.

The last constraint is specific to k-Star Transversal, and it is easy to see that it is a valid relaxation: Given a feasible integral solution $y : V_G \mapsto \{0,1\}$, the last constraint is vacuously satisfied when $y_u = x_u = 1$, and if not, it requires that at least $\deg(u) - k + 2$ vertices should be picked in the transversal so that there is no copy of k-Star in the induced subgraph centered on u. The first phase proceeds as the following.

- Let $S \leftarrow \emptyset$.
- Repeat the following until the size of S does not increase in one iteration.
  - Solve the above Sherali-Adams hierarchy for $V_G \setminus S$ – it means to solve the above LP with additional constraints $x_v = 1$ for all $v \in S$, which also implies $x_{u,v} = x_u$ for $v \in S, u \in V_G$. Denote this LP by $\mathsf{SA}(S)$.
  - $S \leftarrow \{v : x_v \geq \frac{1}{\alpha}\}$, where $\alpha := 10$.

We need to establish three properties from the first phase:

- The size of S is close to that of the optimal k-Star Transversal.
- Maximum degree of the subgraph induced by $V_G \setminus S$ is small.
- The remaining solution has small fractional values – $x_v < \frac{1}{\alpha}$ for all $v \in V_G \setminus S$.

The final property is satisfied by the procedure. The following two lemmas establish the other two properties.

▶ **Lemma 12.** *Let* Frac *be the optimal value of* $\mathsf{SA}(\emptyset)$. *When the above procedure terminates,* $|S| \leq \alpha \, \mathsf{Frac}$.

Proof. Assume that the above loop iterated l times, and for $i = 0, ..., l$, let S_i be S after the ith loop such that $S_0 = \emptyset, ..., S_l = S$. We use induction from the last iteration. Let Frac_i be the optimal fractional solution to $\mathsf{SA}(S_i)$ minus $|S_i|$ such that $\mathsf{Frac} = \mathsf{Frac}_0$.

We first establish $|S_l| - |S_{l-1}| \leq \alpha \, \mathsf{Frac}_{l-1}$. This is easy to see because, when x is the optimal fraction solution to $\mathsf{SA}(S_{l-1})$,

$$|S_l| - |S_{l-1}| = |\{v \notin S_{l-1} : x_v \geq \frac{1}{\alpha}\}| \leq \alpha \, \mathsf{Frac}_{l-1} \, .$$

For $i = l-2, l-1, ..., 0$, we show that $|S_l| - |S_i| \leq \alpha \, \mathsf{Frac}_i$. Let x be the optimal fraction solution to $\mathsf{SA}(S_i)$, and x' be the solution obtained by *partially rounding* x in the following way.

- $x'_v = 1$ if $v \in S_i$. Otherwise, $x'_v = x_v$.
- $x'_{u,v} = x'_u$ $(v \in S_i)$, x'_v $(u \in S_i)$, or $x_{u,v}$ otherwise.

It is easy to check that it is a feasible solution to $\mathsf{SA}(S_{i+1})$ (intuitively, *rounding up* only helps feasibility), so its value is

$$|S_i| + \sum_{v \notin S_i, x_v < \frac{1}{\alpha}} x_v \geq |S_i| + \mathsf{Frac}_{i+1},$$

which implies

$$\mathsf{Frac}_i = \sum_{v \notin S_i, x_v \geq \frac{1}{\alpha}} x_v + \sum_{v \notin S_i, x_v < \frac{1}{\alpha}} x_v \geq \frac{1}{\alpha}(|S_{i+1}| - |S_i|) + \mathsf{Frac}_{i+1} \, .$$

Finally, we have

$$\begin{aligned} & |S_l| - |S_i| \\ = & (|S_l| - |S_{i+1}|) + (|S_{i+1}| - |S_i|) \\ \leq & \alpha \, \mathsf{Frac}_{i+1} + (|S_{i+1}| - |S_i|) \\ \leq & \alpha \, \mathsf{Frac}_i, \end{aligned}$$

where the first inequality follows from the induction hypothesis. This completes the induction. ◀

▶ **Lemma 13.** *After the termination, every vertex has degree at most $2k$ in the subgraph induced by $V_G \setminus S$.*

Proof. We prove that at least one vertex is added to S if the subgraph induced by $V_G \setminus S$ has a vertex of degree more than $2k$. Fix one such iteration, and let S_1 and S_2 be S before and after the iteration respectively. Let G' be the subgraph of G induced by $V_G \setminus S_1$. If the subgraph induced by $V_G \setminus S_2$ does not have any vertex with degree more than $2k$, we are done. Otherwise, fix one such vertex $u \in V_G \setminus S_2$. Note that the degree of u in G' is also more than $2k$.

We show that at least one neighbor v of u satisfies $v \notin S_1$ but $v \in S_2$. Let x be the optimal fractional solution to $\mathsf{SA}(S_1)$ and consider the following constraint for u.

$$\sum_{v:(u,v)\in E_G} (x_v - x_{u,v}) \geq (\mathsf{deg}(u) - k + 2)(1 - x_u).$$

Let $\mathsf{Nbr}(u)$ and $\mathsf{Nbr}'(u)$ be the set of neighbors of u in G and G' respectively, and $\mathsf{deg}'(u) = |\mathsf{Nbr}'(u)|$. Note that $\mathsf{Nbr}'(u) = \mathsf{Nbr}(u) \setminus S_1$, and for $v \in \mathsf{Nbr}(u) \cap S_1$, $x_v = 1$ and $x_{u,v} = x_u$. Therefore, the above constraint is equivalent to

$$\sum_{v:\mathsf{Nbr}(u)\cap S_1} (1 - x_u) + \sum_{v:\mathsf{Nbr}'(u)} (x_v - x_{u,v}) \geq (\mathsf{deg}(u) - k + 2)(1 - x_u)$$
$$\Leftrightarrow \sum_{v:\mathsf{Nbr}'(u)} (x_v - x_{u,v}) \geq (\mathsf{deg}'(u) - k + 2)(1 - x_u).$$

The fact that $u \notin S_2$ implies that $x_u < \frac{1}{\alpha}$, which implies

$$\sum_{v\in\mathsf{Nbr}'(u)} x_v$$
$$\geq \sum_{v\in\mathsf{Nbr}'(u)} (x_v - x_{u,v}) \geq (1 - \frac{1}{\alpha})(\mathsf{deg}'(u) - k) = \qquad (1 - \frac{1}{\alpha})\,\mathsf{deg}'(u)(1 - \frac{k}{\mathsf{deg}'(u)}).$$

Therefore, there is one $v \in \mathsf{Nbr}'(u)$ with $x_v \geq (1 - \frac{1}{\alpha})(1 - \frac{k}{\mathsf{deg}'(u)}) \geq \frac{9}{10} \cdot \frac{1}{2} > \frac{1}{\alpha}$. v satisfies $v \notin S_1$ but $v \in S_2$. ◀

A.2 Constrained Set Multicover

The first phase returns a set S whose size is at most α times the optimal solution and the subgraph induced by $V_G \setminus S$ has maximum degree at most $2k$. As above, let G' be the subgraph induced by $V_G \setminus S$, $\mathsf{Nbr}(u), \mathsf{Nbr}'(u)$ be the neighbors of u in G and G' respectively, and $\mathsf{deg}(u) = |\mathsf{Nbr}(u)|, \mathsf{deg}'(u) = |\mathsf{Nbr}'(u)|$. The remaining task is to find a small subset $F \subseteq V_G \setminus S$ such that the subgraph of G' (and G) induced by $V_G \setminus (S \cup F)$ has no vertex of degree at least $k - 1$. We reduce the remaining problem to the *Constrained Set Multicover* problem defined below.

▶ **Definition 14.** Given an set system $U = \{e_1, ..., e_n\}$, a collection of subsets $\mathcal{C} = \{C_1, ..., C_m\}$, and a positive integer r_e for each $e \in U$, the Constrained Set Multicover problem asks to find the smallest subcollection (each set must be used at most once) such that each element e is covered by at least r_e times.

Probably the most natural greedy algorithm does the following:

- Pick a set C with the largest cardinality (ties broken arbitrarily).

- Set $r_e \leftarrow r_e - 1$ for $e \in C$. If $r_e = 0$, remove it from U. For each $C \in \mathcal{C}$, let $C \leftarrow C \cap U$.
- Repeat while U is nonempty.

Constrained Set Cover has the following standard LP relaxation, and Rajagopalan and Vazirani [46] showed that the greedy algorithm gives an integral solution whose value is at most H_d (i.e. the dth harmonic number) times the optimal solution to the LP, where d is the maximum set size.

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{C \in \mathcal{C}} z_C & \\ \text{subject to} & \displaystyle\sum_{C : e \in C} z_C \geq r_e & e \in U \\ & 0 \leq z_C \leq 1 & C \in \mathcal{C} \end{array}$$

Our remaining problem, k-Star Transversal on G', can be thought as an instance of Constrained Set Cover in the following way: $U := \{u \in V_G \setminus S : \mathsf{deg}'(u) \geq k - 1\}$ with $r_u := \mathsf{deg}'(u) - k + 2$, and for each $v \in V_G \setminus S$, add $\mathsf{Nbr}'(v) \cap U$ to $\mathcal{C}$. Intuitively, this formulation requires at least r_u neighbors be picked in the transversal whether u is picked or not. This is not a valid reduction because the optimal solution of the above formulation can be much more than the optimal solution of our problem. However, at least one direction is clear (any feasible solution to the above formulation is feasible for our problem), and it suffices to show that the above LP admits a solution whose value is close to the optimum of our problem. The LP relaxation of the above special case of Constrained Set Cover is the following:

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{v \in V_G \setminus S} z_v & \\ \text{subject to} & \displaystyle\sum_{v : v \in \mathsf{Nbr}'(u)} z_v \geq \mathsf{deg}'(u) - k + 2 & u \in U \\ & 0 \leq z_v \leq 1 & v \in V_G \setminus S \end{array}$$

Consider the last iteration of the first phase where we solved $\mathsf{SA}(S)$. Let x be the optimal solution to $\mathsf{SA}(S)$ and $\mathsf{Frac} := \sum_v x_v - |S|$. Note that $x_v < \frac{1}{\alpha}$ when $v \notin S$. Define $\{y_v\}_{v \in V \setminus S}$ such that $y_v := 2x_v$.

▶ **Lemma 15.** *$\{y_v\}$ is a feasible solution to the above LP for Constrained Set Cover.*

Proof. By construction $0 \leq y_v < \frac{2}{\alpha}$, so it suffices to check for each $u \in U$,

$$\sum_{v : v \in \mathsf{Nbr}'(u)} y_v \geq \mathsf{deg}'(u) - k + 2.$$

Fix $u \in U$. Recall that Sherali-Adams constraints on x imply that

$$\begin{aligned} & \sum_{v : \mathsf{Nbr}'(u)} (x_v - x_{u,v}) \geq (\mathsf{deg}'(u) - k + 2)(1 - x_u) \\ \Rightarrow & \sum_{v : \mathsf{Nbr}'(u)} x_v \geq (\mathsf{deg}'(u) - k + 2)(1 - x_u) \\ \Rightarrow & \sum_{v : \mathsf{Nbr}'(u)} 2x_v \geq \mathsf{deg}'(u) - k + 2, \end{aligned}$$

where the last line follows from the fact that $1 - \frac{1}{\alpha} > \frac{1}{2}$. ◀

Therefore, Constrained Set Cover LP admits a feasible solution of value $2\,\mathsf{Frac}$, and the greedy algorithm gives a k-Star Transversal F with $|F| \leq 2 \cdot \mathsf{Frac} \cdot H_{2k}$. Since Frac is at most the size of the optimal k-Star Transversal for G' (and clearly G), $|S \cup F|$ is at most $O(\log k)$ times the size of the smallest k-Star Transversal of G.

Towards a Characterization of Approximation Resistance for Symmetric CSPs

Venkatesan Guruswami[*] **and Euiwoong Lee**[†]

Carnegie Mellon University
Pittsburgh, PA 15213, USA
`guruswami@cmu.edu`, `euiwoonl@cs.cmu.edu`

Abstract

A Boolean constraint satisfaction problem (CSP) is called approximation resistant if independently setting variables to 1 with some probability α achieves the best possible approximation ratio for the fraction of constraints satisfied. We study approximation resistance of a natural subclass of CSPs that we call *Symmetric Constraint Satisfaction Problems (SCSPs)*, where satisfaction of each constraint only depends on the number of true literals in its scope. Thus a SCSP of arity k can be described by a subset $S \subseteq \{0, 1, \ldots, k\}$ of allowed number of true literals.

For SCSPs without negation, we conjecture that a simple sufficient condition to be approximation resistant by Austrin and Håstad is indeed necessary. We show that this condition has a compact analytic representation in the case of symmetric CSPs (depending only on the gap between the largest and smallest numbers in S), and provide the rationale behind our conjecture. We prove two interesting special cases of the conjecture, (i) when S is an interval (i.e., $S = \{i \mid l \leq i \leq r\}$ for some $l \leq r$) and (ii) when S is even (i.e., $s \in S \Leftrightarrow k - s \in S$). For SCSPs with negation, we prove that the analogous sufficient condition by Austrin and Mossel is necessary for the same two cases, though we do not pose an analogous conjecture in general.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Constraint Satisfaction Problems, Approximation resistance

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.305

1 Introduction

Constraint Satisfaction Problems (CSPs) are among the most fundamental and well-studied class of optimization problems. Given a fixed integer k and a predicate $Q \subseteq \{0, 1\}^k$, an instance of CSP(Q) *without negation* is specified by a set of variables $X = \{x_1, \ldots, x_n\}$ on the domain $\{0, 1\}$ and a set of constraints $\mathcal{C} = \{C_1, \ldots, C_m\}$, where each constraint $C_j = (x_{j,1}, \ldots, x_{j,k})$ is a k-tuple of variables. An assignment $X \mapsto \{0, 1\}$ satisfies C_j if $(x_{j,1}, \ldots, x_{j,k}) \in Q$. For an instance of CSP(Q) *with negation*, each constraint C_j is additionally given offsets $(b_{j,1}, \ldots, b_{j,k}) \in \{0, 1\}^k$ and is satisfied if $(x_{j,1} \oplus b_{j,1}, \ldots, x_{j,k} \oplus b_{j,k}) \in Q$ where $\oplus$ denotes the addition in $\mathbb{F}_2$. The goal is to find an assignment that satisfies as many constraints as possible.

CSPs contain a large number of famous problems such as Max-SAT (with negation), and Max-Cut / Max-Set-Splitting (without negation) by definition. They have always played a crucial role in the theory of computational complexity, as many breakthrough results such as the NP-completeness of 3SAT, the Probabilistically Checkable Proofs (PCP) theorem, and the Unique Games Conjecture (UGC) study hardness of a certain CSP.

[*] Supported in part by NSF grant CCF-1115525.
[†] Supported by a Samsung Fellowship and NSF grant CCF-1115525.

Based on these works, recent works on approximability of CSPs focus on characterizing every CSP according to its approximation resistance. We define *random assignments* to be the class of algorithms that assign $x_i \leftarrow 1$ with probability α independently. A CSP is called *approximation resistant*, if for any $\epsilon > 0$, it is NP-hard to have a $(\rho^* + \epsilon)$-approximation algorithm, where ρ^* is the approximation ratio achieved by the best random assignment. Even assuming the UGC, the complete characterzation of approximation resistance has not been found, and previous works either change the notion of approximation resistance or study a subclass of CSPs to find a characterization, and more general results tend to suggest more complex characterizations.

This work considers a natural subclass of CSPs where a predicate Q is *symmetric* – for any permutation $\pi : [k] \mapsto [k]$, $(x_1, \ldots, x_k) \in Q$ if and only if $(x_{\pi(1)}, \ldots, x_{\pi(k)}) \in Q$. Equivalently, for every such Q, there exists $S \subseteq [k] \cup \{0\}$ such that $(x_1, \ldots, x_k) \in Q$ if and only if $(x_1 + \cdots + x_k) \in S$. Let SCSP($S$) denote such a symmetric CSP. While this is a significant restriction, it is a natural one that still captures the following fundamental problems, such as Max-SAT, Max-Not-All-Equal-SAT, t-out-of-k-SAT (with negation), and Max-Cut, Max-Set-Splitting, Discrepancy minimization (without negation). Except the work of Austrin and Håstad [2], many works on this line focused CSPs with negation, while we feel that the aforementioned problems without negation have a very natural interpretation as (hyper)graph coloring and are worth studying.

There is a simple sufficient condition to be approximation resistant due to Austrin and Mossel [4] with negation, and due to Austrin and Håstad [2] without negation. For SCSPs, we show that these simple sufficient conditions can be further simplfied and understood more intuitively, and suggest that they might also be necessary for and thus precisely characterize approximation resistance. We prove it for two natural special cases (which capture all problems mentioned in the last paragraph) for both SCSPs with / without negation, and provide reasons that we believe this is true at least for SCSPs without negation.

1.1 Related Work

Given the importance of CSPs and the variety of problems that can be formulated as a CSP, it is a natural task to classify all CSPs according to their computational complexity for some well-defined task. For the task of deciding satisfiability (i.e., finding an assignment that satisfies every constraint if there is one), the work of Schaefer [14] gave a complete characterization on the Boolean domain in 1978.

However, such a classification seems much harder when we study approximability of CSPs. Since the seminal work of Håstad [11], many natural problems have been proven to be approximation resistant. These examples include Max-3SAT / Max-3LIN (with negation) and Max-4-Set-Splitting (without negation), and for Boolean CSPs of arity 3, putting together the hardness results of [11] with the algorithmic results of Zwick [16], it is known that a CSP is approximation resistant if and only if it is implied by parity. However, characterizing approximation resistance of every CSP for larger arity k is a harder task. The Ph.D. thesis of Hast [10] is devoted to this task for $k = 4$, and succeeds to classify 354 out of 400 predicates.

The advent of the Unique Games Conjecture (UGC) [12], though it is not as widely believed as $\mathbf{P} \neq \mathbf{NP}$, revived the hope to classify every CSP according to its approximation resistance. For CSPs with negation, the work of Austrin and Mossel [4] gave a simple sufficient condition to be approximation resistant, namely the existence of a balanced pairwise independent distribution that is supported on the satisfying assignments of the predicate. The work of Austrin and Håstad [2] proved a similar sufficient condition for CSPs without negation, and that if this condition is not met, this predicate (both with /

without negation) is *useful* for some polynomial optimization – for every such Q, there is a k-variate polynomial $p(y_1, \ldots, y_k)$ such that if we are given an instance of CSP(Q) that admits a $(1-\epsilon)$-satisfying assignment, the altered problem, where we change each constraint C_j's payoff from $\mathbb{I}[(x_{j,1} \oplus b_{j,1}, \ldots, x_{j,k} \oplus b_{j,k}) \in Q]$ (where $\mathbb{I}[\cdot]$ is the indicator function) to $p(x_{j,1} \oplus b_{j,1}, \ldots, x_{j,k} \oplus b_{j,k})$, admits an approximation algorithm that does better than any random assignment.

Predicates that don't admit a pairwise independent distribution supported on their satisfying assignments can be expressed as the sign of a quadratic polynomial (see [2]). This motivates the study of the approximability of such predicates, though it is known that there are approximation resistant predicates that can be expressed as a quadratic threshold function and thus the sufficient condition of Austrin and Mossel [4] is not necessary for approximation resistance. Still this motivates the question of understanding which quadratic threshold functions can be approximated non-trivially.

Cheraghchi, Håstad, Isaksson, and Svensson [8] studied the simpler case of predicates which are the sign of a linear function with no constant term, obtaining algorithms beating the random assignment threshold of 1/2 in some special cases. Austrin, Benabbas, and Magen [1] conjecture that every such predicate can be approximated better than a factor 1/2 and is therefore not approximation resistant. They prove that predicates that are the sign of *symmetric* quadratic polynomials with no constant term are not approximation resistant.

Assuming the UGC, the work of Austrin and Khot [3] gave a characterization of approximation resistance for even k-partite CSPs, and Khot, Tulsiani, and Worah [13] gave a characterization of *strong approximation resistance* for general CSPs – strong approximation resistance roughly means hardness of finding an assignment that deviates from the performance of the random assignment in either direction (i.e., it is hard to also find an assignment saisfying a noticeably smaller fraction of constraints than the random assignment). These two works are notable in studying approximation resistance of general CSPs, but their characterizations become more complicated, which they suggest is necessary.

Without the UGC, even the existence of pairwise independent distribution supported on the predicate is not known to be sufficient for approximation resistance. Another line of work shows partial results either by using a stronger condition [7], or by using a restricted model of computation (e.g., Sherali-Adams or Lasserre hierarchy of convex relaxations) [15, 6, 5].

1.2 Our Results

Our work was initially motivated by a simple observation that for symmetric CSPs, the sufficient condition to be approximation resistant by Austrin and Håstad [2] admits a more compact and intuitive two-dimensional description in $\mathbb{R}^2$.

Fix a positive integer k and denote $[k] = \{1, 2, \ldots, k\}$. For $s \in [k] \cup \{0\}$, let $P(s) \in \mathbb{R}^2$ be the point defined by $P(s) := (\frac{s}{k}, \frac{s(s-1)}{k(k-1)})$. For any s, $P(s)$ lies on the curve $y = \frac{k}{k-1}x^2 - \frac{x}{k-1}$, which is slightly below the curve $y = x^2$ for $x \in [0, 1]$. Given a subset $S \subseteq [k] \cup \{0\}$, let $P_S := \{P(s) : s \in S\}$ and $\mathsf{conv}(P_S)$ be the convex hull of P_S. For symmetric CSPs, the condition of Austrin and Håstad depends on whether this convex hull intersects a certain curve or a point.

For SCSP(S) without negation, the condition becomes whether $\mathsf{conv}(P_S)$ intersects the curve $y = x^2$. If we let s_{min} and s_{max} be the minimum and maximum number in S respectively, by convexity of $y = \frac{k}{k-1}x^2 - \frac{x}{k-1}$, it is equivalent to that the line passing $P(s_{min})$ and $P(s_{max})$ and $y = x^2$ intersect, which is again equivalent to (see Lemma A.4)

$$\frac{(s_{max} + s_{min} - 1)^2}{k-1} \geq \frac{4 s_{max} s_{min}}{k}. \tag{1}$$

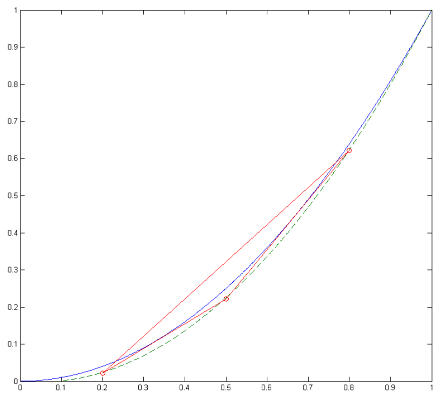


Figure 1 An example when $k = 10$ and $S = \{2, 5, 8\}$. The solid curve is $y = x^2$ and the dashed curve is $y = \frac{k}{k-1}x^2 - \frac{x}{k-1}$, where all $P(s)$ lie. In this case the triangle $\mathsf{conv}(P_S)$ intersects $y = x^2$, so SCSP(S) is approximation resistant.

A simple calculation shows that the above condition is implied by $(s_{max} - s_{min}) \geq \sqrt{2(s_{max} + s_{min})}$ which in turn holds if $(s_{max} - s_{min}) \geq 2\sqrt{k}$. This means that SCP$(S)$ is approximation resistant unless s_{min} and s_{max} are very close. See Figure 1 for an example. We conjecture that this simple condition completely characterizes approximation resistance of symmetric CSPs without negation. Note that we exclude the cases where S contains 0 or k, since without negation, a trivial deterministic strategy to give the same value to every variable satisfies every constraint.

▶ **Conjecture 1.1.** *For $S \subseteq [k-1]$, SCSP(S) without negation is approximation resistant if and only if* (1) *holds.*

The hardness claim, the "if" part, is currently proved only under the UGC, but our focus is on the algorithmic claim that the violation of (1) leads to an approximation algorithm that outperforms the best random assignment. Even though we were not formally able to prove Conjecture 1.1, we explain the rationale behind the conjecture and we prove it for the following two natural special cases in Section 2:

1. S is an *interval*: S contains every integer from s_{min} to s_{max}.
2. S is *even*: $s \in S$ if and only if $k - s \in S$.

▶ **Theorem 1.2.** *If $S \subseteq [k-1]$ and S is either an interval or even, SCSP(S) without negation is approximation resistant if and only if* (1) *holds (the hardness claim, i.e., the "if" part, is under the Unique Games conjecture).*

For SCSP(S) with negation, the analogous condition is whether $\mathsf{conv}(P_S)$ contains a single point $(\frac{1}{2}, \frac{1}{4})$ or not. While it is tempting to pose a conjecture similar to Conjecture 1.1, we refrain from doing so due to the lack of evidence compared to the case without negation. However, we prove the following theorem which shows that the analogous characterization works at least for the two special cases introduced above.

▶ **Theorem 1.3.** *If $S \subset [k] \cup \{0\}$ and S is either an interval or even, SCSP(S) with negation is approximation resistant if and only if* $\mathsf{conv}(P_S)$ *contains* $(\frac{1}{2}, \frac{1}{4})$ *(the hardness claim, i.e., the "if" part, is under the Unique Games conjecture).*

1.3 Techniques

We mainly focus on SCSPs without negation, and briefly sketch why the violation of (1) might lead to an approximation algorithm that outperforms the best random assignment. Let α^* be the probability that the best random assignment uses, and ρ^* be the expected fraction of constraints satisfied by it. Our algorithms follow the following general framework: sample correlated random variables $X_1, \ldots, X_n$, where each X_i lies in $[-\alpha^*, 1-\alpha^*]$, and independently round $x_i \leftarrow 1$ with probability $\alpha^* + X_i$.

Fix one constraint $C = (x_1, \ldots, x_k)$ (for SCSPs with negation, additionally assume that offsets are all 0). Using symmetry, the probability that it is satisfied by the above strategy can be expressed as

$$\rho^* + \sum_{l=1}^{k} c_l \mathop{\mathbb{E}}_{I \in \binom{[k]}{l}} [\prod_{i \in I} X_i].$$

For some coefficients $\{c_l\}_{l \in [k]}$. These coefficients c_l can be expressed by the following two ways.

- Let $f(\alpha) : [0,1] \mapsto [0,1]$ the probability that a constraint is satisfied by a random assignment with probability α. c_l is proportional to $f^{(l)}(\alpha^*)$, the l'th derivative of f evaluated at α^*.
- Let $Q = \{(x_1, \ldots, x_k) \in \{0,1\}^k : (x_1 + \cdots + x_k) \in S\}$ be the predicate associated with S. When $\alpha^* = \frac{1}{2}$, c_l is proportional to the Fourier coefficient $\hat{Q}(T)$ with $|T| = l$.

Given this observation, α^* for SCSPs without negation has nice properties since it should be a global maximum in the interval $[0,1]$. In particular, it should be a local maximum so that $c_1 = f'(\alpha) = 0$ and $c_2, f''(\alpha) \leq 0$. By modifying an algorithm by Austrin and Håstad [2], we prove that we can sample $X_1, \ldots, X_n$ such that the average second moment $\mathbb{E}[X_i X_j]$ is strictly negative if (1) does not hold. By scaling X_i's so that the product of at least three X_i's becomes negligible, this idea results in an approximation algorithm that outperforms the best random assignment, except the degenerate case where $c_2 = f''(\alpha^*) = 0$ even though α^* is a local maximum. This is the main rationale behind Conjecture 1.1 and we elaborate this belief more in Section 2. It is notable that our conjectured characterization for the case without negation only depends on the minimum and the maximum number in S, while α^* depends on other elements.

For SCSPs with negation where α^* is fixed to be $\frac{1}{2}$, the situation becomes more complicated since c_1 and $f'(\alpha)$ are not necessarily zero and there are many ways that $\mathsf{conv}(P_S)$ does not contain $(\frac{1}{2}, \frac{1}{4})$ (in the case of SCSPs without negation, the slope of the line separating $\mathsf{conv}(P_S)$ and $y = x^2$ is always positive, but it is not the case here). Therefore, a complete characterization requires understanding interactions among c_1, c_2, and the separating line. We found that the somewhat involved method of Austrin, Benabbas, and Magen [1] gives a way to sample these $X_1, \ldots, X_n$ with desired first and second moments to prove our results when S exhibits additional special structures, but believe that a new set of ideas are required to give a complete characterization.

1.4 Organization

In Section 2, we study SCSPs without negation. We further elaborate our characterization in Section 2.1, and provide an algorithm in Section 2.2. We study SCSPs with negation in Section 3.

2 Symmetric CSPs without negation

2.1 A 2-dimensional characterization

Fix k and $S \subseteq [k-1]$. Our conjectured condition to be approximation resistant is that $\mathsf{conv}(P_S)$ intersects the curve $y = x^2$, which is equivalent to (1). Austrin and Håstad [2] proved that this simple condition is sufficient to be approximation resistant.

▶ **Theorem 2.1** ([2]). *Let $S \subseteq [k-1]$ be such that* (1) *holds. Then, assuming the Unique Games Conjecture, SCSP(S) without negation is approximation resistant.*

They studied general CSPs and their condition is more complicated than stated here. See Appendix A to see how it is simplified for SCSPs. We conjecture that for SCSPs, this condition is indeed equivalent to approximation resistance.

▶ **Conjecture 2.2** (Restatement of Conjecture 1.1). *For $S \subseteq [k-1]$, SCSP(S) without negation is approximation resistant if and only if* (1) *holds.*

To provide our rationale behind the conjecture, we define the function $f : [0,1] \mapsto [0,1]$ to be the probability that one constraint is satisfied by the random assignment that gives $x_i \leftarrow 1$ independently with probability α.

$$f(\alpha) = \sum_{s \in S} \binom{k}{s} \alpha^s (1-\alpha)^{k-s}$$

Let $\alpha^* \in [0,1]$ be a value that maximizes $f(\alpha)$, and $\rho^* := f(\alpha^*)$. There might be more than one α with $f(\alpha) = \rho^*$. In Section 2.2, we prove that S is not approximation resistant if there exists one such α^* with a negative second derivative.

▶ **Theorem 2.3.** *$S \subseteq [k-1]$ be such that* (1) *does not hold and there exists $\alpha^* \in [0,1]$ such that $f(\alpha^*) = \rho^*$ and $f''(\alpha^*) < 0$. Then, there is a randomized polynomial time algorithm for SCSP(S) that satisfies strictly more than ρ^* fraction of constraints in expectation.*

Since $f(0) = f(1) = 0 < \rho^*$, every $\alpha \in [0,1]$ with $f(\alpha) = \rho^*$ must be a local maximum, so it should have $f'(\alpha) = 0$ and $f''(\alpha) \leq 0$. If α is a local maximum, $f''(\alpha) = 0$ also implies $f'''(\alpha) = 0$, so ruling out this degeneracy at a global maximum gives the complete characterization!

Ruling out this degeneracy at a global maximum does not seem to be closely related to general shape of $f(\alpha)$ or S. It might still hold even if $f(\alpha)$ has multiple global maxima, or S satisfies (1) so that SCSP(S) is approximation resistant.

However, examples in Figure 2 led us to believe that the condition (1) is also related to general shape of f. When S contains two numbers l and r with $l + r = k$, as two numbers become far apart, f becomes unimodal to bimodal, and the transition happens exactly when (1) starts to hold. Furthermore, the degenerate case $f'(\alpha^*) = f''(\alpha^*) = 0$ happens when (1) holds with equality. Intuitively, when two numbers l and r are far apart, it is a better strategy to focus on only one of them (i.e. $\alpha^* \approx \frac{l}{k}$ or $\frac{r}{k}$) so f is bimodal, but if (1) does not hold and l and r are close enough, it is better to target in the middle to satisfy both l and r with reasonability probability so that f is unimodal with a large negative curvature at α^*.

Having more points between l and r seems to strengthen the above intuition, and removing the assumption that $l + r = k$ only seems to add algebraic complication without hurting the intuition. Thus, we propose the following stronger conjecture that implies Conjecture 1.1.

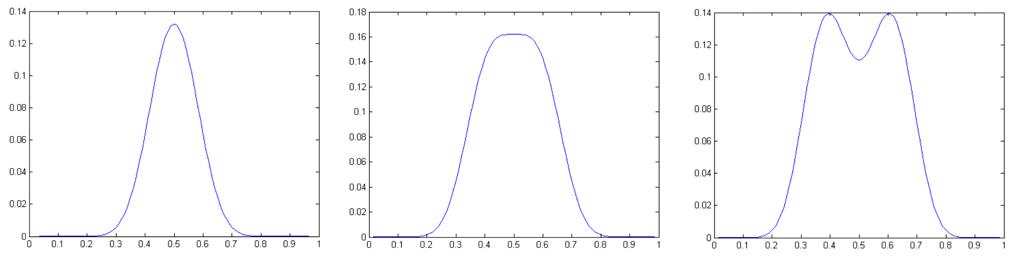


Figure 2 Examples for $k = 36$. **Left**: $S = \{18\}$, (1) is not satisfied, unimodal with $\alpha^* = \frac{1}{2}$, $f''(\frac{1}{2}) < 0$. **Middle**: $S = \{15, 21\}$, (1) is satisfied with equality, unimodal with $\alpha^* = \frac{1}{2}$, but $f''(\frac{1}{2}) = 0$. **Right**: $S = \{14, 22\}$, (1) is satisfied with slack, bimodal with two α^*, but $f''(\alpha^*) < 0$.

▶ **Conjecture 2.4.** *If* (1) *does not hold,* $f(\alpha)$ *is unimodal in* $[0, 1]$ *with the unique maximum at* α^**, and* $f''(\alpha^*) < 0$.

While we are unable to formally prove Conjecture 2.4 for every S, we establish it for the case when S is either an interval (Section 2.3) or even (Section 2.4), thus proving Theorem 1.2.

2.2 Algorithm

Let $\alpha^* \in [0, 1]$ be such that $f(\alpha^*) = \rho^*$ and $f''(\alpha^*) < 0$. Furthermore, suppose that S does not satisfy (1). We give a randomized approximation algorithm which is guaranteed to satisfy strictly more than ρ^* fraction of constraints in expectation, proving Theorem 2.3. Let $D := D(k)$ be a large constant determined later. Our strategy is the following.

1. Sample $X_1, \dots, X_n$ from some correlated multivariate normal distribution where each X_i has mean 0 and variance at most σ^2 for some $\sigma := \sigma(k)$.
2. For each $i \in [n]$, set
$$X'_i = \begin{cases} -D\alpha^* & \text{if } X_i < -D\alpha^* \\ D(1-\alpha^*) & \text{if } X_i > D(1-\alpha^*) \\ X_i & \text{otherwise} \end{cases}$$
so that $\alpha^* + \frac{X'_i}{D}$ is always in $[0, 1]$.
3. Set $x_i \leftarrow 1$ independently with probability $\alpha^* + \frac{X'_i}{D}$.

Fix one constraint C and suppose that $C = (x_1, \dots, x_k)$. We consider a multivariate polynomial

$$g(y_1, \dots, y_k) := \sum_{T \subseteq [k], |T| \in S} \prod_{i \in T} (\alpha^* + \frac{y_i}{D}) \prod_{i \in [k] \setminus T} (1 - \alpha^* - \frac{y_i}{D}).$$

$g(X'_1, \dots, X'_k)$ is equal to the probability that the constraint C is satisfied. By symmetry, for any $1 \le i_1 < \dots < i_l \le k$, the coefficient of a monomial $y_{i_1} y_{i_2} \dots y_{i_l}$ only depends on l. Let c_l be this coefficient.

▶ **Lemma 2.5.** $c_l = \frac{(k-l)!}{k! D^l} f^{(l)}(\alpha^*)$.

Proof. Note that $g(y, y, \dots, y) = f(\alpha^* + \frac{y}{D})$, which has the Taylor expansion

$$\sum_{l=0}^{k} \frac{f^{(l)}(\alpha^*)}{l!} (\frac{y}{D})^l.$$

Since g is multilinear, by symmetry, the coefficient of a monomial $y_{i_1}y_{i_2}\dots y_{i_l}$ in $g(y_1,\dots,y_k)$ is equal to the coefficient of y^l in $f(\alpha^*+\frac{y}{D})$ divided by $\binom{k}{l}$, which is $c_l=\frac{(k-l)!}{k!D^l}f^{(l)}(\alpha^*)$. ◀

We analyze the overall performance of this algorithm. Let $\mathcal{D}_l$ be the distribution on $\binom{[n]}{l}$ where we sample a constraint C uniformly at random, sample l distinct variables from $\binom{C}{l}$, and output their indices. We prove the following lemma, which implies that by taking large D, the effect of truncation from X_i to X'_i and the contribution of monomials of degree greater than two become small.

▶ **Lemma 2.6.** *The expected fraction of constraints satisfied by the above algorithm is at least*

$$\rho^*+c_2\binom{k}{2}\mathop{\mathbb{E}}_{(i,j)\sim\mathcal{D}_2}[X_iX_j]-O_k(\frac{1}{D^3})=\rho^*+\frac{f''(\alpha^*)}{2D^2}\mathop{\mathbb{E}}_{(i,j)\sim\mathcal{D}_2}[X_iX_j]-O_k(\frac{1}{D^3}),$$

where $O_k(\cdot)$ is hiding constants depending on k.

Proof. Note that as long as S does not contain 0 or k, $\alpha^*\in[\frac{1}{k},1-\frac{1}{k}]$. For any $1\le l\le k$ and $1\le i_1<\dots<i_l\le k$, we apply Lemma B.1 (set $D\leftarrow\frac{D}{k}$),

$$|\,\mathbb{E}[\prod_{j=1}^{l}X_{i_j}]-\mathbb{E}[\prod_{j=1}^{l}X_{i_j}]|\le 2^l\cdot\sigma^l\cdot l!\cdot e^{-D/kl}.$$

If we expand $f(\alpha)=\sum_{l=0}^{k}a_l\alpha^l$, each coefficient a_l has magnitude at most 2^k, which means that $|f^{(l)}(\alpha^*)|$ is bounded by $k2^kk!$. Therefore, any $|c_l|$ is at most $k2^kk!$. Let c_{max} be this quantity. Summing over this error for all monomials, the probability that a fixed constraint $C=\{x_1,\dots,x_k\}$ is satisfied is

$$\begin{aligned}\mathbb{E}[g(X'_1,\dots,X'_k)]\ge\ &\mathbb{E}[g(X_1,\dots,X_k)]-c_{max}\cdot 2^{2k}\cdot\sigma^k\cdot k!\cdot e^{-D/k^2}\\ =\ &\rho^*+\sum_{l=1}^{k}c_l\sum_{1\le i_1<\dots<i_l\le k}X_{i_1}X_{i_2}\dots X_{i_l}-O_k(e^{-D/k^2})\\ =\ &\rho^*+\sum_{l=1}^{k}c_l\sum_{1\le i_1<\dots<i_l\le k}X_{i_1}X_{i_2}\dots X_{i_l}-O_k(e^{-D/k^2})\end{aligned}$$

Averaging over m constraints, the expected fraction of satisfied constraints is at least

$$\begin{aligned}&\rho^*+\sum_{l=1}^{k}c_l\binom{k}{l}\mathop{\mathbb{E}}_{(i_1,\dots,i_l)\sim\mathcal{D}_l}[X_{i_1}\dots X_{i_l}]-O_k(e^{-D/k^2})\\ =\ &\rho^*+c_2\binom{k}{2}\mathop{\mathbb{E}}_{(i_1,i_2)\sim\mathcal{D}_2}[X_{i_1}X_{i_2}]+\sum_{l=3}^{k}c_l\binom{k}{l}\mathop{\mathbb{E}}_{(i_1,\dots,i_l)\sim\mathcal{D}_l}[X_{i_1}\dots X_{i_l}]-O_k(e^{-D/k^2})\\ =\ &\rho^*+c_2\binom{k}{2}\mathop{\mathbb{E}}_{(i_1,i_2)\sim\mathcal{D}_2}[X_{i_1}X_{i_2}]-O_k(\frac{1}{D^3}),\end{aligned}$$

where the first equality follows from the fact that $\mathbb{E}[X_i]=0$ for all i. Recall that $c_l=\frac{(k-l)!}{k!D^l}f^{(l)}(\alpha^*)$ so that $|c_l|=O_k(\frac{1}{D^l})$. ◀

Therefore, if we have a way to sample $X_1,\dots,X_n$ such that each X_i has mean 0 and variance at most σ^2, and $\mathbb{E}_{(i,j)\sim\mathcal{D}_2}[X_iX_j]<-\delta$ for some $\delta:=\delta(k)>0$, taking D large enough ensures that the algorithm satisfies strictly more than ρ^* fraction of constraints. We now show how to do such a sampling.

We assume that for some $\epsilon := \epsilon(k) > 0$, the given instance admits a solution that satisfies $(1-\epsilon)$ fraction of constraints. Otherwise, the random assignment with probability α^* guarantees the approximation ratio of $\frac{\rho^*}{1-\epsilon}$. The following lemma completes the proof of Theorem 2.3.

▶ **Lemma 2.7.** *Suppose that S does not satisfy (1). For sufficiently small $\epsilon, \delta > 0$ and sufficiently large σ all depending only on k, given an instance of SCSP(S) where $(1-\epsilon)$ fraction of constraints are simultaneously satisfiable, it is possible to sample $X_1, \dots, X_n$ from a multivariate normal distribution such that each X_i has mean 0 and variance bounded by σ^2, and $\mathbb{E}_{(i,j)\sim\mathcal{D}_2}[X_iX_j] < -\delta$.*

Proof. Recall that (1) is equivalent to the fact that the line ℓ passing $P(s_{min})$ and $P(s_{max})$ intersects the curve $y = x^2$. Let a be the value that the vector $(a, -1)$ is orthogonal to ℓ. a is strictly positive since ℓ has a positive slope. If ℓ and $y = x^2$ do not intersect, there is a line with the same slope as ℓ that strictly separates $y = x^2$ and $\{P(s) : s \in S\}$ – in other words, there exists $c \in \mathbb{R}$ such that

- $ax - y + c > \gamma > 0$ for $(x, y) \in \{P(s) : s \in S\}$.
- $ax - x^2 + c < 0$ for any $x \in \mathbb{R} \Rightarrow c < \frac{-a^2}{4}$.

Consider a constraint $C = (x_1, \dots, x_k)$. Since $(\mathbb{E}_{i\in[k]}[x_i], \mathbb{E}_{i\neq j\in[k]}[x_ix_j]) = P(x_1 + \cdots + x_k)$, if C is satisfied,

$$a \mathop{\mathbb{E}}_{i\in[k]}[x_i] - \mathop{\mathbb{E}}_{i\neq j\in[k]}[x_ix_j] + c > \gamma.$$

Let

$$\eta := -\min_{x_1,\dots,x_k\in\{0,1\}} \left(a \mathop{\mathbb{E}}_{i\in[k]}[x_i] - \mathop{\mathbb{E}}_{i\neq j\in[k]}[x_ix_j] + c \right).$$

We solve the following semidefinite programm (SDP):

$$\begin{aligned} \text{maximize} \quad & a \mathop{\mathbb{E}}_{i\in\mathcal{D}_1}[\langle v_0, v_i\rangle] - \mathop{\mathbb{E}}_{i,j\in\mathcal{D}_2}[\langle v_i, v_j\rangle] + c \\ \text{subject to} \quad & ||v_0|| = 1 \\ & \langle v_i, v_0\rangle = ||v_i||^2 \quad \text{for all } i \in [n] \end{aligned}$$

Note that $\langle v_i, v_0\rangle = ||v_i||^2$ implies $||v_i|| \leq 1$. For any assignment to $x_1, \dots, x_n$, setting $v_i = x_iv_0$ satisfies that $x_i = \langle v_0, v_i\rangle$ and $x_ix_j = \langle v_i, v_j\rangle$. Since at least $(1-\epsilon)$ fraction of constraints can be simultaneously satisfied, the optimum of the above SDP is at least $(1-\epsilon)\gamma - \epsilon\eta$. Given $\gamma > 0$ and η, take sufficiently small $\epsilon, \delta > 0$ such that $(1-\epsilon)\gamma - \epsilon\eta = \delta$. There are finitely many S (thus γ and η) for each k, so ϵ and δ can be taken to depend only on k. Given vectors $v_0, v_1, \dots, v_n$, we sample $X_1, \dots, X_n$ by the following simple procedure:

1. Sample a vector g whose coordinates are independent standard normal.
2. Let $X_i = \langle g, v_i - \frac{a}{2}v_0\rangle$.

It is clear that $\mathbb{E}[X_i] = 0$ for each i, and $\mathbb{E}[X_i^2] = ||v_i - \frac{a}{2}v_0||^2 \leq (a+1)^2 + 1$, so taking $\sigma := \sigma(k)$ large enough ensures that the variance of each X_i is bounded by σ^2. We now compute the second moment.

$$\begin{aligned}
& \mathop{\mathbb{E}}_{i,j\sim\mathcal{D}_2}[X_iX_j] \\
= & \mathop{\mathbb{E}}_{i,j\sim\mathcal{D}_2}[\langle v_i - \frac{a}{2}v_0, v_j - \frac{a}{2}v_0\rangle] \\
= & \mathop{\mathbb{E}}_{i,j\sim\mathcal{D}_2}[\langle v_i, v_j\rangle] - a\mathop{\mathbb{E}}_{i\in\mathcal{D}_1}[\langle v_i, v_0\rangle] + \frac{a^2}{4} \\
< & \mathop{\mathbb{E}}_{i,j\sim\mathcal{D}_2}[\langle v_i, v_j\rangle] - a\mathop{\mathbb{E}}_{i\in\mathcal{D}_1}[\langle v_i, v_0\rangle] - c \\
\le & -((1-\epsilon)\gamma - \epsilon\eta) = -\delta,
\end{aligned}$$

where the first inequality follows from $c < -\frac{a^2}{4}$ and the second follows from the optimality of our SDP. ◀

2.3 Case of Interval S

We study properties of $f(\alpha)$ when S is an interval – $S = \{s_{min}, s_{min}+1, \ldots, s_{max}-1, s_{max}\}$, and prove Conjecture 2.4 for this case. One notable fact is that as long as S is interval, the conclusion of Conjecture 2.4 is true even if S does satisfy (1) and becomes approximation resistant.

▶ **Lemma 2.8.** *Suppose $S \subseteq [k-1]$ is an interval. Then, $f(\alpha)$ is unimodal in $[0,1]$ with the unique maximum at α^* and $f''(\alpha^*) < 0$.*

Proof. Let $l := s_{min}$ and $r = s_{max}$. Given

$$f(\alpha) = \sum_{s=l}^{r}\binom{k}{s}\alpha^s(1-\alpha)^{k-s}$$

and

$$f'(\alpha) = \sum_{s=l}^{r}\binom{k}{s}\Big(s\alpha^{s-1}(1-\alpha)^{k-s} - (k-s)\alpha^s(1-\alpha)^{k-s-1}\Big),$$

since $\binom{k}{s}(k-s) = \binom{k}{s+1}(s+1)$, we have

$$f'(\alpha) = \binom{k}{l}l\alpha^{l-1}(1-\alpha)^{k-l} - \binom{k}{r}(k-r)\alpha^r(1-\alpha)^{k-r-1}.$$

If $0 < \alpha < 1$, setting $\beta := \frac{\alpha}{1-\alpha}$ gives a unique non-zero solution to $f'(\beta) = 0$. This proves the unimodality. For the second derivative,

$$\begin{aligned}
f''(\alpha) = & \binom{k}{l}l(l-1)\alpha^{l-2}(1-\alpha)^{k-l} - \binom{k}{l}l(k-l)\alpha^{l-1}(1-\alpha)^{k-l-1} + \\
& \binom{k}{r}(k-r)(k-r-1)\alpha^r(1-\alpha)^{k-r-2} - \binom{k}{r}r(k-r)\alpha^{r-1}(1-\alpha)^{k-r-1} \\
= & \binom{k}{l}l\alpha^{l-2}(1-\alpha)^{k-l-1}\Big((l-1)(1-\alpha) - (k-l)\alpha\Big) + \\
& \binom{k}{r}(k-r)\alpha^{r-1}(1-\alpha)^{k-r-2}\Big((k-r-1)\alpha - r(1-\alpha)\Big).
\end{aligned}$$

Since $\frac{l-1}{k-1} < \frac{l}{k} \leq \alpha^* \leq \frac{r}{k} < \frac{r}{k-1}$,

$$(l-1)(1-\alpha^*) - (k-l)\alpha^* = (l-1) - (k-1)\alpha^* < 0$$

and

$$(k-r-1)\alpha^* - r(1-\alpha^*) = (k-1)\alpha^* - r < 0,$$

so that $f''(\alpha^*) < 0$. ◀

2.4 Case of Even S

We study properties of $f(\alpha)$ when S is even – $s \in S$ if and only if $k-s \in S$, and prove Conjecture 2.4 for this case. We first simplify (1) for this setting. If we let $l := s_{min}$ and $r := s_{max} = k-l$, (1) is equivalent to

$$\frac{(l+r-1)^2}{k-1} \geq \frac{4lr}{k} \quad \Leftrightarrow \quad k(k-1) \geq 4lr \quad \Leftrightarrow \quad (r-l)^2 \geq k.$$

Therefore, (1) is equivalent to

$$r - l \geq \sqrt{k} \tag{2}$$

▶ **Lemma 2.9.** *Suppose $S \subseteq [k-1]$ is even. If (2) does not hold, $f(\alpha)$ is unimodal in $[0,1]$ with the unique maximum at $\alpha^* = \frac{1}{2}$ and $f''(\alpha^*) < 0$.*

Proof. Given a even S, let $S_1 = \{s \in S : s \leq k/2\}$. When we write f_S to denote the dependence of f on S, we can decompose $f_S(\alpha) = \sum_{s \in S_1} f_{\{s,k-s\}}(\alpha)$, so the following claim proves the lemma. ◀

▶ **Claim 2.10.** *Let $l \leq \frac{k}{2}$ and $r = k-l$ such that $r - l < \sqrt{k} \Leftrightarrow k(k-1) < 4lr$. Let $S = \{l, r\}$. f is unimodal with the unique maximum at $\frac{1}{2}$, and $f''(\frac{1}{2}) < 0$.*

Proof. Note that f is symmetric around $\alpha = 1/2$. If there exists a local maximum at $\alpha' \in (0, 1/2)$, f also has a local maximum at $(1-\alpha')$ with the same value, so there must exist a local minimum in $(\alpha', 1-\alpha')$. In particular, there is $\alpha \in (\alpha', 1-\alpha')$ such that $f'(\alpha) = 0$ and $f''(\alpha) \geq 0$. We prove that such α cannot exist.

$$\begin{aligned}
& f'(\alpha) = 0 \\
\Leftrightarrow \quad & \binom{k}{l}\alpha^{l-1}(1-\alpha)^{r-1}(l-k\alpha) + \binom{k}{r}\alpha^{r-1}(1-\alpha)^{l-1}(r-k\alpha) = 0 \\
\Leftrightarrow \quad & \frac{\binom{k}{l}\alpha^{l-1}(1-\alpha)^{r-1}}{\binom{k}{r}\alpha^{r-1}(1-\alpha)^{l-1}} = \frac{\binom{k}{l}(1-\alpha)^{r-l}}{\binom{k}{r}\alpha^{r-l}} = -\frac{(k\alpha - r)}{(k\alpha - l)}
\end{aligned}$$

Similarly,

$$\begin{aligned}
& f''(\alpha) \geq 0 \\
\Leftrightarrow \quad & \frac{\binom{k}{l}(1-\alpha)^{r-l}}{\binom{k}{r}\alpha^{r-l}} \geq -\frac{r(r-1)(1-\alpha)^2 - 2rl\alpha(1-\alpha) + l(l-1)\alpha^2}{l(l-1)(1-\alpha)^2 - 2rl\alpha(1-\alpha) + r(r-1)\alpha^2}.
\end{aligned}$$

By symmetry, we can assume $\alpha \geq \frac{1}{2}$, so that $(k\alpha - l) \geq 0$ and $l(l-1)(1-\alpha)^2 - 2rl\alpha(1-\alpha) + r(r-1)\alpha^2 \geq 0$.

$$\begin{aligned}
&\frac{(k\alpha - r)}{(k\alpha - l)} \leq \frac{r(r-1)(1-\alpha)^2 - 2rl\alpha(1-\alpha) + l(l-1)\alpha^2}{l(l-1)(1-\alpha)^2 - 2rl\alpha(1-\alpha) + r(r-1)\alpha^2} \\
\Leftrightarrow\ & (k\alpha - r)(l(l-1)(1-\alpha)^2 - 2rl\alpha(1-\alpha) + r(r-1)\alpha^2) \\
&\leq (k\alpha - l)(r(r-1)(1-\alpha)^2 - 2rl\alpha(1-\alpha) + l(l-1)\alpha^2) \\
\Leftrightarrow\ & \alpha^2(l^3 - r^3 - (l^2 - r^2) + rl(l-r) - 2k(l^2 - r^2) + 2k(l-r)) + \\
&\qquad \alpha(k(l^2 - r^2) - k(l-r)) - rl(l-r) \leq 0 \\
\Leftrightarrow\ & \alpha^2(-k^2 + k) + \alpha(k^2 - k) - rl \geq 0 \qquad \text{divide by } (l-r) \text{ and use } l + r = k
\end{aligned}$$

However, $\alpha^2(-k^2+k) + \alpha(k^2-k) - rl$ has a negative leading coefficient and its discriminant is

$$(k^2 - k)^2 - 4rl(k^2 - k) = (k^2 - k)(k^2 - k - 4rl) < 0$$

by the assumption of the claim. ◀

We do not formally prove the converse, but Figure 2 shows examples where it is tight. When (2) holds with equality, f still has the unique local maximum at $\frac{1}{2}$ but $f''(\frac{1}{2}) = 0$, and even when (2) holds with small slack, two local maxima start to appear. This phenomenon is one of the main reasons that we pose Conjecture 2.4. Though we were not able to formally prove for the general case, we believe that the violation of (1) not only allows us to sample random variables with desired second moments but also ensures that $f(\alpha)$ is a nice unimodal curve.

3 Approximability of symmetric CSPs with negation

Fix k and $S \subset [k] \cup \{0\}$. In this section, we consider SCSP(S) with negation and prove Theorem 1.2. Note that in this section we allow S to contain 0 or k. For example, famous Max-3SAT is 3-SCSP($\{1,2,3\}$). We still exclude the trivial case $S = [k] \cup \{0\}$.

The condition we are interested in is whether $\mathsf{conv}(P_S)$ contains $(\frac{1}{2}, \frac{1}{4})$. In SCSPs with negation, the sufficient condition of Austrin and Mossel on general CSPs to be approximation resistant becomes equivalent to it. See Appendix A to see the equivalence.

▶ **Theorem 3.1** ([2]). *Fix k and let $S \subset [k] \cup \{0\}$ be such that $\mathsf{conv}(P_S)$ contains $(\frac{1}{2}, \frac{1}{4})$. Then, assuming the Unique Games Conjecture, SCSP(S) with negation is approximation resistant.*

On the other hand, we now show that the algorithm of Austrin et al. [1], which is inspired by Hast [10], can be used to show that if S is an interval or even and $\mathsf{conv}(P_S)$ does not contain $(\frac{1}{2}, \frac{1}{4})$, SCSP($S$) is not approximation resistant.

Let $f : \{0,1\}^k \mapsto \{0,1\}$ be the function such that $f(x_1, \ldots, x_k) = 1$ if and only if $(x_1 + \cdots + x_k) \in S$. Define the inner product of two functions as $\langle f, g \rangle = \mathbb{E}_{x \in \{0,1\}^k}[f(x)g(x)]$, and for $T \subseteq [k]$, let $\chi_T(x_1, \ldots, x_k) = \prod_{i \in T}(-1)^{x_i}$. It is well known that $\{\chi_T\}_{T \subseteq [k]}$ form an orthonormal basis and every function has a unique *Fourier expansion* with respect to this basis,

$$f = \sum_{T \subseteq [k]} \hat{f}(T)\chi_T, \qquad \hat{f}(T) := \langle f, \chi_T \rangle.$$

Define

$$f^{=d}(x) = \sum_{|T|=d} \hat{f}(S)\chi_T(x).$$

The main theorem of Austrin et al. [1] is

▶ **Theorem 3.2** ([1]). *Suppose that there exists $\eta \in \mathbb{R}$ such that*

$$\frac{2\eta}{\sqrt{2\pi}} f^{=1}(x) + \frac{2}{\pi} f^{=2}(x) > 0 \tag{3}$$

for every $x \in f^{-1}(1)$. Then there is a randomized polynomial time algorithm that approximates SCSP(S) better than the random assignment in expectation.

We compute $f^{=1}$ and $f^{=2}$.

$$\hat{f}(\{1\}) = \langle f, \chi_{\{1\}}\rangle = \frac{1}{2^k}\sum_{s\in S}\left(\binom{k-1}{s} - \binom{k-1}{s-1}\right)$$

$$\hat{f}(\{1,2\}) = \langle f, \chi_{\{1,2\}}\rangle = \frac{1}{2^k}\sum_{s\in S}\left(\binom{k-2}{s} - 2\binom{k-2}{s-1} + \binom{k-2}{s-2}\right)$$

By symmetry, $\hat{f}_T =: \hat{f}_1$ is the same for all $|T| = 1$ and $\hat{f}_T =: \hat{f}_2$ is the same for all $|T| = 2$. If we let $s = x_1 + \cdots + x_k$,

$$f^{=1}(x) = \hat{f}_1 \sum_{i\in[k]} (-1)^{x_i} = k\hat{f}_1 \mathop{\mathbb{E}}_{i\in[k]}[-2x_i + 1] = k\hat{f}_1(-2\frac{s}{k} + 1)$$

$$f^{=2}(x) = \hat{f}_2 \sum_{i\neq j} (-1)^{x_i + x_j} = \binom{k}{2}\hat{f}_2 \mathop{\mathbb{E}}_{i\neq j}[(-2x_i+1)(-2x_j+1)] = \binom{k}{2}\hat{f}_2(4\frac{s(s-1)}{k(k-1)} - 4\frac{s}{k} + 1).$$

3.1 When S is an interval

Let $S = \{l, l+1, \ldots, r-1, r\}$. If $r \le \frac{k}{2}$, we have $(\frac{-2s}{k} + 1) \le 0$ for all $s \in S$, so choosing η either large enough or small enough ensures (3). Similarly, if $l \ge \frac{k}{2}$, (3) holds. Therefore, we assume that $l < \frac{k}{2}$ and $r > \frac{k}{2}$, and compute $\hat{f}_2$.

$$\begin{aligned}\hat{f}_2 &= \frac{1}{2^k}\sum_{s=l}^{r}\left(\binom{k-2}{s} - 2\binom{k-2}{s-1} + \binom{k-2}{s-2}\right)\\ &= \frac{1}{2^k}\left(\binom{k-2}{l-2} - \binom{k-2}{l-1} + \binom{k-2}{r} - \binom{k-2}{r-1}\right)\end{aligned}$$

Since $\binom{k-2}{l-1} > \binom{k-2}{l-2}$ for $0 < l < \frac{k}{2}$ and $\binom{k-2}{r-1} > \binom{k-2}{r}$ for $\frac{k}{2} < r < k$, $\hat{f}_2 < 0$ except when $l = 0$ and $r = k$ (i.e., $S = [k] \cup \{0\}$).

If $\mathsf{conv}(P_S)$ does not contain $(\frac{1}{2}, \frac{1}{4})$, there exist $\alpha, \beta \in \mathbb{R}$ such that for any $(a,b) \in \mathsf{conv}(P_S)$,

$$\alpha(a - \frac{1}{2}) + \beta(b - \frac{1}{4}) > 0.$$

If k is even, $s := \frac{k}{2} \in S$ and $P(s) = (\frac{1}{2}, \frac{s-1}{2(k-1)})$ where $\frac{s-1}{2(k-1)} < \frac{1}{4}$, which implies $\beta < 0$ since the above inequality should hold for all $s \in S$. When k is odd (let $k = 2s+1$), s and $s+1$ should be in S and

$$\frac{1}{2}\Big(P(s) + P(s+1)\Big) = (\frac{1}{2}, \frac{s^2}{k(k-1)}),$$

where $\frac{s^2}{k(k-1)} < \frac{1}{4}$. Therefore, we can conclude $\beta < 0$ in any case. For any $x \in f^{-1}(1)$ with $s = x_1 + \cdots + x_k$ and $P(s) = (a, b)$,

$$\begin{aligned}
&\frac{2\eta}{\sqrt{2\pi}} f^{=1}(x) + \frac{2}{\pi} f^{=2}(x) \\
= \;& \frac{2\eta}{\sqrt{2\pi}} k\hat{f}_1(-2a+1) + \frac{2}{\pi}\binom{k}{2}\hat{f}_2(4b-4a+1) \\
= \;& \frac{8}{\beta\pi}\binom{k}{2}\hat{f}_2\left(\frac{\frac{2\eta}{\sqrt{2\pi}}k\hat{f}_1}{\frac{8}{\beta\pi}\binom{k}{2}\hat{f}_2}(-2a+1) + \beta(b-a+\frac{1}{4})\right) \\
= \;& \frac{8}{\beta\pi}\binom{k}{2}\hat{f}_2\left((-\frac{\alpha+\beta}{2})(-2a+1) + \beta(b-a+\frac{1}{4})\right) \quad \text{by adjusting } \eta \text{ so that } \frac{\frac{2\eta}{\sqrt{2\pi}}k\hat{f}_1}{\frac{8}{\beta\pi}\binom{k}{2}\hat{f}_2} = -\frac{\alpha+\beta}{2} \\
= \;& \frac{8}{\beta\pi}\binom{k}{2}\hat{f}_2\left(\alpha(a-\frac{1}{2}) + \beta(b-\frac{1}{4})\right) \\
> \;& 0.
\end{aligned}$$

Therefore, (3) is satisfied if S is an interval and $\mathsf{conv}(S)$ does not contain $(\frac{1}{2}, \frac{1}{4})$.

3.2 When S is even

Given S, let $Q \in \{0,1\}^k$ be the predicate associated with S and $f : \{0,1\}^k \mapsto \{0,1\}$ be the indicator function of Q. We want to show that when S is even,

$$\frac{2\eta}{\sqrt{2\pi}} f^{=1}(x) + \frac{2}{\pi} f^{=2}(x) > 0$$

is satisfied for any $x \in f^{-1}(1)$. When S is even,

$$\hat{f}_1 = \frac{1}{2^{k+1}} \sum_{s \in S} \left(\binom{k-1}{s} - \binom{k-1}{s-1} + \binom{k-1}{k-s} - \binom{k-1}{k-s-1}\right) = 0.$$

We compute the sign of the contribution of each s to $\hat{f}_2$.

$$\begin{aligned}
& \binom{k-2}{s} - 2\binom{k-2}{s-1} + \binom{k-2}{s-2} \geq 0 \\
\Leftrightarrow \quad & (k-s)(k-s-1) - 2s(k-s) + s(s-1) \geq 0 \\
\Leftrightarrow \quad & 4s^2 - 4sk + k^2 - k \geq 0 \\
\Leftrightarrow \quad & s \leq \frac{k-\sqrt{k}}{2} \text{ or } s \geq \frac{k+\sqrt{k}}{2}
\end{aligned}$$

We also consider the line passing $P(s)$ and $P(k-s)$. If we denote $t = k - s$, Its slope is

$$\frac{\frac{t(t-1)-s(s-1)}{k(k-1)}}{\frac{t-s}{k}} = \frac{t^2 - s^2 - (t-s)}{(k-1)(t-s)} = 1,$$

and the value of this line at $\frac{1}{2}$ is at least $\frac{1}{4}$ when

$$\begin{aligned}
& \frac{s(s-1) + (k-s)(k-s-1)}{2k(k-1)} \geq \frac{1}{4} \\
\Leftrightarrow \quad & 2s(s-1) + 2(k-s)(k-s-1) \geq k(k-1) \\
\Leftrightarrow \quad & s \leq \frac{k-\sqrt{k}}{2} \text{ or } s \geq \frac{k+\sqrt{k}}{2}.
\end{aligned}$$

Intuitively, if we consider the line of slope 1 that passes $(\frac{1}{2}, \frac{1}{4})$, $P(s)$ is below this line if $s \in (\frac{k-\sqrt{k}}{2}, \frac{k+\sqrt{k}}{2})$. Let $S_1 = S \cap \{0, 1, \ldots, \lceil \frac{k}{2} \rceil\}$. If S_1 contains a value $s_1 \leq \frac{k-\sqrt{k}}{2}$ and a value $s_2 \geq \frac{k-\sqrt{k}}{2}$ (including the case $s_1 = s_2 = \frac{k-\sqrt{k}}{2}$ is an integer in S_1), the line passing $P(s_1)$ and $P(k - s_1)$ passes a point $(\frac{1}{2}, t_1)$ for some $t_1 \geq \frac{1}{4}$ and the line passing $P(s_2)$ and $P(k - s_2)$ passes a point $(\frac{1}{2}, t_2)$ for some $t_2 \leq \frac{1}{4}$. Therefore, $\mathsf{conv}(P_S)$ contains a point $(\frac{1}{2}, \frac{1}{4})$ and S becomes balanced pairwise independent. We consider the remaining two cases.

1. $s < \frac{k-\sqrt{k}}{2}$ for all $s \in S_1$: $\hat{f}_2 > 0$ and for all $s \in S$, $-(\frac{s}{k} - \frac{1}{2}) + (\frac{s(s-1)}{k(k-1)} - \frac{1}{4}) > 0$. Therefore, for any $x \in f^{-1}$ with $s = x_1 + \cdots + x_k$,

$$\begin{aligned} & \frac{2\eta}{\sqrt{2\pi}} f^{=1}(x) + \frac{2}{\pi} f^{=2}(x) \\ = \ & \frac{2}{\pi} f^{=2}(x) \\ = \ & \frac{2}{\pi} \binom{k}{2} \hat{f}_2 (4\frac{s(s-1)}{k(k-1)} - 4\frac{s}{k} + 1) \\ > \ & 0. \end{aligned}$$

2. $s > \frac{k-\sqrt{k}}{2}$ for all $s \in S_1$: $\hat{f}_2 < 0$ and for all $s \in S$, $-(\frac{s}{k} - \frac{1}{2}) + (\frac{s(s-1)}{k(k-1)} - \frac{1}{4}) < 0$. Similarly as above, for any $x \in f^{-1}$ with $s = x_1 + \cdots + x_k$, (3) is satisfied.

References

1 Per Austrin, Siavosh Benabbas, and Avner Magen. On quadratic threshold CSPs. *Discrete Mathematics & Theoretical Computer Science*, 14(2):205–228, 2012.

2 Per Austrin and Johan Håstad. On the usefulness of predicates. *ACM Transactions on Computation Theory*, 5(1):1:1–1:24, May 2013.

3 Per Austrin and Subhash Khot. A characterization of approximation resistance for even k-partite CSPs. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 187–196, 2013.

4 Per Austrin and Elchanan Mossel. Approximation resistant predicates from pairwise independence. *Computational Complexity*, 18(2):249–271, 2009.

5 Boaz Barak, Siu On Chan, and Pravesh Kothari. Sum of squares lower bounds from pairwise independence. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, STOC '15, 2015. To appear.

6 Siavosh Benabbas, Konstantinos Georgiou, Avner Magen, and Madhur Tulsiani. SDP gaps from pairwise independence. *Theory of Computing*, 8(1):269–289, 2012.

7 Siu On Chan. Approximation resistance from pairwise independent subgroups. In *Proceedings of the 45th annual ACM Symposium on Theory of Computing*, STOC '13, pages 447–456, 2013.

8 Mahdi Cheraghchi, Johan Håstad, Marcus Isaksson, and Ola Svensson. Approximating linear threshold predicates. *ACM Trans. Comput. Theory*, 4(1):2:1–2:31, 2012.

9 Wing-Sum Cheung. Generalizations of Hölder's inequality. *International Journal of Mathematics and Mathematical Sciences*, 26(1):7–10, 2001.

10 G Hast. *Beating a random assignment. KTH, Stockholm.* PhD thesis, Ph. D Thesis, 2005.

11 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, July 2001.

12 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th annual ACM Symposium on Theory of Computing*, STOC '02, pages 767–775, 2002.

13 Subhash Khot, Madhur Tulsiani, and Pratik Worah. A characterization of strong approximation resistance. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 634–643, 2014.
14 Thomas Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th annual ACM Symposium on Theory of Computing*, STOC '78, pages 216–226, 1978.
15 Madhur Tulsiani. CSP gaps and reductions in the Lasserre hierarchy. In *Proceedings of the 41st annual ACM Symposium on Theory of Computing*, STOC '09, pages 303–312, 2009.
16 Uri Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '98, pages 201–210, 1998.

A Austrin-Håstad Condition for Symmetric CSPs

This section explains how the condition of Austrin-Håstad [2] is simplified for SCSPs. They studied general CSPs where a predicate Q is a subset of $\{0,1\}^k$. Note that given $S \subseteq [k] \cup \{0\}$, SCSP($S$) is equivalent to CSP($Q$) where

$$Q = \{(x_1, \ldots, x_k) \in \{0,1\}^k : (x_1 + \cdots + x_k) \in S\} \tag{4}$$

Given Q, their general definition of *pairwise independence* and *positive correlation* is given below.

▶ **Definition A.1.** Q is *balanced pairwise independent* if there is a distribution μ supported on Q such that $\Pr_\mu[x_i = 1] = \frac{1}{2}$ for every $i \in [k]$ and $\Pr_\mu[x_i = x_j = 1] = \frac{1}{4}$ for every $1 \le i < j \le k$.

▶ **Definition A.2.** Q is *positively correlated* if there is a distribution μ supported on Q and $p, \rho \in [0,1]$ with $\rho \ge p^2$ such that $\Pr_\mu[x_i = 1] = p$ for every $i \in [k]$ and $\Pr_\mu[x_i = x_j = 1] = \rho$ for every $1 \le i < j \le k$.

We formally prove that their definitions have simpler descriptions in $\mathbb{R}^2$ for symmetric CSPs. Recall that given $s \in [k] \cup \{0\}$,

$$P(s) = (\frac{s}{k}, \frac{s(s-1)}{k(k-1)}) \in \mathbb{R}^2 \qquad \text{and} \qquad P_S := \{P(s) : s \in S\}\ .$$

▶ **Lemma A.3.** *Let $S \subseteq [k] \cup \{0\}$ and Q be obtained by (4). Q is pairwise independent if and only if $\mathsf{conv}(P_S)$ contains $(\frac{1}{2}, \frac{1}{4})$, and Q is positively correlated if and only if $\mathsf{conv}(P_S)$ intersects the curve $y = x^2$.*

Proof. We first prove the second claim of the lemma. Let Q be positively correlated with parameters p, ρ ($\rho \ge p^2$) and the distribution μ such that $\Pr_\mu[x_i = 1] = p$ for all i, $\Pr_\mu[x_i = x_j = 1] = \rho$ and for all $i < j$. Let ν be the distribution of $x_1 + \cdots + x_k$ where $(x_1, \ldots, x_k)$ are sampled from μ.

$$(p, \rho) = (\mathop{\mathbb{E}}_i[x_i], \mathop{\mathbb{E}}_{i<j}[x_i x_j]) = (\mathop{\mathbb{E}}_{s\sim\nu}[\frac{s}{k}], \mathop{\mathbb{E}}_{s\sim\nu}[\frac{s(s-1)}{k(k-1)}]) = \mathop{\mathbb{E}}_{s\sim\nu}[P(s)],$$

proving that positive correlation of Q implies $(p, \rho) \in \mathsf{conv}(P_S)$. Since $P(s)$ is strictly below the curve $y = x^2$ for any $s \in [k-1]$ and (p, ρ) is on or above this curve, $\mathsf{conv}(P_S)$ must intersect $y = x^2$.

Suppose that $\mathsf{conv}(P_S)$ intersects the curve $y = x^2$. There exists a distribution ν on S such that $\mathbb{E}_{s\sim\nu}[P(s)] = (p, p^2)$. Let μ_s be the distribution on $\{0,1\}^k$ that uniformly samples a string with exactly s 1's. Let μ be the distribution where s is sampled from ν and $(x_1, \dots, x_k)$ is sampled from μ_s. By definition, $\Pr_\mu[x_i = 1]$ and $\Pr_\mu[x_i = x_j = 1]$ do not depend on choice of indices,

$$\Pr_\mu[x_1 = 1] = \mathop{\mathbb{E}}_\mu[x_1] = \mathop{\mathbb{E}}_{s\sim\nu}\mathop{\mathbb{E}}_{x\sim\mu_s}[x_1] = \mathop{\mathbb{E}}_{s\sim\nu}[\frac{s}{k}] = p$$
$$\Pr_\mu[x_1 = x_2 = 1] = \mathop{\mathbb{E}}_\mu[x_1 x_2] = \mathop{\mathbb{E}}_{s\sim\nu}\mathop{\mathbb{E}}_{x\sim\mu_s}[x_1 x_2] = \mathop{\mathbb{E}}_{s\sim\nu}[\frac{s(s-1)}{k(k-1)}] = p^2,$$

implying that $(p, p^2) \in \mathsf{conv}(P_S)$.

The proof of the first claim is similar except that the curve $y = x^2$ is replaced by $(\frac{1}{2}, \frac{1}{4})$. ◀

▶ **Lemma A.4.** $\mathsf{conv}(P_S)$ *intersects the curve $x = y^2$ if and only if*

$$\frac{(s_{max} + s_{min} - 1)^2}{k-1} \geq \frac{4 s_{max} s_{min}}{k}\ .$$

Proof. Let $l = s_{min}$ and $r = s_{max}$. The line passing $P(l)$ and $P(r)$ has a slope $\frac{\frac{r(r-1)-l(l-1)}{k(k-1)}}{\frac{r-l}{k}} = \frac{r+l-1}{k-1}$ and a y-intercept b such that

$$\frac{l(l-1)}{k(k-1)} = \frac{r+l-1}{k-1} \cdot \frac{l}{k} + b \Leftrightarrow b = \frac{l(l-1) - l(r+l-1)}{k(k-1)} = \frac{-lr}{k(k-1)}.$$

This line intersects $y = x^2$ if and only if

$$x^2 = \frac{r+l-1}{k-1}x - \frac{lr}{k(k-1)}$$

has a real root, which is equivalent to

$$(\frac{r+l-1}{k-1})^2 - \frac{4lr}{k(k-1)} \geq 0 \Leftrightarrow \frac{(r+l-1)^2}{k-1} \geq \frac{4lr}{k}.$$ ◀

B Technical Proof

▶ **Lemma B.1.** *Let $Y_1, \dots, Y_l$ be sampled from a multivariate normal distribution where each Y_i has mean 0 and variance at most σ^2. Let $Y'_1, \dots, Y'_l$ be such that*

$$Y'_i = \begin{cases} Y_i & \text{if } |Y_i| \leq D \\ D & \text{if } Y_i > D \\ -D & \text{if } Y_i < -D \end{cases}$$

Then, for large enough D,

$$|\mathbb{E}[\prod_{i=1}^{l} Y_i] - \mathbb{E}[\prod_{i=1}^{l} Y'_i]| \leq 2^l \cdot \sigma^l \cdot l! \cdot e^{-D/l}.$$

Proof. For each $i \in [l]$, let $Y''_i = Y'_i - Y_i$. Take D large enough so that

$$\mathbb{E}[|Y''_i|^l] = 2\int_{y=D}^{\infty} (y-D)^l \phi(y) \leq 2\int_{y=D}^{\infty} y^l \phi(y) \leq e^{-D}.$$

Also each Y_i, a normal random variable with mean 0 and variance σ, satisfies $\mathbb{E}[|Y_i|^l] \le \sigma^l \cdot l!$. We have

$$
\begin{aligned}
|\,\mathbb{E}[\prod_{i=1}^{l} Y_i] - \mathbb{E}[\prod_{i=1}^{l} Y_i']| &= \left| \sum_{T\subseteq[l], T\neq[l]} \mathbb{E}[\prod_{i\in T} Y_i \prod_{i\notin T} Y_i''] \right| \\
&\le \sum_{T\subseteq[l], T\neq[l]} \prod_{i\in T} (\mathbb{E}[|Y_i|^l])^{1/l} \prod_{i\notin T} (\mathbb{E}[|Y_i''|^l])^{1/l} \\
&\quad \text{(by Generalized Hölder's inequality [9])} \\
&\le 2^l \cdot \sigma^l \cdot l! \cdot e^{-D/l}\ . \qquad \blacktriangleleft
\end{aligned}
$$

Sequential Importance Sampling Algorithms for Estimating the All-terminal Reliability Polynomial of Sparse Graphs

David G. Harris*[1] and Francis Sullivan[2]

1 Department of Applied Mathematics, University of Maryland, College Park, MD 20742, USA
davidgharris29@hotmail.com

2 IDA/Center for Computing Sciences, Bowie, MD 20715, USA
fran@super.org

Abstract

The all-terminal reliability polynomial of a graph counts its connected subgraphs of various sizes. Algorithms based on sequential importance sampling (SIS) have been proposed to estimate a graph's reliability polynomial. We show upper bounds on the relative error of three sequential importance sampling algorithms. We use these to create a hybrid algorithm, which selects the best SIS algorithm for a particular graph G and particular coefficient of the polynomial.

This hybrid algorithm is particularly effective when G has low degree. For graphs of average degree ≤ 11, it is the fastest known algorithm; for graphs of average degree ≤ 45 it is the fastest known polynomial-space algorithm. For example, when a graph has average degree 3, this algorithm estimates to error ϵ in time $O(1.26^n \epsilon^{-2})$.

Although the algorithm may take exponential time, in practice it can have good performance even on medium-scale graphs. We provide experimental results that show quite practical performance on graphs with hundreds of vertices and thousands of edges. By contrast, alternative algorithms are either not rigorous or are completely impractical for such large graphs.

1998 ACM Subject Classification G.2.1 Combinatorics, G.2.2 Graph theory, G.3 Probability and statistics

Keywords and phrases All-terminal reliability, sequential importance sampling

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.323

1 Introduction

Let G be a connected undirected multi-graph with vertex set V and edge set E. We define $\operatorname{Rel}(p)$, the all-terminal reliability polynomial of G, to be the probability that the graph remains connected when edges are removed independently with probability p. This function is a polynomial which can be factored as

$$\operatorname{Rel}(p) = \sum_{i=0}^{m} N_i (1-p)^i p^{m-i}$$

where N_i is the number of connected subgraphs of G with i edges. This polynomial has various physical applications, for example determining the reliability of a computer network or power grid.

* Research supported in part by NSF Awards CNS-1010789 and CCF-1422569.

Exactly computing the reliability polynomial is known to be #P-complete [20]. The best-known algorithm at this time is due to [3]; it runs in 2^n time and 2^n space, or 3^n time and polynomial space. While this is a great theoretical achievement, and works well on small graphs ($n \approx 40$), it is completely impractical for larger graphs.

In this paper, we give an algorithm targeting sparse graphs (graphs with average degree $\alpha = O(1)$), running in polynomial space and exponential time, to estimate the reliability polynomial coefficients N_t to some relative error ϵ. When α is a small constant, the algorithm may be significantly faster than [3]. When $\alpha \leq 11$, this algorithm runs faster than the exponential-space variant of [3], plus our algorithm is polynomial-space; in fact, our algorithm is the fastest known method in this regime. When $\alpha \leq 45$, it runs faster than the polynomial-space variant of [3]. Furthermore, our algorithm can be easily parallelized — absolutely no communication or memory is required between different processing elements.

Furthermore, in practice this algorithm seems to scale much better than the theoretical guarantees would imply. We have conducted extensive experiments on relatively large graphs, up to hundreds of vertices. The algorithm can achieve a 10% error rate on all coefficients after a minutes' run-time. This put real-world networks within reach. By contrast, the existing algorithms such as [3] simply cannot be run on such large graphs — these algorithms will use more space or time than exists on any computer.

1.1 Background

A variety of approaches have been proposed to estimate the reliability polynomial, or fragments of it, for a graph. Some algorithms seek to compute $\mathrm{Rel}(p_0)$ for a fixed probability p_0, such as [10] or [16]. The problem of estimating $\mathrm{Rel}(p_0)$ is related to the problem of estimating the reliability polynomial coefficients N_i, but they are not equivalent especially in terms of evaluating the relative error. For example, Karger's algorithm [16] gives a fully-polynomial relative approximation scheme (fpras) for the problem of estimating $1 - \mathrm{Rel}(p_0)$.

Other algorithms, for example [19], [20] can estimate N_t in polynomial time, but only for a narrow range of value of t, such as $t = O(1)$ or $t = m - n - O(1)$. [5] gives an algorithm to transform a given dense graph G into a relatively sparse graph G', with only $O(n \log n)$ edges, which has approximately the same reliability. There is no known polynomial-time algorithm to estimate arbitrary coefficients.

Many heuristic algorithms have been studied to approximate the all-terminal reliability. For example, [4] discusses an algorithm based on Monte-Carlo-Markov-Chain sampling of connected subgraphs. For the special case of directed acyclic graphs, [18] describes an algorithm based on Monte-Carlo sampling for the closely related problem of estimating $s - t$ connectivity. Algorithms based on sequential importance sampling are described in [2],[14],[8]. While these approximations can be effective in practice, typically they do not have rigorous complexity bounds. This can be especially problematic for estimation algorithms: if the algorithm is run for too few iterations, then it may produce a wrong estimate despite outwardly appearing to run successfully.

1.2 The Tutte polynomial

The reliability polynomial is a special case of the Tutte polynomial, a very powerful graph invariant. While the reliability polynomial counts only the connected subgraphs of G, the Tutte polynomial also encodes information about the decomposition of G into its connected components.

In [3], Bjorklund et al. gave an algorithm running in 2^n time and space, or time 3^n and polynomial space, to exactly compute all the Tutte polynomial coefficients. The Tutte polynomial can be specialized to compute the reliability polynomial, and in fact the algorithm of [3] provides the fastest algorithm, in general, for computing the reliability polynomial. Under appropriate complexity-theoretic assumptions, the Tutte polynomial cannot be computed in time $\exp(o(n))$ [9], or even approximated in polynomial-time [12]. (It is possible that the reliability polynomial may be a tractable special-case, though).

For classes of sparse graphs, alternative algorithms may be still faster. For example, the algorithm of Bjorklund et al. itself can be specialized for graphs of maximum degree Δ; on such a graph, the algorithm has running time $\exp((2^{\Delta+1}-1)^{1/(\Delta+1)}n + o(1))$. Other algorithms have been proposed to regular graphs (such as [7]) or bounded-degree graphs (such as [11]), although these appear to be dominated by the algorithm of [3].

The algorithm of Bjorklund et al. represents a breakthrough from the theoretical point of view, in reality is impractical graphs of even moderate size. For example, experiments in [3] describe a computation time of four days to solve a graph with 22 vertices.

1.3 Our contribution

In this paper, we will propose a new algorithm to estimate the reliability polynomial coefficients N_t for graphs whose *average degree* α is smal. The running time of this algorithm will have the form $(\chi_\alpha + o_\alpha(1))^n/\epsilon^2$, where ϵ is the desired relative error and χ_α is a parameter depending on α. The space required by the algorithm is polynomial in all parameters. This algorithm has a number of advantages over the existing ones, both from a theoretical and a practical point of view.

First, the theory: for small α, the algorithm may be significantly faster than Bjorklund et al. or any other known algorithm. As we will see, this algorithm will be faster than any known algorithm for $\alpha \leq 11$. At this point, the algorithm of Bjorklund et al. becomes faster, although the latter requires exponential space. Our algorithm will be faster than any known *polynomial-space* algorithm for $\alpha \leq 45$.[1] Note that we are considering the most general class of sparse graphs, while other algorithms of [3], [7], [11] were specialized to more restrictive classes of sparse graphs (most prominently, bounded-degree).

Second, in practice this algorithm seems to scale much better than the theoretical guarantees would imply. We have conducted extensive experiments on medium-scale graphs, up to hundreds of vertices. The algorithm can achieve a 10% error rate on all coefficients after a minutes' run-time. This puts real-world networks within reach, whereas the alternative exponential-time algorithms would never finish their computations on such large graphs.

There is a synergy between the theoretical analysis and the practical performance. Certain key parameters must be tuned accurately for our algorithm to guarantee good performance. These parameters are difficult to set empirically, and if they are set incorrectly, the algorithm can appear to run perfectly well but nevertheless return inaccurate results. By setting these parameters in accordance with the worst-case analysis, we tend to achieve results that are accurate in practice. This is despite the fact that the worst-case analysis, for such large graphs, is not directly relevant to practical computations.

[1] This is likely a conservative estimate of the worst-case behavior of our algorithm; an accurate estimate would require solving some open problems in extremal graph theory.

2 General approach

The algorithm we propose is based on sequential-importance sampling (SIS). Three SIS approaches were proposed in [2],[14],[8]. Experimental evidence showed that these algorithms could give quite effective estimates, at least in certain parameter ranges.

The SIS algorithms can be divided into two types. The first, which we refer to as top-down, start by removing edges from the initial graph G and counting the connectivity of the resulting subgraph. The second, which we refer to as bottom-up, start from a spanning tree of G and add edges. On any particular graph, the top-down algorithms tend to have better relative variance when the t is large; the bottom-up algorithms tend to have better relative variance when t is small. (Recall N_t is the coefficient of interest.)

There is a natural strategy to combine the strengths of these algorithms: for any given graph G and any desired coefficient N_t, run all three algorithms in parallel and select the one which gives the best estimate. Unfortunately, there is not any generic method to determine which statistic has lowest variance, and in fact exponentially many samples may be required to calculate a sample variance accurately. Thus, exponential time might be incurred in simply accumulating enough statistical information in order to select the appropriate algorithm.

In this paper, we compute upper bounds on the precision of the three algorithms. These upper bounds play two roles. First, they allow us to find upper bounds on the running time required to achieve any desired level ϵ of relative accuracy. The main probabilistic tool we use in this paper is to estimate the relative variance of the estimate

$$\operatorname{rv}(F) = \mathbf{E}[F^2]/\mathbf{E}[F]^2.$$

As a consequence of the Chebyshev inequality, one achieves the desired precision with high probability after extracting $\Theta(\operatorname{rv}(F)/\epsilon^2)$ independent samples. Hence, if we can bound the relative variance of *any* of the SIS algorithms as $(\chi + o(1))^n$, this implies that the running time of the resulting hybrid algorithm will also be $(\chi + o(1))^n/\epsilon^2$. We will see that for a fixed value of the average degree α, we can achieve good bounds on χ_α.

The second, less-obvious function of these upper bounds, is that they allow us to create a hybrid algorithm, which computes the upper bounds and publishes the statistic with smallest upper-bound on precision. That is, for any fixed graph G, these bounds give us a sensible strategy on which coefficients to estimate with the top-down algorithm and which coefficients to estimate with a bottom-up algorithm.

In practice, these SIS algorithms can be far more accurate than the worst-case analysis would predict. We will examine some examples of this in Section 5. Thus, we can evaluate this hybrid algorithm empirically, and we see that it can give useful estimates for graphs of moderate size (hundreds of vertices), while the exact exponential-time algorithms are impractical for tens of vertices.

Once we estimate N_t, we can automatically estimate $\operatorname{Rel}(p)$ for any fixed value of p. For, if we estimate $\widehat{N_t}$ up to relative error ϵ for all t, we may then estimate

$$\widehat{\operatorname{Rel}(p)} = \sum_i \widehat{N_i} p^i (1-p)^{m-i} \tag{1}$$

and we note that all the summands in (1) are positive.

Despite the advantages of our algorithm, we note that it is ultimately much more specialized than that of Bjorklund et al., for three reasons. First, we only estimate the coefficients instead of computing them exactly, and we do so probabilistically instead of deterministically. Second, this algorithm applies only to the reliability polynomial, which is a special case of the Tutte polynomial. Third, the complexity of our algorithm is exponential in the number of edges, and hence for dense graphs it is slower.

2.1 Notation

We are given an connected undirected graph G, which may have multiple edges or self-loops. We will use m, n to refer to the number of edges and vertices in the graph G. Our algorithm seeks to estimate N_t, and t is always the index of the coefficient we are interested in. The average degree α is given by $\alpha = 2m/n$. Note that as G is connected, we must have $\alpha \geq 2 - 2/n$.

In this paper, we will only be concerned with the case that α is a small constant. Hence, we will assume throughout that $\alpha \leq Cn$, where C is some large fixed constant. Together with the restriction that $\alpha \geq 1$, this implies that any asymptotics in m can be reduced to equivalent asymptotics in n. We will make this assumption $m = \Theta(n)$ throughout this paper.

We let $K = m - n + 1$ and let $k = t - n + 1$. Note that spanning trees of G have $n - 1$ edges, and so k counts the distance of t from its maximal value K. This is particularly useful for algorithms which add edges to spanning trees. We let $\kappa(G)$ denote the number of (labelled) spanning trees of G. Note that using the Kirchoff formula, $\kappa(G)$ can be computed in polynomial time.

We will frequently use the entropy function in estimating binomial coefficients. To simplify the notation in these estimates we define $l(x) = x \ln x$. By Stirling's formula, for any $c \in [0, 1]$ we have

$$\exp(n(-l(1-c) - l(c)) - o(1)) \leq \binom{n}{cn} \leq \exp(n(-l(1-c) - l(c)) + o(1)) \tag{2}$$

We will always use the notation $\beta = t/n$. Note that $\beta \in [1 - 1/n, \alpha/2]$.

2.2 The Kruskal-Katona Theorem

We will frequently need to lower-bound N_t in terms of $\kappa(G)$. A key technical tool to do so comes from the Krusal-Katona Theorem [17]. This is a basic combinatorial principle which gives bounds on the number of objects in a family of sets which is upward-closed. Graph connectivity has this property, as if a graph H is connected and $H' \supseteq H$ then H' must be connected as well. We will use a simplified version of this pricniple, which is slightly less accurate than the full Kruskal-Katona bound but it adequate for our purposes.[2]

▶ **Theorem 1** ([17]). *Let m' be the unique integer such that*

$$\binom{m'}{K} \leq \kappa(G) < \binom{m'+1}{K}.$$

Then for all $t \geq n - 1$ we have

$$N_t \geq \binom{m'}{m-t}.$$

[2] We contrast our use of the Kruskal-Katona Theorem to that of [1]. [1] uses this bound, or variants of it, to estimate N_t itself. We use this bound to estimate the error committed by our SIS algorithms, but these algorithms do not themselves refer to the Kruskal-Katona bounds in any way.

We can explain the intuition behind this result. If we completely ignore graph connectivity, then there are exactly $\binom{m}{m-t}$ subgraphs with t edges. Thus, the Kruskal-Katona Theorem states that the number of connected subgraphs and the number of subgraphs have a similar behavior as a function of t.

We will use throughout the notation

$$\gamma' = m'/n$$

Note that $K \leq m' \leq m$, so that $\gamma \in [\alpha/2 - 1 - 1/n, \alpha/2]$.

3 The basic algorithms

We begin by describing two very simple statistics to estimate N_t. These statistics are accurate for different ranges of the parameter t. We then discuss how to select the best statistic for a given value of t.

3.1 Top-down algorithm bounds

Let us first consider a top-down estimate, which starts with the original graph and removes edges until it reaches a disconnected graph. Two such algorithms are discussed in [2], [15], which use a variety of heuristics to select which edge to remove. These heuristics are somewhat difficult to analyze, although they are very useful in practice, so for this paper we will consider a simplified algorithm. The following algorithm, which we refer to as TOPDOWN, has strictly larger relative variance compared to [2], [15]:

1. Select uniformly at random a subgraph $H \subseteq G$ with t edges.
2. Check if H is connected.
3. If H is connected estimate $\widehat{N_t} = \binom{m}{t}$; otherwise estimate $\widehat{N_t} = 0$.

Note that this algorithm is not really a "top-down" algorithm any more, since we could produce H by adding edges as well as removing edges.

The statistic produced by TOPDOWN is clearly an unbiased estimator for N_t.

▶ **Proposition 2.** *Define*

$$f_T = l(\alpha/2) - l(\beta) - l(\gamma) + l(\beta + \gamma - \alpha/2)$$

Then TOPDOWN has relative variance $\exp(n(f_T + o(1)))$.

Proof. TOPDOWN is a Bernoulli random variable with probability $N_t/\binom{m}{t}$, so it has relative variance $\frac{\binom{m}{t}}{N_t}$. By Theorem 1, this is at most $\frac{\binom{m}{t}}{\binom{m'}{m-t}}$. Recalling that $t = \beta n, m = alphan/2, m' = \gamma n$, e apply Stirling's formula (2). (We note here that $m = \Theta(n)$, so all asymptotic terms can be reduced to $o(n)$). This gives the estimate:

$$\mathrm{rv}_T \leq \exp(n(f_T + o(1)))$$

◀

We note one key difference between this algorithm, based on estimating the reliability coefficients, and similar algorithms such as [21] which seek to compute the coefficients exactly. In this case, we are doing a Monte-Carlo estimation of the number of connected subgraphs, so our algorithm is more efficient when the subgraphs are numerous. Enumerative algorithms, by contrast, must explore each subgraph individually, and so they are more efficient when the subgraphs are few.

3.2 Bottom-up, no edge weighting

The bottom-up algorithm is based on starting with a random spanning tree and adding edges to it. This type of algorithm is discussed in [8], [14]. In Section 4, we will consider more sophisticated variants which use add edges with a non-uniform probability. However, as a warm-up exercise, we consider the following simple algorithm which we refer to as BOTTOMUP:

1. Choose a spanning tree T uniformly at random from G. (This can be done efficiently as described in e.g. [22])
2. Choose $k = t - n + 1$ edges uniformly at random from $G - T$, and add these edges to obtain a connected subgraph $H \subseteq G$.
3. Estimate $\widehat{N_t} = \frac{\kappa(G)\binom{K}{k}}{\kappa(H)}$

Recall that $\kappa(G)$ can be computed in polynomial time, so this is a polynomial-time algorithm.

▶ **Proposition 3.** *Define*

$$f_B = -l(\beta - 1) - l(1 - \alpha/2 + \gamma) + l(\beta - \alpha/2 + \gamma)$$

Then algorithm BOTTOMUP is an unbiased estimator with relative variance at most $\exp(n(f_B + o(1)))$.

Proof. Let H be a connected subgraph of G with t edges. Then BOTTOMUP selects H with probability $p_H = \frac{\kappa(H)}{\kappa(G)\binom{K}{k}}$. Integrating over all such H, we see that the expected value of $\widehat{N_t}$ is given by

$$\mathbf{E}[\widehat{N_t}] = \sum_H p_H \frac{\kappa(G)\binom{K}{k}}{\kappa(H)} = \sum_H 1 = N_t$$

as desired.

Next, we can estimate

$$\mathbf{E}[\widehat{N_t}^2] = \sum_H p_H \Big(\frac{\kappa(G)\binom{K}{k}}{\kappa(H)}\Big)^2 = N_t \kappa(G)\binom{K}{k} \mathbf{E}_H[1/\kappa(H)] \leq N_t \kappa(G)\binom{K}{k}$$

Hence, the relative variance is given by

$$\text{rv} \leq \frac{N_t \kappa(G)\binom{K}{k}}{N_t^2} \leq \frac{m\binom{m'}{K}\binom{K}{k}}{\binom{m'}{m-t}}$$

Applying Stirling's formula (2), and recalling $m = \Theta(n)$, shows that this is at most $\exp(n(f_B + o(1)))$, as desired. ◀

3.3 Hybrid algorithm

Now define the hybrid algorithm which, for any graph G and any desired coefficient t, computes f_T and f_B. If $f_T < f_B$, then this algorithm outputs TOPDOWN; otherwise it outputs BOTTOMUP. By Propositions 2, 3, this hybrid algorithm satisfies

$$\text{rv} \leq \exp(n \min(f_T, f_B) + o(n))$$

Let us examine how to bound the quantity $\min(f_T, f_B)$ as a function of β, γ. For any fixed γ, f_T is an decreasing function of β and f_B is a increasing function of β, reaching extreme values $f_T = 0$ at $\beta = \alpha/2$ and $f_B = 0$ at $\beta = 1$. Hence the maximum value of $\min(f_T, f_B)$ occurs at the point where $f_T = f_B$.

We seek to maximize f_B subject to $f_T = f_B$. In general, there is no closed form solution. However, for any given value of α we can numerically optimize this as follows. Given any value of γ, there is a unique value β such that $f_T = f_B$, which can be found to arbitrary accuracy via bisection. This essentially reduces the search to a single variable γ. One can verify that the resulting function f_T is unimodal in γ, and hence the maximum value of γ can be found again by a golden-section search strategy. This allows us to find a value β^*, γ^* maximizing $\min(f_T, f_B)$ to an arbitrary accuracy. The resulting value χ_α is the worst-case for the running time, as a function of α. That is, when α is constant and $n \to \infty$, we have $\mathrm{rv} = \exp(n\chi_\alpha + o(n))$ for graphs with average degree $\leq \alpha$.

The following table shows bounds on χ_α for various values of α. Note that unlike algorithms which restrict to regular or bounded-degree graphs, there is no restriction on the integrality of α.

α	χ_α	α	χ_α
3	1.32	15	2.34
4	1.51	20	2.55
6	1.76	25	2.72
8	1.93	30	2.87
10	2.08	35	2.99

For $\alpha \leq 8$, this algorithm is strictly faster than [3]; for $\alpha \leq 35$ this algorithm is faster than the polynomial-space variant of Bjorklund et al. We will improve this still further in the next section.

4 Bottom-up, edge weighting

We consider a variant of the bottom-up algorithm in which a weighting function is used to select the edges to add to the spanning tree, as described in [14]. Again without concerning ourselves with polynomial efficiency, we summarize this algorithm as

1. Select a spanning tree T uniformly at random
2. For $i = 1, \dots, k$, repeat the following:
   3. Randomly select an edge e_i to add to T. We use the probability distribution P_i (which is conditioned on $e_1, \dots, e_{i-1}, T$) to select the edge e_i, and this probability distribution P_i is given by
   $$P_i(e') \propto \kappa(T \cup e_1 \dots \cup e_{i-1} \cup e')^{-\rho}$$
4. Estimate
$$\widehat{N_t} = \frac{\kappa(G) P_1(e_1) P_2(e_2) \dots P_k(e_k)}{k! \kappa(H)}$$

This is a generalization of the algorithm of [14], in that we allow a weighting factor $\rho \in [0, 1]$. (In the algorithm of [14], $\rho = 1$). Note that for $\rho = 0$, there is a uniform distribution on the new edge e_i, and so this reduces to the unweighted bottom-up algorithm of Section 3.2.

Experimental evidence in [14] suggested that this algorithm had better variance for estimating N_t for small values of t. In this section, we examine this algorithm rigorously and show an upper bound which is better than that of Section 3.2.

For any connected subgraph $H \subseteq G$, we define the notations for this section:

$$\lambda_H(e) = \kappa(H \cup e)/\kappa(H)$$
$$S_H = \sum_{e \in G-H} \lambda_H(e)^{-\rho}$$

and we define

$$H_i = T \cup e_1 \cup \dots \cup e_{i-1}$$

where T is the spanning tree selected at step (1) of this algorithm. Thus, we have $P_i(e) = \lambda_{H_i}(e)/S_{H_i}$ for $i = 1, \dots, k$.

We first show that this is an unbiased estimator. Any $T, e_1, \dots, e_k$ is selected with probability $p = \frac{1}{\kappa(G)} P_1(e_1) P_2(e_2) \dots P_k(e_k)$. Integrating over $T, e_1, \dots, e_k$, we compute the expected value:

$$\begin{aligned}\mathbf{E}[\widehat{N_t}] &= \sum_{T, e_1, \dots, e_k} p \times \frac{1}{\kappa(H) k! p} \\ &= \sum_H \sum_{\substack{T \subseteq H \\ e_1, \dots, e_k \in H - T}} \frac{1}{k! \kappa(H)} \\ &= \sum_H \frac{1}{k! \kappa(H)} \kappa(H) k! = N_t\end{aligned}$$

so the statistic is unbiased. Next, the mean square is given by

$$\begin{aligned}\mathbf{E}[\widehat{N_t}^2] &= \sum_H \frac{\kappa(G)}{k!^2 \kappa(H)^2} \sum_{\substack{T \subseteq H \\ e_1, \dots, e_k \in H - T}} S_{H_1} S_{H_2} \dots S_{H_k} \lambda_{H_1}(e_1)^\rho \dots \lambda_{H_k}(e_k)^\rho \\ &= \sum_H \frac{\kappa(G)}{k!^2 \kappa(H)^2} \sum_{\substack{T \subseteq H \\ e_1, \dots, e_k \in H - T}} S_{H_1} S_{H_2} \dots S_{H_k} \Big(\frac{\kappa(H_2)}{\kappa(H_1)}\Big)^\rho \Big(\frac{\kappa(H_3)}{\kappa(H_2)}\Big)^\rho \cdots \Big(\frac{\kappa(H)}{\kappa(H_k)}\Big)^\rho \\ &= \sum_H \frac{\kappa(G)}{k!^2 \kappa(H)^2} \sum_{\substack{T \subseteq H \\ e_1, \dots, e_k \in H - T}} S_{H_1} S_{H_2} \dots S_{H_k} \kappa(H)^\rho\end{aligned}$$

To interpret this quantity, consider the following random process. We select a connected subgraph H with t edges, uniformly at random among all such subgraphs. Next, we select a random spanning tree T of H and a random permutation of the remaining edges $e_1, \dots, e_k \in H - T$. We then output the random variable R given by

$$R = \frac{S_{H_1} \dots S_{H_k}}{k! \kappa(H)^{1-\rho}}.$$

We see now that $\mathbf{E}[\widehat{N_t}^2] \leq N_t \kappa(G) \mathbf{E}[R]$, and hence the relative variance of the weighted bottom-up estimate is given by

$$\text{rv} \leq \frac{N_t \kappa(G) \mathbf{E}[R]}{N_t^2} \leq \frac{\kappa(G) \mathbf{E}[R]}{\binom{m'}{m-t}}$$

Thus, it suffices to estimate $\mathbf{E}[R]$. We will in fact show an upper bound on R. We begin with a simple estimate. For any graph H, we have $\lambda_H(e) \geq 1$. Thus $S_{H_i} \leq K - i$. Noting that $\rho \leq 1$ and that $\kappa(H) \geq 1$, we have $R \leq \binom{K}{k}$. This simple estimate leads to the same bound as in Section 3.2.

Lemma 4 improves on this estimate as follows:

▶ **Lemma 4.** *Let* $\rho \in [0,1]$.
Define the function

$$A_\rho(y) = \int_0^y (1 - (\frac{x}{1+x})^\rho) dx.$$

Then, for any connected subgraph $H \subseteq G$ *with* βn *edges, we have* $R \leq \exp(n(f_{B2} + o(1)))$, *for*

$$f_{B2} = \max_{\phi \in [0, \alpha/2 - \beta]} -(1-\rho)\Big(l(\gamma) - l(\alpha/2 - 1) - l(1 - \alpha/2 + \gamma) - l(1+\phi) + l(\phi)\Big) - l(\beta - 1) + l\big(\alpha/2 - 1 - A_\rho(\phi)\big) - l\big(\alpha/2 - \beta - A_\rho(\phi)\big)$$

Proof. See Appendix B. ◀

This then implies that the relative variance of the bottom-up algorithm is bounded by $\mathrm{rv} = \exp((f_{B2} + o(1))n)$. Note that to compute f_{B2} itself requires a numerical maximization over $\phi \in [0, \alpha/2 - \beta]$.

As before, for any given α we seek β^*, γ^* so that the resulting upper bound $\min(f_T, f_{B2})$ is maximized. This expression is too complicated for us to solve in closed form, or even to prove that all relevant functions have the appropriate smoothness to allow a rigorous numerical analysis. However, for any fixed $\rho \in [0,1]$ we can approximately solve this numerically. Using off-the-shelf numerical libraries, we optimize f_{B2} subject to $f_T = f_{B2}$. We can furthermore set ρ to *minimize* the resulting f_{B2}.

For any average degree α, we select an optimal parameter ρ^*. The following table shows various values of α as well as the corresponding ρ^* and χ_α:

α	ρ^*	χ_α	α	ρ^*	χ_α
3	0.71	1.26	12	0.84	2.01
4	0.74	1.41	15	0.85	2.15
6	0.79	1.62	20	0.87	2.34
8	0.81	1.78	30	0.89	2.64
10	0.83	1.90	40	0.90	2.87
11	0.83	1.96	45	0.91	2.97

For $\alpha \leq 11$, this algorithm is strictly faster than [3]; for $\alpha \leq 45$ this algorithm is faster than the polynomial-space variant of [3].

5 Practical Performance

One key advantage of this algorithm is that it can be used on real-world graphs up to hundreds of vertices. In this case, the worst-case analysis would indicate exponentially low accuracy. However, in practice the accuracy may be much better than this.

For our first test case, we generated Erdős-Renyi graphs of average degree 10 and ran the algorithm as specified in Appendix A. (Qualitatively similar results are seen for other edge densities). We tabulate the estimated relative error as well as the running time of a single iteration. For the most part, implementing this algorithm requires only slight modifications to the codes of [14],[15]; see these for more details. Figure 1 lists the estimated relative error of this algorithm.

The relative variance is clearly growing exponentially with n, but the rate of growth (about 1.05^n) is much slower than the bound of 1.89^n as indicated in the worst-case analysis. Hence for graphs of moderate size $n \approx 200$ this algorithm remains quite practical.

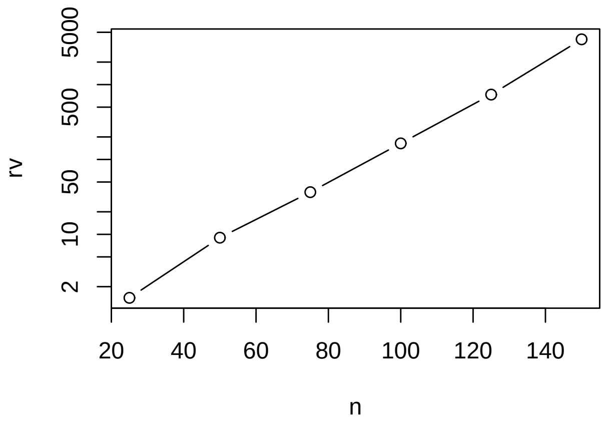


Figure 1 Relative error for Erdős-Renyi graphs. Note logarithmic vertical scale.

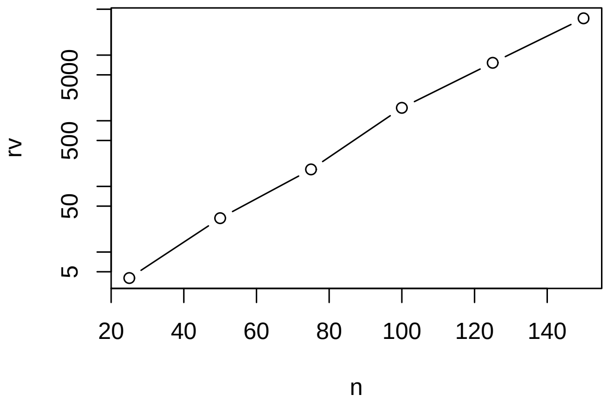


Figure 2 Relative error for Barabasi-Albert graphs. Note logarithmic vertical scale.

A similar result was seen for Barabasi-Albert random graphs, again of average degree 10, as shown in Figure 2.

The relative variance is significantly higher in this case, but the rate of increase remains slowly exponential, about 1.075^n.

The running times of these algorithms are relatively modest, and growing at a rate of about $n^{1.5}$, as indicated in Figure 3.

Recall that to achieve a relative error ϵ, we must repeat this algorithm for a number of samples $T = \text{rv} \times \epsilon^{-2}$. In order to achieve a relative error of say 10% on the Erdős-Renyi graph with $n = 150$, we would need to run for approximately 200000 iterations; this would entail a running time of about 300 seconds. (And furthermore this work could be completely parallelized). Hence this algorithm provides a quite practical method for estimating graph reliability on medium-scale graphs.

By way of comparison, [13] implemented an optimized version of an algorithm to exactly compute the Tutte polynomial. This program requires days of computations for graphs with only ~ 20 vertices and ~ 100 edges.

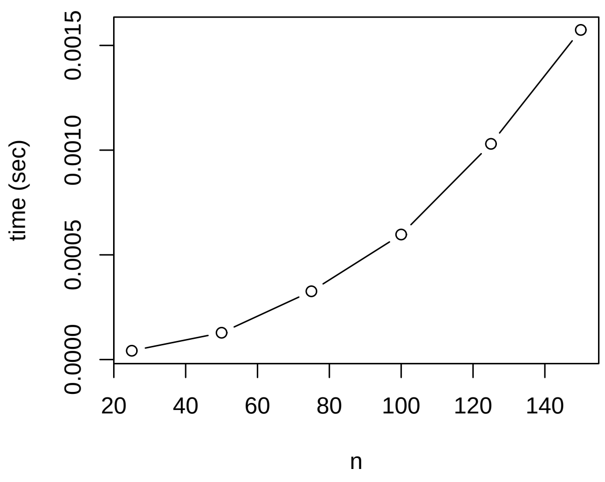


Figure 3 Running time for a single sample on an Erdős-Renyi graph.

5.1 Possible further improvements

The parameters (specifically, the choice of β) suggested by our worst-case analysis are not optimal for these sample graphs. By choosing parameters better, we could reduce the error by orders of magnitude. There seem to be three main reasons these bounds are not tight in practice. First, the top-down algorithm of [15] is much more accurate than the simple top-down algorithm we analyze in this paper.

Second, our estimate for the accuracy of the bottom-up algorithm is too conservative. The bottom-up error should be discounted by a factor of $\mathbf{E}[1/\kappa(H)]$, where H is the subgraph chosen by the bottom-up algorithm. In the worst case, the expected value of this term might be very large if some subgraphs H have many spanning trees and some have few. In practice, all the subgraphs H tend to have about the same number of spanning trees, and so $\mathbf{E}[1/\kappa(H)]$ is small.

Third, our method of setting the parameter ρ depends on estimating $\kappa(H \cup e_1 \cdots \cup e_i)$ where H is a subgraph of G and e_i are edges in $G - H$. It is currently an open problem in graph theory to determine tight bounds in this case. We are forced to use an upper bound for $\kappa(H \cup e_1 \cdots \cup e_i)$ which is much larger than necessary. This causes us to set ρ to an excessively large value.

6 Conclusion

We have shown exponential bounds on the relative variance of three SIS algorithms for estimating the graph reliability polynomial. These bounds are simple computable functions of G. By choosing the algorithm which minimizes the upper bound, we define a hybrid algorithm with worst-case relative variance $O(\chi_\alpha^n)$. Hence with $O(\chi_\alpha^n/\epsilon^2)$ work, one can, with high probability, estimate the graph reliability polynomial to relative error ϵ. Although this is exponential, we believe it is the fastest known algorithm for estimating the graph reliability polynomial when the average degree α is small.

Note that this bound on relative variance depended on bounding the number of spanning trees of certain sparse graphs. As this is an open problem in graph theory, the bounds we use are far from tight. It is likely that the true behavior of this algorithm is much better than the indicated values of χ_α.

In practice, these SIS algorithms tend to exhibit exponential relative variance on real-world graphs [14], [15]; however, the errors increase much slower than the worst-case analysis predicts. Hence, on many medium-scale graphs ($n \approx 200$) these algorithms can give a quite practical approach to estimate the graph reliability. In these cases exact, exponential-time algorithms such as [3] are absolutely infeasible.

Acknowledgments. Thanks to Isabel Beichl and Aravind Srinivasan for helpful comments and revisions.

References

1 Michael O. Ball and J. Scott Provan. Bounds on the reliablity polynomial for shellable independence systems. *SIAM Journal on Algebraic and Discrete Methods*, 3:166–181, 1982.
2 Isabel Beichl, Brian Cloteaux, and Francis Sullivan. An approximation algorithm for the coefficients of the reliability polynomial. *Congressus Numerantium*, 197:143–151, 2010.
3 Andreas Bjorklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Computing the Tutte polynomial in vertex-exponential time. *Foundations of Computer Science (FOCS)*, pages 677–686, 2008.
4 Adams L. Buchsbaum and Milena Mihail. Monte Carlo and Markov chain techniques for network reliability and sampling. *Networks*, 25:117–130, 1995.
5 Shiri Chechik, Yuval Emek, Boaz Patt-Shamir, and David Peleg. Sparse reliable graph backbones. *Automata, Languages, and Processing*, pages 261–272, 2010.
6 Andrew Chen. On graphs with large numbers of spanning trees. *PhD dissertation for Michigan State University Department of Computer Science*, 2005.
7 Fan RK Chung and Ronald L. Graham. On the cover polynomial of a digraph. *Journal of Combinatorial Theory, Series B*, 65:273–290, 1995.
8 Charles J. Colbourn, Bradley M. Debroni, and Wendy J. Myrold. Estimating the coefficients of the reliability polynomial. *Congress Numerantium*, 62:217–223, 1988.
9 Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlén. Exponential time complexity of the permanent and Tutte polynomial. *ACM Transactions on Algorithms*, 10-4, 2014.
10 George S. Fishman. A Monte Carlo sampling plan for estimating network reliability. *Operations Research*, 34:581–594, 1986.
11 Heidi Gebauer and Yoshio Okamoto. Fast exponential-time algorithms for the forest counting in graph classes. *Theory of Computing Australasian Symposium*, 65:63–69, 2007.
12 Leslie Ann Goldberg and Mark Jerrum. Inapproximability of the Tutte polynomial. *Information and Computation*, 206-7:908–929, 2008.
13 Gary Haggard, David J. Pearce, and Gordon Royle. Computing Tutte polynomials. *ACM Transactions on Mathematical Software*, 37-3 Article # 24, 2010.
14 David G. Harris, Francis Sullivan, and Isabel Beichl. Linear algebra and sequential importance sampling for network reliability. *Winter Simulation Conference*, 2011.
15 David G. Harris, Francis Sullivan, and Isabel Beichl. Fast sequential importance sampling to estimate the graph reliability polynomial. *Algorithmica*, 68-4:916–939, 2014.
16 David Karger. A randomized fully polynomial time approximation scheme for the all terminal network reliability problem. *SIAM Journal on Computing*, 29:11–17, 1996.
17 Joseph B. Kruskal. The number of simplices in a complex. *Mathematical Optimization Techniques*, 1963.
18 Marco Laumanns and Rico Zenklusen. High-confidence estimation of small s-t reliabilities in directed acylic networks. *Networks*, 57-4:367–388, 2011.
19 Wendy Myrvold. Counting k-component forests of a graph. *Networks*, 22:647–652, 1992.

20 J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12:777–788, 1983.

21 Kyoko Sekine, Hiroshi Imai, and Seiichiro Tani. Computing the Tutte polynomial of a graph of moderate size. *Lecture Notes in Computer Science*, 1004:224–233, 1995.

22 David Bruce Wilson. Generating random spanning trees more quickly than the cover time. *ACM Symposium on Theory of Computing (STOC)*, pages 296–303, 1996.

A Full algorithm

For completeness, we include a pseudo-code of the entire hybrid algorithm. Suppose we are given a graph G, and wish to estimate N_t up to relative error ϵ.

```
1. Compute $\alpha = m/n$ and $\beta = t/n$.
2. Using the Kirchoff formula, compute the number of spanning trees $\kappa(G)$. Find $m'$ such
   that $\binom{m'}{m-n} = \kappa(G)$ and let $\gamma = m'/n$.
3. Compute the bound on the top-down algorithm $f_T$.
4. Find the critical $h^*$ for $r(h, \alpha, \beta, \gamma)$. Use this to obtain the bottom-up bound $f_{B2}$.
5. If $f_T < f_{B2}$, draw the following statistic $F_1$ for $T = \exp(n(f_T + c))/\epsilon^2$ iterations:
   6. Select a subgraph $H \subseteq G$ uniformly at random among subgraphs with $t$ edges.
   7. Check if $H$ is connected
   8. If $H$ is connected set $F_1 = \binom{m}{t}$ else set $F_1 = 0$.
9. Else if $f_{B2} \leq f_T$, draw the following statistic $F_2$ for $T = \exp(n(f_{B2} + c))/\epsilon^2$ iterations:
   10. Select a spanning tree $H$ uniformly at random
   11. Successively select edges $e_1, \dots, e_k$ to add to $H$. At stage $i$, select edge $e_i$ with
       probability $P_i$ given by
       $P_i(e') \propto 1/\kappa(H \cup e_1 \dots \cup e_{i-1} \cup e')^{\rho^*}$
   12. Estimate
       $F_2 = \frac{\kappa(G) P_1(e_1) P_2(e_2) \dots P_k(e_k)}{k! \kappa(H)}$
13. Average the $T$ samples of the appropriate statistic and output this sample mean.
```

This algorithm estimates N_t within relative error ϵ with probability at least $3/4$ for n sufficiently large; furthermore, the worst-case running time of this algorithm is $(\chi_\alpha + o(1))^n/\epsilon^2$.

In practice, we use the algorithm of [2],[15] for the top-down estimation instead of the indicated steps (5) — (8). It is almost as fast as the simple Monte Carlo top-down estimation, and it is as least as accurate (in the worst-case) while being much more accurate in practice.

B Proof of Lemma 4

The heart of Lemma 4 is to show an upper bound on the quantity $S_H = \sum_{e \in G-H} \lambda_H(e)^{-\rho}$. We begin with two elementary propositions concerning the number of spanning trees in various subgraphs.

▶ **Proposition 5.** *For any graph H and edges $e_1, \dots, e_i \notin H$, we have*

$$\kappa(H \cup e_1 \dots \cup e_i) \leq \kappa(H) \binom{n-1+i}{n-1}$$

Proof. Any spanning tree T of $H \cup e_1 \dots \cup e_i$ may be formed as follows: choose a spanning tree T' of H, add the edges $e_1, \dots, e_i$, and extract a spanning tree T of $T' \cup e_1 \cup \dots \cup e_i$. ◀

We note that Proposition 5 can be improved exponentially. However, the formulas are quite complicated and the improvement is slight, so we elect to take this simple estimate. As discussed in [6], the state of research is very poor for estimating $\kappa(H \cup e_1 \cdots \cup e_i)$, even when H is itself a spanning tree. Preliminary results for small graphs seem to indicate that the true upper bound is much smaller than $\kappa(H)\binom{n-1+i}{n-1}$. If so, this would lead to much better bounds on the behavior of our algorithm.

▶ **Proposition 6.** *For any graph H and edges $e_1, \dots, e_i \notin H$, we have*

$$\kappa(H \cup e_1 \cdots \cup e_i) \leq \lambda_H(e_1) \cdots \lambda_H(e_i)$$

Proof. It suffices to show that $\lambda_{H \cup e'}(e) \leq \lambda_H(e)$ for any $H, e, e' \in G - H$. We recall the Kirchoff matrix-tree theorem used to count the number of spanning trees of a graph. Let A_G be the adjacency matrix of G, and let D be a diagonal matrix whose ith entry is the degree of vertex v_i. The Kirchoff formula states that $\kappa(G) = \det(D - A_G)_{11}$, the minor of $D - A_G$ obtained by removing the first row and column.

When we update H by adding edge e', we must update λ_H to the new $\lambda_{H \cup e'}$. For any edge $e = \langle i, j \rangle$ we define δ_e to be the column vector is $+1$ in coordinate i, is -1 in coordinate j, and is zero elsewhere. Observe that when edge e' is added to G, the matrix L changes by $\delta_{e'}\delta_{e'}^T$:

$$L_{H \cup e'} = L_H + \delta_{e'}\delta_{e'}^T.$$

Now let us examine how to update λ:

$$\begin{aligned}
\lambda_{H \cup e'}(e) &= \kappa(H \cup e \cup e')/\kappa(H \cup e') \\
&= \det(L_{H \cup e \cup e'})/\det(L_{H \cup e'}) \\
&= \det(L_H + \delta_e\delta_{e'}^T + \delta_e\delta_e^T)/\det(L_H + \delta_{e'}\delta_e^T) \\
&= \det\Big(I + (L_H + \delta_{e'}\delta_{e'}^T)^{-1}\delta_e\delta_e^T\Big) \\
&= 1 + \delta_e^T(L_H + \delta_{e'}\delta_{e'}^T)^{-1}\delta_e \\
&= 1 + \delta_e^T\big(L_H^{-1} - \frac{uu^T}{1 - \delta_e^T u}\big)\delta_e \qquad \text{where } u = L_H^{-1}\delta_{e'} \\
&= 1 + \delta_e^T L_H^{-1}\delta_e - \frac{(\delta_e^T u)^2}{1 - \delta_e^T u} \\
&= \lambda_H(e) - \frac{(\delta_e^T u)^2}{\lambda_H(e')} \leq \lambda_H(e)
\end{aligned}$$

◀

▶ **Proposition 7.** *Suppose $H \subseteq G$ is a subgraph with t edges. Suppose $s \in \mathbf{Z}_+$ satisfies $\kappa(H)\binom{n-1+s}{n-1} < \kappa(G)$. Then $s \leq m - t$.*

Proof. Suppose that $s > m - t$, then we would have:

$$\begin{aligned}
\frac{\kappa(G)}{\kappa(H)\binom{n-1+s}{n-1}} &< \frac{\kappa(G)}{\kappa(H)\binom{n-1+m-t}{n-1}} \\
&\leq \frac{\kappa(H)\binom{n-1+(m-t)}{n-1}}{\kappa(H)\binom{n-1+m-t}{n-1}} \qquad \text{by Proposition 5} \\
&\leq 1
\end{aligned}$$

contradicting our hypothesis on s. ◀

We can combine Propositions 5 and 6 to bound S_H for subgraphs $H \subseteq G$:

▶ **Lemma 8.** *Let $H \subseteq G$ be a connected subgraph with $t = n - 1 + k$ edges. Suppose $s \in \mathbf{Z}_+$ satisfies $\kappa(H)\binom{n-1+s}{n-1} < \kappa(G)$. Define*

$$A_\rho(y) = \int_0^y (1 - (\frac{x}{1+x})^\rho)dx$$

Then we have

$$\sum_{e \in G-H} \lambda_H(e)^{-\rho} \leq K + 1 - k - nA_\rho(s/n)$$

Proof. Observe that by Proposition 7, we have $s \leq m - t$.

Let $e_1, \dots, e_{m-t}$ enumerate the edges of $G - H$ sorted by decreasing order of λ_H, so that $\lambda_H(e_1) \geq \lambda_H(e_2) \geq \dots \geq \lambda_H(e_{m-t})$. To simplify the notation, write $\lambda_i = \lambda_H(e_i)$. By Propositions 5, 6, for any $i \leq m - t$, we have that

$$\begin{aligned}\kappa(G) &= \kappa(H \cup e_1 \dots \cup e_i \cup e_{i+1} \cup \dots \cup e_{m-t}) \\ &\leq \kappa(H \cup e_{i+1} \dots \cup e_{m-t})\binom{n-1+i}{n-1} \\ &\leq \kappa(H)\lambda_{i+1} \dots \lambda_{m-t}\binom{n-1+i}{n-1}\end{aligned}$$

Define $\lambda_{m-t+1} = 1$. Then $\vec{\lambda}$ satisfies the following system of constraints for $i = 1, \dots, m-t$:

$$\lambda_i \dots \lambda_{m-t} \geq \frac{\kappa(G)}{\kappa(H)\binom{n-2+i}{n-1}} \qquad \text{(Constraint } C_i)$$
$$\lambda_i \geq \lambda_{i+1}$$

Hence, it suffices to maximize on $S' = \sum_{i=1}^{m-t} \lambda_i^{-\rho}$ subject to these constraints. By compactness, such a maximum exists.

We first claim that in any such maximum, $\vec{\lambda}$ must satisfy $\lambda_i > \lambda_{i+1}$ for $i = 1, \dots, s$ strictly. Suppose that we have a block of equalities of the form $\lambda_i = \dots = \lambda_k$, where $i \leq s$ is minimal and k is maximal. We assume for simplicity that $i > 1$ (the case in which $i = 1$ is essentially identical.) Let $\eta = \lambda_i = \dots = \lambda_k$.

There are two ways in which k could be maximal. First, it might be that $k = m - t + 1$. In this case, we have $\lambda_i = \dots = \lambda_{m-t} = 1$. But then constraint C_i states that

$$1 \geq \frac{\kappa(G)}{\kappa(H)\binom{n-2+i}{n-1}}$$

which implies that

$$\kappa(G) \leq \frac{\kappa(H)}{\binom{n-2+i}{n-1}} \leq \frac{\kappa(H)}{\binom{n-2+s}{n-1}}$$

which contradicts the definition of s.

The other case is that we have $\lambda_k > \lambda_{k+1}$ for $k \leq m - t$. We claim that in this case, it must be that constraints $C_{i+1}, \dots, C_k$ are slack. For, suppose that constraint C_j is tight for some j in the range $i + 1, \dots, k$. Collecting all the terms other than $\lambda_{j-1}, \lambda_j, \lambda_{j+1}$ into a single constant c gives us the constraints:

$$\binom{n-2+(j-1)}{n-1}\eta^2 \geq c \qquad (C_{j-1})$$

$$\binom{n-2+j}{n-1}\eta = c \qquad (C_j)$$
$$\binom{n-2+j+1}{n-1} \geq c \qquad (C_{j+1})$$

(Note that constraint (C_j) is an equality.)

From constraint C_{j-1}, we can eliminate η to obtain that

$$c \geq \frac{\binom{n-2+j}{n-1}^2}{\binom{n-2+(j-1)}{n-1}}$$

and, substituting this into constraint C_{j+1} we obtain:

$$\frac{\binom{n-2+(j+1)}{n-1}\binom{n-2+(j-1)}{n-1}}{\binom{n-2+j}{n-1}^2} \geq 1$$

which reduces to

$$\frac{(j+n-1)(j-1)}{j(j+n-2)} \geq 1$$

which is a contradiction.

We have shown that if $\lambda_i = \dots = \lambda_k$ then constraints $C_{i+1}, \dots, C_k$ must be slack. Now divide λ_k by δ and multiply λ_i by δ for some $\delta > 1$. For δ sufficiently small, this does not change the sorted order of $\lambda_1, \dots, \lambda_{m-t}$. Furthermore, this only affects the constraints $C_{i+1}, \dots, C_k$, which are slack, and thus for δ sufficiently small all constraints remain satisfied. As $\lambda_i = \lambda_k$ this modification increases S', which is a contradiction.

We have thus shown that $\lambda_i > \lambda_{i+1}$ for $i = 1, \dots, s$.

We next claim that all of the constraints $C_1, \dots, C_s$ are tight. For, if C_i was slack for some $i \geq 2$, then we could multiply λ_{i-1} by δ and divide λ_i by δ for sufficiently small $\delta > 1$. As $\lambda_{i-2} > \lambda_{i-1} > \lambda_i > \lambda_{i+1}$, for δ sufficiently small this does not affect the sorted order of $\vec{\lambda}$, preserves all constraints, and increases S'. (For $i = 1$, simply divide λ_i by δ.)

We have now shown that when S' is maximized then all constraints $C_1, \dots, C_s$ must be tight. Dividing constraint C_i by C_{i+1} yields $\lambda_i = \frac{n+i-1}{i}$ for $i = 1, \dots, s-1$. We also must have $\lambda_i \geq 1$ for $i = s, \dots, m-t$, so we have

$$\begin{aligned}
S_H \leq S' &\leq (m-t-s+1) + \sum_{j=1}^{s-1} (\frac{n+j-1}{n})^{-\rho} \\
&\leq (K-k-s+1) + \int_{j=0}^{s} (\frac{j}{n+j})^{\rho} dj \\
&= (K-k-s+1) + n\int_{x=0}^{s/n} (\frac{x}{1+x})^{\rho} dx \qquad \text{setting } x = j/n \\
&= K+1-k-nA_\rho(s/n).
\end{aligned}$$

◀

▶ **Corollary 9.** *Suppose S satisfies $\kappa(H)\binom{n-1+s}{n-1} < \kappa(G)$. Let $H \subseteq G$ be a subgraph of G and T a spanning tree of H and $e_1, \dots, e_k$ enumerate the edges of $H - T$ (in any order). Then for $i = 1, \dots, k$ we have*

$$S_{T \cup e_1 \dots \cup e_{i-1}} \leq K + 1 - i - nA_\rho(s/n)$$

Proof. By Lemma 8 we have $S_{H_i} \leq K + 1 - i - nA_\rho(s_i/n)$ where s_i is maximal such that $\kappa(H_i)\binom{n-1+s_i}{n-1} < \kappa(G)$. Observe that $\kappa(H_i) \leq \kappa(H)$ and so $s_i \geq s$. Thus $S_{H_i} \leq K + 1 - i - nA_\rho(s/n)$ as claimed. ◀

We now pass to the limit, bounding the asymptotic growth of R.

▶ **Lemma 10.** *Let $\rho \in [0,1]$.*

Define the function

$$A_\rho(y) = \int_0^y (1 - (\frac{x}{1+x})^\rho)dx.$$

Then, for any connected subgraph $H \subseteq G$ with βn edges, we have $R \leq \exp(n(r + o(1)))$, for

$$r = \max_{\phi\in[0,\alpha/2-\beta]} -(1-\rho)\Big(l(\gamma) - l(\alpha/2-1) - l(1-\alpha/2+\gamma) - l(1+\phi) + l(\phi)\Big)$$
$$- l(\beta-1) + l\big(\alpha/2 - 1 - A_\rho(\phi)\big) - l\big(\alpha/2 - \beta - A_\rho(\phi)\big)$$

Proof. Let $h = \ln(\kappa(H))/n$. Let s be maximal such that $\kappa(H)\binom{n-1+s}{n-1} < \kappa(G)$, and let $\phi = s/n$. By Corollary 9, we have $S_{H_i} \leq K - i - nA_\rho(\phi)$ for $i = 1, \dots, k$. Also, by Proposition 7, we have $s \leq m - t$ and so $\phi \leq \alpha/2 - \beta$.

So

$$\ln R \leq \ln(K - nA(\phi)) + \cdots + \ln(K - k - nA(\phi)) - (\rho - 1)\ln\kappa(H) - \ln(k!)$$

We have that s is maximal such that $\binom{n-1+s}{n-1} < \kappa(G)/\kappa(H)$. Hence $\binom{n-1+s}{n-1}$ is within a factor of n of $\kappa(G)/\kappa(H)$, so that

$$\ln\kappa(G)/\kappa(H) - \ln n \leq \ln\binom{n-1+s}{n-1}$$

We apply Stirling's formula (2), and divide by n to get that

$$l(\phi+1) - l(\phi) \geq \frac{\ln\kappa(G)/\kappa(H)}{n} - o(1)$$

Now note that, by definition of m' and γ, we have that

$$\kappa(G) \geq \binom{m'}{K} = \exp(n(l(\gamma) - l(\alpha/2-1) - l(1-\alpha/2+\gamma)) - o(1))$$

Thus it follows that

$$l(\phi+1) - l(\phi) \geq l(\gamma) - l(\alpha/2-1) - l(1-\alpha/2+\gamma) - h - o(1)$$

Now we have:

$$\begin{aligned}
\ln R &\leq \ln(K + 1 - nA(\phi)) + \cdots + \ln(K + 1 - k - nA(\phi)) - (1-\rho)\ln\kappa(H) - \ln(k!)\\
&\leq \ln\binom{K + 1 - nA(\phi + o(1))}{k} - (1-\rho)h + o(n)\\
&\leq \ln\binom{K - nA(\phi) + o(n)}{k}\\
&\qquad - (1-\rho)\Big(l(\phi+1) - l(\phi) - l(\gamma) + l(\alpha/2-1) + l(1-\alpha/2-\gamma)\Big) + o(n)\\
&\leq r + o(n) \qquad \text{as } \phi \in [0, \alpha/2-\beta]
\end{aligned}$$

◀

Improved NP-inapproximability for 2-Variable Linear Equations[*]

Johan Håstad[1], Sangxia Huang[1], Rajsekar Manokaran[2], Ryan O'Donnell[3], and John Wright[3]

1 **School of Computer Science and Communication**
KTH Royal Institute of Technology
Stockholm, Sweden
`johanh@csc.kth.se`, `huang.sangxia@gmail.com`

2 **Computer Science and Engineering Department**
IIT Madras
Chennai, India
`rajsekar@gmail.com`

3 **Department of Computer Science**
Carnegie Mellon University
Pittsburgh, PA, USA
`{odonnell,jswright}@cs.cmu.edu`

Abstract

An instance of the 2-Lin(2) problem is a system of equations of the form "$x_i + x_j = b \pmod 2$". Given such a system in which it's possible to satisfy all but an ϵ fraction of the equations, we show it is NP-hard to satisfy all but a $C\epsilon$ fraction of the equations, for any $C < \frac{11}{8} = 1.375$ (and any $0 < \epsilon \leq \frac{1}{8}$). The previous best result, standing for over 15 years, had $\frac{5}{4}$ in place of $\frac{11}{8}$. Our result provides the best known NP-hardness even for the Unique-Games problem, and it also holds for the special case of Max-Cut. The precise factor $\frac{11}{8}$ is unlikely to be best possible; we also give a conjecture concerning analysis of Boolean functions which, if true, would yield a larger hardness factor of $\frac{3}{2}$.

Our proof is by a modified gadget reduction from a pairwise-independent predicate. We also show an inherent limitation to this type of gadget reduction. In particular, any such reduction can never establish a hardness factor C greater than 2.54. Previously, no such limitation on gadget reductions was known.

1998 ACM Subject Classification F.2.0 Analysis of Algorithms and Problem Complexity

Keywords and phrases approximability, unique games, linear equation, gadget, linear programming

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.341

1 Introduction

The well known constraint satisfaction problem (CSP) 2-Lin(q) is defined as follows: Given n variables $x_1, \dots, x_n$, as well as a system of equations (constraints) of the form "$x_i + x_j = b \pmod q$" for constants $b \in \mathbb{Z}_q$, the task is to assign values from $\mathbb{Z}_q$ to the variables so

[*] Abridged Version. Johan Håstad and Sangxia Huang are supported by ERC Advanced Investigator grant 226203 and Swedish Research Council. Ryan O'Donnell and John Wright are supported by NSF grants CCF-0747250 and CCF-1116594 and a grant from the MSR–CMU Center for Computational Thinking. John Wright is also supported by a Simons Fellowship in Theoretical Computer Science.

that there are as few unsatisfied constraints as possible. It is known [15, 17] that, from an approximability standpoint, this problem is equivalent to the notorious Unique-Games problem [14]. The special case of $q = 2$ is particularly interesting and can be equivalently stated as follows: Given a "supply graph" G and a "demand graph" H over the same set of vertices V, partition V into two parts so as to minimize the total number of cut supply edges and uncut demand edges. The further special case when the supply graph G is empty (i.e., every equation is of the form $x_i - x_j = 1 \pmod 2$) is equivalent to the Max-Cut problem.

Let's say that an algorithm guarantees an (ϵ, ϵ')*-approximation* if, given any instance in which the best solution falsifies at most an ϵ-fraction of the constraints, the algorithm finds a solution falsifying at most an ϵ'-fraction of the constraints. If an algorithm guarantees $(\epsilon, C\epsilon)$-approximation for every ϵ then we also say that it is a *factor-C approximation.*

We remark here that we are prioritizing the so-called "Min-Deletion" version of the 2-Lin(2) problem. We feel it is the more natural parameterization. For example, in the more traditional "Max-2-Lin(2)" formulation, the discrepancy between known algorithms and NP-hardness involves two quirky factors, 0.878 and 0.912. However, this disguises what we feel is the really interesting question – the same key open question that arises for the highly analogous Sparsest-Cut problem: Is there an efficient $(\epsilon, O(\epsilon))$-approximation, or even one that improves on the known $(\epsilon, O(\sqrt{\log n})\epsilon)$- and $(\epsilon, O(\sqrt{\epsilon}))$-approximations?

The relative importance of the "Min-Deletion" version is even more pronounced for the 2-Lin(q) problem. As we describe below, this version of the problem is essentially equivalent to the highly notorious Unique-Games problem. By way of contrast, the traditional maximization approximation factor measure for Unique-Games is not particularly interesting – it's known [10] that there is no constant-factor approximation for "Max-Unique-Games", but this appears to have no relevance for the Unique Games Conjecture.

1.1 History of the problem

No efficient $(\epsilon, O(\epsilon))$-approximation algorithm for 2-Lin(2) is known. The best known efficient approximation guarantee with no dependence on n dates back to the seminal work of Goemans and Williamson:

▶ **Theorem 1** ([11]). *There is a polynomial-time* $(\epsilon, \frac{2}{\pi}\sqrt{\epsilon} + o(\epsilon))$*-approximation algorithm for* 2-Lin(2)*.*

Allowing the approximation to depend on n, we have the following result building on [3]:

▶ **Theorem 2** ([1]). *There is a polynomial-time factor-*$O(\sqrt{\log n})$ *approximation for* 2-Lin(2)*.*

Generalizing Theorem 1 to 2-Lin(q), we have the following result of Charikar, Makarychev, and Makarychev:

▶ **Theorem 3** ([7]). *There is a polynomial time* $(\epsilon, C_q\sqrt{\epsilon})$*-approximation for* 2-Lin(q) *(and indeed for* Unique-Games*), for a certain* $C_q = \Theta(\sqrt{\log q})$*.*

The question of whether or not this theorem can be improved is known to be essentially equivalent to the influential Unique Games Conjecture of Khot [14]:

▶ **Theorem 4.** *The Unique Games Conjecture implies ([15, 17]) that for all sufficiently small* $\epsilon > 0$*,* $(\epsilon, \frac{2}{\pi}\sqrt{\epsilon} + o(\epsilon))$*-approximating* 2-Lin(2) *is* NP*-hard, and for general* q*,* $(\epsilon, \Omega(\sqrt{\log q})\sqrt{\epsilon})$*-approximating* 2-Lin($q$) *is* NP*-hard. On the other hand ([19]), if there exists* $q = q(\epsilon)$ *such that* $(\epsilon, \omega(\sqrt{\epsilon}))$*-approximating* 2-Lin($q$) *is* NP*-hard then the Unique Games Conjecture holds.*

The recent work of Arora, Barak, and Steurer has also emphasized the importance of subexponential-time algorithms in this context:

▶ **Theorem 5** ([2]). *For any $\beta \geq \frac{\log\log n}{\log n}$ there is a $2^{O(qn^\beta)}$-time algorithm for $(\epsilon, O(\beta^{-3/2})\sqrt{\epsilon})$-approximating* 2-Lin$(q)$*. For example, there is a constant $K < \infty$ and an $O(2^{n^{0.001}})$-time algorithm for $(\epsilon, K\sqrt{\epsilon})$-approximating* 2-Lin$(q)$ *for any $q = n^{o(1)}$.*

Finally, we remark that there is an *exact* algorithm for 2-Lin(2) by [21] that runs in time roughly 1.73^n.

The known NP-hardness results for 2-Lin(q) are rather far from the known algorithms. It follows easily from the PCP Theorem that for any q, there exists $C > 1$ such that factor-C approximation of 2-Lin(q) is NP-hard. However, getting an explicit value for C has been a difficult task. In 1995, Bellare, Goldreich, and Sudan [5] introduced the Long Code testing technique, which let them prove NP-hardness of approximating 2-Lin(2) to factor of roughly 1.02. Around 1997, Håstad [13] gave an optimal inapproximability result for the 3-Lin(2) problem; combining this with the "automated gadget" results of Trevisan et al. [20] allowed him to establish NP-hardness of factor-C approximation for any $C < \frac{5}{4}$. By including the "outer PCP" results of Moshkovitz and Raz [16] we may state the following more precise theorem:

▶ **Theorem 6** ([13]). *Fix any $C < \frac{5}{4}$. Then it is* NP*-hard to $(\epsilon, C\epsilon)$-approximate* 2-Lin(2) *(for any $0 < \epsilon \leq \epsilon_0 = \frac{1}{4}$). In fact ([16]), there is a reduction with quasilinear blowup; hence $(\epsilon, C\epsilon)$-approximation on size-N instances requires $2^{N^{1-o(1)}}$ time assuming the Exponential Time Hypothesis (ETH).*

Since 1997 there had been no improvement on this hardness factor of $\frac{5}{4}$, even for the (presumably much harder) 2-Lin(q) problem. We remark that Håstad [13] showed the same hardness result even for Max-Cut (albeit with a slightly smaller ϵ_0) and that O'Donnell and Wright [18] showed the same result for 2-Lin(q) (even with a slightly larger ϵ_0, namely $\epsilon_0 \to \frac{1}{2}$ as $q \to \infty$).

1.2 Our results and techniques

In this work we give the first known improvement to the factor-$\frac{5}{4}$ NP-hardness for 2-Lin(2) from [13]:

▶ **Theorem 7.** *Fix any $C < \frac{11}{8}$. Then it is* NP*-hard to $(\epsilon, C\epsilon)$-approximate* 2-Lin(2) *(for any $0 < \epsilon \leq \epsilon_0 = \frac{1}{8}$). Furthermore, the reduction takes* 3-Sat *instances of size n to* 2-Lin(2) *instances of size $n^{7+o(1)}$; hence $(\epsilon, C\epsilon)$-approximating* 2-Lin(2) *instances of size N requires at least $2^{N^{1/7-o(1)}}$ time assuming the ETH.*

▶ Remark. The power 7 in the size of the reduction comes from Chan's hardness reduction for the 7-ary Hadamard predicate [6].

We sketch the proof of this theorem in Section 3. The same theorem also holds in the special case of Max-Cut (albeit with some smaller, inexplicit value of ϵ_0). Proofs for both results can be found in the full version of the paper.

Our result is a gadget reduction from the "7-ary Hadamard predicate" CSP, for which Chan [6] recently established an optimal NP-inapproximability result. In a sense our Theorem 7 is a direct generalization of Håstad's Theorem 6, which involved an optimal gadget reduction from the "3-ary Hadamard predicate" CSP, namely 3-Lin(2). That said, we should emphasize some obstacles that prevented this result from being obtained 15 years ago.

First, we employ Chan's recent approximation-resistance result for the 7-ary Hadamard predicate. In fact, what's crucial is not its approximation-resistance, but rather the stronger fact that it's a *useless* predicate, as defined in the recent work [4]. That is, given a nearly-satisfiable instance of the CSP, it's NP-hard to assign values to the variables so that the distribution on the 7-tuples of the constraints is noticeably different from the uniform distribution.

Second, although in principle our reduction fits into the "automated gadget" framework of Trevisan et al. [20], in practice it's completely impossible to find the necessary gadget automatically, since it would involve solving a linear program with $\sim 2^{120}$ constraints. Instead we had to construct and analyze our gadget by hand. On the other hand, by also constructing an appropriate LP dual solution, we are able to show the following.

▶ **Theorem 8** (Informally stated). *Our gadget achieving factor-$\frac{11}{8}$ NP-hardness for 2-Lin(2) is* optimal *among gadget reductions from Chan's 7-ary Hadamard predicate hardness.*

In spite of Theorem 8, it seems extremely unlikely that factor-$\frac{11}{8}$ NP-hardness for 2-Lin(2) is the end of the line. Indeed, we view Theorem 7 as more of a "proof of concept" illustrating that the longstanding factor-$\frac{5}{4}$ barrier can be broken; we hope to see further improvements in the future. In particular, in Section 4 we present a candidate NP-hardness reduction from high-arity useless CSPs that we believe may yield NP-hardness of approximating 2-Lin(2) to any factor below $\frac{3}{2}$. The analysis of this reduction eventually depends on a certain conjecture regarding analysis of Boolean functions that we were unable to resolve; thus we leave it as an open problem.

Finally, in Section 5 we show an inherent limitation of the method of gadget reductions from pairwise-independent predicates. We prove that such reductions can never establish an NP-hardness factor better than $\frac{1}{1-e^{-1/2}} \approx 2.54$ for $(\varepsilon, C\varepsilon)$-approximation of 2-Lin(2). We believe that this highlights a serious bottleneck in obtaining hardness results matching the performance of algorithms for this problem as most optimal NP-inapproximability results involve pairwise-independent predicates.

2 Preliminaries

▶ **Definition 9.** Given $x, y \in \{-1,1\}^n$, the *Hamming distance* between x and y, denoted $d_H(x,y)$, is the number of coordinates i where x_i and y_i differ. Similarly, if $f, g : V \to \{-1,1\}$ are two functions over a variable set V, then the Hamming distance $d_H(f,g)$ between them is the number of inputs x where $f(x)$ and $g(x)$ disagree.

▶ **Definition 10.** A *predicate* on n variables is a function $\phi : \{-1,1\}^n \to \{0,1\}$. We say that $x \in \{-1,1\}^n$ *satisfies* ϕ if $\phi(x) = 1$ and otherwise that it *violates* ϕ.

▶ **Definition 11.** Given a predicate $\phi : \{-1,1\}^n \to \{0,1\}$, $\mathsf{Sat}(\phi)$ is the set of satisfying assignments.

▶ **Definition 12.** A set $S \subseteq \{-1,1\}^n$ is a *balanced pairwise-independent subgroup* if it satisfies the following properties:

1. S forms a group under bitwise multiplication.
2. If $\boldsymbol{x}$ is selected from S uniformly at random, then $\Pr[\boldsymbol{x}_i = 1] = \Pr[\boldsymbol{x}_i = -1] = \frac{1}{2}$ for any $i \in [n]$. Furthermore, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are independent for any $i \neq j$.

A predicate $\phi : \{-1,1\}^n \to \{0,1\}$ *contains a balanced pairwise-independent subgroup* if there exists a set $S \subseteq \mathsf{Sat}(\phi)$ which is a balanced pairwise-independent subgroup.

▶ **Definition 13.** For a subset $S \subseteq [n]$, the parity function $\chi_S : \{-1,1\}^n \to \{-1,1\}$ is defined as $\chi_S(x) := \prod_{i \in S} x_i$.

▶ **Definition 14.** The Had_k predicate has $2^k - 1$ input variables, one for each nonempty subset $S \subseteq [k]$. The input string $\{x_S\}_{\emptyset \neq S \subseteq [k]}$ satisfies Had_k if for each S, $x_S = \chi_S(x)$.

▶ **Fact 15.** *The* Had_k *predicate contains a balanced pairwise-independent subgroup. (In fact, the whole set* $\mathsf{Sat}(\mathsf{Had}_k)$ *is a balanced pairwise-independent subgroup.)*

Given a predicate $\phi : \{-1,1\}^n \to \{0,1\}$, an instance $\mathcal{I}$ of the Max-ϕ CSP is a variable set V and a distribution of ϕ-constraints on these variables. To sample a constraint from this distribution, we write $\mathcal{C} \sim \mathcal{I}$, where $\mathcal{C} = ((x_1, b_1), (x_2, b_2), \ldots, (x_n, b_n))$. Here the x_i's are in V and the b_i's are in $\{-1,1\}$. An assignment $A : V \to \{-1,1\}$ satisfies the constraint $\mathcal{C}$ if

$$\phi\left(b_1 \cdot A(x_1), b_2 \cdot A(x_2), \ldots, b_n \cdot A(x_n)\right) = 1.$$

We define several measures of assignments and instances.

▶ **Definition 16.** The *value* of A on $\mathcal{I}$ is just $\mathsf{val}(A;\mathcal{I}) := \Pr_{\mathcal{C} \sim \mathcal{I}}[A \text{ satisfies } \mathcal{C}]$, and the value of the instance $\mathcal{I}$ is $\mathsf{val}(\mathcal{I}) := \max_{\text{assignments } A} \mathsf{val}(A;\mathcal{I})$. We define $\mathsf{uval}(A;\mathcal{I}) := 1 - \mathsf{val}(A;\mathcal{I})$ and similarly $\mathsf{uval}(\mathcal{I})$.

▶ **Definition 17.** Let $(=) : \{-1,1\}^2 \to \{0,1\}$ be the equality predicate, i.e. $(=)(b_1, b_2) = 1$ iff $b_1 = b_2$ for all $b_1, b_2 \in \{-1,1\}$. We will refer to the Max-$(=)$ CSP as the *Max*-2-Lin(2) CSP. Any constraint $\mathcal{C} = ((x_1, b_1), (x_2, b_2))$ in a Max-2-Lin(2) instance tests "$x_1 = x_2$" if $b_1 \cdot b_2 = 1$, and otherwise tests "$x_1 \neq x_2$".

Typically, a hardness of approximation result will show that given an instance $\mathcal{I}$ of the Max-ϕ problem, it is NP-hard to tell whether $\mathsf{val}(\mathcal{I}) \geq c$ or $\mathsf{val}(\mathcal{I}) \leq s$, for some numbers $c > s$. A stronger notion of hardness is *uselessness*, first defined in [4], in which in the second case, not only is $\mathsf{val}(\mathcal{I})$ small, but any assignment to the variables A appears "uniformly random" to the constraints. To make this formal, we will require a couple of definitions.

▶ **Definition 18.** Given two probability distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ on some set S, the total variation distance d_{TV} between them is defined to be $d_{TV}(\mathcal{D}_1, \mathcal{D}_2) := \sum_{e \in S} \frac{1}{2} |\mathcal{D}_1(e) - \mathcal{D}_2(e)|$.

▶ **Definition 19.** Given a Max-ϕ instance $\mathcal{I}$ and an assignment A, denote by $\mathcal{D}(A, \mathcal{I})$ the distribution on $\{-1,1\}^n$ generated by first sampling $((x_1, b_1), \ldots, (x_n, b_n)) \sim \mathcal{I}$ and then outputting $(b_1 \cdot A(x_1), \ldots, b_n \cdot A(x_n))$.

The work of [6] showed uselessness for a wide range of predicates, including the Had_k predicate.

▶ **Theorem 20** ([6])**.** *Let* $\phi : \{-1,1\}^n \to \{0,1\}$ *contain a balanced pairwise-independent subgroup. For every* $\epsilon > 0$*, given an instance* $\mathcal{I}$ *of Max-*ϕ*, it is* NP*-hard to distinguish between the following two cases:*

- (Completeness): $\mathsf{val}(\mathcal{I}) \geq 1 - \epsilon$.
- (Soundness): *For every assignment* A*,* $d_{TV}\left(\mathcal{D}(A, \mathcal{I}), \mathcal{U}_n\right) \leq \epsilon$*, where* $\mathcal{U}_n$ *is the uniform distribution on* $\{-1,1\}^n$*.*

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & -1 \\ 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & -1 \\ -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 & -1 \end{bmatrix}$$

Figure 1 The Had_3-matrix. The rows are the satisfying assignments of Had_3.

2.1 Gadgets

The work of Trevisan et al [20] gives a generic methodology for constructing gadget reductions between two predicates. In this section, we review this with an eye towards our eventual Had_k-to-2-Lin(2) gadgets.

Suppose $\phi : \{-1,1\}^n \to \{0,1\}$ is a predicate one would like to reduce to another predicate $\psi : \{-1,1\}^m \to \{0,1\}$. Set $K := |\mathsf{Sat}(\phi)|$. We begin by arranging the elements of $\mathsf{Sat}(\phi)$ as the rows of a $K \times n$ matrix, which we will call the *ϕ-matrix.* An example of this is done for the Had_3 predicate in Figure 1.

The columns of this matrix are elements of $\{-1,1\}^K$. Naming this set $V := \{-1,1\}^K$, we will think of V as the set of possible variables to be used in a gadget reduction from ϕ to ψ. One of the contributions of [20] was to show that the set V is sufficient for any such gadget reduction, and that any gadget reduction with more than 2^K variables has redundant variables which can be eliminated.

Of these variables, the n variables found as the columns of the ϕ-matrix are special; they correspond to n of the variables in the original ϕ instance and are therefore called *generic primary* variables. We will call them $v_1, v_2, \ldots, v_n$, where they are ordered by their position in the ϕ-matrix. The remaining variables are called generic *auxiliary* variables. For example, per Figure 1, $(1,1,1,1,-1,-1,-1,-1)$ and $(1,-1,-1,1,-1,1,1,-1)$ are generic primary variables in any gadget reducing from ϕ, but $(-1,-1,1,-1,1,-1,1,-1)$ is always a generic auxiliary variable.

On top of the variables V will be a distribution of ψ constraints. As a result, a gadget $\mathcal{G}$ is just an instance of the Max-ψ CSP using the variable set V. As above, we will associate $\mathcal{G}$ with the distribution of ψ constraints and write $\mathcal{C} \sim \mathcal{G}$ to sample a constraint from this distribution. Given an assignment $A : V \to \{0,1\}$, the goal is for $\mathcal{G}$ to be able to detect whether the values A assigns to the generic primary variables satisfy the ϕ predicate. For shorthand, we will say that A *satisfies* ϕ when

$$\phi\left(A(v_1), A(v_2), \ldots, A(v_n)\right) = 1.$$

On the other hand, A *fails to satisfy* ϕ when this expression evaluates to 0. Of all assignments, we are perhaps most concerned with the *dictator* assignments. The i-th dictator assignment, written $d_i : \{-1,1\}^K \to \{-1,1\}$, is defined so that $d_i(x) = x_i$ for all $x \in \{-1,1\}^K$. The following fact shows why the dictator assignments are so important:

▶ **Fact 21.** *Each dictator assignment d_i satisfies ϕ.*

Proof. The string $((v_1)_i, (v_2)_i, \ldots, (v_n)_i)$ is the i-th row of the ϕ-matrix, which, by definition, satisfies ϕ. ◀

At this point, we can now give the standard definition of a gadget. Typically, one constructs a gadget so that the dictator assignments pass with high probability, whereas every assignment which fails to satisfy ϕ passes with low probability. This is formalized in the following definition, which is essentially from [20]:

▶ **Definition 22** (Old definition)**.** A (c, s)-*generic gadget* reducing Max-ϕ to Max-ψ is a gadget $\mathcal{G}$ satisfying the following properties:

- (Completeness): For every dictator assignment d_i, $\mathsf{uval}(d_i; \mathcal{G}) \le c$.
- (Soundness): For any assignment A which fails to satisfy ϕ, $\mathsf{uval}(A; \mathcal{G}) \ge s$.

We use uval as our focus is on the deletion version of 2-Lin(2). We include the word *generic* in this definition to distinguish it from the specific type of gadget we will use to reduce Had_k to 2-Lin(2). See Section 2.3 for details.

This style of gadget reduction is appropriate for the case when one is reducing from a predicate for which one knows an inapproximability result and nothing else. However, in our case we are reducing from predicates containing a balanced pairwise-independent subgroup, and Chan [6] has shown *uselessness* for this class of predicates (see Theorem 20). As a result, we can relax the (Soundness) condition in Definition 22; when reducing from this class of predicates, it is sufficient to show that this (Soundness) condition holds for *distributions* of assignments which *appear random on the generic primary variables.* In the following paragraph we expand on what this means.

Denote by $\mathcal{A}$ a distribution over assignments A. The value of $\mathcal{A}$ is just the average value of an assignment drawn from $\mathcal{A}$, i.e. $\mathsf{val}(\mathcal{A}; \mathcal{G}) := \mathbf{E}_{A\sim\mathcal{A}}\, \mathsf{val}(A; \mathcal{G})$, and similarly for $\mathsf{uval}(\mathcal{A}; \mathcal{G})$. We say that $\mathcal{A}$ is *random on the generic primary variables* if the tuple

$$(A(v_1), A(v_2), \ldots, A(v_n))$$

is, over a random $A \sim \mathcal{A}$, distributed as a *uniformly* random element of $\{-1, 1\}^n$.

▶ **Definition 23.** Denote by $\mathsf{R}^{gen}(\phi)$ the set of distributions which are *(uniformly)* random on the generic primary variables.

Our key definition is the following, which requires that our gadget only does well against distributions in $\mathsf{R}^{gen}(\phi)$.

▶ **Definition 24** (New definition)**.** A (c, s)-*generic gadget* reducing Max-ϕ to Max-ψ is a gadget $\mathcal{G}$ satisfying the following properties:

- (Completeness): For every dictator assignment d_i, $\mathsf{uval}(d_i; \mathcal{G}) \le c$.
- (Soundness): For any $\mathcal{A} \in \mathsf{R}^{gen}(\phi)$, $\mathsf{uval}(\mathcal{A}; \mathcal{G}) \ge s$.

The following proposition is standard, and we sketch its proof for completeness.

▶ **Proposition 25.** *Suppose there exists a (c, s)-generic gadget reducing Max-ϕ to Max-ψ, where Max-ϕ is any predicate containing a balanced pairwise-independent subgroup. Then for all $\epsilon > 0$, given an instance $\mathcal{I}$ of Max-ψ, it is* NP*-hard to distinguish between the following two cases:*

- (Completeness): $\mathsf{uval}(\mathcal{I}) \le c + \epsilon$.
- (Soundness): $\mathsf{uval}(\mathcal{I}) \ge s - \epsilon$.

Proof sketch. Let $\mathcal{I}$ be an instance of the Max-ϕ problem produced via Theorem 20. To dispense with some annoying technicalities, we will assume that every constraint $\mathcal{C}$ in the support of $\mathcal{I}$ is of the form $\mathcal{C} = ((x_1, 1), \ldots, (x_n, 1))$. Construct an instance $\mathcal{I}'$ of Max-ψ as follows: for each constraint $\mathcal{C} = ((x_1, 1), \ldots, (x_n, 1))$ in the support of $\mathcal{I}$, add in a copy of $\mathcal{G}$ – call it $\mathcal{G}_\mathcal{C}$ – whose total weight is scaled down so that it equals the weight of $\mathcal{C}$. Further, identify the primary variables $v_1, \ldots, v_n$ of $\mathcal{G}_\mathcal{C}$ with the variables $x_1, \ldots, x_n$.

Completeness: In this case, there exists an assignment A to the variables of $\mathcal{I}$ which violates at most an ϵ-fraction of the constraints. We will extend this to an assignment for all the variables of $\mathcal{I}'$ as follows: for any constraint $\mathcal{C} = ((x_1, 1), \ldots, (x_n, 1))$ which A satisfies, there is some dictator assignment to the variables of $\mathcal{G}_{\mathcal{C}}$ which agrees with A on the primary variables $v_1, \ldots, v_n$. Set A to also agree with this dictator assignment on the auxiliary variables in $\mathcal{G}_{\mathcal{C}}$. Regardless of how A is extended in the remaining $\mathcal{G}_{\mathcal{C}}$'s, it now labels a $(1 - \epsilon)$-fraction of the $\mathcal{G}$ gadgets in $\mathcal{I}'$ with a dictator assignment, meaning that $\mathsf{uval}(A; \mathcal{I}') \leq (1 - \epsilon) \cdot c + \epsilon \cdot 1 \leq c + \epsilon$.

Soundness: Let A be an assignment to the variables in $\mathcal{I}'$. Consider the distribution $\mathcal{A}$ of assignments to the gadget $\mathcal{G}$ generated as follows: sample $\mathcal{C} \sim \mathcal{I}$ and output the restriction of A to the variables of $\mathcal{G}_{\mathcal{C}}$. Because the distribution $(A(x_1), \ldots, A(x_n))$ is ϵ-close to uniform in total variation distance, $\mathcal{A}$ is ϵ-close in total variation distance to some distribution $\mathcal{A}' \in \mathsf{R}^{gen}(\phi)$. As a result, $\mathsf{uval}(\mathcal{A}; \mathcal{G}) \geq \mathsf{uval}(\mathcal{A}'; \mathcal{G}) - \epsilon \geq s - \epsilon$. But then $\mathsf{uval}(\mathcal{A}; \mathcal{G}) = \mathsf{uval}(A; \mathcal{I})$, which is therefore bounded below by $s - \epsilon$. ◀

2.2 Reducing into 2-Lin(2)

In this section, we consider gadgets which reduce into the 2-Lin(2) predicate. We show several convenient simplifying assumptions that can be made in this case.

▶ **Definition 26.** An assignment $A : \{-1, 1\}^K \to \{-1, 1\}$ is *folded* if $A(x) = -A(-x)$ for all $x \in \{-1, 1\}^K$. Here $-x$ is the bitwise negation of x. In addition, a distribution $\mathcal{A}$ is folded if every assignment in its support is folded.

The following proposition shows that when designing a gadget which reduces into 2-Lin(2), it suffices to ensure that its (Soundness) condition holds for folded distributions. The proof is standard.

▶ **Proposition 27.** *For some predicate ϕ, suppose $\mathcal{G}$ is a gadget reducing Max-ϕ to Max-2-*Lin*(2) which satisfies the following two conditions:*

- (Completeness): *For every dictator assignment d_i,* $\mathsf{uval}(d_i; \mathcal{G}) \leq c$.
- (Soundness): *For any* folded $\mathcal{A} \in \mathsf{R}^{gen}(\phi)$, $\mathsf{uval}(\mathcal{A}; \mathcal{G}) \geq s$.

*Then there exists a (c, s)-generic gadget reducing Max-ϕ to Max-2-*Lin*(2).*

Proof. For each pair of antipodal points x and $-x$ in $\{-1, 1\}^K$, pick one (say, x) arbitrarily, and set

$$\mathsf{canon}(x) := \mathsf{canon}(-x) := x.$$

This is the canonical variable associated to x and $-x$. The one constraint is that if either x or $-x$ is one of the generic primary variables, then it should be chosen as the canonical variable associated to x and $-x$. Now, let $\mathcal{G}'$ be the gadget whose constraints are sampled as follows:

1. Sample a constraint $A(x_1) \cdot A(x_2) = b$ from $\mathcal{G}$.
2. For $i \in \{1, 2\}$, set $b_i = 1$ if $\mathsf{canon}(x_i) = x_i$ and $b_i = -1$ otherwise.
3. Output the constraint $A(\mathsf{canon}(x_1)) \cdot A(\mathsf{canon}(x_2)) = b \cdot b_1 \cdot b_2$.

We claim that $\mathcal{G}'$ is a (c, s)-gadget reducing Max-ϕ to Max-2-Lin(2). To see this, set $\mathsf{is\text{-}canon}(x)$ to be 1 if $\mathsf{canon}(x) = x$ and (-1) otherwise. Then the probability that an assignment A fails on $\mathcal{G}'$ is the same as the probability that the assignment $A'(x) := \mathsf{is\text{-}canon}(x) \cdot A(\mathsf{canon}(x))$ fails on $\mathcal{G}$. For any dictator function d_i, $d_i(x) = \mathsf{is\text{-}canon}(x) \cdot d_i(\mathsf{canon}(x))$ for all x. Therefore,

d_i fails $\mathcal{G}'$ with probability c. Next, it is easy to see that for any assignment A, A' is folded and, due to our restriction on $\mathsf{canon}(\cdot)$, A' agrees with A on the generic primary variables. Thus, given a distribution $\mathcal{A} \in \mathsf{R}^{gen}(\phi)$, $\mathcal{A}$ fails on $\mathcal{G}'$ with the same probability that some folded distribution in $\mathsf{R}^{gen}(\phi)$ fails on $\mathcal{G}$, which is at least s. ◀

▶ **Proposition 28.** *For fixed values of c and s, let $\mathcal{G}$ be a gadget satisfying the (Completeness) and (Soundness) conditions in the statement of Proposition 27. Then there exists another gadget satisfying these conditions which only uses equality constraints.*

Proof. Let $\mathcal{G}'$ be the gadget which replaces each constraint in $\mathcal{G}$ of the form $x \neq y$ with the constraint $x = -y$. If A is a folded assignment,

$$A(x) \neq A(y) \iff A(x) = A(-y).$$

Thus, for every folded assignment A, $\mathsf{val}(A;\mathcal{G}) = \mathsf{val}(A,\mathcal{G}')$. As the (Completeness) and (Soundness) conditions in Proposition 27 only concern folded assignments, $\mathcal{G}'$ satisfies these conditions. ◀

This means that sampling from $\mathcal{G}$ can be written as $(x, y) \sim \mathcal{G}$, meaning that we have sampled the constraint "$x = y$".

2.3 The Had_k-to-2-Lin Gadget

Now we focus on our main setting, which is constructing a Had_k-to-2-Lin(2) gadget. Via Section 2.2, we need only consider how well the gadget does against folded assignments.

The Had_k predicate has $2^k - 1$ variables. In addition, it has $K := 2^k$ satisfying assignments, one for each setting of the variables $x_{\{1\}}$ through $x_{\{k\}}$. It will often be convenient to take an alternative (but equivalent) viewpoint of the variable set $V := \{-1,1\}^K$ as the set of k-variable Boolean functions, i.e.

$$V = \left\{ f \mid f : \{-1,1\}^k \to \{-1,1\} \right\}.$$

The Had_k matrix is a $2^k \times (2^k - 1)$ matrix whose rows are indexed by strings in $\{-1,1\}^k$ and whose columns are indexed by nonempty subsets $S \subseteq [k]$. The (x, S)-entry of this matrix is $\chi_S(x)$. This can be verified by noting that for any $x \in \{-1,1\}^k$,

$$\left(\chi_{\{1\}}(x), \chi_{\{2\}}(x), \ldots, \chi_{\{k\}}(x), \chi_{\{1,2\}}(x), \ldots, \chi_{\{1,2,\ldots,k\}}(x), \right)$$

is a satisfying assignment of the Had_k predicate. As a result, for each $S \neq \emptyset$, χ_S is a column in the Had_k matrix. Therefore, these functions are the generic primary variables. However, it will be convenient to consider a larger set of functions to be primary. For example, because we plan on using our gadget on folded assignments, χ_S and $-\chi_S$ will always have opposite values, and so the $-\chi_S$'s should also be primary variables. In addition, it is a little unnatural to have every parity function but one be a primary variable, so we will include the constant function $\chi_\emptyset$ and its negation $-\chi_\emptyset$ in the set of primary variables. In total, we have the following definition.

▶ **Definition 29.** The *primary variables* of a Had_k-to-2-Lin(2) gadget are the functions $\pm\chi_S$, for any $S \subseteq [k]$. The remaining functions are auxiliary variables.

To account for the inclusion of $\chi_\emptyset$ as a primary variable, we will have to modify some of our definitions from Section 2.1. We begin by defining a modification to the Had_k predicate.

▶ **Definition 30.** The Had_k^* predicate has 2^k input variables, one for each subset $S \subseteq [k]$. The input string $\{x_S\}_{S\subseteq[k]}$ satisfies Had_k^* if for each S, $x_S = x_\emptyset \cdot \prod_{i\in S} x_{\{i\}}$.

In other words, if $x_\emptyset = 1$, then the remaining variables should satisfy the Had_k predicate, and if $x_\emptyset = -1$, then their negations should. We will say that A *satisfies the* Had_k^* *predicate* if

$$\mathsf{Had}_k^*\left(A\left(\chi_\emptyset\right), A\left(\chi_{\{1\}}\right), \ldots, A\left(\chi_{\{k\}}\right), A\left(\chi_{\{1,2\}}\right), \ldots, A\left(\chi_{[k]}\right)\right) = 1.$$

Otherwise, *A fails to satisfy the* Had_k^* *predicate.* We say that $\mathcal{A}$ is *random on the primary variables* if the tuple

$$\left(A\left(\chi_\emptyset\right), A\left(\chi_{\{1\}}\right), \ldots, A\left(\chi_{\{k\}}\right), A\left(\chi_{\{1,2\}}\right), \ldots, A\left(\chi_{[k]}\right)\right)$$

is, over a random $A \sim \mathcal{A}$, distributed as a uniformly random element of $\{-1,1\}^K$.

▶ **Definition 31.** Denote by $\mathsf{R}(\mathsf{Had}_k)$ the set of folded distributions which are uniformly random on the primary variables.

▶ **Definition 32.** A (c,s)*-gadget* reducing Max-Had_k to Max-2-Lin(2) is a gadget $\mathcal{G}$ satisfying the following properties:

- (Completeness): For every dictator assignment d_i, $\mathsf{uval}(d_i;\mathcal{G}) \le c$.
- (Soundness): For any $\mathcal{A} \in \mathsf{R}(\mathsf{Had}_k)$, $\mathsf{uval}(\mathcal{A};\mathcal{G}) \ge s$.

▶ **Proposition 33.** *The following two statements are equivalent:*

1. *There exists a (c,s)-gadget reducing Max-*Had_k *to Max-*2-Lin(2)*.*
2. *There exists a (c,s)-generic gadget reducing Max-*Had_k *to Max-*2-Lin(2)*.*

Proof. We prove the two directions separately.

(1) ⇒ (2): Let $\mathcal{G}$ be a (c,s)-gadget reducing Max-Had_k to Max-2-Lin(2). We claim that for any folded $\mathcal{A} \in \mathsf{R}^{gen}(\mathsf{Had}_k)$, $\mathsf{uval}(\mathcal{A};\mathcal{G}) \ge s$. To see this, consider the distribution $\mathcal{A}' \in \mathsf{R}(\mathsf{Had}_k)$ which samples $A \sim \mathcal{A}$ and outputs either A or $-A$, each with half probability. Then $\mathsf{uval}(\mathcal{A}';\mathcal{G}) = \mathsf{uval}(\mathcal{A};\mathcal{G})$, and furthermore we know that $\mathsf{uval}(\mathcal{A};\mathcal{G}) \ge s$. As a result, $\mathcal{G}$ satisfies the (Completeness) and (Soundness) conditions in the statement of Proposition 27, meaning there exists a (c,s)-generic gadget reducing Max-Had_k to Max-2-Lin(2).

(2) ⇒ (1): Let $\mathcal{G}$ be a (c,s)-*generic* gadget reducing Max-Had_k to Max-2-Lin(2). Let $\mathcal{A} \in \mathsf{R}(\mathsf{Had}_k)$, and for $b \in \{-1,1\}$, write $\mathcal{A}_{(b)}$ for $\mathcal{A}$ conditioned on the variable $\chi_\emptyset$ being assigned the value b. Then $b \cdot \mathcal{A}_{(b)}$ (by which we mean the distribution where we sample $A \sim \mathcal{A}_{(b)}$ and output $b \cdot A$) is in $\mathsf{R}^{gen}(\mathsf{Had}_k)$ for both $b \in \{-1,1\}$, and so $\mathsf{uval}\left(b\cdot\mathcal{A}_{(b)};\mathcal{G}\right) \ge s$. As $\mathsf{uval}(\mathcal{A}_{(b)};\mathcal{G}) = \mathsf{uval}\left(b\cdot\mathcal{A}_{(b)};\mathcal{G}\right)$, $\mathsf{uval}(\mathcal{A};\mathcal{G}) \ge s$, and so $\mathcal{G}$ is a (c,s)-gadget reducing Max-Had_k to Max-2-Lin(2). ◀

Combining this with Proposition 25, we have the following corollary.

▶ **Corollary 34.** *Suppose there exists a (c,s)-gadget reducing Max-*Had_k *to Max-*2-Lin(2)*. Then for all $\epsilon > 0$, given an instance $\mathcal{I}$ of Max-*2-Lin(2)*, it is* NP*-hard to distinguish between the following two cases:*

- (Completeness): $\mathsf{uval}(\mathcal{I}) \le c + \epsilon$.
- (Soundness): $\mathsf{uval}(\mathcal{I}) \ge s - \epsilon$.

2.4 Reducing to the length-one case

When constructing good gadgets, we generally want dictators to pass with as high of probability as possible. By Proposition 28, we can assume that our gadget operates by sampling an edge (x, y) and testing equality between the two endpoints. Any such edge of Hamming distance i will be violated by $\frac{i}{K}$ of the dictator assignments. Intuitively, then, if we want to dictators to pass with high probability, we should concentrate the probability mass of our gadget $\mathcal{G}$ on edges of low Hamming distance. The following proposition shows that this is true in the extreme: so long as we are only concerned with maximizing the quantity $\frac{s}{c}$, we can always assume that $\mathcal{G}$ is entirely supported on edges of Hamming distance one.

▶ **Proposition 35.** *Suppose there exists a (c, s)-gadget $\mathcal{G}$ reducing Max-*Had_k *to Max-*2-Lin(2)*. Then there exists a (c', s')-gadget reducing Max-*Had_k *to Max-*2-Lin(2) *using only length-one edges for which*

$$\frac{s'}{c'} \geq \frac{s}{c}.$$

Proof. For each $i \in \{1, \dots, K\}$, let p_i be the probability that an edge sampled from $\mathcal{G}$ has length i, and let $\mathcal{G}_i$ denote the distribution of $\mathcal{G}$ conditioned on this event. Then sampling from $\mathcal{G}$ is equivalent to first sampling a length i with probability p_i, and then sampling an edge from $\mathcal{G}_i$.

Let $Q = 1 \cdot p_1 + 2 \cdot p_2 + \ldots + K \cdot p_K$, and for each $i \in \{1, \dots, K\}$ define $q_i = \frac{i \cdot p_i}{Q}$. It is easy to see that the q_i's form a probability distribution. Now we may define the new gadget $\mathcal{G}'$ as follows:

1. Sample a length i with probability q_i.
2. Sample $(x, y) \sim \mathcal{G}_i$.
3. Pick an arbitrary shortest path $x = x_0, x_1, \dots, x_i = y$ through the hypercube $\{-1, 1\}^K$.
4. Output a uniformly random edge (x_j, x_{j+1}) from this path.

Note that $\mathcal{G}'$ only uses length-one edges. Let $\mathcal{G}'_i$ denote the distribution of $\mathcal{G}'$ conditioned on i being sampled in the first step. (Note that $\mathcal{G}'_i$ is defined in a way that is different from the way $\mathcal{G}_i$ is defined.)

Let $A : \{-1, 1\}^K \to \{-1, 1\}$ be any assignment. Then

$$\mathsf{uval}(A; \mathcal{G}) = \sum_{i=1}^{K} p_i \cdot \mathsf{uval}(A; \mathcal{G}_i), \qquad \text{and} \qquad \mathsf{uval}(A; \mathcal{G}') = \sum_{i=1}^{K} q_i \cdot \mathsf{uval}(A; \mathcal{G}'_i).$$

We can relate $\mathsf{uval}(A; \mathcal{G}'_i)$ to $\mathsf{uval}(A; \mathcal{G}_i)$ as follows: if A assigns different values to the endpoints of the edge $(x, y) \sim \mathcal{G}$, then on any shortest path $x = x_0, x_1, \dots, x_i = y$ through the hypercube $\{-1, 1\}^K$, A must assign different values to at least one of the edges (x_j, x_{j+1}). As a result, every time A errs on $\mathcal{G}_i$, it must err at least a $(1/i)$-fraction of the time on $\mathcal{G}'_i$. This means that:

$$\mathsf{uval}(A; \mathcal{G}'_i) \geq \frac{\mathsf{uval}(A; \mathcal{G}_i)}{i}. \tag{1}$$

In the case when A is a dictator function, Equation (1) becomes an equality. This is because $x = x_0, x_1, \dots, x_i = y$ is a shortest path between x and y through the hypercube $\{-1, 1\}^K$. If A assigns the same values to x and y, then it will assign the same values to all of $x_0, x_1, \dots, x_i$. If, on the other hand, it assigns different values to x and y, then it will assign different values to the endpoints of exactly one edge (x_j, x_{j+1}).

Now we can use this relate $\mathsf{uval}(A;\mathcal{G}')$ to $\mathsf{uval}(A;\mathcal{G})$:

$$\begin{aligned}\mathsf{uval}(A;\mathcal{G}') &= \sum_{i=1}^{K} q_i \cdot \mathsf{uval}(A;\mathcal{G}'_i)\\ &\geq \sum_{i=1}^{K}\left(\frac{i\cdot p_i}{Q}\right)\cdot\frac{\mathsf{uval}(A;\mathcal{G}_i)}{i}\\ &= \frac{1}{Q}\sum_{i=1}^{K} p_i \cdot \mathsf{uval}(A;\mathcal{G}_i)\\ &= \frac{1}{Q}\mathsf{uval}(A;\mathcal{G}). \end{aligned} \tag{2}$$

Here the inequality follows from the definition of q_i and Equation (1). As Equation (1) is an equality in the case when A is a dictator function, we have that $\mathsf{uval}(A;\mathcal{G}') = \frac{1}{Q}\mathsf{uval}(A;\mathcal{G})$ in this case.

Let $\mathcal{A} \in \mathsf{R}(\mathsf{Had}_k)$ maximize $\mathsf{val}(\mathcal{A};\mathcal{G}')$, and let d_i be any dictator function. Then

$$\frac{s'}{c'} = \frac{\mathsf{uval}(\mathcal{A};\mathcal{G}')}{\mathsf{uval}(d_i;\mathcal{G}')} \geq \frac{\frac{1}{Q}\mathsf{uval}(\mathcal{A};\mathcal{G})}{\frac{1}{Q}\mathsf{uval}(d_i;\mathcal{G})} = \frac{\mathsf{uval}(\mathcal{A};\mathcal{G})}{\mathsf{uval}(d_i;\mathcal{G})} \geq \frac{s}{c}.$$

Here the first inequality is by Equation (2) (and the fact that it is an equality for dictators), and the second inequality follows from the fact that $\mathsf{uval}(\mathcal{A},\mathcal{G}) \geq s$ and $\mathsf{uval}(d_i,\mathcal{G}) = c$. ◀

2.5 Linear programs

One of the key insights of the paper [20] is that optimal gadgets (as per Definition 22) can be computed by simply solving a linear program. Fortunately, the same holds for computing optimal gadgets as per Definition 32. In our case, the appropriate linear program (taking into account Proposition 35) is:

$$\begin{aligned} \max \quad & s\\ \text{s.t.} \quad & \mathsf{uval}(\mathcal{A};\mathcal{G}) \geq s, \quad \forall \mathcal{A} \in \mathsf{R}(\mathsf{Had}_k),\\ & \mathcal{G} \text{ is a gadget supported on edges of length one.}\end{aligned}$$

As written, this linear program has an (uncountably) infinite number of constraints, but this can fixed by suitably discretizing the set $\mathsf{R}(\mathsf{Had}_k)$. This is not so important for us, as even after performing this step, the linear program is simply too large to ever be feasible in practice. What is important for us is that we can take its dual; doing so yields the following linear program:

► **Definition 36.** The *dual LP* is defined as

$$\begin{aligned} \min \quad & s\\ \text{s.t.} \quad & \Pr_{A\sim\mathcal{A}}[A(x) = A(y)] \leq s, \quad \forall \text{ edges } (x,y) \text{ of length one,} && (3)\\ & \mathcal{A} \in \mathsf{R}(\mathsf{Had}_k). && (4)\end{aligned}$$

The dual linear program shows us that we can upper-bound the soundness of any gadget with the value s by exhibiting a distribution on assignments in $\mathsf{R}(\mathsf{Had}_k)$ which passes each length-one edge with probability at least s. Moreover, strong LP duality tells us that the optimum values of the two LPs are the same. Hence, we can prove a *tight* upper bound by exhibiting the right distribution. We do this in Section 3 for gadgets reducing Max-Had_3 to Max-2-Lin(2).

2.6 The Had_3 gadget

In this section, we will prove some structural results about the hypercube $\{-1,1\}^8$ which are relevant to any Had_3-to-2-Lin(2) gadget. The results of this section will be useful for Section 3.

Given a string $x \in \{-1,1\}^n$ and subset of strings $B \subseteq \{-1,1\}^n$, we define the distance of x to B as $d_H(x,B) := \min_{y\in B} d_H(x,y)$.

▶ **Proposition 37.** *The vertex set $V = \{-1,1\}^8$ can be partitioned as $V = V_0 \uplus V_1 \uplus V_2$, in which V_0 is the set of primary variables, and $V_i = \{x \in V \mid d_H(x, V_0) = i\}$, for $i = 1, 2$.*

▶ **Proposition 38.** $|V_0| = 16,\quad |V_1| = 128,$ *and* $|V_2| = 112$.

▶ **Proposition 39.**
- *Each $x \in V_0$ has eight neighbors in V_1.*
- *Each $x \in V_1$ has one neighbor in V_0 and seven neighbors in V_2.*
- *Each $x \in V_2$ has eight neighbors in V_1. Furthermore, there are four elements of V_0 which are Hamming distance two away from x.*

▶ **Proposition 40.** *Let $f \in V_2$, and let g_1, g_2, g_3, and g_4 be the four elements of V_0 which are Hamming distance two away from f. Then for any $x \in \{-1,1\}^3$, three of the g_i's have the same value and one has a different value, and $f(x) = \mathsf{sign}(g_1(x) + g_2(x) + g_3(x) + g_4(x))$.*

Proof of Propositions 37, 38, 39, and 40. In this proof, we will take the viewpoint of V as the set of 3-variable Boolean functions. The primary variables are of the form $\pm\chi_S$, where $S \subseteq [3]$. There are 16 such functions, and so $|V_0| = 16$.

Let f' differ from one of the primary variables on a single input. From above, it must be at least distance 3 from any of the other primary variables. This immediately implies that f''s seven other neighbors are in V_2. There are $16 \cdot 8 = 128$ distinct ways of constructng f', and so $|V_1| = 128$.

This leaves $256 - 16 - 128 = 112$ variables in V not yet accounted for. We will now show a method for constructing 112 different elements of V_2; by the pigeonhole principle, this shows that V can be partitioned as Proposition 37 guarantees. Given three primary variables $b_1\chi_{S_1}$, $b_2\chi_{S_2}$, and $b_3\chi_{S_3}$, where $b_1, b_2, b_3 \in \{-1,1\}$, set $b_4 := -b_1 \cdot b_2 \cdot b_3$ and $S_4 := S_1 \Delta S_2 \Delta S_3$. Consider the function $f : \{-1,1\}^3 \to \{-1,1\}$ defined as

$$f(x) := \mathsf{sign}\left(b_1\chi_{S_1}(x) + b_2\chi_{S_2}(x) + b_3\chi_{S_3}(x) + b_4\chi_{S_4}(x)\right).$$

Our claim is that f is distance-2 from each of the $b_i\chi_{S_i}$'s. First, to see that this $\mathsf{sign}(\cdot)$ is well-defined, note that by definition, $\prod_{i=1}^4 b_i\chi_{S_i}(x) = -1$ for all $x \in \{-1,1\}^3$. As a result, for any x, three of the $b_i\chi_{S_i}(x)$'s have the same value, while the other one has a different value. This means that

$$\sum_{i=1}^4 b_i\chi_{S_i}(x) = 2 \cdot \mathsf{sign}\left(\sum_{i=1}^4 b_i\chi_{S_i}(x)\right).$$

for all x. Thus, the correlation of any of the $b_i\chi_{S_i}$'s with f is

$$\mathop{\mathbf{E}}_{\boldsymbol{x}}[f(\boldsymbol{x}) \cdot b_i\chi_{S_i}] = \frac{1}{2}\mathop{\mathbf{E}}_{\boldsymbol{x}}\left[\left(\sum_{i=1}^4 b_i\chi_{S_i}(x)\right) \cdot b_i\chi_{S_i}\right] = \frac{1}{2}.$$

In other words, $\Pr_{\boldsymbol{x}}[f(\boldsymbol{x}) = b_i\chi_{S_i}] = \frac{3}{4}$ for each $i \in \{1,\ldots,4\}$.

There are 8 neighbors of f; each $b_i\chi_{S_i}$ neighbors two of them. As a result, all of f's neighbors are in V_1. In addition, since they are neighbors to the $b_i\chi_{S_i}$'s, they can't be neighbors for any of the other primary variables. This means that the only variables in V_0 that f is distance 2 from are the $b_i\chi_{S_i}$'s.

There are $2 \cdot \binom{8}{3} = 112$ ways of selecting the $b_i\chi_{S_i}$'s. As there are only 112 variables in V which are not in either V_0 or V_1, all of the remaining variables in V must be contained in V_2, and they must all be generated in the manner above. ◀

▶ **Proposition 41.** *Let $B = \mathsf{sat}(\mathsf{Had}_3^*)$. Then*

$$\Pr_{\boldsymbol{x}}[d_H(\boldsymbol{x}, B) = 0] = \frac{1}{16}, \quad \Pr_{\boldsymbol{x}}[d_H(\boldsymbol{x}, B) = 1] = \frac{1}{2}, \quad \textit{and} \quad \Pr_{\boldsymbol{x}}[d_H(\boldsymbol{x}, B) = 2] = \frac{7}{16},$$

where $\boldsymbol{x}$ is a uniformly random element of $\{-1,1\}^8$.

Proof. This can be proven using a proof similar to Proposition 38. Alternatively, we can just show a direct correspondence between the setting here and the setting in Proposition 38, as follows.

The input to Had_3^* is a set of bits $\{x_S\}_{S\subseteq[k]}$, which can also be thought of as the function $f : \mathcal{P}(\{1,2,3\}) \to \{-1,1\}$ in which $f(S) := x_S$. The satisfying assignments are then any function of the form $S \to b \cdot \chi_S(x)$, where $b \in \{-1,1\}$ and $x \in \{-1,1\}^3$ are both fixed. For a string $x \in \{-1,1\}^3$, let $\alpha(x)$ be the corresponding set, i.e. $\alpha(S)_i = -1$ if and only if $i \in S$. For any function $f : \mathcal{P}(\{1,2,3\}) \to \{-1,1\}$, we can associate it with the function $\alpha(f) : \{-1,1\}^3 \to \{-1,1\}$ defined by $\alpha(f)(x) := f(\alpha(x))$ for all x. Then α maps any satisfying assignment to Had_3^* into one of the primary variables in V_0, and more generally, $d_H(f, B) = i$ if and only if $\alpha(f) \in V_i$. The proposition therefore follows by applying Proposition 38 and by noting that $\frac{16}{256} = \frac{1}{16}$, $\frac{128}{256} = \frac{1}{2}$, and $\frac{112}{256} = \frac{7}{16}$. ◀

▶ **Proposition 42.**

1. *Let $f, g \in V_0$ be a pair of distinct affine functions. Then either $d_H(f,g) = 8$, or $d_H(f,g) = 4$.*
2. *For any $x, y \in \{-1,1\}^3$, $x \neq y$, $b_x, b_y \in \{-1,1\}$, the number of functions $f \in V_0$ such that $f(x) = b_x$ is 8, and the number of functions $f \in V_0$ such that $f(x) = b_x$ and $f(y) = b_y$ is 4.*

Proof. Proof of (1): Let $f = b_f\chi_S$, and $g = b_g\chi_T$. Then $\mathbf{E}[fg] = b_f b_g \mathbf{E}[\chi_{S\Delta T}]$ where Δ is the symmetric difference of two sets. If $f = -g$, then clearly $d_H(f,g) = 8$. Now we assume that $f \neq \pm g$, and therefore $S \neq T$. Then $\mathbf{E}[\chi_{S\Delta T}] = 0$. This completes the proof.

Proof of (2): Consider function $f(x) = a_0 + a_1x_1 + a_2x_2 + a_3x_3$. Construct a linear system where a_0, a_1, a_2, a_3 are the variables and $f(x) = b_x$ and $f(y) = b_y$ are the constraints. The result follows from working out the size of the solution space. ◀

2.7 Reducing to Max-Cut

▶ **Definition 43.** Let $(\neq) : \{-1,1\}^2 \to \{0,1\}$ be the inequality predicate, i.e. $(\neq)(b_1,b_2) = 1$ iff $b_1 \neq b_2$ for all $b_1, b_2 \in \{-1,1\}$. The Max-Cut CSP is the special case of the Max-$(\neq)$ CSP in which every constraint $\mathcal{C} = ((x_1,b_1),(x_2,b_2))$ satisfies $b_1 = b_2 = 1$. In other words, every constraint is of the form "$x_1 \neq x_2$".

▶ **Proposition 44.** *For some predicate ϕ, suppose $\mathcal{G}$ is (c,s)-generic gadget reducing Max-ϕ to Max-*2-Lin(2)*. Then there exists a (c',s')-gadget reducing Max-ϕ to* Max-Cut *satisfying*

$$\frac{s'}{c'} = \frac{s}{c}.$$

Proof. Suppose the vertex set of $\mathcal{G}$ is $V = \{-1,1\}^K$. Let $\mathcal{G}'$ be the gadget which operates as follows:

1. With probability $1 - \frac{1}{2^{K-1}}$, pick $x \in \{-1,1\}^K$ and output the constraint "$x \neq -x$".
2. Otherwise, sample $\mathcal{C} \sim \mathcal{G}$. If $\mathcal{C}$ is of the form "$x \neq y$", output "$x \neq y$". If $\mathcal{C}'$ is of the form "$x = y$", output "$x \neq -y$".

Any folded assignment A fails $\mathcal{G}$' with probability at most $\frac{1}{2^{K-1}}$. Any assignment A which is *not* folded fails $\mathcal{G}$' with probability at least $\frac{1}{2^{K-1}}$. As a result, we can always assume that any assignment is folded.

Now, if A is folded, then for any $x, y \in \{-1,1\}^K$, $A(x) = A(y)$ if and only if $A(x) \neq A(-y)$. As a result, $\mathsf{uval}(A;\mathcal{G}') = \mathsf{uval}(A;\mathcal{G})/2^{k-1}$. Thus, $c' = c/2^{k-1}$, $s' = s/2^{k-1}$, and so $s'/c' = s/c$. ◀

3 The factor-11/8 gadget and its optimality

In this section, we prove the following main theorem.

▶ **Theorem 45.** *There is a* $(\frac{1}{8}, \frac{11}{64})$*-gadget reducing* Had_3 *to* 2-Lin(2)*. By a simple padding argument, this implies that for any* $C < \frac{11}{8}$*, it is* NP*-hard to achieve a factor-*C *approximation for both the Max-*2-Lin(2) *and the* Max-Cut *CSPs.*

Furthermore, the value of the LP in (3) is $\frac{11}{64}$*. This means that for every* (c,s)*-gadget reducing Max-*Had_3 *to Max-*2-Lin(2)*,* $\frac{s}{c} \leq \frac{11}{8}$*. In other words, the gadget we construct is* optimal *among gadget reductions from Chan's* 7*-ary Hadamard predicate.*

The result of NP-hardness for Max-2-Lin(2) and Max-Cut follows from discussions in Section 2. We now focus on the gadget construction part. First, we give the construction of our $(\frac{1}{8}, \frac{11}{64})$-gadget reducing Had_3 to 2-Lin(2). Recall that the set of variables in our gadget is the set of Boolean functions on 3 variables. Let $V_0 := \{\pm\chi_S\}_{S\subseteq[k]}$, V_1 be the set of 3-variable Boolean functions that are at distance 1 from some function in V_0, and V_2 be those at distance 2 from some function in V_0.

We will assign a non-negative weight to each constraint in the gadget. Our gadget will then sample each constraint with probability equal to its weight normalized by the weight of the entire gadget. As argued in Proposition 35, the gadget will only use constraints on functions at distance 1. For $f, g \in V$ with $d_H(f,g) = 1$, the weight of the edge $\{f,g\}$ is 5 if and only if either $f \in V_0$ or $g \in V_0$, and otherwise the weight is 1. The total weight of the edges in $\mathcal{G}$ is $5 \times 128 + 896 = 1536$.

To prove completeness, the fact that the dictators pass with probability $\frac{7}{8}$ follows immediately from the fact that $\mathcal{G}$ only uses edges of length one. The soundness is proved by case analysis on the effect of different partial assignments based on the structure of the gadget. Since there are no edges between variables in V_1, whenever we have a partial assignment to the variables in V_0 and V_2, we can complete it optimally by giving assignments greedily to variables in V_1. We then argue that given any folded partial assignment to variables in V_0, there is some greedy heuristics for assigning values to variables in V_2 that always achieves optimum.

To establish optimality of our gadget, we construct an optimal solution to the dual LP given in (3). Our goal is to construct $\mathcal{A} \in \mathsf{R}(\mathsf{Had}_3)$, i.e. a folded distribution of assignments which is random on the primary variables. For $i \in \{0,1,2\}$, denote by $\mathsf{R}_i(\mathsf{Had}_3)$, the set of distributions $\mathcal{A}_i$ such that over a random assignment $A \sim \mathcal{A}_i$, the string $(\{A(\chi_S)\}_{S\subseteq[k]})$ is distributed like a uniformly random element of $\{0,1\}^8$ that is at distance i from satisfying the Had_3 predicate. We construct three separate distributions $\mathcal{A}_0$, $\mathcal{A}_1$, and $\mathcal{A}_2$ with the property that $\mathcal{A}_i \in \mathsf{R}_i(\mathsf{Had}_3)$ for each $i \in \{0, 1, 2\}$. Then, we set $\mathcal{A} = \frac{1}{16}\mathcal{A}_0 + \frac{1}{2}\mathcal{A}_1 + \frac{7}{16}\mathcal{A}_2$. We study the value of $\mathcal{A}_0$, $\mathcal{A}_1$ and $\mathcal{A}_2$ on edges between V_0 and V_1, and edges between V_1 and V_2, and show that the values of $\mathcal{A}$ on both sets of edges are $\frac{53}{64}$. We complete the proof by noting that $1 - \frac{53}{64} = \frac{11}{64}$, the number guaranteed by the theorem.

A complete proof of Theorem 45 can be found in Section 3 and 4 of the full paper.

4 A candidate factor-3/2 hardness reduction

Herein we present an interesting problem concerning analysis of Boolean functions. We make a conjecture about its solution which, if true, implies NP-hardness of factor-$(\frac{3}{2} - \delta)$ approximating 2-Lin(2) for any $\delta > 0$.

▶ **Definition 46.** Let $g : \{-1, 1\}^n \to \{-1, 1\}$ be an odd function (i.e., $g(-x) = -g(x)$). The *Game Show*, played with *Middle Function* g, works as follows. There are two players: the *Host* and the *Contestant*. Before the game begins, the Host secretly picks a uniformly random monotone path $\boldsymbol{\pi}$ from $(1, 1, \dots, 1)$ to $(-1, -1, \dots, -1)$ in the Hamming cube. (We say that a path is monotone if at each step, a 1 is changed to a -1. Equivalently, $\boldsymbol{\pi}$ is a uniformly random permutation on $[n]$.) The Host also secretly picks $\boldsymbol{T} \sim \text{Binomial}(n, \frac{1}{2})$. We define the *secret half-path* to be the sequence of the first $\boldsymbol{T}$ edges along $\boldsymbol{\pi}$: $(x_0, x_1), (x_1, x_2), \dots, (x_{\boldsymbol{T}-1}, x_{\boldsymbol{T}})$. Note that $x_{\boldsymbol{T}}$ is uniformly distributed on $\{-1, 1\}^n$.

The Game now begins, with the current *time* being $t = 0$, the current *point* being $x_0 = (1, 1, \dots, 1)$, and the current *function* being $\widetilde{g} = g$. (The current function will always be $\pm g$.) At each time step $t = 0, 1, 2, \dots$, the Host asks whether the Contestant would like to *negate* the current function, meaning replace $\widetilde{g}$ with $-\widetilde{g}$. If the Contestant does not negate the current function there is no cost. However, if the Contestant elects to negate the current function, the Contestant must pay a cost of $w(t) := \frac{1}{(1-t/n)^2}$. After the Contestant makes the decision, the Host reveals to the Contestant what the $(t+1)$-th point on the secret half-path is, and increases the time by 1.

As soon as time $\boldsymbol{T}$ is reached, the Game ends. At this instant, if $\widetilde{g}(x_{\boldsymbol{T}}) \neq 1$, then the Contestant incurs a further cost of $w(\boldsymbol{T})$. (It's as though the Contestant is now obliged to negate $\widetilde{g}$.) Thus one can think of the Contestant's goal throughout the Game as trying to ensure that $\widetilde{g}(x_{\boldsymbol{T}})$ will equal 1, while trying to minimize the total cost incurred by all negations.

We define $\text{cost}(g)$ to be the least expected cost that a Contestant can achieve when the Game Show is played with Middle Function g. For $g : \{-1, 1\}^n \to \{-1, 1\}$ and "negation pattern" $b \in \{-1, 1\}^n$, we write g^{+b} to denote the function defined by $g^{+b}(x) = g(b_1x_1, \dots, b_nx_n)$.

Roughly speaking, our conjecture about the Game Show is that for every odd g, the average value of $\text{cost}(g^{+b})$ over all b is at least $\frac{3}{2}$. To be precise, we need to be concerned with averaging over merely pairwise-independent distributions on b.

▶ **Game Show Conjecture.** *Let $g : \{-1, 1\}^n \to \{-1, 1\}$ be odd and let $\mathcal{D}$ be any distribution on $\{-1, 1\}^n$ which is pairwise-independent and symmetric (meaning* $\Pr_{\mathcal{D}}[\boldsymbol{b}] = \Pr_{\mathcal{D}}[-\boldsymbol{b}]$*). Then* $\mathbf{E}_{\boldsymbol{b}\sim\mathcal{D}}[\text{cost}(g^{+\boldsymbol{b}})] \geq \frac{3}{2} - o_n(1)$.

Our motivation for making the Game Show Conjecture is the following result:

▶ **Theorem 47.** *Suppose the Game Show Conjecture is true. Then it is* NP*-hard to approximate* 2-Lin(2) *(and hence also* Max-Cut*) to factor* $\frac{3}{2} - \delta$ *for any* $\delta > 0$.

The proof of the above theorem can be found in the complete version of the paper.

We remark that given a Middle Function g, in some sense it is "easy" to determine the Contestant's best strategy. It can done with a dynamic program, since the Game Show is essentially a 2-Lin(2) instance on a tree graph. Nevertheless, we have been unable to prove the conjecture. A discussion about some of our efforts in proving the conjecture can be found in the full paper.

5 Limitations of gadget reductions

In this section, we show a limitation to proving inapproximability using gadget reductions from balanced pairwise-independent predicates: that is, predicates ϕ that admit a set $S \subseteq \mathsf{sat}(\phi)$ satisfying Property 2 in Definition 12. We show that gadget reductions from ϕ to 2-Lin(2) can not prove inapproximability larger than a factor-2.54 for the deletion version. Note that this applies to the Had_k predicates and to a broader class of predicates that do not necessarily admit a natural group operation.

▶ **Theorem 48.** *Let $\mathcal{G}$ be a (c, s)-generic gadget reducing Max-ϕ to Max-*2-Lin(2)*, where ϕ admits a balanced pairwise-independent set. Then*

$$\frac{s}{c} \le \frac{1}{1 - e^{-1/2}} \approx 2.54.$$

Proof. As before, K is the number of satisfying assignments of ϕ. Recall that the vertex set of $\mathcal{G}$ is $V = \{-1, 1\}^K$. Further, via Propositions 27 and 28, we need only consider folded assignments to these variables, and we can assume $\mathcal{G}$ only uses $(=)$-constraints. Finally, via Proposition 35, we can assume that every $(=)$-constraint used by $\mathcal{G}$ is between two variables x and y which are Hamming distance one from each other. Let P be the set of generic primary variables, let $-P$ be their negations, and let $P^{\pm} = P \cup (-P)$ denote the union of the two. Since ϕ is balanced pairwise-independent, we have a set $S \subseteq [K]$ so that for i picked uniformly at random from S, $\Pr_i[u_i = v_i] = 1/2$ for distinct primary variables $u, v \in P$.

Define the similarity between x and y to be $\mathsf{sim}(x, y) := \Pr_i[x_i = y_i]$ and set $\mathsf{sim}(x, P^{\pm}) := \max_{y \in P^{\pm}} \mathsf{sim}(x, y)$. Pairwise-independence allows us to claim that any variable x is strongly similar (i.e. has similarity $> \frac{3}{4}$) with at most one variable $y \in P^{\pm}$; define y to be x's *closest* primary variable.

▶ **Fact 49.** *For any $x \in V$, if $\mathsf{sim}(x, y) > \frac{3}{4}$ for some $y \in P^{\pm}$, then $\mathsf{sim}(x, y') < \frac{3}{4}$ for all other $y' \in P^{\pm}$.*

Proof. If x has $\mathsf{sim}(x, y_1) > \frac{3}{4}$ and $\mathsf{sim}(x, y_2) \ge \frac{3}{4}$ for $y_1, y_2 \in P^{\pm}$, then

$$\mathsf{sim}(y_1, y_2) \ge \mathsf{sim}(y_1, x) + \mathsf{sim}(x, y_2) - 1 > \frac{1}{2},$$

contradicting the assumption on ϕ. ◀

This fact allows us to design the following "threshold-rounding" procedure to construct a distribution $\mathcal{A} \in \mathsf{R}^{gen}(\phi)$. Let $C = \frac{2}{e^2 - e^{3/2}}$, and $\mathcal{D}$ be a distribution over $[3/4, 1]$ with probability density function $\mathcal{D}(t) = C \cdot e^{2t}$, for $t \in [3/4, 1]$.

1. Pick a random assignment to the primary variables.
2. Pick a number $t \sim \mathcal{D}$. For any variable $x \in V$, call x type 1 if $\mathsf{sim}(x, P^{\pm}) > t$ and type 2 otherwise.

3. Assign all type-1 variables the value of their closest primary variable.
4. Pick a uniformly random dictator d_i and set all the type-2 variables to agree with this dictator.
5. Output the resulting assignment.

Note that the assignments are folded and are random on the primary variables. We analyse the performance of this assignment. Let (x, y) be an edge in $\{-1,1\}^K$ of Hamming weight one. If both $\mathsf{sim}(x, P^{\pm}), \mathsf{sim}(y, P^{\pm}) \le \frac{3}{4}$, then regardless of the value of t, x and y will both always be type-2 variables, in which case $\mathcal{A}$ violates the edge between them with the probability of a random dictator, which is $\frac{1}{K} \le \frac{1}{1-e^{-1/2}} \cdot \frac{1}{K}$.

On the other hand, suppose WLOG that $\mathsf{sim}(x, P^{\pm}) > \mathsf{sim}(y, P^{\pm})$ and that $\mathsf{sim}(x, P^{\pm}) > \frac{3}{4}$. If we set $s := \mathsf{sim}(y, P^{\pm})$, then $\mathsf{sim}(x, P^{\pm}) = s + \frac{1}{K}$. Because y is distance one from x, $s \ge \frac{3}{4}$. Not only that, if y has a closest primary variable, then that variable is the same as x's closest primary variable (this is by Fact 49). Now, to calculate the probability that $\mathcal{A}$ violates (x, y), there are three cases:

1. If $t \in \left[\frac{3}{4}, s\right)$, then x and y are assigned the value of the same variable in $P^{\pm}$, so (x, y) is never violated in this case.
2. If $t \in \left[s, s + \frac{1}{K}\right)$, then y's value is chosen according to a uniformly random dictator assignment, meaning that it is a uniformly random ±1-bit. independent from x's value In this case, (x, y) is violated with probabiltiy $\frac{1}{2}$.
3. If $t \in \left[s + \frac{1}{K}, 1\right]$, then both x and y are assigned values according to a random dictator, in which case (x, y) is violated with probability $\frac{1}{K}$.

In total,

$$\begin{aligned}\Pr[\mathcal{A} \text{ violates } (x,y)] &= \frac{1}{2} \cdot \Pr_{t\sim\mathcal{D}}[t \in [s, s+1/K)] + \frac{1}{K} \cdot \Pr_{t\sim\mathcal{D}}[t \in [s+1/K, 1)] \\ &= \frac{1}{2}\int_s^{s+\frac{1}{K}} Ce^{2t}\mathrm{d}t + \frac{1}{K}\int_{s+\frac{1}{K}}^{1} Ce^{2t}\mathrm{d}t \\ &\le \frac{1}{2} \cdot \frac{Ce^{2s+2/K}}{K} + \frac{1}{K}\int_{s+\frac{1}{K}}^{1} Ce^{2t}\mathrm{d}t \\ &= \frac{Ce^2}{2K} = \frac{1}{1-e^{-1/2}} \cdot \frac{1}{K},\end{aligned}$$

as promised. Here the inequality follows from the fact that e^{2t} is an increasing function. As $\mathcal{G}$ only uses length-one edges, $c = \frac{1}{K}$. We have just shown that $\mathsf{uval}(\mathcal{A}; \mathcal{G}) \le \frac{1}{1-e^{-1/2}} \cdot \frac{1}{K}$. Because $\mathcal{A} \in \mathsf{R}^{gen}(\phi)$, we conclude that $\frac{s}{c} \le \frac{1}{1-e^{-1/2}}$. ◀

6 Conclusion

As mentioned, we view our factor-$\frac{11}{8}$ NP-hardness result more as a proof of concept, illustrating that the longstanding barrier of factor-$\frac{5}{4}$ NP-hardness for Max-Cut/2-Lin(2)/Unique-Games can be broken. There are quite a few avenues for further work:

- Derive a better NP-hardness result for 2-Lin(2) by reduction from Had_4. As one can always embed a Had_3-based gadget into a Had_4-based gadget, this will always yield a hardness of at least $\frac{11}{8}$. But presumably the optimal Had_4-based gadget will do slightly better.
- Since our analysis of the optimal Had_3 gadget is already somewhat complicated, it might be challenging to analyze the Had_4 case explicitly. A weaker but more plausible goal would be to prove (perhaps indirectly) that there *exists* a $\delta_0 > 0$ such that the optimal

Had_4 gadget achieves factor-$(\frac{11}{8}+\delta_0)$ NP-hardness. This would at least definitely establish that $\frac{11}{8}$ is not the "correct answer" either.

- Prove the Game Show Conjecture which would yield the improved NP-hardness factor of $\frac{3}{2}$. It may also be simpler to try to prove a non-optimal version of the conjecture, yielding *some* hardness factor better than $\frac{11}{8}$ but worse than $\frac{3}{2}$.
- For Max-Cut, our work establishes NP-hardness of $(\epsilon, C\epsilon)$-approximation for any $C < \frac{11}{8}$, but only for $\epsilon \leq \epsilon_0$ where ϵ_0 is some not-very-large constant (see full version for details). It would be nice to get a Max-Cut gadget yielding a larger ϵ_0, like the $\epsilon_0 = \frac{1}{8}$ we have for 2-Lin(2).
- A recent result of Gupta, Talwar, and Witmer [12] showed NP-hardness of approximating the (closely related) Non-Uniform Sparsest Cut problem to factor-$\frac{17}{16}$, by designing a gadget reduction from the old $(\frac{4}{21}, \frac{5}{21})$-approximation hardness of Håstad [13]. A natural question is whether one can use ideas from this paper to make a direct reduction from Had_2 or Had_3 to Non-Uniform Sparsest Cut, improving the NP-hardness factor of $\frac{17}{16}$.
- We are now in the situation (similar to the situation prior to [18]) wherein the best NP-hardness factor we know how to achieve for 2-Lin(q) (or Unique-Games) is achieved by taking $q = 2$. In fact, we don't know how to achieve an NP-hardness factor better than $\frac{5}{4}$ for 2-Lin(q) for any $q > 2$, even though 2-Lin(q) is presumably *harder* for larger q. Can this be remedied?
- In light of the limitations described in Section 5, it makes sense to seek alternative methodology of establishing improved NP-hardness for 2-CSPs. An example showing that this is not at all hopeless comes from the decade-old work of Chlebík and Chlebíková [8], which shows NP-hardness of approximating 2-Sat(-Deletion) to factor $8\sqrt{5} - 15 \approx 2.8885$. Their result is essentially a small tweak to the Vertex-Cover hardness of Dinur and Safra [9] and thus indeed uses a fairly radical methodology for establishing two-bit CSP-hardness, namely direct reduction from a specialized Label-Cover-type problem.

Acknowledgments. The authors would like to warmly thank Per Austrin for his assistance with computer analysis of the $\frac{11}{8}$-gadget.

References

1 Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $O(\sqrt{\log n})$ approximation algorithms for Min-Uncut, Min-2CNF-Deletion, and directed cut problems. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 573–581, 2005.

2 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for Unique Games and related problems. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, pages 563–572, 2010.

3 Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 222–231, 2004.

4 Per Austrin and Johan Håstad. On the usefulness of predicates. *ACM Trans. Comput. Theory*, 5(1):1:1–1:24, 2013.

5 Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and non-approximability – towards tight results. *SIAM Journal of Computing*, 27(3):804–915, 1998.

6 Siu On Chan. Approximation resistance from pairwise independent subgroups. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, pages 447–456, 2013.

7 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for Unique Games. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 205–214, 2006.

8 Miroslav Chlebík and Janka Chlebíková. On approximation hardness of the minimum 2SAT-DELETION problem. In *Proceedings of the 29th Annual International Symposium on Mathematical Foundations of Computer Science*, pages 263–273, 2004.

9 Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.

10 Uriel Feige and Daniel Reichman. On systems of linear equations with two variables per equation. In *Proceedings of the 7th Annual International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 117–127, 2004.

11 Michel Goemans and David Williamson. A 0.878 approximation algorithm for MAX-2SAT and MAX-CUT. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 422–431, 1994.

12 Anupam Gupta, Kunal Talwar, and David Witmer. Sparsest Cut on bounded treewidth graphs: algorithms and hardness results. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, pages 281–290, 2013.

13 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

14 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 767–775, 2002.

15 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for Max-Cut and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007.

16 Dana Moshkovitz and Ran Raz. Two-query PCP with subconstant error. *Journal of the ACM*, 57(5):29, 2010.

17 Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. *Annals of Mathematics*, 171(1):295–341, 2010.

18 Ryan O'Donnell and John Wright. A new point of NP-hardness for Unique-Games. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 289–306, 2012.

19 Anup Rao. Parallel repetition in projection games and a concentration bound. *SIAM Journal of Computing*, 40(6):1871–1891, 2011.

20 Luca Trevisan, Gregory Sorkin, Madhu Sudan, and David Williamson. Gadgets, approximation, and linear programming. *SIAM Journal on Computing*, 29(6):2074–2097, 2000.

21 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.

A Tight Approximation Bound for the Stable Marriage Problem with Restricted Ties

Chien-Chung Huang[1], Kazuo Iwama[2], Shuichi Miyazaki[3], and Hiroki Yanagisawa[4]

1 **Department of Computer Science and Engineering, Chalmers University of Technology**
Göteborg, Sweden
`huangch@chalmers.se`

2 **School of Informatics, Kyoto University**
Kyoto, Japan
`iwama@kuis.kyoto-u.ac.jp`

3 **Academic Center for Computing and Media Studies, Kyoto University**
Kyoto, Japan
`shuichi@media.kyoto-u.ac.jp`

4 **IBM Research - Tokyo, IBM Japan**
Tokyo, Japan
`yanagis@jp.ibm.com`

Abstract

The problem of finding a maximum cardinality stable matching in the presence of ties and unacceptable partners, called MAX SMTI, is a well-studied NP-hard problem. The MAX SMTI is NP-hard even for highly restricted instances where (i) ties appear only in women's preference lists and (ii) each tie appears at the end of each woman's preference list. The current best lower bounds on the approximation ratio for this variant are 1.1052 unless P=NP and 1.25 under the unique games conjecture, while the current best upper bound is 1.4616. In this paper, we improve the upper bound to 1.25, which matches the lower bound under the unique games conjecture. Note that this is the first special case of the MAX SMTI where the tight approximation bound is obtained. The improved ratio is achieved via a new analysis technique, which avoids the complicated case-by-case analysis used in earlier studies. As a by-product of our analysis, we show that the integrality gap of natural IP and LP formulations for this variant is 1.25. We also show that the unrestricted MAX SMTI cannot be approximated with less than 1.5 unless the approximation ratio of a certain special case of the minimum maximal matching problem can be improved.

1998 ACM Subject Classification G.2.1 Combinatorics

Keywords and phrases stable marriage with ties and incomplete lists, approximation algorithm, integer program, linear program relaxation, integrality gap

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.361

1 Introduction

The stable marriage problem [12, 27] is a classical combinatorial problem introduced by Gale and Shapley in their celebrated seminal paper [9]. An input of this problem includes two sets; a set of men and a set of women. Each man submits a preference list that orders women according to his preference, and similarly each woman submits her preference list. Given these lists, the problem is to find a stable matching, a matching without any *blocking pairs*,

Table 1 Four problem settings of MAX SMTI

Two-sided-ties case (2T)		One-sided-ties case (1T)	
$m_1 : w_2\ w_1$	$w_1 : m_1\ m_2$	$m_1 : w_2\ w_1$	$w_1 : m_1$
$m_2 : (w_1\ w_2)\ w_3$	$w_2 : m_1\ m_2$	$m_2 : w_2\ w_3$	$w_2 : (m_1\ m_2)\ m_3$
$m_3 : w_3$	$w_3 : (m_2\ m_3)$	$m_3 : w_3\ w_2$	$w_3 : m_2\ m_3$
Restricted two-sided-ties case (R2T)		**Restricted one-sided-ties case (R1T)**	
$m_1 : w_2\ w_1$	$w_1 : m_1\ m_2$	$m_1 : w_1\ w_3$	$w_1 : m_1\ m_3$
$m_2 : w_1\ (w_2\ w_3)$	$w_2 : m_1\ m_2$	$m_2 : w_2\ w_3$	$w_2 : m_2$
$m_3 : w_3$	$w_3 : (m_2\ m_3)$	$m_3 : w_3\ w_1$	$w_3 : m_2\ (m_1\ m_3)$

where a blocking pair is that of a man and a woman who might elope (a formal definition is given in Sec. 2). There are several variants of the problem in terms of the format of preference lists. In its most general setting, preference lists may contain ties (i.e., two or more men indifferent to woman w may be included in a tie of w's list), and may be incomplete (i.e., a man who is not acceptable to w is missing in her list). It is known that there exists at least one stable matching in any instance, but there may exist stable matchings of different sizes. The problem of finding a maximum cardinality stable matching in this setting (called the Maximum Stable Marriage problem with Ties and Incomplete lists, or MAX SMTI for short) is NP-hard [18, 28]; therefore, the approximability of the MAX SMTI has been intensively studied.

The MAX SMTI has been studied for various settings (see Table **??**). The most general setting is the two-sided-ties problem (2T), where ties may appear in both men's and women's lists. The second setting is the one-sided-ties problem (1T), where ties can appear only in women's lists (i.e., men's preference lists are strictly ordered and may not include ties). The third setting is the restricted one-sided-ties problem (R1T), where, in addition to the second setting, ties can appear only at the end of women's preference lists. The R1T was first studied by Irving and Manlove [17], inspired by an actual application for the Scottish Foundation Allocation Scheme (SFAS) [16]. The SFAS is designed to assign residents to hospitals under the condition that a resident (a man in our marriage case) submits a strictly-ordered preference list while a hospital (a woman) submits a preference list that may contain one tie of arbitrary length at the end of the list. In this paper, we also consider another natural setting, the restricted two-sided ties problem (R2T), which was not studied before. In R2T, ties can appear in both sexes' lists, but the position must be the end of the lists.

Let us review previous results on the approximability of the MAX SMTI for 2T, 1T, and R1T. It is easy to see that any algorithm that produces a stable matching is a 2-approximation algorithm, but the existence of a $(2 - \epsilon)$-approximation algorithm for the MAX SMTI is nontrivial. For 2T, after several attempts to obtain $(2 - o(1))$-approximation algorithms [19, 20], a 1.875-approximation algorithm was first presented in [21]. Later, Király [25] improved it to 1.6667 and McDermid [30] to 1.5, which is the current best approximation ratio. Recently, Király [26] and Paluch [31] presented simpler linear time algorithms with the same approximation ratio of 1.5. On the negative side, it is known that 1.1379-approximation is NP-hard and 1.3333-approximation is UG-hard via a reduction from the vertex cover problem [35]. (Here "UG-hard" means that there is no better approximation algorithm if the unique games conjecture [24] is true.) Also, an integrality gap for 2T is shown to be at least $1.5 - o(1)$ [22] for a natural integer programming formulation, which rules out the possibility of using some current techniques (e.g., rounding and primal-dual algorithms) to show $(1.5 - \epsilon)$-approximation.

Table 2 Upper and lower bounds on approximation ratio and lower bounds on integrality gap (new results discussed in this paper are in bold)

	2T	R2T	1T	R1T
Upper bounds on approximation ratio	1.5 [30]	1.5 [30]	1.4616 [5]	1.4616 [5] → **1.25**
Lower bounds on integrality gap	$1.5 - o(1)$ [22]	– → **1.3333**	1.3678 [22] → $\mathbf{1.5 - o(1)}$	– → **1.25**
UG lower bounds on approximation ratio	1.3333 [35]	– → **1.3333**	1.25 [13, 35]	1.25 [13, 35]
Lower bounds assuming MMM-Bi-APM is hard to approximate	– → **1.5**	–	–	–

It is still open whether there exists a $(1.5 - \epsilon)$-approximation algorithm for 2T and R2T, but this problem is resolved for 1T and R1T by providing a 1.4706-approximation algorithm that uses a linear programming (LP) relaxation [22]. Huang and Kavitha [15] improved the approximation ratio to 1.4667 by developing a linear time algorithm, and Radnai [32] tightened the analysis of this algorithm to show 1.4643-approximation. Very recently, Dean and Jalasutram [5] showed that the algorithm presented in [22] provides the current best approximation ratio 1.4616 through an analysis using the idea of factor-revealing LP. On the negative side of 1T and R1T, it is known that 1.1052-approximation is NP-hard and $(1.25 - \epsilon)$-approximation is UG-hard, via a reduction from the vertex cover problem [13, 35]. The integrality gap of the natural IP formulation is known to be at least 1.3678 for 1T [22], but there is no known lower bound on the integrality gap for R1T.

Our contributions

Our contributions (and previous results) are summarized in Table 2.

Our first contribution is to provide a tight upper bound of 1.25 for R1T. This is the first upper bound result for the MAX SMTI that matches a UG lower bound. Our algorithm is LP-based, which is almost the same as that in [22], but in this paper we introduce a novel analysis, which not only avoids the tedious case-analysis used in earlier studies, but also significantly improves the approximation ratio. In all previous analyses of the current algorithms, we create a bipartite graph G that is the union of two matchings M^* and M, where M^* is a largest stable matching and M is a stable matching obtained from an approximation algorithm (see Fig. 1). It is easy to see that the number of short paths in this graph is directly related to the approximation factor. Indeed, Király [25] showed that his algorithm does not create length-three paths for 1T, and McDermid [30] did the same for 2T, both achieving a 1.5 upper bound. The analyses of the algorithms discussed in [15, 22] bounded the number of length-five paths for 1T, which led to breaking the 1.5 barrier. Unfortunately, natural extension of this approach to longer paths seems to have a quick limit since there is no obvious way of getting rid of complicated case analysis. In fact, the current best bound for 1T [5], resulting in an improvement from 1.4643 to 1.4616, requires a computer-assisted proof to bound the numbers of length-seven and length-nine paths.

In this paper, we come back to a more direct and standard approach in the analysis of an LP-based approximation algorithm. That is, we just apply a formula relating the

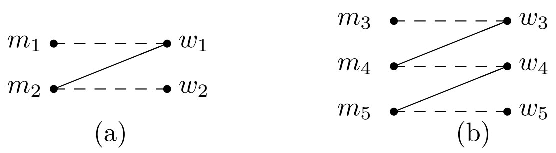


Figure 1 Illustrations of (a) a length-three path and (b) a length-five path in the union graph G, where solid edges represent pairs from M and dashed edges represent pairs from M^*.

LP-relaxed (optimum) value and the optimum integral value. A notable difference between our new analysis and the old one [22] is that we partition M into three sets of pairs based on M and M^* in the old analysis, whereas we do so based on M only in the new analysis. This difference allows us to avoid handling long paths (such as length-seven and length-nine paths). One might be curious about an extension of this approach to 1T or even 2T, which is our obvious future goal.

Our second contribution is to give a 1.3333-UG lower bound for R2T, which is obtained by modifying the reduction used to show the same UG lower bound for 2T [35]. This result implies that R2T is strictly harder than R1T under the unique games conjecture, while such a separation is unknown for 2T and 1T. For R2T, we also show that the integrality gap of the natural IP formulation is at least 1.3333.

Our third contribution is to show a new lower bound on the integrality gap for 1T. Specifically, we construct an instance of 1T whose integrality gap is at least $1.5 - o(1)$. This result suggests that the integrality gap of 1.5 for 2T is no longer a convincing bad sign for improving the current 1.5 upper bound because we already have a better approximation ratio for 1T (1.4616) than the integrality gap. Note that our new integrality gap of $1.5 - o(1)$ does not contradict the upper bounds of less than 1.5 [5, 22] since the technique used in the algorithms of these studies is not a simple LP rounding.

Our final contribution is to give support to the inapproximability of 2T by relating 2T to the minimum maximal matching problem (MMM). The MMM is a classical optimization problem, which asks us to find a maximal matching with minimum cardinality in a given undirected graph. This problem is known to be NP-hard even for a very restricted class of graphs (including bipartite graphs) [10, 14, 36]. It is also known that the MMM is equivalent to the minimum edge dominating set problem (MEDS) with respect to approximability [36], that is, there exists an α-approximation algorithm for the MMM if and only if there exists an α-approximation algorithm for the MEDS. So far, the approximability of the MMM (and equivalently that of the MEDS) has been extensively studied [2, 3, 8, 11, 29], but none achieved a $(2 - \epsilon)$-approximation for any constant $\epsilon > 0$ even on bipartite graphs. Regarding the inapproximability, the current best lower bound under P$\neq$NP is 7/6 for general graphs [4]. Based on the reduction in [18], we show that a $(1.5 - \epsilon)$-approximation algorithm for 2T for a constant $\epsilon > 0$ implies a $(2 - \epsilon')$-approximation algorithm for the MMM on bipartite graphs with an almost-perfect matching (which we call the MMM-Bi-APM) for a constant $\epsilon' > 0$. Note that there is no known $(2 - \epsilon')$-approximation algorithm, even for the MMM-Bi-APM.

2 Preliminaries

We now give notations, most of which are taken from [22]. An instance I of the MAX SMTI is composed of n men, n women, and each person's preference list that may be incomplete and may include ties. If a person p includes a person q (of the opposite sex) in p's preference list, we say that q is *acceptable* to p. Without loss of generality, we assume that a man m is acceptable to w if and only if w is acceptable to m. A matching M is a set of pairs (m, w)

such that m is acceptable to w and vice versa, and each person appears at most once in M. If $(m, w) \in M$, we say that m (w) is *matched in* M, and write $M(m) = w$ and $M(w) = m$. If p does not appear in M, we say that p is *single in* M. If m strictly prefers w_i to w_j, we write $w_i \succ_m w_j$. If w_i and w_j are tied in m's list (including the case in which $w_i = w_j$), we write $w_i =_m w_j$. The statement $w_i \succeq_m w_j$ is true if and only if $w_i \succ_m w_j$ or $w_i =_m w_j$. We use similar notations for women's preference lists. We say that m and w form a *blocking pair* for a matching M (or simply, (m, w) *blocks* M) if the following three conditions are met: (i) $M(m) \neq w$ but m and w are acceptable to each other, (ii) $w \succ_m M(m)$ or m is single in M, and (iii) $m \succ_w M(w)$ or w is single in M. A matching M is called *stable* if there is no blocking pair for M.

The MAX SMTI is the problem of finding the largest stable matching. The following IP formulation of MAX SMTI instance I, denoted as $IP(I)$, is a generalization of the one for the original stable marriage problem given in [33, 34]. For each pair (m, w), we introduce a binary variable $x_{m,w}$.

$$\begin{aligned}
&\text{Maximize:} && \sum_i \sum_j x_{i,j} && \\
&\text{Subject to:} && \sum_i x_{i,w} \leq 1 && \forall w && (1)\\
& && \sum_j x_{m,j} \leq 1 && \forall m && (2)\\
& && \sum_{j \succeq_m w} x_{m,j} + \sum_{i \succeq_w m} x_{i,w} - x_{m,w} \geq 1 && \forall (m,w) \in A && (3)\\
& && x_{m,w} = 0 && \forall (m,w) \notin A && (4)\\
& && x_{m,w} \in \{0,1\} && \forall (m,w) && (5)
\end{aligned}$$

Here, A is the set of mutually acceptable pairs, that is, $(m, w) \in A$ if and only if m and w are acceptable to each other. In this formulation, "$x_{m,w} = 1$" is interpreted as "m and w are matched," and "$x_{m,w} = 0$" otherwise. Thus the objective function is equal to the size of a matching. Note that Constraint (3) ensures that (m, w) is not a blocking pair. When $x_{m,w} = 1$, all three terms of the left-hand side are 1; hence, Constraint (3) is satisfied. When $x_{m,w} = 0$, either the first or the second term of the left-hand side must be 1, which implies that m (respectively w) must be matched with a partner as good as w (respectively m). The notation $LP(I)$ denotes the linear program relaxation of $IP(I)$ in which Constraint (5) is replaced with "$0 \leq x_{m,w} \leq 1$."

3 Approximation Algorithm for R1T

3.1 Algorithm GSA-LP

We now describe our approximation algorithm GSA-LP for instance I in which the men's lists are strict and the women's lists may contain ties. This algorithm is a simpler version of the algorithm given in [22], whose pseudo-code is given in Algorithm 1. In this algorithm, we maintain a variable p_m (initially set to one), which stores the current position for m in his preference list, and another priority value f_m (initially set to zero) for each m.

The GSA-LP algorithm consists of a sequence of proposals from men to women, as the standard Gale-Shapley algorithm. When a woman receives proposals from two men, she keeps the better one and rejects the other. If two men are in the same tie, the woman chooses

Algorithm 1 GSA-LP (Gale-Shapley Algorithm with LP solution)

```
Input: An SMTI instance $I$
Output: A matching $M$
 1: Formulate the given instance $I$ as an integer program $IP(I)$
 2: Solve its LP relaxation $LP(I)$ and obtain an optimal solution $x^*(= \{x^*_{i,j}\})$
 3: Let $M := \emptyset$
 4: Set $f_m := 0$ and $p_m := 1$ for each man $m$
 5: while there exists an $m$ such that ($m$ is single in $M$) and ($f_m \leq 3$) do
 6:    Let $m$ be an arbitrary such man
 7:    if $p_m$ is no larger than the length of $m$'s preference list then
 8:       Let $w$ be the $p_m$-th woman of $m$'s preference list
 9:       if $m$ has not proposed to $w$ yet then
10:          Set $f_m := f_m + x^*_{m,w}$ and $p_m := 1$
11:       else
12:          Set $p_m := p_m + 1$
13:       end if
14:       // Let $m$ propose to $w$
15:       if $w$ is single in $M$ then
16:          Set $M := M \cup \{(m, w)\}$
17:       else if $m \succ_w M(w)$ or ($m =_w M(w)$ and $f_m > f_{M(w)}$) then
18:          Set $M := M \cup \{(m, w)\} \setminus \{(M(w), w)\}$
19:       end if
20:    else
21:       Set $f_m := f_m + 2$ and $p_m := 1$ // $m$ goes to the second round
22:    end if
23: end while
24: return $M$
```

the man with the larger priority value f_m (if two values are the same, she keeps the current partner).

Intuitively, there are two rounds of proposals for each m. In the first round, whenever m sends a proposal to w for the first time, the priority value f_m is increased by $x^*_{m,w}$ (Lines 9–10). When he is rejected by w (either immediately or later after once accepted), he restarts his sequence of proposals from the top of his list. Note that in the restarted sequence of proposals, women he proposes to are not new until w. Up to that woman, f_m does not change and the restart does not happen. If m has proposed to all of the women in his list and he is still single, then f_m increases by 2 and m goes to his second round (Lines 20–21). In the second round, m sends a sequence of proposals from the top of his list again. Meanwhile, f_m does not change and restart never happens (that is, m sends at most one proposal to each woman in the second round). It is not hard to see that this algorithm runs in polynomial time and outputs a stable matching based on a similar argument in [22].

This algorithm is designed so that, if w is eventually matched to m at the termination of the algorithm, and there is another man m' who proposed to w at least once and is tied with m in her preference list (i.e., $m =_w m'$), then we have the following inequality

$$f_{m'} = \sum_{j' \succ_{m'} w, j' \notin S} x^*_{m',j'} \leq \sum_{j \succeq_m l(m), j \notin S} x^*_{m,j} = f_m, \tag{6}$$

where $l(m)$ denotes the woman least preferred by m among those who have received a proposal from m during the algorithm. Here, f_m and $f_{m'}$ refer to the values at the termination of the algorithm. Also note that, when m proposes to the woman at the end of his list for the first time, $f_m \leq 1$. If m is single in M at the termination of the algorithm, he has proposed to all the women in his list, and $f_m > 1$ holds.

3.2 Analysis of Approximation Ratio

3.2.1 Overview of Analysis

Our analysis is similar to the LP-based analysis used in [22], but the major difference between these two approaches is that our analysis does not use the bipartite graph on which older analyses (e.g., [15, 22, 25, 30]) heavily rely. Let us fix an instance I, and let M be the stable matching output from GSA-LP. We partition M into P, R, and T. Specifically, P is the set of pairs $(m, w) \in M$ such that $f_m > 1$ at the end of the algorithm, T is the set of pairs $(m, w) \in M$ such that $f_m \leq 1$, w has a tie, and m is contained in her tie, and $R = M \setminus (P \cup T)$. Let S be the set of men and women who are single in M.

Now we analyze the approximation ratio of GSA-LP under the assumption that ties can appear only at the end of women's preference lists. Recall that $x^*_{i,j}$ is the value of $x_{i,j}$ for the optimum solution x^* of $LP(I)$. Note that if $x^*_{m,w} > 0$ for $m, w \in S$, then $(m, w) \in A$ by Constraint (4) of $LP(I)$, so (m, w) is a blocking pair for M. This contradicts the stability of M; hence, $\sum_{i,j \in S} x^*_{i,j} = 0$. Now, let us define the value $x^*(X)$ for a subset $X \subseteq M$ as:

$$x^*(X) = \sum_{(m,w)\in X} \left(\sum_j x^*_{m,j} + \sum_i x^*_{i,w} + \sum_{j \in S} x^*_{m,j} + \sum_{i \in S} x^*_{i,w} \right).$$

It is not difficult to see that $x^*(P) + x^*(R) + x^*(T) = 2 \sum_i \sum_j x^*_{i,j}$, since $\sum_{i,j \in S} x^*_{i,j} = 0$. Note that $|M^*|$ and $\sum_i \sum_j x^*_{i,j}$ are the optimal values for $IP(I)$ and $LP(I)$ respectively, where M^* is an optimal solution of I (that is, one of the maximum stable matchings of I). Hence we have that $|M^*| \leq \sum_i \sum_j x^*_{i,j} = (x^*(P) + x^*(R) + x^*(T))/2$. We later prove the following key lemma.

▶ **Lemma 1.** $x^*(P) + x^*(R) + x^*(T) \leq \frac{5}{2}(|P| + |R| + |T|)$.

From this, we have that $|M^*| \leq (x^*(P) + x^*(R) + x^*(T))/2 \leq \frac{5}{4}(|P| + |R| + |T|) = \frac{5}{4}|M|$, and Theorem 2 follows:

▶ **Theorem 2.** *The approximation ratio of GSA-LP is at most $5/4$ for R1T.*

Remarks on Integrality Gap

The proof of Theorem 2 implies

$$\sum_i \sum_j x^*_{i,j} \leq \frac{5}{4}|M| \leq \frac{5}{4}|M^*|,$$

and this means that the integrality gap of $IP(I)$ is at most $5/4$. This result is tight because the integrality gap of $IP(I)$ is at least $5/4$, which is shown in Threorem 17.

3.2.2 Proof Sketch of Lemma 1

For readability, we first give a simpler proof of Lemma 1 for a special case in which two conditions (which we explain shortly) hold. The full proof (without conditions) is included in Sec. 3.2.3. We first define the following symbols (see also Fig. 2).

$$\begin{array}{ll}
m_p : \underbrace{\cdots w_p \cdots}_{\pi} & w_p : \underbrace{\cdots m_p \cdots}_{p} \\[1em]
m_r : \underbrace{\cdots}_{\rho}\underbrace{w_r \cdots}_{\overline{\rho}} & w_r : \underbrace{\cdots m_r}_{r}\underbrace{\cdots}_{\overline{r}} \\[1em]
m_t : \underbrace{\cdots w_t \cdots l(m_t)}_{\tau}\underbrace{\cdots}_{\overline{\tau}} & w_t : \underbrace{\cdots\cdots(\cdots m_t \cdots)}_{t}
\end{array}$$

Figure 2 Illustrations of symbols π, ρ, $\overline{\rho}$, τ, $\overline{\tau}$, p, r, $\overline{r}$, and t for pairs $(m_p, w_p) \in P$, $(m_r, w_r) \in R$, and $(m_t, w_t) \in T$.

$\pi = \{(m, j) \in A \mid (m, w) \in P, j \notin S\}$,
$p = \{(i, w) \in A \mid (m, w) \in P, i \notin S\}$,
$\rho = \{(m, j) \in A \mid (m, w) \in R, j \succ_m w, j \notin S\}$,
$\overline{\rho} = \{(m, j) \in A \mid (m, w) \in R, w \succeq_m j, j \notin S\}$,
$r = \{(i, w) \in A \mid (m, w) \in R, i \succeq_w m, i \notin S\}$,
$\overline{r} = \{(i, w) \in A \mid (m, w) \in R, m \succ_w i, i \notin S\}$,
$\tau = \{(m, j) \in A \mid (m, w) \in T, j \succeq_m l(m), j \notin S\}$,
$\overline{\tau} = \{(m, j) \in A \mid (m, w) \in T, l(m) \succ_m j, j \notin S\}$, and
$t = \{(i, w) \in A \mid (m, w) \in T, i \notin S\}$.

For $X \in \{\pi, \rho, \overline{\rho}, \tau, \overline{\tau}, p, r, \overline{r}, t\}$, $Y \in \{\pi, \rho, \overline{\rho}, \tau, \overline{\tau}\}$, and $Z \in \{p, r, \overline{r}, t\}$, we define $\sigma(X)$ and $\sigma(Y, Z)$ as

$$\sigma(X) = \sum_{(m,w)\in X} x^*_{m,w} \quad\text{and}\quad \sigma(Y, Z) = \sum_{(m,w)\in Y\cap Z} x^*_{m,w}.$$

The two conditions we use in this section are (I) $P = \emptyset$ and (II) $\sigma(\rho)/|R| \leq \sigma(\tau)/|T|$. Condition (II) is introduced to avoid using Inequality (6) in the analysis, which can make the proof significantly simpler. Inequality (6) implies that we have $f_{m'} \leq f_m$ for any $(m, w) \in T$ and $(m', w') \in R$ such that both m and m' have proposed to w during the course of GSA-LP. The intuitive meaning of Condition (II) is that $f_{m'} \leq f_m$ holds *on average* if we choose $(m, w) \in T$ and $(m', w') \in R$ uniformly at random. Note that $f_m \approx \sigma(\tau)/|T|$ and $f_{m'} \approx \sigma(\rho)/|R|$ for m and m' selected in this manner. Note also that Condition (II) does not hold in general because, in its interpretation, we do not guarantee that m' proposes to w (which is the case for Inequality (6)).

First, we prove several useful lemmas. Note that the claims similar to these lemmas are already proven in [22]. Lemma 3 is immediate from the definition since $\sigma(\pi) = \sigma(p) = 0$ from Condition (I). Lemma 4 also holds under Condition (I), but Lemmas 5–7 hold without any condition. Among these lemmas, Lemma 7 plays a key role in the analysis for R1T because it uses the restriction that each woman can contain a tie at the end of her preference list.

▶ **Lemma 3.** *(Under Condition (I))* $\sigma(\rho) + \sigma(\overline{\rho}) + \sigma(\tau) + \sigma(\overline{\tau}) = \sigma(r) + \sigma(\overline{r}) + \sigma(t)$.

▶ **Lemma 4.** *(Under Condition (I))* $\sigma(r) = \sigma(\overline{\rho}, r) + \sigma(\overline{\tau}, r)$.

Proof. By definition and Condition (I), we have $\sigma(r) = \sigma(\rho, r) + \sigma(\overline{\rho}, r) + \sigma(\tau, r) + \sigma(\overline{\tau}, r)$. To prove this lemma, we show that $\sigma(\rho, r) + \sigma(\tau, r) = 0$. If this does not hold, then there is a pair $(i, j) \in (\rho \cup \tau) \cap r$. Such (i, j) satisfies $M(j) \neq i$ by the definitions of ρ, τ, and r. Also, $i =_j M(j)$ holds because $(i, j) \in r$ means that $i \succeq_j M(j)$ and $(i, j) \in \rho \cup \tau$ means that i must have proposed to j and woman j rejected the proposal, which implies $M(j) \succeq_j i$. However, $(M(j), j) \in R$ means that $M(j)$ is not included in j's tie; a contradiction. ◀

▶ **Lemma 5.** *For any* $(m,w)\in M$*,*

$$\sum_{j\succ_m w, j\in S} x^*_{m,j}=0 \quad \textit{and} \quad \sum_{i\succ_w m, i\in S} x^*_{i,w}=0.$$

Proof. For the left equation, suppose that there is a woman j such that $j\in S$ and $j\succ_m w$. Then j must include m in her list. Hence, (m,j) blocks M, which contradicts the stability of M. Therefore, no such j exists; hence, $\sum_{j\succ_m w, j\in S} x^*_{m,j}$ actually sums up an empty set of variables. The right equation can also be validated in a similar way. ◀

▶ **Lemma 6.** $\sigma(\rho)+\sigma(r)\geq |R|$.

Proof. For each $(m,w)\in R$, we have

$$\sum_{j\succ_m w, j\notin S} x^*_{m,j}+\sum_{i\succeq_w m, i\notin S} x^*_{i,w}=\sum_{j\succ_m w} x^*_{m,j}+\sum_{i\succeq_w m} x^*_{i,w}\geq 1.$$

The equality comes from Lemma 5 and the fact that m is not in a tie of w's list because $(m,w)\in R$. The inequality comes from Constraint (3) of LP formulation. By adding this inequality for all $(m,w)\in R$, we have $\sigma(\rho)+\sigma(r)\geq|R|$. ◀

▶ **Lemma 7.** *For any* $(m,w)\in T$*,*

$$\sum_{i\in S} x^*_{i,w}=0.$$

Proof. Since $(m,w)\in T$, there is no man i such that $m\succ_w i$ because m is included in w's tie, which is located at the end of the list. If $i\in S$ and $i\succ_w m$, then $x^*_{i,w}=0$ by Lemma 5. We show that there is no i such that $i\in S$ and $i=_w m$. Suppose on the contrary that $i(\in S)$ and m are tied in w's list. Since i is single in M, i must have proposed to w with $f_i>1$. By the definition of T, $f_m\leq 1$ when m proposed to w. Therefore, it is impossible that m, rather than i, is matched with w in M; a contradiction. This completes the proof. ◀

Now we are ready to give the proof of Lemma 1. Recall that $P=\emptyset$ by Condition (I); hence, $x^*(P)=0$. Our goal here is $x^*(R)+x^*(T)\leq\frac{5}{2}(|R|+|T|)$. If $\sigma(\rho)>|R||T|/(|R|+|T|)$, then we have

$$\begin{aligned}
x^*(R)+x^*(T) &= \sum_{(m,w)\in R}\left(2\sum_j x^*_{m,j}+2\sum_i x^*_{i,w}-\sum_{j\notin S}x^*_{m,j}-\sum_{i\notin S}x^*_{i,w}\right)\\
&\quad+\sum_{(m,w)\in T}\left(2\sum_j x^*_{m,j}+2\sum_i x^*_{i,w}-\sum_{j\notin S}x^*_{m,j}-\sum_{i\notin S}x^*_{i,w}\right)\\
&\leq 4|R|+4|T|-\sigma(\rho)-\sigma(\overline{\rho})-\sigma(\tau)-\sigma(\overline{\tau})-\sigma(r)-\sigma(\overline{r})-\sigma(t)\\
&= 4|R|+4|T|-2\sigma(\rho)-2\sigma(\overline{\rho})-2\sigma(\tau)-2\sigma(\overline{\tau}) && \text{(by Lemma 3)}\\
&\leq 4|R|+4|T|-2\sigma(\rho)-2\sigma(\overline{\rho},r)-2\sigma(\tau)-2\sigma(\overline{\tau},r)\\
&= 4|R|+4|T|-2\sigma(\rho)-2\sigma(\tau)-2\sigma(r) && \text{(by Lemma 4)}\\
&\leq 2|R|+4|T|-2\sigma(\tau) && \text{(by Lemma 6)}\\
&\leq 2|R|+4|T|-2\frac{|T|}{|R|}\sigma(\rho) && \text{(by Condition (II))}\\
&< 2|R|+4|T|-\frac{2|T|^2}{|R|+|T|} && \text{(since } \sigma(\rho)>|R||T|/(|R|+|T|))\\
&\leq \frac{5}{2}(|R|+|T|).
\end{aligned}$$

Otherwise (i.e., $\sigma(\rho) \le |R||T|/(|R|+|T|)$), then we have

$$\begin{aligned}
x^*(R)+x^*(T) &= \sum_{(m,w)\in R}\left(2\sum_j x^*_{m,j} + 2\sum_i x^*_{i,w} - \sum_{j\notin S} x^*_{m,j} - \sum_{i\notin S} x^*_{i,w}\right) \\
&\quad + \sum_{(m,w)\in T}\left(2\sum_j x^*_{m,j} + \sum_{i\notin S} x^*_{i,w} - \sum_{j\notin S} x^*_{m,j}\right) && \text{(by Lemma 7)} \\
&\le 4|R| - \sigma(\rho) - \sigma(\overline{\rho}) - \sigma(r) - \sigma(\overline{r}) + 2|T| + \sigma(t) - \sigma(\tau) - \sigma(\overline{\tau}) \\
&= 4|R| + 2|T| - 2\sigma(r) - 2\sigma(\overline{r}) && \text{(by Lemma 3)} \\
&\le 4|R| + 2|T| - 2(|R| - \sigma(\rho)) && \text{(by Lemma 6)} \\
&\le 2|R| + 2|T| + \frac{2|R||T|}{|R|+|T|} && \text{(since } \sigma(\rho) \le /R//T//(/R/+/T/)) \\
&\le \frac{5}{2}(|R|+|T|).
\end{aligned}$$

◀

3.2.3 Full Proof of Lemma 1

In this section, we give a full proof of Lemma 1 (without Conditions (I) and (II)). Recall that in Sec. 3.2.2, we defined nine symbols such as π and p. For the full proof, we need three more symbols: For each $(m,w) \in T$, let

$\tau_m = \{(m,j) \in A \mid j \succeq_m l(m), j \notin S\}$,
$\overline{\tau_m} = \{(m,j) \in A \mid l(m) \succ_m j, j \notin S\}$, and
$t_w = \{(i,w) \in A \mid i \notin S\}$.

The following Lemmas 8–10 are unconditional counterparts of Lemmas 3 and 4 in Sec. 3.2.2.

▶ **Lemma 8.** (i) *For any w, $\sigma(t_w) = \sigma(\pi,t_w) + \sigma(\rho,t_w) + \sigma(\overline{\rho},t_w) + \sigma(\tau,t_w) + \sigma(\overline{\tau},t_w)$ and* (ii) *$\sigma(t) = \sigma(\pi,t) + \sigma(\rho,t) + \sigma(\overline{\rho},t) + \sigma(\tau,t) + \sigma(\overline{\tau},t)$.*

Proof. By definition, $t_w = (\pi \cap t_w) \cup (\rho \cap t_w) \cup (\overline{\rho} \cap t_w) \cup (\tau \cap t_w) \cup (\overline{\tau}, t_w)$, and the five intersections in the right-hand side are mutually disjoint. This proves (i). (ii) can be proved similarly. ◀

▶ **Lemma 9.** (i) *$\sigma(\overline{r}) \ge \sigma(\pi,\overline{r}) + \sigma(\rho,\overline{r}) + \sigma(\tau,\overline{r})$.* (ii) *$\sigma(\overline{\rho}) \ge \sigma(\overline{\rho},r) + \sigma(\overline{\rho},t)$.* (iii) *$\sigma(\overline{\tau_m}) \ge \sigma(\overline{\tau_m},r)$.* (iv) *$\sigma(p) \ge \sigma(\pi,p) + \sigma(\rho,p) + \sigma(\tau,p)$.* (v) *$\sigma(\tau_m) \ge \sigma(\tau_m,t)$.* (vi) *$\sigma(\overline{\tau_m}) \ge \sigma(\overline{\tau_m},r) + \sigma(\overline{\tau_m},t)$.*

Proof. (i) By definition, $\sigma(\overline{r}) = \sigma(\pi,\overline{r}) + \sigma(\rho,\overline{r}) + \sigma(\overline{\rho},\overline{r}) + \sigma(\tau,\overline{r}) + \sigma(\overline{\tau},\overline{r}) \ge \sigma(\pi,\overline{r}) + \sigma(\rho,\overline{r}) + \sigma(\tau,\overline{r})$. (ii)–(vi) can be proved similarly. ◀

▶ **Lemma 10.** (i) *$\sigma(\rho) = \sigma(\rho,p) + \sigma(\rho,\overline{r}) + \sigma(\rho,t)$.* (ii) *$\sigma(\pi) = \sigma(\pi,p) + \sigma(\pi,\overline{r}) + \sigma(\pi,t)$.* (iii) *$\sigma(\tau) = \sigma(\tau,p) + \sigma(\tau,\overline{r}) + \sigma(\tau,t)$.* (iv) *$\sigma(r) = \sigma(\overline{\rho},r) + \sigma(\overline{\tau},r)$.*

Proof. (i) By definition, $\sigma(\rho) = \sigma(\rho,p) + \sigma(\rho,r) + \sigma(\rho,\overline{r}) + \sigma(\rho,t)$. Now we show $\sigma(\rho,r) = 0$. If this does not hold, then there is a pair $(i,j) \in \rho \cap r$. Pair $(i,j) \in \rho$ implies that $(i,M(i)) \in R$ and $j \succ_i M(i)$, and $(i,j) \in r$ implies that $(M(j),j) \in R$ and $i \succeq_j M(j)$. Since $(M(j),j) \in R$, i and $M(j)$ are not tied in j's preference list; hence, $i \succ_j M(j)$. Since $j \succ_i M(i)$, i proposed to j during the algorithm. Hence, j must be matched with i or more a preferable man, which contradicts $i \succ_j M(j)$. Therefore, no such (i,j) exists and $\sigma(\rho,r) = 0$.

(ii) By definition, $\sigma(\pi) = \sigma(\pi, p) + \sigma(\pi, r) + \sigma(\pi, \overline{r}) + \sigma(\pi, t)$. We show that $\sigma(\pi, r) = 0$. If not, there are a man i and a woman j such that $(i, M(i)) \in P$, $(M(j), j) \in R$, and $i \succeq_j M(j)$. It is impossible that $i =_j M(j)$ because $i \neq M(j)$ since $(i, M(i)) \in P$ and $(M(j), j) \in R$, and i and $M(j)$ are not tied in j's preference list by the definition of R. Therefore $i \succ_j M(j)$. Also, by the definition of P, $f_i > 1$ at the end of the algorithm; hence, i must have proposed to all of the women in his list, especially, to j. Using a similar argument to the proof of part (i), we have a contradiction, implying that $\sigma(\pi, r) = 0$.

(iii) By definition, $\sigma(\tau) = \sigma(\tau, p) + \sigma(\tau, r) + \sigma(\tau, \overline{r}) + \sigma(\tau, t)$. By noting that m proposes to all the women from the top of the list to $l(m)$, we can show that $\sigma(\tau, r) = 0$ using a similar argument as the proofs of parts (i) and (ii).

(iv) By definition, $\sigma(r) = \sigma(\pi, r) + \sigma(\rho, r) + \sigma(\overline{\rho}, r) + \sigma(\tau, r) + \sigma(\overline{\tau}, r)$. We already proved that $\sigma(\pi, r) = 0$, $\sigma(\rho, r) = 0$, and $\sigma(\tau, r) = 0$. ◀

The following Lemma 11 is an elaborate version of Lemma 7.

▶ **Lemma 11.** (i) *For any $(m, w) \in P$, $\sum_{j \in S} x^*_{m,j} = 0$.* (ii) *For any $(m, w) \in T$, $\sum_{i \in S} x^*_{i,w} = 0$.*

Proof. (i) $(m, w) \in P$ implies that m has proposed to all the women in his list; hence, they are matched in M. Therefore, there is no (m, j) such that $(m, w) \in P$ and $j \in S$.

(ii) Since $(m, w) \in T$, there is no man i such that $m \succ_w i$ because w includes m in her tie at the end of her preference list. If $i \in S$ and $i \succ_w m$, then $x^*_{i,w} = 0$ by Lemma 5. We show that there is no i such that $i \in S$ and $i =_w m$. Suppose on the contrary that $i (\in S)$ and m are tied in w's list. Since i is single in M, i must have proposed to w with $f_i > 1$. By the definition of T, $f_m \leq 1$ when m proposed to w. Therefore, it is impossible that m, rather than i, is matched with w in M; a contradiction. This completes the proof. ◀

In the subsequent three lemmas, we bound $x^*(P)$, $x^*(R)$, and $x^*(T)$, which will lead to the proof of Lemma 15.

▶ **Lemma 12.** $x^*(P) \leq 2|P| + \sigma(\pi, \overline{r}) + \sigma(\pi, t) - \sigma(\rho, p) - \sigma(\tau, p)$.

Proof. We have

$$\begin{aligned}
x^*(P) &= \sum_{(m,w)\in P} \left(\sum_{j} x^*_{m,j} + \sum_{i} x^*_{i,w} + \sum_{j \in S} x^*_{m,j} + \sum_{i \in S} x^*_{i,w} \right) \\
&= \sum_{(m,w)\in P} \left(\sum_{j \notin S} x^*_{m,j} + 2\sum_{i} x^*_{i,w} - \sum_{i \notin S} x^*_{i,w} \right) \\
&\leq \sigma(\pi) + 2|P| - \sigma(p) \\
&\leq \sigma(\pi) + 2|P| - \sigma(\pi, p) - \sigma(\rho, p) - \sigma(\tau, p) \\
&= 2|P| + \sigma(\pi, \overline{r}) + \sigma(\pi, t) - \sigma(\rho, p) - \sigma(\tau, p).
\end{aligned}$$

The first equality is the definition of $x^*(P)$, and the second equality is from Lemma 11(i). The first inequality is from the definitions of $\sigma(\pi)$ and $\sigma(p)$, and the fact that $\sum_{(m,w)\in P} \sum_i x^*_{i,w} \leq |P|$ by Constraint (1) of LP formulation. The second inequality is from Lemma 9(iv). The last equality is from Lemma 10(ii). ◀

▶ **Lemma 13.** $x^*(R) \leq 2|R| + \sigma(\rho) - \sigma(\overline{\rho}, t) + \sigma(\overline{\tau}, r) - \sigma(\overline{r})$.

Proof. We have

$$\begin{aligned}
x^*(R) &= \sum_{(m,w)\in R}\left(\sum_j x^*_{m,j} + \sum_i x^*_{i,w} + \sum_{j\in S} x^*_{m,j} + \sum_{i\in S} x^*_{i,w}\right)\\
&= \sum_{(m,w)\in R}\left(2\sum_j x^*_{m,j} + 2\sum_i x^*_{i,w} - \sum_{j\notin S} x^*_{m,j} - \sum_{i\notin S} x^*_{i,w}\right)\\
&\leq 2|R| + 2|R| - (\sigma(\rho)+\sigma(\overline{\rho})) - (\sigma(r)+\sigma(\overline{r}))\\
&\leq 2|R| + \sigma(\rho) - \sigma(\overline{\rho}, r) - \sigma(\overline{\rho}, t) + \sigma(r) - \sigma(\overline{r})\\
&= 2|R| + \sigma(\rho) - \sigma(\overline{\rho}, t) + \sigma(\overline{\tau}, r) - \sigma(\overline{r}).
\end{aligned}$$

The first equality is the definition of $x^*(R)$. The first inequality is from Constraints (1) and (2) of LP formulation and definitions of $\sigma(\rho)$, $\sigma(\overline{\rho})$, $\sigma(r)$, and $\sigma(\overline{r})$. The last inequality is from Lemmas 9(ii) and 6. The last equality is from Lemma 10(iv). ◀

▶ **Lemma 14.**

$$x^*(T) \leq 2|T| - \sigma(\overline{\tau}, r) - \sigma(t) + \sum_{(m,w)\in T} \min\{2-\sigma(\tau_m), 2\sigma(t_w) - (\sigma(\tau_m, t) + \sigma(\overline{\tau_m}, t))\}.$$

Proof. By definition,

$$x^*(T) = \sum_{(m,w)\in T}\left(\sum_j x^*_{m,j} + \sum_i x^*_{i,w} + \sum_{j\in S} x^*_{m,j} + \sum_{i\in S} x^*_{i,w}\right).$$

We will bound the quantity inside the parentheses in two ways. First, for each $(m,w)\in T$, we have

$$\begin{aligned}
&\sum_j x^*_{m,j} + \sum_i x^*_{i,w} + \sum_{j\in S} x^*_{m,j} + \sum_{i\in S} x^*_{i,w}\\
&\quad\leq 1 + 1 + (1 - \sigma(\tau_m) - \sigma(\overline{\tau_m})) + (1 - \sigma(t_w))\\
&\quad\leq 4 - (\sigma(\tau_m) + \sigma(\overline{\tau_m}, r)) - \sigma(t_w). \qquad (7)
\end{aligned}$$

The first inequality is from Constraints (1) and (2) of LP formulation and definitions of $\sigma(\tau_m)$, $\sigma(\overline{\tau_m})$, and $\sigma(t_w)$. The last inequality is from Lemma 9(iii).

Next, for each $(m,w)\in T$, we have

$$\begin{aligned}
&\sum_j x^*_{m,j} + \sum_i x^*_{i,w} + \sum_{j\in S} x^*_{m,j} + \sum_{i\in S} x^*_{i,w}\\
&\quad= 2\sum_j x^*_{m,j} + \sum_{i\notin S} x^*_{i,w} - \sum_{j\notin S} x^*_{m,j}\\
&\quad\leq 2 + \sigma(t_w) - (\sigma(\tau_m) + \sigma(\overline{\tau_m}))\\
&\quad\leq 2 + \sigma(t_w) - (\sigma(\tau_m, t) + \sigma(\overline{\tau_m}, r) + \sigma(\overline{\tau_m}, t)). \qquad (8)
\end{aligned}$$

The equality comes from Lemma 11(ii) and the last inequality is from (v) and (vi) of Lemma 9.

Combining Inequalities (7) and (8), we have

$$\begin{aligned}
x^*(T) &= \sum_{(m,w)\in T} \left(\sum_{j} x^*_{m,j} + \sum_{i} x^*_{i,w} + \sum_{j\in S} x^*_{m,j} + \sum_{i\in S} x^*_{i,w} \right) \\
&\le \sum_{(m,w)\in T} \min\{4 - (\sigma(\tau_m) + \sigma(\overline{\tau_m}, r)) - \sigma(t_w), \\
&\qquad\qquad 2 + \sigma(t_w) - (\sigma(\tau_m, t) + \sigma(\overline{\tau_m}, r) + \sigma(\overline{\tau_m}, t))\} \\
&= 2|T| - \sigma(\overline{\tau}, r) - \sigma(t) \\
&\quad + \sum_{(m,w)\in T} \min\{2 - \sigma(\tau_m), 2\sigma(t_w) - (\sigma(\tau_m, t) + \sigma(\overline{\tau_m}, t))\}.
\end{aligned}$$

◀

To simplify notation, let $\delta_{m,w} = 2(\sigma(\tau, t_w) + \sigma(\overline{\tau}, t_w) - \sigma(\tau_m, t) - \sigma(\overline{\tau_m}, t))$ for each $(m, w) \in T$.

▶ **Lemma 15.** $x^*(P) + x^*(R) + x^*(T) \le$

$$2|P| + 2|R| + 2|T| + \sum_{(m,w)\in T} \min\{2 - 2\sigma(\tau_m), 2\sigma(\pi, t_w) + 2\sigma(\rho, t_w) + \delta_{m,w}\}.$$

Proof. Starting from Lemmas 12, 13, and 14, we have the sequence of deformations of the formula. To help following the deformations, we give underlines to the terms that are used for the deformation.

$$\begin{aligned}
&x^*(P) + x^*(R) + x^*(T) \\
&\le 2|P| + 2|R| + 2|T| + (\sigma(\pi, \overline{r}) + \sigma(\pi, t) - \sigma(\rho, p) - \sigma(\tau, p)) \\
&\quad +(\sigma(\rho) - \sigma(\overline{\rho}, t)\underline{+\sigma(\overline{\tau}, r)} - \sigma(\overline{r})) \\
&\quad \underline{-\sigma(\overline{\tau}, r)} - \sigma(t) + \sum_{(m,w)\in T} \min\{2 - \sigma(\tau_m), 2\sigma(t_w) - (\sigma(\tau_m, t) + \sigma(\overline{\tau_m}, t))\} \\
&= 2|P| + 2|R| + 2|T| + (\sigma(\pi, \overline{r}) + \sigma(\pi, t) - \sigma(\rho, p) - \sigma(\tau, p)) \\
&\quad +(\sigma(\rho)\underline{-\sigma(\overline{\rho}, t)} - \sigma(\overline{r})) \\
&\quad -\sigma(t) + \sum_{(m,w)\in T} \min\{2 - \sigma(\tau_m), 2\sigma(t_w) - (\sigma(\tau_m, t) + \sigma(\overline{\tau_m}, t))\} \\
&\le 2|P| + 2|R| + 2|T| + (\underline{\sigma(\pi, \overline{r}) + \sigma(\pi, t)} - \sigma(\rho, p) - \sigma(\tau, p)) + (\sigma(\rho)\underline{-\sigma(\overline{r})}) \\
&\quad \underline{-\sigma(t)} + \sum_{(m,w)\in T} \min\{2 - \sigma(\tau_m), 2\sigma(t_w) - (\sigma(\overline{\rho}, t_w) + \sigma(\tau_m, t) + \sigma(\overline{\tau_m}, t))\} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Use Lemmas 8(ii) and 9(i).)} \\
&\le 2|P| + 2|R| + 2|T| - \underline{\sigma(\rho, p)} - \sigma(\tau, p) + (\underline{\sigma(\rho) - \sigma(\rho, \overline{r})} - \sigma(\tau, \overline{r})) \\
&\quad -\sigma(\overline{\rho}, t)\underline{-\sigma(\rho, t)} - \sigma(\overline{\tau}, t) - \sigma(\tau, t) \\
&\quad + \sum_{(m,w)\in T} \min\{2 - \sigma(\tau_m), 2\sigma(t_w) - (\sigma(\overline{\rho}, t_w) + \sigma(\tau_m, t) + \sigma(\overline{\tau_m}, t))\} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Use Lemma 10(i).)}
\end{aligned}$$

$$\begin{aligned}
&= 2|P|+2|R|+2|T|\underline{-\sigma(\tau,p)-\sigma(\tau,\overline{r})-\sigma(\overline{\rho},t)-\sigma(\overline{\tau},t)-\sigma(\tau,t)}\\
&\quad + \sum_{(m,w)\in T}\min\{2-\sigma(\tau_m), 2\sigma(t_w)-(\sigma(\overline{\rho},t_w)+\sigma(\tau_m,t)+\sigma(\overline{\tau_m},t))\}\\
&\hspace{20em}\text{(Use Lemma 10(iii).)}\\
&\le 2|P|+2|R|+2|T|\\
&\quad + \sum_{(m,w)\in T}\min\{2-2\sigma(\tau_m), \underline{2\sigma(t_w)}-(\underline{2\sigma(\overline{\rho},t_w)}+2\sigma(\tau_m,t)+2\sigma(\overline{\tau_m},t))\}\\
&\hspace{20em}\text{(Use Lemma 8(i).)}\\
&= 2|P|+2|R|+2|T|+\sum_{(m,w)\in T}\min\{2-2\sigma(\tau_m),\\
&\qquad 2\sigma(\pi,t_w)+2\sigma(\rho,t_w)+2(\sigma(\tau,t_w)+\sigma(\overline{\tau},t_w)-\sigma(\tau_m,t)-\sigma(\overline{\tau_m},t))\}.
\end{aligned}$$

For the last inequality, we also used the inequality $\min\{a,b\}-c \le \min\{a-x, b-y\}$, where $x \le c$ and $y \le c$. ◀

▶ **Lemma 16.**

$$\sum_{(m,w)\in T}\min\{2-2\sigma(\tau_m), 2\sigma(\pi,t_w)+2\sigma(\rho,t_w)+\delta_{m,w}\} \le \frac{1}{2}(|P|+|R|+|T|).$$

Proof. For each pair $(m,w)\in T$, let $P(w)=\{i \mid (i,M(i))\in P, i =_w m\}$, $R(w)=\{i \mid (i,M(i))\in R, i =_w m, w \succ_i M(i)\}$, and $PR(w)=P(w)\cup R(w)$. Then it is not difficult to see that

$$\begin{aligned}
\sigma(\pi,t_w)+\sigma(\rho,t_w) &= \sum_{(i,M(i))\in P} x^*_{i,w} + \sum_{(i,M(i))\in R, w\succ_i M(i)} x^*_{i,w}\\
&= \sum_{(i,M(i))\in P, i=_w m} x^*_{i,w} + \sum_{(i,M(i))\in R, w\succ_i M(i), i=_w m} x^*_{i,w}\\
&= \sum_{i\in PR(w)} x^*_{i,w}. \qquad (9)
\end{aligned}$$

The first equality is from the definitions of $\sigma(\pi,t_w)$ and $\sigma(\rho,t_w)$ for $(m,w)\in T$. For the second equality, first note that there is no man i such that $m \succ_w i$ because $(m,w)\in T$. Also, note that any i considered in the summation has proposed to w during the execution of the algorithm; hence, there is no i such that $i \succ_w m$ and $x^*_{i,w} > 0$. Therefore, considering only i such that $i =_w m$ suffices. The last equality is from the definition of $PR(w)$.

For w such that $(m,w)\in T$ and a man $i\in PR(w)$, we define $\pi_i=\{(i,j)\in A\}$ if $i\in P(w)$, $\rho_i=\{(i,j)\in A \mid j \succ_i M(i)\}$ if $i\in R(w)$, and $\pi\rho_i=\pi_i\cup\rho_i$ for $i\in PR(w)$. Then, define $\nu_{i,w}=x^*_{i,w}/\sigma(\pi\rho_i)$ and

$$\nu_w = \sum_{i\in PR(w)} \nu_{i,w}.$$

Now, for each $(m,w)\in T$, we have

$$\sigma(\tau_m)=\sum_{j\succeq_m l(m)} x^*_{m,j} \ge \max_{i\in PR(w)}\sigma(\pi,\rho_i) \ge \sum_{i\in PR(w)}\frac{\nu_{i,w}}{\nu_w}\sigma(\pi,\rho_i) = \frac{1}{\nu_w}\sum_{i\in PR(w)} x^*_{i,w}. \qquad (10)$$

The first equality is due to the definition of τ_m and the fact that m has proposed to any woman j such that $j \succeq_m l(m)$. For the first inequality, we used Inequality (6). More specifically, we used the fact that each man $i \in PR(w)$ must have proposed to w with the f-value of at least

$$\sigma(\pi, \rho_i) = \sum_{(i,j)\in\pi\rho_i} x^*_{i,j},$$

but m, who proposed to w with the f-value

$$\sum_{j \succeq_m l(m)} x^*_{m,j},$$

is eventually matched to w. For the second inequality, we used the fact that a (weighted) average of $\sigma(\pi, \rho_i)$ over $i \in PR(w)$ is no more than the maximum over $i \in PR(w)$.

For $(m, w) \in T$ such that

$$\sum_{i\in PR(w)} x^*_{i,w} \leq \frac{\nu_w}{2(1+\nu_w)}(2 - \delta_{m,w}),$$

we have

$$2\sigma(\pi, t_w) + 2\sigma(\rho, t_w) + \delta_{m,w} = 2\sum_{i\in PR(w)} x^*_{i,w} + \delta_{m,w} \leq \frac{2\nu_w + \delta_{m,w}}{1+\nu_w} \leq \frac{1+\nu_w}{2} + \delta_{m,w}. \quad (11)$$

The first equality comes from Equation (9). For the last inequality, we used $\frac{\delta_{m,w}}{1+\nu_w} \leq \delta_{m,w}$, and the fact that $2x/(1+x) \leq (1+x)/2$ holds for any $x \geq 0$.

For $(m, w) \in T$ such that

$$\sum_{i\in PR(w)} x^*_{i,w} \geq \frac{\nu_w}{2(1+\nu_w)}(2 - \delta_{m,w}),$$

we have, from Inequality (10)

$$2 - 2\sigma(\tau_m) \leq 2 - \frac{2}{\nu_w}\sum_{i\in PR(w)} x^*_{i,w} \leq 2 - \frac{2-\delta_{m,w}}{1+\nu_w} = \frac{2\nu_w + \delta_{m,w}}{1+\nu_w} \leq \frac{1+\nu_w}{2} + \delta_{m,w}. \quad (12)$$

Therefore, we have

$$\begin{aligned}
&\sum_{(m,w)\in T} \min\{2 - 2\sigma(\tau_m), 2\sigma(\pi, t_w) + 2\sigma(\rho, t_w) + \delta_{m,w}\} \\
&\leq \sum_{(m,w)\in T}\left(\frac{1+\nu_w}{2} + \delta_{m,w}\right) \\
&= \sum_{(m,w)\in T} \frac{1+\nu_w}{2} + 2\sum_{(m,w)\in T}(\sigma(\tau, t_w) + \sigma(\overline{\tau}, t_w) - \sigma(\tau_m, t) - \sigma(\overline{\tau_m}, t)) \\
&= \sum_{(m,w)\in T} \frac{1+\nu_w}{2} \\
&= \frac{1}{2}\sum_{(m,w)\in T}\left(1 + \sum_{i\in PR(w)} \nu_{i,w}\right) \\
&= \frac{|T|}{2} + \frac{1}{2}\sum_{(i,M(i))\in P\cup R}\ \sum_{(m,w)\in T} \nu_{i,w} \\
&\leq \frac{|P| + |R| + |T|}{2}.
\end{aligned}$$

The first inequality comes from Inequalities (11) and (12). For the second equality, note that

$$\sum_{(m,w)\in T} (\sigma(\tau, t_w) + \sigma(\overline{\tau}, t_w)) = \sigma(\tau, t) + \sigma(\overline{\tau}, t) = \sum_{(m,w)\in T} (\sigma(\tau_m, t) + \sigma(\overline{\tau_m}, t))$$

by the definitions of $\sigma(\tau, t_w)$, $\sigma(\overline{\tau}, t_w)$, $\sigma(\tau_m, t)$, and $\sigma(\overline{\tau_m}, t)$. For the last equality, we exchanged the order of summation. The last inequality is due to the fact that for each $(i, M(i)) \in P \cup R$,

$$\sum_{(m,w)\in T} \nu_{i,w} \leq 1.$$

◀

Combining Lemmas 15 and 16, we can easily obtain Lemma 1. ◀

4 Lower Bounds

In this section, we show several results related to the inapproximability of the MAX SMTI. We first show three lower bounds on the integrality gap of the IP formulation given in Sec. 2, though the proof of Theorem 18 is omitted due to limitations of space.

▶ **Theorem 17.** *The integrality gap of the IP formulation given in Sec. 2 is at least* 1.25 *for R1T.*

Proof. We show an R1T instance I_1 whose integrality gap is (at least) 1.25.

m_1:	w_1	w_1:	$m_2\ m_3\ m_1$
m_2:	$w_2\ w_1$	w_2:	$(m_2\ m_3)$
m_3:	$w_2\ w_1\ w_3$	w_3:	m_3

One of the largest stable matchings for I_1 is $M^* = \{(m_2, w_1), (m_3, w_2)\}$. There is a feasible fractional solution x for $LP(I_1)$ such that $x_{m_1,w_1} = x_{m_2,w_1} = x_{m_2,w_2} = x_{m_3,w_2} = x_{m_3,w_3} = 0.5$. Hence, the integrality gap is at least $(5 \times 0.5)/|M^*| = 1.25$. ◀

▶ **Theorem 18.** *The integrality gap of the IP formulation given in Sec. 2 is at least* 1.3333 *for R2T.*

▶ **Theorem 19.** *The integrality gap of the IP formulation given in Sec. 2 is at least* $1.5 - o(1)$ *for 1T.*

Proof. We show a 1T instance I_3 whose integrality gap is $1.5 - o(1)$.

m_1:	$w_1\ w_1'$	w_1:	$(m_1\ m_2\ m_3\ \cdots\ m_k)\ m_1'$
m_2:	$w_1\ w_2\ w_2'$	w_2:	$(m_2\ m_3\ \cdots\ m_k)\ m_1'$
m_3:	$w_1\ w_2\ w_3\ w_3'$	w_3:	$(m_3\ \cdots\ m_k)\ m_3'$
$\vdots$			
m_{k-1}:	$w_1\ w_2\ w_3\ \cdots\ w_{k-1}\ w_{k-1}'$	w_{k-1}:	$(m_{k-1}\ m_k)\ m_{k-1}'$
m_k:	$w_1\ w_2\ w_3\ \cdots\ w_{k-1}\ w_k\ w_k'$	w_k:	$m_k\ m_k'$
m_1':	w_1	w_1':	m_1
$\vdots$			
m_k':	w_k	w_k':	m_k

The largest stable matching M^* for this instance is $\{(m_1, w_1), (m_2, w_2), \ldots, (m_k, w_k)\}$. There is a feasible fractional solution x for $LP(I_3)$ such that $x_{m_i,w_j} = 1/k$, $x_{m_i,w'_i} = 1 - i/k$, and $x_{m'_j,w_j} = (j-1)/k$ for all of the pairs $(i, j) \in \{1, 2, \ldots, k\}^2$. Hence, the integrality gap is at least

$$LP(I_3)/|M^*| = \left(k + \sum_{j=1}^{k} \frac{j-1}{k}\right) /k = 3/2 - o(1).$$

◀

The lower bound of Theorem 19 is an improvement over the previous bound of 1.3678 [22]. This result rules out some current techniques to obtain an approximation algorithm with a factor of $1.5 - \epsilon$ for 1T.

Second, we show a relation between the general 2T case of the MAX SMTI and a special case of the minimum maximal matching problem (MMM-Bi-APM(ϵ)) with respect to inapproximability, which is formally written as Theorem 21.

▶ **Definition 20.** The MMM-Bi-APM(ϵ) (for given ϵ such that $0 < \epsilon < 1/2$) is the problem to find a minimum maximal matching on a given balanced bipartite graph $G = (U, V, E)$ ($|U| = |V|$) that contains a matching M of size at least $(1 - \epsilon)|U|$ ($= (1 - \epsilon)|V|$).

▶ **Theorem 21.** *If the MMM-Bi-APM(ϵ) is* NP*-hard to approximate to within a factor of $2 - \epsilon$, then the MAX SMTI with two-sided ties (2T) is* NP*-hard to approximate to within a factor of $3/2 - O(\epsilon)$.*

Proof. We show that, if there is an approximation algorithm with approximation ratio $\alpha = 3/(2 + \epsilon + 2\epsilon(1 - \epsilon)) = 3/2 - O(\epsilon)$ for the MAX SMTI with two-sided ties (2T), then there is a $(2 - \epsilon)$-approximation algorithm for the MMM-Bi-APM(ϵ).

To show this, let $G = (U, V, E)$ such that $|U| = |V| = n$ be a balanced bipartite graph, an input of the MMM-Bi-APM(ϵ). Let $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. We construct an instance I_G of MAX SMTI as follows. Let $k = \lfloor (1 + \epsilon)n/2 \rfloor$. I_G consists of $n + k$ men $u_i (1 \leq i \leq n)$ and $x_i (1 \leq i \leq k)$, and $n + k$ women $v_i (1 \leq i \leq n)$ and $w_i (1 \leq i \leq k)$. Each man u_i corresponds to a vertex u_i of U, and each woman v_i corresponds to a vertex v_i of V. Hereafter, we do not distinguish between the names of these persons and vertices. The preference lists are given in the following. For a vertex $v \in U \cup V$, $N(v)$ denotes the set of vertices incident to v, and $[N(v)]$ denotes an arbitrary ordering of vertices in $N(v)$.

$$\begin{array}{llcll}
u_1: & ([N(u_1)])\ w_1\ \cdots\ w_k & \qquad & v_1: & ([N(v_1)])\ x_1\ \cdots\ x_k \\
\vdots & \vdots & & \vdots & \vdots \\
u_n: & ([N(u_n)])\ w_1\ \cdots\ w_k & & v_n: & ([N(v_n)])\ x_1\ \cdots\ x_k \\
x_1: & v_1\ \cdots\ v_n & & w_1: & u_1\ \cdots\ u_n \\
\vdots & \vdots & & \vdots & \vdots \\
x_k: & v_1\ \cdots\ v_n & & w_k: & u_1\ \cdots\ u_n
\end{array}$$

Let M^* be a minimum maximal matching of $G = (U, V, E)$. Then $|M^*| \geq (1-\epsilon)n/2$ by the assumption that G contains a matching of size at least $(1-\epsilon)n$. Next, it is easy to see that the above MAX SMTI instance I_G has a stable matching of size $|M^*| + (2n - 2|M^*|) = 2n - |M^*|$. (If $(u_i, v_j) \in M^*$ then include the pair (u_i, v_j). If u_i is unmatched in M^*, then include $(u_i, w_{i'})$ for some i', and similarly if v_j is unmatched in M^*, include $(x_{j'}, v_j)$ for some j'.) Therefore, if we have an α-approximation algorithm for the MAX SMTI, then this algorithm produces a matching M of size at least $(2n - |M^*|)/\alpha$. Let T be the set of pairs in M between people in U and V. First, it is easy to see that T is a maximal matching of G since if G is not maximal then M contains a blocking pair. Next, note that the size of M is exactly $2n - |T|$, which implies that $2n - |T| \geq (2n - |M^*|)/\alpha$. Hence we have $|T|/|M^*| \leq 2(1 - 1/\alpha)n/|M^*| + 1/\alpha \leq 4(1 - 1/\alpha)/(1 - \epsilon) + 1/\alpha$, where we use $|M^*| \geq (1 - \epsilon)n/2$ for the last inequality. Therefore, if there is an algorithm with approximation ratio $\alpha = 3/(2 + \epsilon + 2\epsilon(1 - \epsilon))$ for the MAX SMTI, then T is an approximate solution for the MMM-Bi-APM(ϵ) with an approximation ratio at most

$$\begin{aligned}
|T|/|M^*| &\leq 4(1 - 1/\alpha)/(1 - \epsilon) + 1/\alpha \\
&= 4(1 - (2 + \epsilon + 2\epsilon(1 - \epsilon))/3)/(1 - \epsilon) + (2 + \epsilon + 2\epsilon(1 - \epsilon))/3 \\
&= \frac{4}{3} - \frac{8}{3}\epsilon + \frac{1}{3}(2 + \epsilon + 2\epsilon(1 - \epsilon)) \\
&= 2 - \frac{5}{3}\epsilon - \frac{2}{3}\epsilon^2 \\
&< 2 - \epsilon.
\end{aligned}$$

◀

The (in)approximability of the MMM-Bi-APM(ϵ) is unknown, but we informally discuss it. It would be easy to construct a $(2 - \epsilon)$-approximation algorithm for the MMM-Bi-APM(ϵ) if we had an approximation algorithm with a constant approximation ratio for *the maximum balanced independent set problem on bipartite graphs*, which asks us to find a largest independent set $U' \cup V'$ such that $U' \subseteq U$, $V' \subseteq V$, and $|U'| = |V'|$ in a given bipartite graph $G = (U, V, E)$. However, this problem is known to be NP-hard and is hard to approximate (does not allow any constant approximation algorithm) under plausible assumptions [1, 6, 7, 23]. Although these results do not immediately rule out the existence of the $(2 - \epsilon)$-approximation algorithm for the MMM-Bi-APM(ϵ), they imply some difficulty of this problem.

Finally, we also show another inapproximability result for R2T, which is formally written as Theorem 22. This result can be obtained by slightly modifying the inapproximability proof for 2T given in [35].

▶ **Theorem 22.** *The MAX SMTI problem in which each person is allowed to include a tie only at the end of the preference list (R2T) is* NP*-hard to approximate with any factor smaller than* 33/29 *and is* UG*-hard to approximate with any factor smaller than* 4/3.

Proof. Yanagisawa [35] used a reduction from the minimum vertex cover problem with a perfect matching (which is UG hard to $(2 - \epsilon)$-approximate) to the 2T problem. In the reduction, he used the following gadget for each edge in a perfect matching.

$$\begin{array}{llcll}
v_j^A: & v_j^b & \qquad & v_i^a: & v_i^B \\
v_i^B: & (e_{ij}^c\ v_j^b)\ v_{i_1}^b \cdots v_{i_{di}}^b\ v_i^a & & v_j^b: & e_{ij}^C\ v_i^B\ v_{j_1}^B \cdots v_{j_{dj}}^B\ v_j^A \\
e_{ij}^C: & e_{ij}^c\ (v_i^b\ v_j^b) & & e_{ij}^c: & (v_j^B\ v_i^B)\ e_{ij}^C \\
v_j^B: & (e_{ij}^c\ v_i^b)\ v_{j_1}^b \cdots v_{j_{dj}}^b\ v_j^a & & v_i^b: & e_{ij}^C\ v_j^B\ v_{i_1}^B \cdots v_{i_{di}}^B\ v_i^A \\
v_i^A: & v_i^b & & v_j^a: & v_j^B
\end{array}$$

By modifying this gadget to the following one, we can satisfy the restriction that all the ties appear at the end of preference lists.

$$\begin{array}{llcll}
v_j^A: & v_j^b & \qquad & v_i^a: & v_i^B \\
v_i^B: & e_{ij}^c\ v_j^b\ v_{i_1}^b \cdots v_{i_{di}}^b\ v_i^a & & v_j^b: & e_{ij}^C\ v_i^B\ v_{j_1}^B \cdots v_{j_{dj}}^B\ v_j^A \\
e_{ij}^C: & e_{ij}^c\ (v_i^b\ v_j^b) & & e_{ij}^c: & (v_j^B\ v_i^B\ e_{ij}^C) \\
v_j^B: & e_{ij}^c\ v_i^b\ v_{j_1}^b \cdots v_{j_{dj}}^b\ v_j^a & & v_i^b: & e_{ij}^C\ v_j^B\ v_{i_1}^B \cdots v_{i_{di}}^B\ v_i^A \\
v_i^A: & v_i^b & & v_j^a: & v_j^B
\end{array}$$

It is easy to see that a matching M is stable for a MAX SMTI instance obtained from the original gadget if and only if M is stable for the MAX SMTI instance obtained from the modified gadget. Hence, the inapproximability result for 2T carries over to R2T. ◀

Acknowledgements. The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by JSPS KAKENHI Grant Numbers 24500013 and 25240002.

References

1 C. Ambühl, M. Mastrolilli, and O. Svensson, "Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut," *SIAM Journal on Computing*, Vol. 40(2), pp. 567-596, 2011.

2 J. Cardinal, S. Langerman, and E. Levy, "Improved approximation bounds for edge dominating set in dense graphs," *Theortical Computer Science*, Vol. 410(8–10), pp. 949–957, 2009.

3 R. Carr, T. Fujito, G. Konjevod, and O. Parekh, "A $2\frac{1}{10}$-approximation algorithm for a generalization of the weighted edge-dominating set problem," *Journal of Combinatorial Optimization*, Vol. 5(3), pp. 317–326, 2001.

4 M. Chlebík and J. Chlebíkova, "Approximation hardness of edge dominating set problems," *Journal of Combinatorial Optimization*, Vol. 11(3), pp. 279–290, 2006.

5 B. Dean and R. Jalasutram, "Factor revealing LPs and stable matching with ties and incomplete lists," In *Proc. of the 3rd International Workshop on Matching Under Preferences*, pp. 42–53, 2015.

6 U. Feige, "Relations between average case complexity and approximation complexity," In *Proc. STOC*, pp. 534–543, 2002.

7 U. Feige and S. Kogan, "Hardness of approximation of the balanced complete bipartite subgraph problem," Technical report MCS04-04, Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel, 2004.

8 T. Fujito and H. Nagamochi, "A 2-approximation algorithm for the minimum weight edge dominating set problem," *Discrete Applied Mathematics*, Vol. 181, pp. 199–207, 2002.

9 D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *Amer. Math. Monthly*, Vol. 69, pp. 9–15, 1962.

10 M. R. Garey and D. S. Johnson, *Computers and Intractability.* W. H. Freeman, San Francisco, 1979.

11 Z. Gotthilf, M. Lewenstein, and E. Rainshmidt, "A $(2 - c\frac{\log n}{n})$-approximation algorithm for the minimum maximal matching problem," In *Proc. WAOA*, pp. 267–278, 2008.

12 D. Gusfield and R. W. Irving, *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, Boston, MA, 1989.

13 M. M. Halldórsson, K. Iwama, S. Miyazaki, and H. Yanagisawa, "Improved approximation results for the stable marriage problem," *ACM Transactions on Algorithms*, Vol. 3(3), Article No. 30, 2007.

14 J. D. Horton and K. Kilakos, "Minimum edge dominating sets," *SIAM Journal on Discrete Mathematics*, Vol. 6(3), pp. 375–387, 1993.

15 C.-C. Huang and T. Kavitha, "An improved approximation algorithm for the stable marriage problem with one-sided ties," In *Proc. IPCO*, LNCS 8494, pp. 297–308, 2014.

16 R. W. Irving, "Matching medical students to pairs of hospitals: a new variation on a well-known theme," In *Proc. ESA*, LNCS 1461, pp. 381–392, 1998

17 R. W. Irving and D. F. Manlove, “Approximation algorithms for hard variants of the stable marriage and hospitals/residents problems,” *Journal of Combinatorial Optimization*, Vol. 16(3), pp. 279–292, 2008.

18 K. Iwama, D. F. Manlove, S. Miyazaki, and Y. Morita, “Stable marriage with incomplete lists and ties,” In *Proc. ICALP*, LNCS 1644, pp. 443–452, 1999.

19 K. Iwama, S. Miyazaki and K. Okamoto, “A $(2 - c \log N/N)$-approximation algorithm for the stable marriage problem, *IEICE Transactions*, Vol. 89-D(8), pp. 2380–2387, 2006.

20 K. Iwama, S. Miyazaki, N. Yamauchi, “A $(2-c\frac{1}{\sqrt{N}})$-approximation algorithm for the stable marriage problem,” *Algorithmica*, Vol. 51(3), pp. 342–356, 2008.

21 K. Iwama, S. Miyazaki, and N. Yamauchi, “A 1.875–approximation algorithm for the stable marriage problem,” In *Proc. SODA*, pp. 288–297, 2007.

22 K. Iwama, S. Miyazaki, and H. Yanagisawa, “A 25/17-approximation algorithm for the stable marriage problem with one-sided ties,” *Algorithmica*, Vol. 68, pp. 758–775, 2014.

23 S. Khot, “Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique,” *SIAM Journal on Computing*, Vol. 36(4), pp. 1025–1071, 2006.

24 S. Khot and O. Regev, “Vertex cover might be hard to approximate to within $2-\epsilon$,” *Journal of Computer and System Sciences*, Vol. 74(3), pp. 335–349, 2008.

25 Z. Király, “Better and simpler approximation algorithms for the stable marriage problem,” *Algorithmica*, Vol. 60, No. 1, pp. 3–20, 2011.

26 Z. Király, “Linear time local approximation algorithm for maximum stable marriage,” *MDPI Algorithms*, Vol. 6(3), pp. 471–484, 2013.

27 D. F. Manlove, *Algorithmics of Matching Under Preferences*, World Scientific Pub Co Inc, 2013.

28 D. F. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, and Y. Morita. “Hard variants of stable marriage,” *Theoretical Computer Science*, Vol. 276(1–2), pp. 261–279, 2002.

29 Y. Matsumoto, N. Kamiyama, and K. Imai, “An approximation algorithm dependent on edge-coloring number for minimum maximal matching problem,” *Information Processing Letters*, Vol. 111(10), pp. 465–468, 2011.

30 E. J. McDermid, “A 3/2-approximation algorithm for general stable marriage,” In *Proc. ICALP*, LNCS 5555, pp. 689–700, 2009.

31 K. E. Paluch, “Faster and simpler approximation of stable matchings,” *MDPI Algorithms*, Vol. 7(2), pp. 189–202, 2014.

32 A. Radnai, “Approximation algorithms for the stable matching problem,” Master’s Thesis, Eötvös Loránd University, 2014.

33 A. E. Roth, U. G. Rothblum, and J. H. V. Vate, “Stable matchings, optimal assignments, and linear programming,” *Mathematics of Operations Research*, Vol. 18(4), pp. 803–828, 1993.

34 C.-P. Teo and J. Sethuraman, “The geometry of fractional stable matchings and its applications,” *Mathematics of Operations Research*, Vol. 23(4), pp. 874–891, 1998.

35 H. Yanagisawa, “Approximation algorithms for stable marriage problems,” Ph. D. Thesis, Kyoto University, 2007.

36 M. Yannakakis, F. Gavril, “Edge dominating sets in graphs,” *SIAM Journal on Applied Mathematics*, Vol. 38, pp. 364–372, 1980.

Designing Overlapping Networks for Publish-Subscribe Systems*

Jennifer Iglesias[†1], Rajmohan Rajaraman[2], R. Ravi[1], and Ravi Sundaram[2]

1 **Carnegie Mellon University**
5000 Forbes Avenue, Pittsburgh, PA, USA
`{jiglesia, ravi}@andrew.cmu.edu`

2 **Northeastern University**
360 Huntington Ave., Boston, MA, USA
`{rraj, koods}@ccs.neu.edu`

Abstract

From the publish-subscribe systems of the early days of the Internet to the recent emergence of Web 3.0 and IoT (Internet of Things), new problems arise in the design of networks centered at producers and consumers of constantly evolving information. In a typical problem, each terminal is a source or sink of information and builds a physical network in the form of a tree or an overlay network in the form of a star rooted at itself. Every pair of pub-sub terminals that need to be coordinated (e.g. the source and sink of an important piece of control information) define an edge in a bipartite demand graph; the solution must ensure that the corresponding networks rooted at the endpoints of each demand edge overlap at some node. This simple overlap constraint, and the requirement that each network is a tree or a star, leads to a variety of new questions on the design of overlapping networks.

In this paper, for the general demand case of the problem, we show that a natural LP formulation has a non-constant integrality gap; on the positive side, we present a logarithmic approximation for the general demand case. When the demand graph is complete, however, we design approximation algorithms with small constant performance ratios, irrespective of whether the pub networks and sub networks are required to be trees or stars.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases Approximation Algorithms, Steiner Trees, Publish-Subscribe Systems, Integrality Gap, VPN.

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.381

1 Introduction

In large Internet publishing systems, a variety of sources of information constantly refresh their content, while a set of subscribers continuously pull this updated information. The recent widespread adoption of the "Internet of Things" and "Web 3.0" tools similarly involves the constant real-time sharing of information between producers of relevant content and

* Research supported in part by ONR grant N000141211001 and NSF grant CCF-1422715.
† This material is based n part upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 2013170941, and by Microsoft Research Graduate Women's Fellowship.

their corresponding consumers. Common examples include syndication systems as well as distributed databases that contain information originating at sources with sinks interested in the most up to date copies.

A natural approach to enable efficient information transfer in such systems is to build a *cost-effective* collection of networks, one for each publisher and supplier: the publishers push their updates to a set of locations via their respective networks, while the subscribers pull the information, refreshed by multiple publishers, from these intermediate nodes using their own networks. Note that each subscriber network needs only to *overlap* those publishers' networks that are of interest. Such interests are represented by an auxiliary bipartite *demand* graph with publishers on one side, subscribers on the other, and edges (of interest) between the two. Since the individual networks are being used for scatter/gather or push/pull operations (by publishers/subscribers respectively) the two natural structures are trees and overlay stars. Trees correspond to situations where the entity (e.g. a pusher, such as Facebook, or a puller, such as the IRS) has sufficient network presence to employ multicast/reverse-multicast while overlay stars correspond to point-to-point communication (e.g. a puller, such as a single user who can't share bandwidth, or has to pull directly from the source).

This basic framework gives rise to a class of problems we have christened *DON* or Design of Overlapping Networks. Given their relevance to developments in the internet ecosystem, these theoretical problems are significant from a practical perspective. Our central goal is to settle the polynomial-time approximability for the most general DON problem, in which we have an arbitrary demand graph, and arbitrary choice of tree or overlay star by each publisher/subscriber. In this paper, we obtain a constant approximation for the special case when all subscribers are interested in all publishers, and a logarithmic approximation for the general case. The latter approximation is, in fact, with respect to the value of a natural linear programming relaxation of the problem. In a contrasting result, we establish a non-constant integrality gap for this linear program. However, the exact approximability status of the general DON problem remains tantalizingly open.

1.1 Problem Definition

In the general DON problem we are given an undirected graph $G = (V(G), E(G))$ with non-negative costs on the edges $c : E \to \mathbb{Z}^+$, subsets of nodes $P, S \subseteq V$ with $P \cap S = \emptyset$ (publishers and subscribers respectively), the type of network to be installed for each publisher and subscriber, $\mathit{Type} : P \cup S \to \{\text{tree}, \text{star}\}$, and an auxiliary demand graph $D = (V(D), E(D))$ where $V(D) = P \cup S$ and $E(D) \subseteq P \times S$ specifying (publisher, subscriber) pairs whose networks are required to overlap (intersect). The goal is to build a collection of networks where each node in $P \cup S$ builds a network rooted at it of the specified type, and for all pairs $(p, s) \in E(D)$ the network rooted at p and the network rooted at s share a node. The objective is to minimize the sum of the costs of all the networks. We assume that the edge costs form a metric: they are symmetric and satisfy the triangle inequality. Any instance of the general DON problem can be split into four sub-instances; this subdivision only loses a factor of 4 in the approximation algorithms. When the type requirement Type is tree (resp., star) for all publishers and subscribers we refer to the problem as *tree-tree DON* (resp. *star-star DON*). We also use *tree-star DON* to refer to the two problem variants where on one side, say the publishers, we are required to build rooted trees while on the other, we are required to build rooted stars. We use the prefixes *general* and *complete* to denote arbitrary and complete demand graphs, respectively, as in general tree-tree DON or complete tree-star DON, etc. Thus, the term general (complete) DON refers to the problem where the demand graph is arbitrary (complete) and the type requirement may vary across terminals.

We denote the installed network by $N = (V, E_N)$; P_i denotes the network rooted at publisher p_i and S_j the network rooted at subscriber s_j. Then, the multi-graph $N = (\cup_{p_i \in P} P_i) \cup (\cup_{s_j \in S} S_j)$ is the (multi-set) union of all the installed networks. The cost of N is the sum of the costs of all the constituent networks, with each edge counted as many times as the number of individual networks they are present in. Recall that the installed networks are operated autonomously by each publisher and subscriber, and thus the cost of an edge needs to be multiplied by the number of such independent networks that build and utilize it in their updates.

1.2 Results and Techniques

We present new algorithms and results for several DON problems.

1. We conjecture that a polynomial-time constant-factor approximation for general DON is not achievable. We present in Section 2.1 an $\Omega(\log \log n)$ integrality gap for a natural LP relaxation of the general tree-tree DON; note that this result also extends to the general DON problem. This integrality gap proof, which is our strongest technical contribution, is based on a novel reduction from a well-studied LP relaxation for the group Steiner problem, applied to a hypercube demand graph instance of DON.
   On the positive side, we present an $O(\log n)$-approximation algorithm for the general DON problem in Section 2.2. The main ingredient of our result is a constant-factor approximation algorithm for tree-star DON on tree metrics, by a careful deterministic rounding of an LP relaxation of the problem. The logarithmic approximation for general DON follows by extending to general metrics using the methods of Bartal and Fakcharoenphol et. al [7, 2] and combining with previously known results for the star-star and tree-tree variants [3].
2. We next study the complete DON variants where the demand graph is complete. We give constant-factor approximation algorithms for all three variants – tree-tree, star-star and tree-star – which together yield a constant-factor approximation for complete DON. Unlike our algorithm for general DON, all of our algorithms for the complete demand case are combinatorial; they combine structural characterizations of near-optimal solutions with interesting connections to access network design and facility location problems.
   a. Our approximation factor for complete tree-tree DON in Section 3.1 is 4.74 via a reduction to asymmetric VPN. The 4.74 comes from the result by Eisenbrand and Grandoni [5] on asymmetric VPN.
   b. Our approximation factor for complete star-star DON in Section 3.2 is 4α, where α is the best approximation factor achieved for uncapacitated facility location; this improves upon the result in [3].
   c. For the complete tree-star-DON problem, we get a $4\rho_{TS}$-approximation in Section 3.3 where ρ_{TS} is the best known approximation for the tree-star Access Network Design problem, which is generalized by the Connected Facility Location or CFL problem (thus $\rho_{TS} \leq \rho_{CFL} \leq 4$ [6]).

1.3 Related Work

Data Dissemination Networks. Our formulation of DON generalizes network data dissemination problems first studied in [3]. Using our terminology, the relevant results of [3] are $O(\log n)$-approximation algorithms for general tree-tree DON and general star-star DON, and a 14.57-approximation for the complete star-star DON. Our work improves the approximation factor for complete star-star DON to under 6 (since the current best approximation for uncapacitated facility location is 1.488 [14]), and presents new results for many other DON problems. The star-star DON problem is also closely related to the minimum-cost 2-spanner problem studied in [4, 13]. In particular, a greedy algorithm essentially along the same lines as an algorithm of [4] yields an $O(\log n)$-approximation for the star-star-DON problem even when the underlying distances do not form a metric.

Network Design. There has been considerable work in network design, which is concerned with the design of network structures that satisfy some connectivity properties and optimize some underlying cost structure [22]. Well-known problems in this area include the minimum Steiner tree, group Steiner tree [8], and general survivable networks [12]. One key distinction between many of these network design problems and DON is that the desired solution in DON is a collection of networks (as opposed to a single network), and each edge contributes to the total cost of the solution as many times as it occurs in the network collection. On the other hand, the goal in many classical network design problems is to build a single network. Note that the problem of building a single minimum-cost network such that every pair of nodes in a given demand graph is connected in the network is exactly the generalized Steiner network problem, for which polynomial-time constant-factor approximations exist [1].

Multicommodity facility location. Another stream of work has addressed the extension of facility location problems to reach clients with additional restrictions on the facility opening costs, to reach facilities more robustly [21], or with the addition of services that facilities open to satisfy the clients with various cost functions governing the installation of services and facilities [17, 19]. The work in [15] arising from publisher-subscriber mechanisms is most closely related to our work, and rather than use a network from each publisher, models the publisher as a commodity that can be supplied at various nodes in the network by installing "facilities" of appropriate costs; the subscribers build minimum-cost networks to reach these facility installations of the appropriate publishers.

Access Networks and Connected Facility Location. Our algorithms for the complete DON problem are connected to to the access network design and facility location problems. In a version of the Access Network Design problem [16], we are given an undirected graph, a root node and nonnegative metric costs on the edges, along with a subset of terminal nodes. The goal is to design a backbone network in the form of a tree or tour which is built with higher speed and higher quality cables, while the terminals access the backbone using direct access edges. Thus the overall network is a backbone rooted Steiner tree (or tour), with access networks that are stars arising from the terminals and ending at the nodes of the backbone. We are given a cost multiplier μ that denotes the cost overhead factor for the backbone compared to the access network and the objective to be minimized is the total cost $\mu \cdot c(\text{backbone network } T) + \sum_{stars\ s} c(s)$. A $\rho_{AN} = 3 + 2\sqrt{2}$-approximation is presented for this problem [16] when the backbone is a ring and the access network is a star. Subsequent work [10, 20, 6] present constant-factor approximations for other generalizations and variants of this problem as well using LP rounding and primal-dual methods. The current best approximation factor for the CFL generalization is $\rho_{CFL} \leq 4$ [6], which also extends to tree-star access network design, i.e. $\rho_{AN} \leq \rho_{CFL} \leq 4$.

Virtual Private Network. The DON problems are also closely related to the VPN and asymmetric VPN problems. The VPN problem can be solved exactly [9], while the asymmetric VPN problem is NP-hard but has a constant approximation [18, 5]. The VPN problems differ from DON as each VPN problem instance seeks only one network, while a DON instance builds multiple networks. Nevertheless, we are able to decompose an approximate solution for asymmetric VPN into multiple networks, and obtain a useful approximation for the complete tree-tree DON problem (Section 3.1). In Section 3.1, we also present a more direct approach to the complete tree-tree DON problem that yields insights into the tree-star case at the cost of a slightly worse approximation ratio (as opposed to the reduction to VPN for the tree-tree case)."

2 DON with General Demands

In this section, we consider approximation algorithms for the general DON problem. We present in Section 2.1 an $\Omega(\log \log n)$ integrality gap for a natural LP relaxation of the general tree-tree DON problem. In Section 2.2, we present an $O(\log n)$-approximation algorithm for general DON.

2.1 Integrality Gap for Tree-Tree DON

In this section, we show that a natural LP formulation for the general tree-tree DON problem has super-constant integrality gap. We note that the same lower bound on integrality gap extends to the appropriate LP for general DON. A natural integer program, $\mathbf{IP_{DON}}$ for the tree-tree DON problem is as follows (all variables are $0-1$): we let $r \in P \cup S$ denote a publisher or subscriber node, and denote its tree N_r; z_e^r is an indicator variable that is 1 iff edge $e \in E(G)$ is in tree N_r; y_h^r is an indicator variable that is 1 iff vertex h is in tree N_r; $x_h^{r,s}$ is an indicator variable that is 1 iff vertex h is in both trees, N_r and N_s. X will refer to a cut which is a subset of vertices of $V(G)$ and $\delta(X)$ will denote the edges between X and its complement $V(G) \setminus X$. The integer program $\mathbf{IP_{DON}}$ for the tree-tree DON has the following nontrivial constraints.

$$\begin{array}{llll}
\min \displaystyle\sum_{r \in V(D), e \in E(G)} c_e z_e^r & & & \\
 & \displaystyle\sum_{e \in \delta(X)} z_e^r & \geq y_h^r & \forall X, r \in X, h \in V(G) \setminus X \\
 & x_h^{r,s} & \leq y_h^r & \forall r, s \in V(D), h \in V(G) \\
 & x_h^{r,s} & \leq y_h^s & \forall r, s \in V(D), h \in V(G) \\
 & \displaystyle\sum_{h \in V(G)} x_h^{r,s} & \geq 1 & \forall (r,s) \in E(D) \\
 & x_h^{r,s}, y_h^r, z_e^r & \geq 0 & \forall r, s \in V(D), e \in E(G), h \in V(G) \\
 & x_h^{r,s}, y_h^r, z_e^r & \text{are integers} & \forall r, s \in V(D), e \in E(G), h \in V(G)
\end{array}$$

$(\mathbf{IP_{DON}})$

The first set of cut covering constraints enforce that the tree rooted at r is connected with all nodes h for which y_h^r is set to one. The fourth set enforces all pairs of terminals r, s in the demand graph must meet in some hub vertex h. The second and third sets enforce that if a node h is used as a hub for a pair, it is required to occur in both these trees. Relaxing the above integer program by dropping the integrality constraints gives us the natural linear program $\mathbf{LP_{DON}}$. Observe that the feasible integral solutions of the linear program are exactly the solutions to the integer program.

▶ **Theorem 1.** *For every sufficiently large n, there exist instances of tree-tree DON with $n = |V(G)|$ for which* $\mathbf{LP_{DON}}$ *has an* $\Omega(\log\log n)$ *integrality ratio.*

Proof. Recall that the integrality ratio of a (minimizing) linear program is the minimum ratio between any feasible integral point and the optimum fractional solution. Our proof will proceed by a reduction from a linear program for the Group Steiner Tree (GST) problem.

Given a tree T with edge costs and a collection of k groups of leaves, $g_1, g_2, \ldots g_k$, the Group Steiner Tree problem is to find a minimum cost subtree such that at least one vertex from every group is connected to the root. In [11] it was shown that a natural linear program for the GST problem has an $\Omega(\log^2 n)$ integrality ratio even when the input metric costs c arise from an underlying tree. Similar to the linear program for the tree-tree DON problem we present the linear program $\mathbf{LP_{GST}}$ as the relaxation of an integer program $\mathbf{IP_{GST}}$ with $0-1$ variables.

$$\begin{aligned} \min \sum_{e\in E(T)} x_e c_e & & & \\ \sum_{e\in\delta(X)} x_e &\geq 1 & & \forall X, g_i : g_i \cap X = \emptyset, r \in X \\ x_e &\geq 0 & & \forall e \\ x_e &\ \text{integer} & & \forall e \end{aligned}$$

$(\mathbf{IP_{GST}})$

Here we give an explanation of what the variables in $\mathbf{IP_{GST}}$ represent: x_e is an indicator variable that is 1 iff edge $e \in E(T)$ is in the solution subtree; and X is a subset of vertices of $V(G)$ referring to a cut and $\delta(X)$ will denote the edges between X and its complement $V(G) \setminus X$. The main cut covering constraints enforce that each group is connected to the root node r. As before, dropping the integrality constraints gives us $\mathbf{LP_{GST}}$.

As stated before Halperin et.al [11] show that $\mathbf{LP_{GST}}$ has an integrality ratio of $\Omega(\log^2 n)$ even on tree metrics when k, the number of groups, is $\Omega(n)$ where $n = |V(T)|$.

Given an instance, T_{GST} of $\mathbf{LP_{GST}}$ with $n = V(T)$ vertices and k groups, we transform it into an instance of Tree-Tree DON, $\mathbf{LP_{DON}}$, with $N = 2^k n = |V(G)|$ vertices in the host graph such that

- corresponding to every fractional solution, of value f_{GST}, of $\mathbf{LP_{GST}}$ there is a fractional solution of value $f_{DON} \leq 2^k f_{GST}$ to $\mathbf{LP_{DON}}$, and
- corresponding to every feasible integral point, of value I_{DON}, of $\mathbf{LP_{DON}}$ there is a feasible integral point of value $I_{GST} \leq \frac{I_{DON}(1+\log k)}{2^k}$ to $\mathbf{LP_{GST}}$.

The transformation is intuitive: we take a "graph product" of the Group Steiner Tree instance with a hypercube of dimension k, where k is the number of groups. We will make 2^k copies of the tree T. Each root r of a copy of T will be either a publisher or subscriber. The demand graph D is a hypercube of dimension k. Now let T_r denote the copy of T rooted at r. Now consider s such that there is an edge (r, s) in the ith dimension in D, and $v \in g_i$. We will connect the copy of v in T_r to the copy of v in T_s with a zero cost edge.

It is easy to see that $f_{DON} \leq 2^k f_{GST}$ for the above transformation - observe that replicating the fractional solution to $\mathbf{LP_{GST}}$ in each of the 2^k copies of T is a valid fractional solution to $\mathbf{LP_{DON}}$.

For the other direction, i.e., to see $I_{GST} \leq \frac{I_{DON}(1+\log k)}{2^k}$ first observe that for edge (r, s) in dimension i of the demand hypercube, at least one of the trees corresponding to r or s must cross dimension i and the only way to cross dimension i is along a 0-cost edge connecting two corresponding group g_i leaves. Now note that any tree N_r (the network that r builds) in a solution to $\mathbf{IP_{DON}}$ can be transformed into a subtree of T by keeping an edge in T if N_r contains the corresponding edge in any copy of T in G. Let the subtree of T so obtained be called the *retract* of N_r. It is easy to see that if N_r ever crosses dimension j then a leaf in group g_j is connected to the root of T in its retract and that the cost of a retract is never more than the cost of the original N_r. By our earlier observation for any edge (r, s) in dimension i at least one of the two retracts of N_r, N_s must connect a node in group g_i to the root. Hence if we select a node in D at random and take its retract then any given group is connected with probability at least 1/2 and it has expected cost $\frac{I_{DON}}{2^k}$. Thus if we take the union of $1 + \log k$ retracts chosen uniformly at random then the resulting subgraph of T has expected cost $\frac{I_{DON}(1+\log k)}{2^k}$ and the probability any given group is not connected to the root is less than $\frac{1}{k}$. Since there are k groups this means there exists a subgraph of T, connecting the root to every group, of cost at most $\frac{I_{DON}(1+\log k)}{2^k}$, i.e., $I_{GST} \leq \frac{I_{DON}(1+\log k)}{2^k}$.

From $f_{DON} \leq 2^k f_{GST}$ and $I_{GST} \leq \frac{I_{DON}(1+\log k)}{2^k}$ it follows that $\frac{I_{DON}}{f_{DON}} \geq \frac{I_{GST}}{f_{GST}}(1 + \log k)$. By [11], when $k = \theta(n)$ we have that $\frac{I_{GST}}{f_{GST}} = \Omega(\log^2 n)$ from which it follows that $\frac{I_{DON}}{f_{DON}} = \Omega(\frac{\log^2 n}{\log k}) = \Omega(\log n)$ but the size of the transformed instance is $N = 2^k n$, i.e., $n = \Omega(\frac{\log N}{\log \log N})$. In other words, the integrality gap is $\frac{I_{DON}}{f_{DON}} = \Omega(\log n) = \Omega(\log(\frac{\log N}{\log \log N})) = \Omega(\log \log N)$ ◀

2.2 $O(\log n)$ approximation

We next show that the general DON problem can be approximated to within an $O(\log n)$ factor in polynomial time. As discussed in Section 1, the general DON problem can be split into three problems: tree-tree DON, star-star DON, and tree-star DON. In previous work, $O(\log n)$-approximation algorithms have been developed for tree-tree DON and star-star DON [3]. We now present an $O(\log n)$-approximation for tree-star DON, implying an $O(\log n)$-approximation for general DON.

Our $O(\log n)$-approximation for tree-star DON is obtained by deriving a constant-factor approximation for the special case of tree metrics, and invoking the standard reduction from general metrics to tree metrics [7]. Our constant-factor approximation algorithm, which rounds an LP relaxation, essentially generalizes a result of [15] on multicommodity facility location from a uniform facility cost case to the case where the facility costs form a tree metric.

▶ **Theorem 2.** *The tree-star DON problem with general demands on tree metrics can be approximated to within a constant factor in polynomial time. This implies an $O(\log n)$-approximation algorithm for general DON on general metrics.*

We first present a linear programming relaxation for the problem. Let T denote the given tree which is our metric. For a publisher j and an edge e of T, let z_e^j represent the extent to which j's tree uses e. For a subscriber i and leaf node v, let y_v^i denote the extent to which i's star visits v. For leaf node u, subscriber i and publisher j such that (i, j) is in the demand graph, let $x_u^{i,j}$ denote the extent to which j meets i at u. Let $d(u, v)$ denote the distance between u and v under the tree metric; abusing notation somewhat, let $d(e)$ denote the distance between the two endpoints of the edge e. Then, we have the following LP.

$$\min \sum_{j,e} z_e^j d(e) + \sum_{i,u} y_u^i d(i,u)$$

$$\begin{aligned} z_e^j &\geq \textstyle\sum_{u:e\in P_{ju}} x_u^{i,j} && \text{for all } i,\ j,\ e \\ \textstyle\sum_u x_u^{i,j} &\geq 1 && \text{for all } (i,j) \text{ in demand graph} \\ y_u^i &\geq x_u^{i,j} && \text{for all } i,\ j,\ u \\ x_u^{i,j}, y_u^i, z_e^j &\geq 0 && \text{for all } i,\ j,\ u,\ e \end{aligned}$$

$(\mathbf{LP_{TS}})$

The first set of constraints guarantees that every edge, e, in j's tree can support the extent that j meets any other node below e. The second set of constraints guarantees that all demands are satisfied. The third set of constraints guarantee that i's star supports how much i must meet with other nodes.

We now present our algorithm. We introduce some useful notation first. Let Y_v^i denote the sum of y_w^i, over all leaves w in the subtree rooted at v. Similarly, let $X_v^{i,j}$ denote the sum of $x_w^{i,j}$, over all leaves w in the subtree rooted at v.

We will identify three different types of hubs in the tree to which a subscriber will build it's star. These different types of hubs are paid for by different parts of the LP and guarantee a connection to the publisher trees in different ways.

1. Solve $\mathbf{LP_{TS}}$ to obtain a fractional solution (x, y, z).
2. For every subscriber i:
   - For every node v such that $Y_v^i \geq 1/3$ and there is no child c of v such that $Y_c^i \geq 2/3$: we mark node v.
   - For each marked node v such that no ancestor of v is marked, we add v to $\sigma(v)$; we refer to v as a type-C hub for i.
   - For every node v such that (a) there is no ancestor of v that is a type-C hub for i, and (b) there are two children c_1 and c_2 of v such that $Y_{c_1}^i \geq 1/3$ and $Y_{c_2}^i \geq 1/3$, we add v to $\sigma(v)$; we refer to v as a type-A hub for i.
   - For every node v that is an ancestor of a type-C hub, we define W_v^i to be sum, over every child c of v that is not an ancestor of a type-C hub, of Y_c^i.
   - For every path p from the root or a type-A hub node to a descendant type-A hub or type-C hub node: we divide p into minimal contiguous segments such that the sum of W_v^i, over all v in the segment, is at least $1/3$; for each such segment, we create a type-B hub for i at the lowest node in the segment.
   - The star network for i connects i to each type-A, -B, and -C hub.
3. For every publisher j, the tree network consists of all edges e such that $z_e^j \geq 1/3$.

We prove Theorem 2 by establishing the following two lemmas.

▶ **Lemma 3.** *For any edge (i, j) in the demand graph, the tree of publisher j overlaps with the star of subscriber i at least one node.*

Proof. Fix publisher j and subscriber i such that (i, j) is an edge in the demand graph. Consider some subtree rooted at a node r_0 such $X_{r_0}^{i,j}$ is at least $1/3$ in the LP solution, while for any child c of r_0, $X_c^{i,j} < 1/3$. Suppose $(r_0, r_1, \ldots, r_f)$ denote the path from r_0 to the root of the tree.

We first show that if there is a type-C hub at a node r_k, then the tree of publisher j includes node r_k. By our algorithm's choice of locating type-C hubs, it follows that

$Y^i_{r_{k-1}} < 2/3$. Therefore, publisher j meets subscriber i to an extent less than 2/3 in the subtree rooted at r_{k-1}. We consider two cases. If j is in the subtree rooted at r_{k-1}, then for the edge $e = (r_{k-1}, r_k)$, $z^j_e \geq 1/3$. Otherwise, since $X^{i,j}_{r_{k-1}} \geq X^{i,j}_{r_0} \geq 1/3$, again we have $z^j_e \geq 1/3$. Thus, in both cases, we ensure that the tree for publisher j contains r_k.

We next show that if there is a type-A or type-B hub for i at a node r_k and there are no hubs for i at any r_g, $0 \leq g < k$, then the tree for j must include r_k. Since there is no type-A hub at any r_g, $0 \leq g < k$, each r_g has at most one child that has a descendant with a type-C hub; if there were two such children, then r_g would have a type-A hub. Furthermore, there must be a type-C hub in the subtree rooted at r_0; if not, then the first ancestor of r_0 to have a hub would have a type-C hub, which would contradict our assumption. So suppose there is a type-C hub in the subtree rooted at r_0, say under the child r_{-1} of r_0. Then, it must be the case that the sum of $W^i_{r_g}$, over $0 \leq g < k$, is at most 1/3, since otherwise we would have a type-B hub at r_g. Furthermore, by the definition of r_0, $X^{i,j}_{r_{-1}} < 1/3$. This implies that j meets i to an extent of 1/3 outside the subtree rooted at r_{k-1} and at least 1/3 inside the subtree rooted at r_{k-1}. Thus, regardless of where j is located for edge $e = (r_{k-1}, r_k)$, we will have $z^j_e \geq 1/3$, ensuring that the tree for publisher j contains r_k. ◀

▶ **Lemma 4.** *The total cost of the tree and star networks is at most twelve times the LP optimal.*

Proof. An edge e is added to the tree of publisher j exactly when $z^j_e \geq 1/3$. Therefore, the cost of the tree network of j is within three times the cost for j in the LP.

We next consider the costs of the subscriber stars. There are three parts to it. The first is the distance to the type-A hubs. If a type-A hub for i is created at a node r, then there exist two children c_1 and c_2 of r such that $Y^i_{c_1}$ and $Y^i_{c_2}$ are both at least 1/3. Clearly, i is either not in the subtree rooted at c_1 or not in the subtree rooted at c_2. In either case, the cost for i in the LP solution for reaching the fractional hubs in one of c_1 or c_2 is at least $d(i,r)/3$. Adding this over all the type-A hubs yields a cost that is at most 3 times the LP cost for i.

If a type-C hub is created at a node r, then we consider two cases: r is a strict ancestor of i and r is not an ancestor of i or $r = i$. When r is not an ancestor of i or $r = i$, then we know that $Y^i_r \geq 1/3$ and all the distances from i to r's subtree are at least $d(i,r)$ so these type-C hub yield a cost that is at most 3 times the LP cost for i. Now if r is a strict ancestor of i, then let c be the child of r whose subtree contains i. We know that $Y^i_c \leq 2/3$. So, we know at least 1/3 of the LP cost associated with i travels distances $d(i,r)$. There is only one C-hub who is a strict ancestor of i. Therefore the type-C hubs yield a cost that is at most 6 times the LP cost for i.

If a type-B hub is created at a node r, then consider the sequence of ancestors a of r, whose W^i_a add up to 1/3. The cost of i reaching the fractional hubs in the LP that contribute to these W^i_a is at least $d(i,r)/3$.

The fractional hubs against which we have charged the type-C and type-B hubs are different, so the cost for the type-B and type-C hubs is at most 3 times the LP cost for i, yielding an $O(1)$-approximation for the overall total cost. ◀

3 DON with Complete Demands

In this section, we present constant factor approximation algorithms for the DON problem when the demand graph is complete. We obtain this result by deriving constant-factor approximations for the three variants – tree-tree, star-star and tree-star – in the following subsections.

3.1 Complete tree-tree DON

The complete tree-tree DON problem has an interesting connection to the asymmetric VPN problem [18, 5], which we can exploit to obtain a constant-factor approximation.

First we introduce the VPN problem. Given a graph G, with edge costs c, and marginals for each vertex, the VPN problem is to build a network of minimum cost such that for any set of pairwise demands which obey the marginals, the flow can be routed on our network. A set of pairwise demands obeys the marginals if the demands a vertex is involved in does not exceed its marginal. One crucial distinction between VPN and the DON problems is that while the VPN problem seeks the design of a single network, DON problems seek networks for every node involved.

Now we define asymmetric VPN. Here the flows are directed, and each vertex has two marginals, one for how much can flow out of the node, and one for how much can flow into the node. We restrict to the case where the terminals allow 1 flow out of the node and no flow in (*sources*), or they allow 1 flow into the node, and no flow out (*sinks*).

It turns out that for asymmetric VPN, there is always a tree solution which is within a constant factor of an optimal solution [18]. We now use this tree solution to get a solution for complete tree-tree DON.

▶ **Lemma 5.** *Given a complete tree-tree DON problem, consider an asymmetric VPN problem with the same input as the DON problem, with the subscribers as sources, and the publishers as sinks. Then, any tree solution for the asymmetric VPN problem can be transformed into a solution of the same cost for the complete tree-tree DON problem.*

Proof. Let T be the tree which is a solution to asymmetric VPN. Since our solution to asymmetric VPN is a tree which is adjacent to all the publishers and subscribers, then every edge in the tree induces a partition of the terminal nodes.

Consider any edge $e \in T$; we decide which trees use e. Let S_1, P_1 be the subscribers and publishers respectively on one side of the partition; likewise let S_2, P_2 be the remaining subscribers and publishers respectively. Now let $a = \min(|S_1|, |P_2|)$ and $b = \min(|P_1|, |S_2|)$. Now a valid demand matrix would be to require a unit flow from a elements of S_1 to a elements of P_2 and to require a unit flow from b elements of S_2 to b elements of P_1. These $a + b$ flows must all cross e since T is a tree, therefore e has multiplicity at least $a + b$.

Now if $|S_1| \leq |P_2|$, then we assign e to be in the trees for the elements of S_1, otherwise e is in the trees for the elements of P_2. Likewise if $|S_2| \leq |P_1|$ we assign e to be in the trees for the elements of S_2, otherwise e is in the trees for the elements of P_1. The number of times we use e is

$$\min(|S_1|, |P_2|) + \min(|S_2|, |P_1|) = a + b$$

So, we don't overuse e.

We next need to show that the edges assigned to a node form a tree. Since the original structure was a tree, we only need to show that the edges assigned to a terminal t are connected. Without loss of generality, suppose that a copy of e was assigned to be in T_s for $s \in S_1$. Let Q be the path in the tree T from e to s. Let $e' \in Q$. Let V' be the vertices on the same side as s of the partition formed by removing e' from T. We know that $S \cap V' \subseteq S_1$. Likewise we know that $P_2 \subseteq P \cap V'^C$. Since $|S_1| \leq |P_2|$ (because $e \in T_s$), then we know $|S \cap V'| \leq |P \cap V'^C|$ where $V'^C = V \setminus V'$. So, e' is assigned to be in the tree for nodes in $S \cap V'$. Therefore we have that $Q \subseteq T_s$. Therefore e is connected to s. Hence the graphs formed by our assignment scheme are connected.

Lastly, we must show that for every $s \in S$ and $p \in P$ then T_s and T_p intersect. Consider $s \in S$ and $p \in P$. Let Q be the path in T from p to s and e be an edge in Q. When we look at the S_1, S_2, P_1, P_2 formed by removing e, then either $s \in S_1$ and $p \in P_2$, or $s \in S_2$ or $p \in P_1$. Without loss of generality, assume $s \in S_1$ and $p \in P_2$. Then e is assigned to be in either the tree for all elements of S_1 or for all elements in P_2. So e is in either T_s or T_p. Since T_s and T_p are connected subtrees of the same tree, and $Q \subseteq T_S \cup T_p$ then T_s and T_p meet at some vertex in Q. Therefore, all demands are satisfied and this is a valid solution to the complete tree-tree DON problem. ◀

Using the current best results on asymmetric VPN which generates a tree solution [5], this provides an 4.74 approximation algorithm for the complete tree-tree DON problem.

Next, we present a direct approximation algorithm which we build on for the tree-star case. This approach has a worse approximation factor than the reduction from asymmetric VPN, but the techniques used are what give us the insight into the tree-star case.

▶ **Theorem 6.** *There is a $4\rho_{TS}$-approximation algorithm for complete tree-tree DON, where ρ_{TS} is the best factor for the tree-star access network design problem.*

In the rest of this subsection, we give a proof of Theorem 6. Given N^*, an optimal solution, let us denote the publisher and subscriber networks by $P_1, P_2, \ldots, P_k$ and $S_1, S_2, \ldots, S_l$ where we index the nodes so that we have $c(P_1) \leq c(P_2) \leq \cdots \leq c(P_k)$ and $c(S_1) \leq c(S_2) \leq \cdots \leq c(S_l)$. Let c_P^* and c_S^* denote the total cost of the publisher and subscriber trees. Let s_j be the subscriber whose network is S_j and let p_i be the publisher whose network is P_i. Note that feasibility requires that $P_i \cap S_j \neq \emptyset$ for all i, j. Let us also assume without loss of generality that $c(P_1) \leq c(S_1)$.

The key transformation of the optimal solution is a reconfiguration of the subscriber networks where we replace each tree S_j for $j \neq 1$ by the direct edge from subscriber node j to subscriber node 1 concatenated with the subscriber tree S_1. In other words, we set $S'_j = \{(s_j, s_1)\} \cup S_1$ for every subscriber $s_j \neq s_1$. Let us assign $S_j = S'_j$.

Note that the modified subscriber trees are still feasible since the original subscriber tree S_1 intersects every publisher tree. We now bound the cost of the additional edge from subscriber j to the subscriber 1, the root of S_1.

▶ **Lemma 7.** *For every subscriber $j \neq 1$, we have $c_{i1} \leq 3c(S_j)$.*

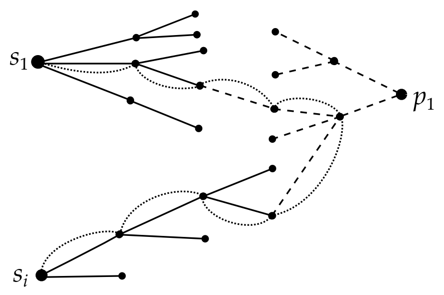


Figure 1 The solid lines show the trees S_i and S_1 and the dashed lines show the tree P_1. The dotted line here is the path from s_i to s_1 through the trees S_i, P_1, and S_1.

Proof. To see this, note that by taking the path from j in S_j to its intersection with P_1 and following it to the intersection of P_1 and S_1 and continuing along S_1 to the subscriber node 1, we have found a path from j to 1 of cost no more than the sum of the costs of S_j, P_1 and S_1. However, since $c(S_j) \geq c(S_1) \geq c(P_1)$, the length of this path is at most $3c(S_j)$. ◀

Given that every subscriber contains the tree S_1, it is particularly simple to design the publisher network P_i' (for publisher p_i) that needs to reach this tree: it will simply be a direct edge that represents the shortest path from the publisher p_i to the tree S_1. The union of all such direct edges gives a collection of stars that end at the subscriber tree S_1. Furthermore since the subscriber tree S_1 is going to be used by every subscriber node, its cost must be counted $|S| = l$ times in the objective.

The resulting problem of finding the best tree for S_1 is exactly the tree-star access network design problem [16] with the root being subscriber 1, the multiplier $M = |S|$ and the terminals being $R = P$, the publisher nodes. Given an optimal solution N^* for the complete tree-tree DON problem, we thus have a solution to the tree-star access network instance of cost at most $c_P^* + |S| \cdot c(S_1)$. We thus have the following lemma.

▶ **Lemma 8.** *For the correct choice of the subscriber node 1 as the root with multiplier $|S|$ and terminals P, there is a solution to the tree-star access network design problem of cost at most $c_P^* + |S| \cdot c(S_1)$.* ◀

Proof of Theorem 6. The approximation algorithm tries every subscriber node as the root of the tree-star access network problem formulated above. By adding the direct edge from each other subscriber to this root, and extending the backbone tree with each such edge, we get a solution to the complete tree-tree DON problem. The algorithm keeps the solution of smallest total cost among all choices of the root subscriber node. The total cost of the solution is the sum of the cost of the tree-star access network design problem and the sum of the costs of the direct edges from the subscribers to the root. By Lemma 7, the latter cost is no more than three times the cost of the tree (with the multiplier of |S|) in the solution to the tree-star access network design problem. By Lemma 8 and the ρ_{TS}-approximation factor for the tree-star access network design problem, we thus obtain a total cost of at most $4\rho_{TS}(c_P^* + |S| \cdot c(S_1))$ which is at most $4\rho_{TS}$ times the cost of N^*. ◀

It is not hard to extend the above methods to the case when the input terminals are partitioned into more than two subsets, say $R = P_1 \cup P_2 \cup \ldots \cup P_k$ and the demand graph is the complete k-partite graph between these k subsets. By considering the partition that has the cheapest tree network in the optimal solution to be in P_1, the above argument can be adapted to give a constant-factor approximation. We omit the details in this extended abstract.

3.2 Complete star-star DON

In this section, we present a constant-factor approximation for complete star-star DON.

▶ **Theorem 9.** *There is a 4α-algorithm for complete star-star DON, where α is the best approximation achievable for metric uncapacitated facility location.*

Our algorithm and the proof of Theorem 9 are based on an argument that there exists a constant-factor approximate solution that has a special structure; our algorithm then computes a constant-factor approximate solution with this special structure.

Given a solution where the publisher network is $P_1, P_2, \ldots, P_k$ and subscriber network is $S_1, S_2, \ldots S_l$, let P_1 be the publisher network of smallest cost and S_1 be the subscriber network of smallest cost without loss of generality. Also, let $\sigma(p_i)$ denote the set of nodes (which we refer to as hubs for p_i) in the star for the ith publisher. Likewise, let $\sigma(s_j)$ be the set of nodes in the star for the jth subscriber. Thus, we can refer to solutions using the maps defined by σ and denote the optimal one by σ^*. We let $c(\sigma)$ be the cost of the solution defined by σ, and $c(\sigma(x))$ denotes the cost of x's star. The next lemma shows a near-optimal solution with a very simple structure.

▶ **Lemma 10.** *There exists a solution σ such that $c(\sigma) \leq 4c(\sigma^*))$, and either $\sigma(s_i) = \sigma(s_j)$ for all pairs of subscribers s_i, s_j and $|\sigma(p_i)| = 1$ for each publisher p_i or $\sigma(p_i) = \sigma(p_j)$ for all pairs of publishers p_i, p_j and $|\sigma(s_i)| = 1$ for each subscriber s_i.*

Proof. Let σ^* be an optimal solution with publisher networks $P_1, P_2, \ldots, P_k$ and subscriber networks $S_1, S_2, \ldots S_l$. Without loss of generality, let $c(P_1) \leq c(S_1)$. Since each subscriber star intersects all publisher stars, we have $d(s_i, s_1) \leq c(S_i) + c(P_1) + c(S_1) \leq 3c(S_i)$. Let C_1 denote the set of publishers that share any hub with p_1. Let p_2 denote the least-cost publisher not in C_1. Let C_2 be the set of all publishers not in C_1 that share any hub with p_2. In general, let p_{j+1} be the least-cost publisher not in $\bigcup_{1 \leq i \leq j} C_i$. Let C_{j+1} denote the set of all publishers not in $\bigcup_{1 \leq i \leq j} C_i$ that share any hub with p_{j+1}. Let h_j denote any hub in $\sigma^*(s_1) \cap \sigma^*(p_j)$.

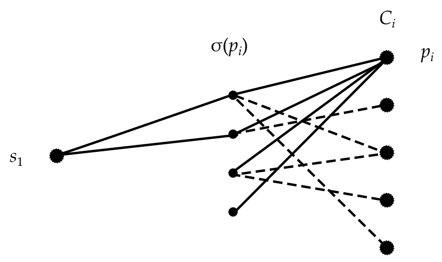


Figure 2 Here is an example of a P_i. We have shown all it's hubs, $\sigma(p_i)$. C_i consists of all those p_j's not in a previous C_k connecting to one of the hubs. Here h_i can be chosen to be either of the top two hubs in $\sigma(P_i)$.

Let $p_{j'}$ be an arbitrary publisher in C_j. We first obtain the following equation $d(p_{j'}, p_j) \leq 2c(P_{j'})$ (owing to a shared hub and the fact that $c(P_j) \leq c(P_{j'})$). By construction, for any two distinct p_i and p_j, we have $\sigma^*(p_i) \cap \sigma^*(p_j) = \emptyset$; i.e., p_i and p_j do not share any hubs. Note that this may not be true of all pairs of publishers in $C_i \times C_j$.

We now consider two cases. In the first case when $c(P_j) \leq d(s_1, h_j)$, we have all subscribers meet all the publishers in cluster C_j at p_j. Consider any subscriber s_i. It meets p_j at some hub, say h^i_j. Its increase in cost for meeting p_j now is at most $c(P_j) \leq d(s_1, h_j)$, which equals one leg of s_1's star. Since two different p_j's do not share any hubs, the h^i_j's (for a given i) are all different. Hence, the total increase in cost for s_i is at most $\sum_j d(s_1, h_j)$, which is at most $c(S_1)$.

If $c(P_j) > d(s_1, h_j)$, then we will have all publishers in C_j, go to s_1. Fix a publisher p'_j in C_j. Its total cost is at most $d(p'_j, p_j) + c(P_j) + d(s_1, h_j) \leq d(p'_j, p_j) + 2c(P_j) \leq 4c(P'_j)$. All the subscribers also go to s_1 to handle this case. We have $d(s_i, s_1) \leq c(S_i) + c(P_1) + c(S_1) \leq 3c(S_i)$.

So overall, we obtain a blowup of at most 4 in the cost for each publisher and each subscriber. We have proved that there exists a solution of cost at most $4 \cdot OPT$ in which every subscriber's star connects to exactly the same set of hubs and every publisher's star is just a line to one of the hubs. ◀

Proof of Theorem 9. Using Lemma 10, we now give a polynomial-time 4α-algorithm where α is the best approximation achievable for the uncapacitated facility location problem.

Our algorithm considers all possible choices for s_1, a linear number (where by symmetry s_1 could be on either side). For a given choice of s_1, we formulate an uncapacitated facility location problem, with the set of publishers as the clients, and the potential facility locations being the publishers and s_1. The cost of opening a facility at any of these nodes is the sum of the distances of all the subscribers to that node. Given a solution to this facility location problem, we obtain a solution to the complete star-star DON problem as follows: each publisher's star is a singleton edge to the facility it is assigned to; each subscriber's star consists of edges to all the open facilities, paid for by the facility costs.

We solve all the linear number of facility location problems, and then the corresponding problems with the roles of subscribers and publishers reversed, and take the best solution. This yields the desired approximation. ◀

3.3 Complete tree-star DON

We now present a constant factor approximation for complete tree-star DON. Without loss of generality, let us suppose that the publishers will build trees, and the subscribers will build stars. The main idea is to show that either the appropriately defined complete star-star DON solution or complete tree-tree DON solution is within a constant factor of optimal.

Let N^* be an optimal solution. Let the trees be indexed $P_1, P_2, \ldots P_k$ and the stars $S_1, \ldots S_\ell$ such that $c(P_1) \leq \cdots \leq c(P_k)$ and $c(S_1) \leq \cdots \leq c(S_k)$.

First consider the case where $c(P_1) \geq c(S_1)$. Note that every P_i and S_j must have a non-empty intersection. Now for every tree P_j we can redirect it to P_1 and then make a copy of P_1. So we will let: $P'_j = \{(p_j, p_1)\} \cup P_1$.

This solution is feasible because P_1 must intersect all the stars. These additions to the solution cost at most $3c(N^*)$, as seen in Lemma 7. Now all the stars can simply take an edge which is the shortest edge to the tree.

The approximation algorithm from this point follows the tree-tree case exactly. In this case, we get that the final solution has cost at most $4\rho_{TS}c(N^*)$. Where ρ_{TS} is the best constant approximation for the tree-star access problem.

Next consider the case that $c(S_1) \geq c(P_1)$. We will now choose p_i's in a similar fashion to the complete star-star DON problem. Let p_1 be the publisher with the smallest cost tree. Let C_1 be all the publishers whose trees meet p_1's tree. Now let p_2 be the smallest tree which does not intersect p_1's tree. Let C_2 be all the publishers not in C_1 who meet p_2's tree. Likewise p_{j+1} will be the smallest tree not in $\cup_{1 \leq i \leq j} C_i$. Let C_{j+1} be all the publishers which intersect p_{j+1}'s tree not in $\cup_{1 \leq i \leq j} C_i$.

Now from hereon, the proof follows that for the complete star-star DON case. Hence we have a solution within a constant factor of optimal where all the stars go to s_1 (the subscriber with star S_1), and some of the publishers: P_{open}. Each tree goes to the nearest node in $S_1 \cup P_{open}$. This establishes the following lemma.

▶ **Lemma 11.** *The complete tree-star DON has an $O(1)$-approximate solution in which either all the subscribers go to some hubs and each tree goes to the nearest hub among a set of one subscriber and some publishers, or where all the publisher trees are identical and all the subscribers go to the closest node in that tree.* ◀

For solving the complete tree-star DON problem, we apply our constant-factor approximation algorithm for the complete tree-tree DON instance, together with our constant-factor algorithm for the complete star-star DON instance, and take the better of the two. This completes our argument showing that complete tree-star DON can be approximated to within a constant factor.

References

1 Ajit Agrawal, Philip N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995.

2 Yair Bartal. On approximating arbitrary metrices by tree metrics. In *STOC*, pages 161–168, 1998.

3 R. C. Chakinala, A. Kumarasubramanian, K. A. Laing, R. Manokaran, C. Pandu Rangan, and R. Rajaraman. Playing push vs pull: models and algorithms for disseminating dynamic data in networks. In *SPAA*, pages 244–253, 2006.

4 Yevgeniy Dodis and Sanjeev Khanna. Designing networks with bounded pairwise distance. In *STOC*, pages 750–759, 1999.

5 Friedrich Eisenbrand and Fabrizio Grandoni. An improved approximation algorithm for virtual private network design. *SODA*, pages 928–932, 2005.

6 Friedrich Eisenbrand, Fabrizio Grandoni, Thomas Rothvoss, and Guido Schafer. Connected facility location via random facility sampling and core detouring. *JCSS*, 76(8):709–726, 2010.

7 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

8 Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000.

9 Navin Goyal, Neil Olver, and F. Bruce Shepherd. The VPN conjecture is true. *J. ACM*, 60(3):17, 2013.

10 Anupam Gupta, Jon M. Kleinberg, Amit Kumar, Rajeev Rastogi, and Bülent Yener. Provisioning a virtual private network: a network design problem for multicommodity flow. *STOC*, pages 389–398, 2001.

11 Eran Halperin, Guy Kortsarz, Robert Krauthgamer, Aravind Srinivasan, and Nan Wang. Integrality ratio for group Steiner trees and directed Steiner trees. *SIAM J. Comput.*, 36(5):1494–1511, 2007.

12 Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

13 Guy Kortsarz and David Peleg. Generating low-degree 2-spanners. *SIAM Journal on Computing*, 27(5):1438–1456, 1998.

14 Shi Li. Approximation algorithms for network routing and facility location problems. *Doctoral Dissertation, Princeton University*, Januray 2014.

15 Laura J. Poplawski and Rajmohan Rajaraman. Multicommodity facility location under group steiner access cost. *SODA*, pages 996–1013, 2011.

16 R. Ravi and F. Sibel Salman. Approximation algorithms for the traveling purchaser problem and its variants in network design. *ESA*, pages 29–40, 1999.

17 R. Ravi and Amitabh Sinha. Approximation algorithms for multicommodity facility location problems. *SIAM J. Discrete Math.*, 24(2):538–551, 2010.

18 Thomas Rothvoss and Laura Sanita. On the complexity of the asymmetric VPN problem. In *APPROX/RANDOM*, pages 326–338, 2009.

19 David B. Shmoys, Chaitanya Swamy, and Retsef Levi. Facility location with service installation costs. In *SODA*, pages 1088–1097, 2004.

20 Chaitanya Swamy and Amit Kumar. Primal-dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.

21 Chaitanya Swamy and David B. Shmoys. Fault-tolerant facility location. *ACM Transactions on Algorithms*, 4(4), 2008.

22 David P Williamson and David B Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

Approximating Dense Max 2-CSPs

Pasin Manurangsi[*1] **and Dana Moshkovitz**[†2]

1 **Dropbox, Inc.**
San Francisco, CA 94107, USA
`pasin@dropbox.com`

2 **Massachusetts Institute of Technology**
Cambridge, MA 02139, USA
`dmoshkov@mit.edu`

Abstract

In this paper, we present a polynomial-time algorithm that approximates sufficiently high-value Max 2-CSPs on sufficiently dense graphs to within $O(N^{\varepsilon})$ approximation ratio for *any* constant $\varepsilon > 0$. Using this algorithm, we also achieve similar results for free games, projection games on sufficiently dense random graphs, and the Densest k-Subgraph problem with sufficiently dense optimal solution. Note, however, that algorithms with similar guarantees to the last algorithm were in fact discovered prior to our work by Feige et al. and Suzuki and Tokuyama.

In addition, our idea for the above algorithms yields the following by-product: a quasi-polynomial time approximation scheme (QPTAS) for satisfiable dense Max 2-CSPs with better running time than the known algorithms.

1998 ACM Subject Classification G.1.6 Optimization, G.2.2 Graph Algorithms

Keywords and phrases Max 2-CSP, Dense Graphs, Densest k-Subgraph, QPTAS, Free Games, Projection Games

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.396

1 Introduction

Maximum constraint satisfaction problem (Max CSP) is a problem of great interest in approximation algorithms since it encapsulates many natural optimization problems; for instance, Max k-SAT, Max-Cut, Max-DiCut, Max k-Lin, projection games, and unique games are all families of Max CSPs. In Max CSP, the input is a set of variables, an alphabet set, and a collection of constraints. Each constraint's domain consists of all the possible assignments to a subset of variables. The goal is to find an assignment to all the variables that satisfies as many constraints as possible.

In this paper, our main focus is on the case where each constraint depends on exactly $k = 2$ variables and the alphabet size is large. This case is intensively researched in hardness of approximation and multi-prover games.

For Max 2-CSP with large alphabet size, the best known polynomial-time approximation algorithm, due to Charikar et al. [10], achieves an approximation ratio of $O((nq)^{1/3})$ where n is the number of variables and q is the alphabet size. On the other hand, it is known that, there is no polynomial-time $2^{\log^{1-\delta}(nq)}$-approximation algorithm for Max 2-CSP unless NP $\not\subseteq$ DTIME$(n^{polylog(n)})$ [16]. Moreover, it is believed that, for some constant $c > 0$, no

[*] Part of this work was completed while the author was at Massachusetts Institute of Technology.
[†] This material is based upon work supported by the National Science Foundation under Grant Number 1218547.

polynomial-time $O((nq)^c)$-approximation algorithm exists for projection games, a family of Max 2-CSP we shall introduce later, unless P = NP [15]. This is also known as the Projection Games Conjecture (PGC). As a result, if the PGC holds, one must study special cases in order to go beyond polynomial approximation ratio for Max 2-CSP.

One such special case that has been particularly fruitful is dense Max 2-CSP where density is measured according to number of constraints, i.e., an instance is δ-dense if there are δn^2 constraints. Note that, for convenience, we always assume that there is at most one constraint on a pair of variables. In other words, we form a simple graph by letting vertices represent the variables and edges represent the constraints. This is the interpretation that we will use throughout the paper. According to this view, δn is the average degree of the graph.

In 1995, Arora, Karger and Karpinski [3] invented a polynomial-time approximation scheme (PTAS) for dense Max 2-CSP when the density δ and alphabet size q are constants. More specifically, for any constant $\varepsilon > 0$, the algorithm achieves an approximation ratio $1+\varepsilon$ and runs in time $O(n^{1/\varepsilon^2})$. Unfortunately, the running time becomes quasi-polynomial time when q is not constant.

Another line of development of such PTASs centers around subsampling technique (e.g. [1, 2, 4]). In summary, these algorithms function by randomly sampling the variables according to some distribution and performing an exhaustive search on the induced instance. Since the sampled set of variables is not too large, the running time is not exponential. However, none of these algorithm achieves polynomial running time for large alphabets. In particular, all of them are stuck at quasi-polynomial running time.

Since none of these algorithms runs in polynomial time for large alphabet, a natural and intriguing question is how good a polynomial-time approximation algorithm can be for dense Max 2-CSPs. In this paper, we partially answer this question by providing a polynomial-time approximation algorithm for dense high-value Max 2-CSPs that achieves $O((nq)^\varepsilon)$ approximation ratio for any constant $\varepsilon > 0$. Moreover, our technique also helps us come up with a quasi-polynomial time approximation scheme for satisfiable Max 2-CSPs with running time asymptotically better than that those from [1, 2, 3, 4].

The central idea of our technique is a trade-off between two different approaches: greedy assignment algorithm and "choice reduction" algorithm. In summary, either a simple greedy algorithm produces an assignment that satisfies many constraints or, by assigning an assignment to just one variable, we can reduce the number of optimal assignment candidates of other variables significantly. The latter is what we call the choice reduction algorithm. By applying this argument repeatedly, either one of the greedy assignments gives a high-value assignment, or we are left with only few candidate labels for each variable. In the latter case, we can then just pick a greedy assignment at the end.

Not only that our technique is useful for Max 2-CSP, we are able to obtain approximation algorithms for other problems in dense settings as well. The first such problem is free games, which can be defined simply as Max 2-CSP on balanced complete bipartite graphs. While free games have been studied extensively in the context of parallel repetition [5, 17] and as basis for complexity and hardness results [1, 9], the algorithm aspect of it has not been researched as much. In fact, apart from the aforementioned algorithms for dense Max 2-CSP that also works for free games, we are aware of only two approximation algorithms, by Aaronson et al. [1] and by Brandao and Harrow [8], specifically developed for free games. Similar to the subsampling lemmas, these two algorithms are PTASs when q is constant but, when q is large, the running times become quasi-polynomial. Interestingly, our result for dense Max 2-CSP directly yields a polynomial-time algorithm that can approximate free games within $O((nq)^\varepsilon)$ factor for any constant $\varepsilon > 0$, which may be the first non-trivial approximation algorithm for free games with such running time.

Secondly, our idea is also applicable for projection games. The projection games problem (also known as Label Cover) is Max 2-CSP on a bipartite graph where, for each assignment to a left vertex of an edge, there is exactly one satisfiable assignment to the other endpoint of the edge. Label Cover is of great significance in the field of hardness of approximation since almost all NP-hardness of approximation results known today are reduced from the NP-hardness of approximation of projection games (e.g. [6, 12]).

The current best polynomial-time approximation algorithm for satisfiable projection games is the authors' with $O((nq)^{1/4})$ ratio [14]. Moreover, as mentioned earlier, if the PGC is true, then, in polynomial time, approximating Label Cover beyond some polynomial ratio is unlikely. In this paper, we exceed this bound on random balanced bipartite graphs with sufficiently high density by proving that, in polynomial time, one can approximate satisfiable projection games on such graphs to within $O((nq)^{\varepsilon})$ factor for any constant $\varepsilon > 0$.

Finally, we show a similar result for Densest k-Subgraph, the problem of finding a size-k subgraph of a given graph that contains as many edges as possible. Finding best polynomial-time approximation algorithm for Densest k-Subgraph(DkS) is an open question in the field of approximation algorithms. Currently, the best known algorithm for DkS achieves an approximation ratio of $O(n^{1/4+\varepsilon})$ for any constant $\varepsilon > 0$ [7]. On the other hand, however, we only know that there is no PTAS for DkS unless P=NP [13].

Even though Densest k-Subgraph on general graphs remains open, the problem is better understood in some dense settings. More specifically, Arora et al. [3] provided a PTAS for the problem when the given graph is dense and $k = \Omega(N)$ where N is the number of vertices of the given graph. Later, Feige et al. [11] and Suzuki and Tokuyama [18] showed that, if we only know that the optimal solution is sufficiently dense, we can still approximate the solution to within any polynomial ratio in polynomial time. Using our approximation algorithm for dense Max 2-CSP, we are able to construct a polynomial-time algorithm for Densest k-Subgraph with similar conditions and guarantees as those of the algorithms from [11] and [18].

The theorems we prove in this paper are stated in Section 3 after appropriate preliminaries in the next section.

2 Preliminaries and Notation

In this section, we formally define the problems we focus on and the notation we use throughout the paper. First, to avoid confusion, let us state the definition of approximation ratio for the purpose of this paper.

▶ **Definition 1.** An approximation algorithm for a maximization problem is said to have an approximation ratio α if the output of the algorithm is at least $1/\alpha$ times the optimal solution.

Note here that the approximation ratio as defined above is always at least one.

Next, before we define our problems, we review the standard notation of density of a graph.

▶ **Definition 2.** A simple undirected graph $G = (V, E)$ is defined to be of density $|E|/|V|^2$.

Moreover, for a graph G and a vertex u, we use $\Gamma^G(u)$ to denote the set of neighbors of u in G. We also define $\Gamma_2^G(u)$ to denote the set of neighbors of neighbors of u in G, i.e., $\Gamma_2^G(u) = \Gamma^G(\Gamma^G(u))$. When it is unambiguous, we will leave out G and simply write $\Gamma(u)$ or $\Gamma_2(u)$.

Now, we will define the problems starting with Max 2-CSP.

▶ **Definition 3.** An instance $(q, V, E, \{C_e\}_{e\in E})$ of Max 2-CSP consists of

- a simple undirected graph (V, E), and
- for each edge $e = (u, v) \in E$, a constraint (or constraint) $C_e : [q]^2 \to \{0, 1\}$ where $[q]$ denotes $\{1, 2, \dots, q\}$.

The goal is to find an assignment (solution) $\varphi : V \to [q]$ that maximizes the number of constraints C_e's that are satisfied, i.e. $C_{(u,v)}(\varphi(u), \varphi(v)) = 1$. In other words, find an assignment $\varphi : \{x_1, \dots, x_n\} \to [q]$ that maximizes $\sum_{(u,v)\in E} C_{(u,v)}(\varphi(u), \varphi(v))$. The value of an assignment is defined as the fraction of edges satisfied by it and the value of an instance is defined as the value of the optimal assignment.

A Max 2-CSP instance $(q, V, E, \{C_e\}_{e\in E})$ is called δ-dense if the graph (V, E) is δ-dense. Throughout the paper, we use n to denote the number of vertices (variables) $|V|$ and N to denote nq, which can be viewed as the size of the problem.

Free games and projection games are specific classes of Max 2-CSP, which can be defined as follows. Note that n, N, density and value are defined in a similar fashion for free games and projection games as well.

▶ **Definition 4.** A free game $(q, A, B, \{C_{a,b}\}_{(a,b)\in A\times B})$ consists of

- Two sets A, B of equal size, and
- for $(a, b) \in A \times B$, a constraint $C_{a,b} : [q]^2 \to \{0, 1\}$.

The goal is to find an assignment $\varphi : A \cup B \to [q]$ that maximizes the number of edges $(a, b) \in A \times B$ that are satisfied, i.e., $C_{a,b}(\varphi(a), \varphi(b)) = 1$.

▶ **Definition 5.** A projection game $(q, A, B, E, \{\pi_e\}_{e\in E})$ consists of

- a simple bipartite graph (A, B, E), and
- for each edge $e = (a, b) \in E$, a "projection" $\pi_e : [q] \to [q]$.

The goal is to find an assignment to the vertices $\varphi : A \cup B \to [q]$ that maximizes the number of edges $e = (a, b)$ that are satisfied, i.e., $\pi_e(\varphi(a)) = \varphi(b)$.

Both free games and projection games can be viewed as special cases of Max 2-CSP. More specifically, free games are simply Max 2-CSPs on complete balanced bipartite graphs.

For projection games, one can view π_e as a constraint $C_e : [q]^2 \to \{0, 1\}$ where $C_e(\sigma_u, \sigma_v) = 1$ if and only if $\pi_e(\sigma_u) = \sigma_v$. In other words, projection game is Max 2-CSP on bipartite graph where an assignment to the endpoint in A of an edge determines the assignment to the endpoint in B.

For convenience, we will define the notation of "optimal assignment" for Max 2-CSP intuitively as follows.

▶ **Definition 6.** For a Max 2-CSP instance $(q, V, E, \{C_e\}_{e\in E})$, for each vertex $u \in V$, let σ_u^{OPT} be the assignment to u in an assignment to vertices that satisfies maximum number of edges, i.e., $\varphi(u) = \sigma_u^{OPT}$ is the assignment that maximizes $\sum_{(u,v)\in E} C_{(u,v)}(\varphi(u), \varphi(v))$. In short, we will sometimes refer to this as "the optimal assignment".

Note that since projection games and free games are families of Max 2-CSP, the above definition also carries over when we discuss them.

Lastly, we define Densest k-Subgraph.

▶ **Definition 7.** In the Densest k-Subgraph problem, the input is a simple graph $G = (V, E)$ of $N = |V|$ vertices. The goal is to find a subgraph of size k that contain maximum number of edges.

3 Summary of Results

We are finally ready to describe our results and how they relate to the previous results. We will start with the main theorem on approximating high-value dense Max 2-CSP.

▶ **Theorem 8** (Main Theorem). *For every constant $\gamma > 0$, there exists a polynomial-time algorithm that, given a δ-dense* Max 2-CSP *instance of value λ, produces an assignment of value $\Omega((\delta\lambda)^{O(1/\gamma)}N^{-\gamma})$ for the instance.*

Note that, when $\delta, \lambda = N^{-o(1)}$, by choosing $\gamma < \varepsilon$, the algorithm can achieve $O((nq)^{\varepsilon})$ approximation ratio for any constant $\varepsilon > 0$.

Since every free game is 1/2-dense, Theorem 8 immediately implies the following corollary.

▶ **Corollary 9.** *For every constant $\gamma > 0$, there exists a polynomial-time algorithm that, given a free game of value λ, produces an assignment of value $\Omega(\lambda^{O(1/\gamma)}N^{-\gamma})$ for the instance.*

Again, note that when $\lambda = N^{-o(1)}$, the algorithm can achieve $O((nq)^{\varepsilon})$ approximation ratio for any constant $\varepsilon > 0$.

The next result is a similar algorithm for projection games on sufficiently dense random graphs as stated below.

▶ **Theorem 10.** *For every constant $\gamma > 0$, there exists a polynomial-time algorithm that, given a satisfiable projection game on a random bipartite graph $(A, B, E) \sim \mathcal{G}(n/2, n/2, p)$ for any $p \geq 10\sqrt{\log n/n}$, produces an assignment of value $\Omega(N^{-\gamma})$ for the instance with probability $1 - o(1)$.*

Note that $\mathcal{G}(n/2, n/2, p)$ is defined in Erdős-Rényi fashion, i.e., the graph contains $n/2$ vertices on each side and, each pair of left and right vertices is included as an edge with probability p independently.

In addition, it is worth noting here that the required density for projection games is much lower than that of Max 2-CSP; our Max 2-CSP algorithm requires the degree to be $\Omega(n/N^{-o(1)})$ whereas the projection games algorithm requires only $\tilde{\Omega}(\sqrt{n})$.

As stated earlier, we are unaware of any non-trivial polynomial-time algorithm for dense Max 2-CSP, free games, or projection games on dense random graphs prior to our algorithm.

Next, we state our analogous result for Densest k-Subgraph.

▶ **Corollary 11.** *For every constant $\gamma > 0$, there exists a polynomial-time algorithm that, given a graph $G = (V, E)$ on N vertices such that its densest subgraph with k vertices is δ-dense, produces a subgraph of k vertices that is $\Omega(\delta^{O(1/\gamma)}N^{-\gamma})$-dense with high probability.*

Note that the density condition is on the optimal solution, not the given graph G. The condition and the algorithm are exactly the same as that of [11] and [18]. However, the techniques are substantially different. While [11] deals combinatorially directly with the given graph G and [18] employs subsampling technique, we simply use our algorithm from Theorem 8 together with a simple reduction from Densest k-Subgraph to Max 2-CSP due to Charikar et al. [10].

Lastly, we also give a quasi-polynomial time approximation scheme for satisfiable dense Max 2-CSP as described formally below.

▶ **Corollary 12** (QPTAS for Dense Max 2-CSP). *For any $1 \geq \varepsilon > 0$, there exists an $(1+\varepsilon)$-approximation algorithm for satisfiable δ-dense* Max 2-CSP *that runs in time $N^{O(\varepsilon^{-1}\delta^{-1}\log N)}$.*

Comparing to the known algorithms, our QPTAS runs faster than QPTASs from [2, 3, 4], each of which takes at least $N^{O(\varepsilon^{-2}\delta^{-1}\log N)}$ time. However, while our algorithm works only for satisfiable instances, the mentioned algorithms work for unsatisfiable instances as well but with an additive error of ε in value instead of the usual multiplicative guarantee of $(1+\varepsilon)$.

4 Proof of The Main Theorem

In this section, we prove the main theorem. In order to do so, we will first show that we do not have to worry about the density δ at all, i.e., it is enough for us to prove the following lemma.

▶ **Lemma 13.** *For every $\gamma > 0$, there exists a polynomial-time algorithm that, given a free game $(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B})$ of value λ', produces an assignment of value $\lambda'^{O(1/\gamma)}q^{-\gamma}$ for the instance.*

The proof of the main theorem based on the lemma above is shown below.

Proof of Theorem 8 based on Lemma 13. The proof is based on putting in "dummy edges" where the constraints are always false regardless of the assignment to make the game more dense. More specifically, given a Max 2-CSP instance $(q, V, E, \{C_e\}_{e\in E})$ of value λ and density δ, we construct a free game $(q', A, B, \{C'_{(a,b)}\}_{(a,b)\in A\times B})$ as follows:

- Let A, B be copies of V and let $q' = q$.
- For each $a \in A$ and $b \in B$, let $C'_{(a,b)} = C_{(a,b)}$ if $(a, b) \in E$. Otherwise, let $C'_{(a,b)} := 0$.

It is not hard to see that, if we assign the optimal assignment of the original instance to the free game, then $\delta\lambda n^2$ edges are satisfied where $n = |V|$. In other words, the value of the free game is at least $\delta\lambda$. Thus, from Lemma 13, for any constant γ, we can find an assignment $\varphi' : A \cup B \to [q']$ of value at least $(\delta\lambda)^{O(1/\gamma)}q^{-\gamma}$ for the free game.

We create an assignment $\varphi : V \to [q]$ based on φ' as follows. For each vertex $v \in V$, let $a_v \in A$ and $b_v \in B$ be the vertices corresponding to v in the free game. Set $\varphi(v)$ to be either $\varphi'(a_v)$ or $\varphi'(b_v)$ with equal probability.

From the above construction, the expected number of edges satisfied by φ in the Max 2-CSP instance is

$$\begin{aligned}
&\mathbb{E}\left[\sum_{(u,v)\in E} C_{(u,v)}(\varphi(u),\varphi(v))\right] \\
&\qquad = \sum_{(u,v)\in E} \mathbb{E}\left[C_{(u,v)}(\varphi(u),\varphi(v))\right] \\
\text{(From our choice of } \varphi) &= \sum_{(u,v)\in E} \frac{1}{4}\left(\sum_{\sigma_u\in\{\varphi'(a_u),\varphi'(b_u)\}}\ \sum_{\sigma_v\in\{\varphi'(a_v),\varphi'(b_v)\}} C_{(u,v)}(\sigma_u,\sigma_v)\right) \\
&\geq \sum_{(u,v)\in E} \frac{1}{4}\left(C_{(u,v)}(\varphi'(a_u),\varphi'(b_v)) + C_{(u,v)}(\varphi'(b_u),\varphi'(a_v))\right) \\
&= \frac{1}{4}\sum_{(u,v)\in E}\left(C_{(u,v)}(\varphi'(a_u),\varphi'(b_v)) + C_{(u,v)}(\varphi'(b_u),\varphi'(a_v))\right) \\
\text{(From definition of } C') &= \frac{1}{4}\sum_{(a,b)\in A\times B} C'_{(a,b)}(\varphi'(a),\varphi'(b)).
\end{aligned}$$

Observe that

$$\sum_{(a,b)\in A\times B} C'_{(a,b)}(\varphi'(a), \varphi'(b))$$

is the value of φ' with respect to the free game, which is at least $(\delta\lambda)^{O(1/\gamma)}q^{-\gamma}$. As a result, we can conclude that φ is of expected value at least $\frac{1}{4}(\delta\lambda)^{O(1/\gamma)}q^{-\gamma} = \Omega((\delta\lambda)^{O(1/\gamma)}N^{-\gamma})$ with respect to the instance $(q, V, E, \{C_e\}_{e\in E})$.

Lastly, we note that while the algorithm above is non-deterministic, the standard derandomization technique via conditional probability can be employed to make the algorithm deterministic without affecting the guarantee on the value of φ, which completes our proof for the main theorem. ◀

Now, we finally give the proof for Lemma 13. As mentioned in the introduction, the main idea of the proof is a trade-off between the greedy algorithm and the choice reduction algorithm. In other words, either the greedy assignment has high value, or we can reduce the number of candidates of the optimal assignment for many variables significantly by assigning only one variable. This argument needs to be applied multiple times to arrive at the result; the more variables we iterate on, the better guarantee we get on the output assignment value.

For the purpose of analysis, we will define our algorithm recursively and use induction to show that the output assignment meets the desired criteria.

Proof of Lemma 13. First, let us define notation that we will use throughout the proof. For a free game $(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B})$, define E^{OPT} to be the set of edges satisfied by $\{\sigma_u^{OPT}\}_{u\in V}$. In other words, $E^{OPT} = \{(u,v) \in E \mid C_{(u,v)}(\sigma_u^{OPT}, \sigma_v^{OPT}) = 1\}$. We also define $\Gamma^{OPT}(u)$ to be the neighborhood of u with respect to (V, E^{OPT}) and let d_u^{OPT} be the degree of u in (V, E^{OPT}), i.e., $d_u^{OPT} = |\Gamma^{OPT}(u)|$. In addition, let $n' = n/2$ be the size of A and B.

We will prove the lemma by induction. Let $P(i)$ represent the following statement: there exists an $O\left((nq)^{2i}\right)$-time algorithm Approx-FreeGame$_i(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S_b\}_{b\in B})$ that takes in a free game instance $(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B})$ of value λ' and a reduced alphabet set S_b for every $b \in B$, and produces an assignment that satisfies at least

$$n'\left(\sum_{b\in B}\left(\frac{d_b^{OPT}}{n'}\right)^{\frac{i+1}{2}}\left(\frac{1}{|S_b|}\right)^{\frac{1}{i}} 1_{\sigma_b^{OPT}\in S_b}\right)$$

edges. Note here that $1_{\sigma_b^{OPT}\in S_b}$ denotes an indicator variable for whether $\sigma_b^{OPT} \in S_b$. Moreover, for convenience, we use the expression $(1/|S_b|)^{\frac{1}{i}}\, 1_{\sigma_b^{OPT}\in S_b}$ to be represent zero when $S_b = \emptyset$.

Before we proceed to the induction, let us note why $P(i)$ implies the lemma. By setting $i = \lceil 1/\gamma \rceil$ and $S_b = [q]$ for every $b \in B$, since $\sigma_b^{OPT} \in S_b$ for every $b \in B$, the number of edges satisfied by the output assignment of the algorithm in $P(i)$ is at least

$$\begin{aligned} n'\sum_{b\in B}\left(\frac{d_b^{OPT}}{n'}\right)^{\frac{i+1}{2}}\left(\frac{1}{q}\right)^{\frac{1}{i}} &= n'\frac{1}{q^{1/i}}\left(\sum_{b\in B}\left(\frac{d_b^{OPT}}{n'}\right)^{\frac{i+1}{2}}\right) \\ \text{(From Hölder's inequality)} &\geq \frac{(n')^2}{q^{1/i}}\left(\frac{1}{n'}\sum_{b\in B}\frac{d_b^{OPT}}{n'}\right)^{\frac{i+1}{2}} \\ &= \frac{(n')^2}{q^{1/i}}\left(\frac{|E^{OPT}|}{(n')^2}\right)^{\frac{i+1}{2}} \end{aligned}$$

$$\text{(Since } |E^{OPT}|/(n')^2 \text{ is the value of the instance)} = \frac{(n')^2}{q^{1/i}}(\lambda')^{\frac{i+1}{2}}$$
$$\text{(From our choice of } i) \geq (n')^2 \frac{\lambda'^{O(1/\gamma)}}{q^\gamma},$$

which is the statement of the lemma.

Now, we finally show that $P(i)$ is true for every $i \in \mathbb{N}$ by induction.

Base Case. The algorithm Approx-FreeGame$_1(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S_b\}_{b\in B})$ is a greedy algorithm that works as follows:

1. For each $a \in A$, assign $\sigma_a^* \in S_a$ that maximizes $\sum_{b\in B} \frac{1}{|S_b|}\left(\sum_{\sigma_b\in S_b} C_{(a,b)}(\sigma_a, \sigma_b)\right)$ to it.
2. For each $b \in B$, assign $\sigma_b^* \in S_b$ that maximizes the number of edges satisfied, i.e., $\sum_{a\in A} C_{(a,b)}(\sigma_a^*, \sigma_b)$, to it.

It is obvious that the algorithm runs in $O(n^2q^2)$ time as desired.

Next, we need to show that the algorithm gives an assignment that satisfies at least

$$n'\left(\sum_{b\in B}\left(\frac{d_b^{OPT}}{n'}\right)\left(\frac{1}{|S_b|}\right)1_{\sigma_b^{OPT}\in S_b}\right) = \sum_{b\in B}\frac{d_b^{OPT}}{|S_b|}1_{\sigma_b^{OPT}\in S_b}$$

edges.

To prove this, observe that, from our choice of σ_b^*, the number of satisfied edges by the output assignment can be bounded as follows.

$$\begin{aligned}
\sum_{b\in B}\sum_{a\in A} C_{(a,b)}(\sigma_a^*, \sigma_b^*) &\geq \sum_{b\in B}\frac{1}{|S_b|}\sum_{\sigma_b\in S_b}\left(\sum_{a\in A} C_{(a,b)}(\sigma_a^*, \sigma_b)\right)\\
&= \sum_{a\in A}\sum_{b\in B}\frac{1}{|S_b|}\left(\sum_{\sigma_b\in S_b} C_{(a,b)}(\sigma_a^*, \sigma_b)\right)\\
\text{(From our choice of } \sigma_a^*) &\geq \sum_{a\in A}\sum_{b\in B}\frac{1}{|S_b|}\left(\sum_{\sigma_b\in S_b} C_{(a,b)}(\sigma_a^{OPT}, \sigma_b)\right)\\
&\geq \sum_{a\in A}\sum_{b\in B}\frac{1}{|S_b|}C_{(a,b)}(\sigma_a^{OPT}, \sigma_b^{OPT})1_{\sigma_b^{OPT}\in S_b}\\
&= \sum_{b\in B}\sum_{a\in A}\frac{1}{|S_b|}C_{(a,b)}(\sigma_a^{OPT}, \sigma_b^{OPT})1_{\sigma_b^{OPT}\in S_b}\\
\text{(From definition of } d_b^{OPT}) &= \sum_{b\in B}\frac{1}{|S_b|}d_b^{OPT}1_{\sigma_b^{OPT}\in S_b}\\
&= \sum_{b\in B}\frac{d_b^{OPT}}{|S_b|}1_{\sigma_b^{OPT}\in S_b}.
\end{aligned}$$

Thus, we can conclude that $P(1)$ is true.

Inductive Step. Let j be any positive integer. Suppose that $P(j)$ holds.

We will now describe Approx-FreeGame$_{j+1}$ based on Approx-FreeGame$_j$ as follows.

1. For each $a \in A$ and $\sigma_a \in S_a$, do the following:
   - **a.** For each $b \in B$, compute $S_b^{a,\sigma_a} = \{\sigma_b \in S_b \mid C_{(a,b)}(\sigma_a, \sigma_b) = 1\}$.
   - **b.** Call Approx-FreeGame$_j(q, A, B, \{C_{(a,b)}\}_{(a,b)\in A\times B}, \{S_b^{a,\sigma_a}\}_{b\in B})$. Let the output assignment be φ^{a,σ_a}.
2. Execute the following greedy algorithm:

  a. For each $a \in A$, assign $\sigma_a^* \in S_a$ to it that maximizes $\sum_{b \in B} \frac{1}{|S_b|} \left(\sum_{\sigma_b \in S_b} C_{(a,b)}(\sigma_a, \sigma_b)\right)$.
  b. For each $b \in B$, assign $\sigma_b^* \in S_b$ to it that maximizes the number of edges satisfied, i.e., maximizes $\sum_{a \in A} C_{(a,b)}(\sigma_a^*, \sigma_b)$.

3. Output an assignment among the greedy assignment and φ^{a,σ_a} for every a, σ_a that satisfies maximum number of edges.

Since every step except the Approx-FreeGame$_j(q, A, B, \{C_{(a,b)}\}_{(a,b) \in A \times B}, \{S_b\}_{b \in B})$ calls takes $O((nq)^2)$ time and we call Approx-FreeGame$_j$ only at most $(nq)^2$ times, we can conclude that the running time of Approx-FreeGame$_{j+1}$ is $O((nq)^{2j+2})$ as desired.

Define R to be $n'\left(\sum_{b \in B}\left(\frac{d_b^{OPT}}{n'}\right)^{\frac{j+2}{2}}\left(\frac{1}{|S_b|}\right)^{\frac{1}{j+1}} 1_{\sigma_b^{OPT} \in S_b}\right)$, our target number of edges we want to satisfy. The only thing left to show is that the assignment output from the algorithm indeed satisfies at least R edges. We will consider two cases.

First, if there exist $a \in A$ and $\sigma_a \in S_b$ such that the output assignment from Approx-FreeGame$_j(q, A, B, \{C_{(a,b)}\}_{(a,b) \in A \times B}, \{S_b^{a,\sigma_a}\}_{b \in B})$ satisfies at least R edges, then it is obvious that the output assignment of Approx-FreeGame$_{j+1}$ indeed satisfies at least R edges as well.

In the second case, for every $a \in A$ and $\sigma_a \in S_a$, the output assignment from Approx-FreeGame$_j(q, A, B, \{C_{(a,b)}\}_{(a,b) \in A \times B}, \{S_b^{a,\sigma_a}\}_{b \in B})$ satisfies less than R edges. For each $a \in A$, since the output assignment from Approx-FreeGame$_j(q, A, B, \{C_{(a,b)}\}_{(a,b) \in A \times B}, \{S_b^{a,\sigma_a^{OPT}}\}_{b \in B})$ satisfies less than R edges, we arrive at the following inequality:

$$\begin{aligned} R &> n'\left(\sum_{b \in B}\left(\frac{d_b^{OPT}}{n'}\right)^{\frac{j+1}{2}}\left(\frac{1}{|S_b^{a,\sigma_a^{OPT}}|}\right)^{\frac{1}{j}} 1_{\sigma_b^{OPT} \in S_b^{a,\sigma_a^{OPT}}}\right) \\ &\geq n'\left(\sum_{b \in \Gamma^{OPT}(a)}\left(\frac{d_b^{OPT}}{n'}\right)^{\frac{j+1}{2}}\left(\frac{1}{|S_b^{a,\sigma_a^{OPT}}|}\right)^{\frac{1}{j}} 1_{\sigma_b^{OPT} \in S_b^{a,\sigma_a^{OPT}}}\right). \end{aligned}$$

Now, observe that, for every $b \in \Gamma^{OPT}(a)$, we have $1_{\sigma_b^{OPT} \in S_b^{a,\sigma_a^{OPT}}} = 1_{\sigma_b^{OPT} \in S_b}$. This is because, from our definition of Γ^{OPT}, $C_{(a,b)}(\sigma_a^{OPT}, \sigma_b^{OPT}) = 1$ for every $b \in \Gamma^{OPT}(a)$, which means that, if σ_b^{OPT} is in S_b, then it remains in $S_b^{a,\sigma_a^{OPT}}$. Thus, the above inequality can be written as follows:

$$R > n'\left(\sum_{b \in \Gamma^{OPT}(a)}\left(\frac{d_b^{OPT}}{n'}\right)^{\frac{j+1}{2}}\left(\frac{1}{|S_b^{a,\sigma_a^{OPT}}|}\right)^{\frac{1}{j}} 1_{\sigma_b^{OPT} \in S_b}\right). \tag{1}$$

We will use inequality (1) later in the proof. For now, we will turn our attention to the number of edges satisfied by the greedy algorithm, which, from our choice of σ_b^*, can be bounded as follows:

$$\begin{aligned} \sum_{b \in B}\sum_{a \in A} C_{(a,b)}(\sigma_a^*, \sigma_b^*) &\geq \sum_{b \in B}\frac{1}{|S_b|}\sum_{\sigma_b \in S_b}\left(\sum_{a \in A} C_{(a,b)}(\sigma_a^*, \sigma_b)\right) \\ &= \sum_{a \in A}\sum_{b \in B}\frac{1}{|S_b|}\left(\sum_{\sigma_b \in S_b} C_{(a,b)}(\sigma_a^*, \sigma_b)\right) \\ \text{(From our choice of } \sigma_a^*) &\geq \sum_{a \in A}\sum_{b \in B}\frac{1}{|S_b|}\left(\sum_{\sigma_b \in S_b} C_{(a,b)}(\sigma_a^{OPT}, \sigma_b)\right) \end{aligned}$$

$$\begin{aligned}
(\text{Since } C_{(a,b)}(\sigma_a^{OPT}, \sigma_b) = 1 \text{ for every } \sigma_b \in S_b^{a,\sigma_a^{OPT}}) &\geq \sum_{a \in A} \sum_{b \in B} \frac{1}{|S_b|} |S_b^{a,\sigma_a^{OPT}}| \\
&= \sum_{a \in A} \sum_{b \in B} \frac{|S_b^{a,\sigma_a^{OPT}}|}{|S_b|} \\
&\geq \sum_{a \in A} \sum_{b \in \Gamma^{OPT}(a)} \frac{|S_b^{a,\sigma_a^{OPT}}|}{|S_b|}.
\end{aligned}$$

Moreover, from inequality (1), we can derive the following inequalities:

$$\begin{aligned}
&R^j \left(\sum_{a \in A} \sum_{b \in \Gamma^{OPT}(a)} \frac{|S_b^{a,\sigma_a^{OPT}}|}{|S_b|} \right) \\
&= \sum_{a \in A} R^j \left(\sum_{b \in \Gamma^{OPT}(a)} \frac{|S_b^{a,\sigma_a^{OPT}}|}{|S_b|} \right) \\
(\text{From (1)}) &\geq (n')^j \sum_{a \in A} \left(\sum_{b \in \Gamma^{OPT}(a)} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j+1}{2}} \left(\frac{1}{|S_b^{a,\sigma_a^{OPT}}|} \right)^{\frac{1}{j}} 1_{\sigma_b^{OPT} \in S_b} \right)^j \\
&\qquad \cdot \left(\sum_{b \in \Gamma^{OPT}(a)} \frac{|S_b^{a,\sigma_a^{OPT}}|}{|S_b|} \right) \\
(\text{Hölder's inequality}) &\geq (n')^j \sum_{a \in A} \left(\sum_{b \in \Gamma^{OPT}(a)} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j}{2}} \left(\frac{1}{|S_b|} \right)^{\frac{1}{j+1}} 1_{\sigma_b^{OPT} \in S_b} \right)^{j+1}
\end{aligned}$$

By applying Hölder's inequality once again, the last term above is at least

$$\begin{aligned}
&(n')^j n' \left(\frac{1}{n'} \sum_{a \in A} \sum_{b \in \Gamma^{OPT}(a)} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j}{2}} \left(\frac{1}{|S_b|} \right)^{\frac{1}{j+1}} 1_{\sigma_b^{OPT} \in S_b} \right)^{j+1} \\
&= \left(\sum_{b \in B} \sum_{a \in \Gamma^{OPT}(b)} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j}{2}} \left(\frac{1}{|S_b|} \right)^{\frac{1}{j+1}} 1_{\sigma_b^{OPT} \in S_b} \right)^{j+1} \\
(\text{Since } d_b^{OPT} = |\Gamma^{OPT}(b)|) &= \left(\sum_{b \in B} d_b^{OPT} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j}{2}} \left(\frac{1}{|S_b|} \right)^{\frac{1}{j+1}} 1_{\sigma_b^{OPT} \in S_b} \right)^{j+1} \\
&= \left(n' \sum_{b \in B} \left(\frac{d_b^{OPT}}{n'} \right)^{\frac{j+2}{2}} \left(\frac{1}{|S_b|} \right)^{\frac{1}{j+1}} 1_{\sigma_b^{OPT} \in S_b} \right)^{j+1} \\
&= R^{j+1}.
\end{aligned}$$

Hence, we can conclude that

$$\sum_{a \in A} \sum_{b \in \Gamma^{OPT}(a)} \frac{|S_b^{a,\sigma_a^{OPT}}|}{|S_b|} \geq R.$$

In other words, our greedy algorithm satisfies at least R edges, which means that $P(j+1)$ is also true for this second case.

As a result, $P(i)$ is true for every positive integer i, which completes the proof for Lemma 13. ◀

5 Approximation Algorithm for Projection Games

In this section, we will present our approximation algorithm for projection games. The main idea of this algorithm is a reduction from projection games on dense random graphs to free games, which we use together with the approximation algorithm for free games from Corollary 9 above to prove Theorem 10. The reduction's properties can be stated formally as follows.

▶ **Lemma 14.** *There is a polynomial-time reduction from a satisfiable projection game $(q, A, B, E, \{\pi_e\}_{e\in E})$ where (A, B, E) is sampled from a distribution $\mathcal{G}(n/2, n/2, p)$ where $p \geq 10\sqrt{\log n/n}$ to a satisfiable free game instance $(q', A', B', \{C_{(a,b)}\}_{(a,b)\in A'\times B'})$ such that, with probability $1 - o(1)$,*
1. *$|A'|, |B'| \leq |A|$ and $q' \leq q$, and*
2. *For any $1 \geq \varepsilon \geq 0$, given an assignment $\varphi' : A' \cup B' \to [q']$ to the free game instance of value ε, one can construct an assignment $\varphi : A \cup B \to [q]$ for the projection game of value $\Omega(\varepsilon)$ in polynomial time.*

Before we describe the reduction, we give a straightforward proof for Theorem 10 based on the above lemma.

Proof of Theorem 10 based on Lemma 14. The proof is simple. First, we use the reduction from Lemma 14 to transform a projection game on dense graph to a free game. Since the approximation ratio deteriorates by only constant factor with probability $1 - o(1)$ in the reduction, we can use the approximation algorithm from Corollary 9 with $\lambda = 1$, which gives us an assignment of value at least $\Omega(1/N^\gamma)$. ◀

To prove the reduction lemma, we use the following two properties of random graphs. We do not prove the lemmas as they follow from a standard Chernoff bound.

▶ **Lemma 15.** *When $p \geq 10\sqrt{\log n/n}$, with probability $1 - o(1)$, every vertex in $G \sim \mathcal{G}(n/2, n/2, p)$ has degree between $np/10$ and $10np$.*

▶ **Lemma 16.** *In $G \sim \mathcal{G}(n/2, n/2, p)$ with $p \geq 10\sqrt{\log n/n}$, with probability $1 - o(1)$, every pair of vertices a, a' on the left has at least $np^2/10$ common neighbors.*

Now, we are ready to prove the reduction lemma. Roughly speaking, the idea of the proof is to "square" the projection game, i.e., use A as the vertices of the new game and, for each pair of vertices in A, add a constriant between them based on their constraints with their common neighbors in the projection game. This can be formalized as follows.

Proof of Lemma 14. The reduction proceeds as follows.
1. Partition A into A_1, A_2 of equal sizes. Then, set $A' \leftarrow A_1, B' \leftarrow A_2$ and $q' \leftarrow q$.
2. For each $a_1 \in A_1, a_2 \in A_2, \sigma_{a_1}, \sigma_{a_2} \in [q]$, let $C_{(a_1,a_2)}(\sigma_{a_1}, \sigma_{a_2})$ to be one if and only if these two assignments agree on every $b \in \Gamma(a_1) \cap \Gamma(a_2)$. In other words, $C_{(a_1,a_2)}(\sigma_{a_1}, \sigma_{a_2}) = 1$ if and only if $\pi_{(a_1,b)}(\sigma_{a_1}) = \pi_{(a_2,b)}(\sigma_{a_2})$ for every $b \in \Gamma(a_1) \cap \Gamma(a_2)$.

It is obvious that the reduction runs in polynomial time, the first condition holds, and the new game is satisfiable. Thus, we only need to prove that, with probability $1 - o(1)$, the second condition is indeed true.

To show this, we present a simple algorithm that, given an assignment $\varphi' : A' \cup B' \to [q']$ of the free game instance of value ε, output an assignment $\varphi : A \cup B \to [q]$ of the projection game of value $\Omega(\varepsilon)$. The algorithm works greedily as follows.

1. For each $a \in A$, let $\varphi(a) \leftarrow \varphi'(a)$.
2. For each $b \in B$, pick $\varphi(b) = \sigma_b^*$ to be the assignment to b that satisfies maximum number of edges, i.e., maximize $|\{a \in \Gamma(b) \mid \pi_{(a,b)}(\varphi(a)) = \sigma_b\}|$.

Trivially, the algorithm runs in polynomial time. Thus, we only need to prove that, with probability $1 - o(1)$, the produced assignment is of value at least $\Omega(\varepsilon)$. To prove this, we will use the properties from Lemma 15 and Lemma 16, which holds with probability $1 - o(1)$.

The number of satisfied edges can be written as follows.

$$\sum_{b\in B}\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi(a))=\varphi(b)} = \sum_{b\in B}\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b^*}.$$

Let d_u be the degree of u in (A, B, E) for every $u \in A \cup B$, i.e. $d_u = |\Gamma(u)|$. We can further rearrange the above expression as follows.

$$\begin{aligned}
&\sum_{b\in B}\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b^*} \\
&= \sum_{b\in B}\left[\frac{1}{d_b}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b^*}\right) d_b\right] \\
&= \sum_{b\in B}\left[\frac{1}{d_b}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b^*}\right)\left(\sum_{a\in\Gamma(b)} 1\right)\right] \\
&= \sum_{b\in B}\left[\frac{1}{d_b}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b^*}\right)\left(\sum_{a\in\Gamma(b)}\sum_{\sigma_b\in[q]} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b}\right)\right] \\
&= \sum_{b\in B}\left[\frac{1}{d_b}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b^*}\right)\left(\sum_{\sigma_b\in[q]}\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b}\right)\right] \\
&= \sum_{b\in B}\left[\frac{1}{d_b}\sum_{\sigma_b\in[q]}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b^*}\right)\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b}\right)\right] \\
\left(\begin{matrix}\text{From the choice}\\ \text{of } \sigma_b^*\end{matrix}\right) &\geq \sum_{b\in B}\left[\frac{1}{d_b}\sum_{\sigma_b\in[q]}\left(\sum_{a\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b}\right)^2\right] \\
&= \sum_{b\in B}\left(\frac{1}{d_b}\sum_{\sigma_b\in[q]}\sum_{a,a'\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b} 1_{\pi_{(a',b)}(\varphi'(a'))=\sigma_b}\right) \\
&= \sum_{b\in B}\left(\frac{1}{d_b}\sum_{a,a'\in\Gamma(b)}\sum_{\sigma_b\in[q]} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b} 1_{\pi_{(a',b)}(\varphi'(a'))=\sigma_b}\right)
\end{aligned}$$

Observe that $\sum_{\sigma_b\in[q]} 1_{\pi_{(a,b)}(\varphi'(a))=\sigma_b} 1_{\pi_{(a',b)}(\varphi'(a'))=\sigma_b} = 1_{\pi_{(a,b)}(\varphi'(a))=\pi_{(a',b)}(\varphi'(a'))}$. Thus, the number of satisfied edges is at least

$$\sum_{b\in B}\left(\frac{1}{d_b}\sum_{a,a'\in\Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a))=\pi_{(a',b)}(\varphi'(a'))}\right).$$

Moreover, from Lemma 15, $d_b \le 10np$ for every $b \in B$ with probability $1 - o(1)$. This implies that, with probability $1 - o(1)$, the output assignment satisfied at least

$$\frac{1}{10np} \sum_{b \in B} \sum_{a,a' \in \Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a)) = \pi_{(a',b)}(\varphi'(a'))}$$

edges.

We can further reorganize this quantity as follows.

$$\begin{aligned}
&\frac{1}{10np} \sum_{b \in B} \sum_{a,a' \in \Gamma(b)} 1_{\pi_{(a,b)}(\varphi'(a)) = \pi_{(a',b)}(\varphi'(a'))} \\
&\qquad \ge \frac{1}{10np} \sum_{b \in B} \sum_{\substack{(a,a') \in A' \times B' \\ \text{s.t. } a,a' \in \Gamma(b)}} 1_{\pi_{(a,b)}(\varphi'(a)) = \pi_{(a',b)}(\varphi'(a'))} \\
&\qquad = \frac{1}{10np} \sum_{(a,a') \in A' \times B'} \sum_{b \in \Gamma(a) \cap \Gamma(a')} 1_{\pi_{(a,b)}(\varphi'(a)) = \pi_{(a',b)}(\varphi'(a'))}.
\end{aligned}$$

Now, observe that, from its definition, if $C_{(a,a')}(\varphi'(a), \varphi'(a'))$ is one, then $1_{\pi_{(a,b)}(\varphi'(a)) = \pi_{(a',b)}(\varphi'(a'))}$ is also one for every $b \in \Gamma(a) \cap \Gamma(a')$. Thus, we have

$$\begin{aligned}
&\frac{1}{10np} \sum_{(a,a') \in A' \times B'} \sum_{b \in \Gamma(a) \cap \Gamma(a')} 1_{\pi_{(a,b)}(\varphi'(a)) = \pi_{(a',b)}(\varphi'(a'))} \\
&\ge \frac{1}{10np} \sum_{(a,a') \in A' \times B'} \sum_{b \in \Gamma(a) \cap \Gamma(a')} C_{(a,a')}(\varphi'(a), \varphi'(a')) \\
&= \frac{1}{10np} \sum_{(a,a') \in A' \times B'} |\Gamma(a) \cap \Gamma(a')| C_{(a,a')}(\varphi'(a), \varphi'(a')).
\end{aligned}$$

From Lemma 16, with probability $1 - o(1)$, $|\Gamma(a) \cap \Gamma(a')| \ge np^2/10$ for every $(a,a') \in A' \times B'$. Hence, we can conclude that the above expression is, with probability $1 - o(1)$, at least

$$\frac{1}{10np} \sum_{(a,a') \in A' \times B'} \frac{np^2}{10} C_{(a,a')}(\varphi'(a), \varphi'(a')) = \frac{p}{100} \sum_{(a,a') \in A' \times B'} C_{(a,a')}(\varphi'(a), \varphi'(a')).$$

Next, note that $\sum_{(a,a') \in A' \times B'} C_{(a,a')}(\varphi'(a), \varphi'(a'))$ is the number of edges satisfied by φ' in the free game, which is at least $\varepsilon |A'||B'| = \varepsilon n^2/16$. Thus, we have

$$\frac{p}{100} \sum_{(a,a') \in A' \times B'} C_{(a,a')}(\varphi'(a), \varphi'(a')) \ge \frac{\varepsilon n^2 p}{1600}.$$

Finally, again from Lemma 15, the total number of edges is at most $5n^2 p$ with probability $1 - o(1)$. As a result, with probability $1 - o(1)$, the algorithm outputs an assignment that satisfies at least $\frac{\varepsilon}{8000} = \Omega(\varepsilon)$ fraction of edges of the projection game instance as desired. ◀

6 Approximation Algorithm for Densest k-Subgraph

The main goal of this section is to prove Corollary 11. As stated previously, we simply use our algorithm from Theorem 8 together with a reduction from Max 2-CSP to DkS from [10]. First, let us start by stating the reduction from Theorem 8, which we rephrase as follows.

▶ **Lemma 17** ([10]). *There exists a randomized polynomial-time algorithm that, given a graph G of N vertices and an integer $k \leq N$, produces an instance $(q, V, E, \{C_e\}_{e \in E})$ of* Max 2-CSP *such that*

- $q \leq N, n = k$, *and*
- *any solution to the instance can be translated in polynomial time to a subgraph of G of k vertices such that the number of edges in the subgraph equals to the number of edges satisfied by the* Max 2-CSP *solution, and*
- *with constant probability, the number of edges satisfied by the optimal solution to the instance is at least $1/100$ times the number of edges in the densest k-subgraph of G.*

We will not show the proof of Lemma 17 here; please refer to Theorem 6 from [10] for the proof. Instead, we will now show how to use the reduction to arrive at the proof of Corollary 11.

Proof of Corollary 11. First, we note that, to prove Corollary 11, it is enough to find a randomized polynomial-time algorithm with similar approximation guarantee to that in Corollary 11 except that the probability of success is a constant (instead of high probability as stated in Corollary 11). This is because we can then repeatedly run this algorithm $\Theta(\log n)$ times and produce the desired result.

The algorithm proceeds as follows:

1. Use the reduction from Lemma 17 on the input graph G and k to produce $(q, V, E, \{C_e\}_{e \in E})$.
2. Run the algorithm from Theorem 8 on $(q, V, E, \{C_e\}_{e \in E})$.
3. Transform the assignment from previous step according to Lemma 17 and output the result.

From the property of the reduction, we know that, with constant probability, the optimal assignment to $(q, V, E, \{C_e\}_{e \in E})$ satisfies $\Omega(\delta k^2)$ edges. If this is the case, we can conclude that the density of (V, E) is $\Omega(\delta)$ and, similarly, that the value of the instance is $\Omega(\delta)$. As a result, the output assignment from step 2 has value at least $\Omega(\delta^{O(1/\gamma)} N^{-\gamma})$. Since the reduction from Lemma 17 preserves the optimum, our algorithm produces a subgraph of density at least $\Omega(\delta^{O(1/\gamma)} N^{-\gamma})$ as well, which concludes our proof for this corollary. ◀

7 QPTAS for Dense Max 2-CSPs

At first glance, it seems that the QPTAS would follow easily for our main theorem. This, however, is not the case as the algorithm in the main theorem always loses at least a constant factor. Instead, we need to give an algorithm that is similar to that of the main theorem but have a stronger guarantee in approximation ratio for satisfiable instances, which can be stated as follows.

▶ **Lemma 18.** *For every positive integer $i > 0$, there exists an $O\left((nq)^{O(i)}\right)$-time algorithm that, for any satisfiable* Max 2-CSP *instance on the complete graph, produces an assignment of value at least $1/q^{1/i}$.*

Lemma 18 can be viewed as a special case of the main theorem when the graph is complete. However, it should be noted that Lemma 18 is more exact in the sense that the guaranteed lower bound of the value of the output assignment is not asymptotic. The proof of this lemma is also similar to that of Lemma 13 except that we need slightly more complicated algorithm and computation to deal with the fact that the underlying graph is not bipartite.

Proof of Lemma 18. We will prove the lemma by induction. Note that throughout the proof, we will not worry about the randomness that the algorithm employs; it is not hard to see that the random assignment algorithms described below can be derandomized via greedy approach so that the approximation guarantees are as good as the expected guarantees of the randomized ones and that we still end up with the same asymptotic running time.

Let $P(i)$ represent the following statement: there exists an $O\left((nq)^{3i}\right)$-time algorithm $\textsc{Approx-CompleteGame}_i(q, V, E, \{C_e\}_{e\in E}, \{S_u\}_{u\in V})$ that takes in a satisfiable $\textsc{Max}$ 2-CSP instance $(q, V, E, \{C_e\}_{e\in V})$ where (V, E) is a complete graph and a reduced alphabet set S_u for every $u \in U$ such that, if $\sigma_u^{OPT} \in S_u$ for every $u \in V$, then the algorithm outputs an assignment of value at least $\left(\prod_{u\in V} \frac{1}{|S_u|}\right)^{\frac{1}{ni}}$.

Observe that $P(i)$ implies the lemma by simply setting $S_u = [q]$ for every $u \in V$.

Base Case. The algorithm $\textsc{Approx-CompleteGame}_1(q, V, E, \{C_e\}_{e\in E}, \{S_u\}_{u\in V})$ is a simple random assignment algorithm. However, before we randomly pick the assignment, we need to first discard the alphabets that we know for sure are not optimal. More specifically, $\textsc{Approx-CompleteGame}_1(q, V, E, \{C_e\}_{e\in E}, \{S_u\}_{u\in V})$ works as follows.

1. While there exist $u, v \in U$ and $\sigma_u \in S_u$ such that $C_{(u,v)}(\sigma_u, \sigma_v) = 0$ for every $\sigma_v \in S_v$, remove σ_u from S_u.
2. For each $u \in V$, pick $\varphi(u)$ independently and uniformly at random from S_u. Output φ.

It is obvious that the algorithm runs in $O(n^3q^3)$ time as desired.

Now, we will show that, if $\sigma_u^{OPT} \in S_u$ for every $u \in V$, then the algorithm gives an assignment that is of value at least $\left(\prod_{u\in V} \frac{1}{|S_u|}\right)^{\frac{1}{n}}$ in expectation.

First, observe that σ_u^{OPT} remains in S_u after step 1 for every $u \in V$. This is because $C_{(u,v)}(\sigma_u^{OPT}, \sigma_v^{OPT}) = 1$ for every $v \neq u$.

Next, Consider the expected number of satisfied edges by the output assignment, which can be rearranged as follows:

$$\begin{aligned}\mathbb{E}\left[\sum_{(u,v)\in E} C_{(u,v)}(\varphi(u), \varphi(v))\right] &= \sum_{(u,v)\in E} \mathbb{E}\left[C_{(u,v)}(\varphi(u), \varphi(v))\right] \\ &= \sum_{(u,v)\in E} \frac{1}{|S_u||S_v|} \sum_{\sigma_u\in S_u} \sum_{\sigma_v\in S_v} C_{(u,v)}(\sigma_u, \sigma_v).\end{aligned}$$

From the condition of the loop in step 1, we know that after the loop ends, for each $\sigma_u \in S_u$, there must be at least one $\sigma_v \in S_v$ such that $C_{(u,v)}(\sigma_u, \sigma_v) = 1$. In other words,

$$\sum_{\sigma_u\in S_u} \sum_{\sigma_v\in S_v} C_{(u,v)}(\sigma_u, \sigma_v) \geq \sum_{\sigma_u\in S_u} 1 = |S_u|.$$

Similarly, we can also conclude that

$$\sum_{\sigma_u\in S_u} \sum_{\sigma_v\in S_v} C_{(u,v)}(\sigma_u, \sigma_v) \geq |S_v|.$$

Thus, we have

$$\sum_{\sigma_u\in S_u} \sum_{\sigma_v\in S_v} C_{(u,v)}(\sigma_u, \sigma_v) \geq \max\{|S_u|, |S_v|\}$$

for every $u \neq v$.

Hence, we can bound the expected number of satisfied edges as follows:

$$\begin{aligned}
\sum_{(u,v)\in E} \frac{1}{|S_u||S_v|} \sum_{\sigma_u\in S_u}\sum_{\sigma_v\in S_v} C_{(u,v)}(\sigma_u,\sigma_v) &\geq \sum_{(u,v)\in E} \frac{1}{|S_u||S_v|}\max\{|S_u|,|S_v|\} \\
&= \sum_{(u,v)\in E} \frac{1}{\min\{|S_u|,|S_v|\}} \\
&\geq \sum_{(u,v)\in E} \frac{1}{\sqrt{|S_u||S_v|}} \\
\text{(A.M. - G.M. inequality)} &\geq |E| \left(\prod_{(u,v)\in E} \frac{1}{\sqrt{|S_u||S_v|}}\right)^{\frac{1}{|E|}} \\
&= |E| \left(\prod_{(u,v)\in E} \frac{1}{\sqrt{|S_u||S_v|}}\right)^{\frac{2}{n(n-1)}} \\
\text{(Each } u\in V \text{ appears in exactly } n-1 \text{ edges)} &= |E| \left(\left(\prod_{u\in V}\frac{1}{|S_u|}\right)^{(n-1)/2}\right)^{\frac{2}{n(n-1)}} \\
&= |E| \left(\prod_{u\in V}\frac{1}{|S_u|}\right)^{1/n},
\end{aligned}$$

which implies that $P(1)$ is true as desired.

Inductive Step. Let j be any positive integer. Suppose that $P(j)$ holds.

We will now describe Approx-CompleteGame$_{j+1}$ based on Approx-CompleteGame$_j$ as follows.

1. Define R to be $\left(\prod_{u\in V}\frac{1}{|S_u|}\right)^{\frac{1}{n(j+1)}}$, our target value we want to achieve.
2. Run the following steps 2(a)i to 2(a)iv until no S_u is modified by neither step 2(a)iv nor step 2(a)ii.
   a. For each $u \in V$ and $\sigma_u \in S_u$, do the following:
      i. For each $v \in V$, compute $S_v^{u,\sigma_u} = \{\sigma_v \in S_v \mid C_{(u,v)}(\sigma_u,\sigma_v) = 1\}$. This is the set of reduced assignments of v if we assign σ_u to u. Note that when $v = u$, let $S_u = \{\sigma_u\}$.
      ii. If $S_v^{u,\sigma_u} = \emptyset$ for some $v \in V$, then remove σ_u from S_u and continue to the next u, σ_u pair.
      iii. Compute $R^{u,\sigma_u} = \left(\prod_{v\in V}\frac{1}{|S_v^{u,\sigma_u}|}\right)^{\frac{1}{nj}}$. If $R' < R$, continue to the next u, σ_u pair.
      iv. Execute Approx-CompleteGame$_j(q, V, E, \{C_e\}_{e\in E}, \{S_v^{u,\sigma_u}\}_{v\in V})$. If the output assignment is of value less than R^{u,σ_u}, then remove σ_u from S_u. Otherwise, return the output assignment as the output to Approx-CompleteGame$_{j+1}$.
3. If the loop in the previous step ends without outputting any assignment, just output a random assignment (i.e. pick $\varphi(u)$ independently and uniformly at random from S_u).

Observe first that the loop can run at most nq times as the total number of elements of S_v's for all $v \in V$ is at most nq. This means that we call Approx-CompleteGame$_j$ at most nq times. Since every step except the Approx-CompleteGame$_j$ calls takes $O((nq)^3)$ time and we call Approx-CompleteGame$_j$ only at most n^2q^2 times, we can conclude that the running time of Approx-CompleteGame$_{j+1}$ is $O((nq)^{3j+3})$ as desired.

The only thing left to show is that the assignment output from the algorithm indeed is of expected value at least R. To do so, we will consider two cases.

First, if step 3 is never reached, the algorithm must terminate at step 2(a)iv. From the return condition in step 2(a)iv, we know that the output assignment is of value at least $R^{u,\sigma_u} \geq R$ as desired.

In the second case where step 3 is reached, we first observe that when we remove s_u from S_u in step 2(a)iv, the instance is still satisfiable. The reason is that, if $\sigma_u = \sigma_u^{OPT}$ is the optimal assignment for u, then σ_v^{OPT} remains in S_v^{u,σ_u} for every $v \in V$. Hence, from our inductive hypothesis, the output assignment from $\text{Approx-CompleteGame}_j(q, V, E, \{C_e\}_{e \in E}, \{S_v^{u,\sigma_u}\}_{v \in V})$ must be of value at least R^{u,σ_u}. As a result, we never remove s_u^{OPT} from S_u, and, thus, the instance remains satisfiable throughout the algorithm.

Moreover, notice that, if $R^{u,\sigma_u} \geq R$ for any u, σ_u, we either remove σ_u from S_u or output the desired assignment. This means that, when step 3 is reached, $R^{u,\sigma_u} < R$ for every $u \in V$ and $\sigma_u \in S_u$.

Now, let us consider the expected number of edges satisfied by the random assignment. Since our graph (V, E) is complete, it can be written as follows.

$$\begin{aligned}
\mathbb{E}\left[\sum_{(u,v)\in E} C_{(u,v)}(\varphi(u), \varphi(v))\right] &= \mathbb{E}\left[\frac{1}{2}\sum_{u \in V}\sum_{\substack{v \in V\\ v \neq u}} C_{(u,v)}(\varphi(u), \varphi(v))\right] \\
&= \frac{1}{2}\sum_{u \in V}\sum_{\substack{v \in V\\ v \neq u}} \mathbb{E}\left[C_{(u,v)}(\varphi(u), \varphi(v))\right] \\
&= \frac{1}{2}\sum_{u \in V}\sum_{\substack{v \in V\\ v \neq u}} \frac{1}{|S_u||S_v|}\left(\sum_{\sigma_u \in S_u}\sum_{\sigma_v \in S_v} C_{(u,v)}(\sigma_u, \sigma_v)\right) \\
\text{(From definition of } S_v^{u,\sigma_u}) &= \frac{1}{2}\sum_{u \in V}\sum_{\substack{v \in V\\ v \neq u}} \frac{1}{|S_u||S_v|}\left(\sum_{\sigma_u \in S_u} |S_v^{u,\sigma_u}|\right) \\
&= \frac{1}{2}\sum_{u \in V}\frac{1}{|S_u|}\sum_{\sigma_u \in S_u}\left(\sum_{\substack{v \in V\\ v \neq u}} \frac{|S_v^{u,\sigma_u}|}{|S_v|}\right) \\
\text{(A.M.-G.M. inequality)} &\geq \frac{1}{2}\sum_{u \in V}\frac{1}{|S_u|}\sum_{\sigma_u \in S_u}(n-1)\sqrt[n-1]{\prod_{\substack{v \in V\\ v \neq u}} \frac{|S_v^{u,\sigma_u}|}{|S_v|}} \\
&= \frac{(n-1)}{2}\sum_{u \in V}\frac{1}{|S_u|}\sum_{\sigma_u \in S_u}\sqrt[n-1]{\frac{\prod_{\substack{v \in n\\ v \neq u}} |S_v^{u,\sigma_u}|}{\prod_{\substack{v \in V\\ v \neq u}} |S_v|}} \\
\text{(From our definition of } R^{u,\sigma_u}, R) &= \frac{(n-1)}{2}\sum_{u \in V}\frac{1}{|S_u|}\sum_{\sigma_u \in S_u}\sqrt[n-1]{\frac{(R^{u,\sigma_u})^{-nj}}{\frac{R^{-n(j+1)}}{|S_u|}}} \\
\text{(Since } R^{u,\sigma_u} < R) &> \frac{(n-1)}{2}\sum_{u \in V}\frac{1}{|S_u|}\sum_{\sigma_u \in S_u}\sqrt[n-1]{|S_u|R^n} \\
&= \frac{(n-1)}{2}\sum_{u \in V}\sqrt[n-1]{|S_u|R^n} \\
&= \frac{(n-1)}{2}R^{n/(n-1)}\left(\sum_{u \in V}\sqrt[n-1]{|S_u|}\right)
\end{aligned}$$

$$\begin{aligned}
(\text{A.M.-G.M. inequality}) &\geq \frac{(n-1)}{2} R^{n/(n-1)} \left(n \sqrt[n(n-1)]{\prod_{u \in V} |S_u|} \right) \\
(\text{From our definition of } R) &= \frac{(n-1)}{2} R^{n/(n-1)} \left(n \sqrt[n(n-1)]{\prod_{u \in V} R^{-n(j+1)}} \right) \\
&= \frac{n(n-1)}{2} R^{(n-1-j)/(n-1)} \\
(\text{Since } R \leq 1 \text{ and } j \geq 0) &\geq \frac{n(n-1)}{2} R.
\end{aligned}$$

Since $\frac{n(n-1)}{2}$ is the number of edges in (V, E), we can conclude that the random assignment is indeed of expected value at least R.

Thus, we can conclude that $P(j+1)$ is true. As a result, $P(i)$ is true for every positive integer i, which completes the proof for Lemma 18. ◀

Next, we will prove Corollary 12 by reducing it to Max 2-CSP on complete graph, and, then plug in Lemma 18 with appropriate i to get the result.

First, observe that, since $\log(1+\varepsilon') = \Omega(\varepsilon')$ for every $1 \geq \varepsilon' > 0$, by plugging in $i = C \log q/\varepsilon'$ for large enough constant C into Lemma 18, we immediately arrive the following corollary.

▶ **Corollary 19.** *For any $1 \geq \varepsilon' > 0$, there exists an $(1+\varepsilon')$-approximation algorithm for satisfiable* Max 2-CSP *on the complete graph that runs in time $N^{O(\varepsilon'^{-1} \log N)}$.*

Now, we will proceed to show the reduction and, thus, prove Corollary 12.

Proof of Corollary 12. First of all, notice that, since $\frac{1}{1+\varepsilon} = 1 - \Theta(\varepsilon)$. It is enough for us to show that there exists an $N^{O(\varepsilon^{-1}\delta^{-1} \log N)}$-time algorithm for satisfiable δ-dense Max 2-CSP that produces an assignment of value at least $1 - \varepsilon$.

On input $(q, V, E, \{C_e\}_{e \in E})$, the algorithm works as follows:

1. Construct a Max 2-CSP instance $(q, V, E', \{C'_e\}_{e \in E'})$ where (V, E') is a complete graph and C'_e is defined as C_e if $e \in E$. Otherwise, $C_e := 1$. In other words, we put in dummy constraints that are always true just to make the graph complete.
2. Run the algorithm from Corollary 19 on $(q, V, E', \{C'_e\}_{e \in E'})$ with $\varepsilon' = \varepsilon\delta$ and output the assignment got from the algorithm.

To see that the algorithm indeed produces an assignment with value $1 - \varepsilon$ for the input instance, first observe that, since $(q, V, E, \{C_e\}_{e \in E})$ is satisfiable, $(q, V, E', \{C'_e\}_{e \in E'})$ is trivially satisfiable. Thus, from Corollary 19, the output assignment has value at least $1/(1+\delta\varepsilon) \geq 1 - \delta\varepsilon$ with respect to $(q, V, E', \{C'_e\}_{e \in E'})$. In other words, the assignment does not satisfy at most $\delta\varepsilon n^2$ edges. Thus, with respect to the input instance, it satisfies at least $\delta n^2 - \delta\varepsilon n^2 = (1-\varepsilon)\delta n^2$ edges. In other words, it is of value at least $1 - \varepsilon$ as desired.

Lastly, note that the running time of this algorithm is determined by that of the algorithm from Corollary 19, which runs in $N^{O(\varepsilon'^{-1} \log N)} = N^{O(\varepsilon^{-1}\delta^{-1} \log N)}$ time as desired. ◀

8 Conclusions and Open Questions

Finally, we conclude by listing the open questions and interesting directions related to the techniques and problems presented here. We also provide our thoughts regarding each question.

- *Can our algorithm be extended to work for* Max k-CSP *for* $k \geq 3$*?* Other algorithms for approximating Max 2-CSP such as those from [2, 3, 4] are applicable for Max k-CSP for any value of k as well. So it is possible that our technique can be employed for Max k-CSP too.
- *Can one also come up with an algorithm that approximates* Max 2-CSP *to within* $O(N^{\varepsilon})$ *factor for any* $\varepsilon > 0$ *for low-value dense* Max 2-CSP*?* Our algorithm needs the value λ to be $N^{-o(1)}$ in order to give such a ratio so it is interesting whether we can remove or relax this condition. However, we do not think that one can remove the condition completely because, with similar technique to the proof of Corollary 12, we can arrive at a reduction from any Max 2-CSP to dense Max 2-CSP where the approximation ratio is preserved but the value decreases. This means that, if we can remove the condition on λ, then we are also able to refute the PGC. This argument nonetheless does not rule out relaxing the condition for λ without removing it completely.
- *Can our QPTAS be extended to unsatisfiable instances?* One of the main disadvantages of our QPTAS is that it requires the instance to be satisfiable. This renders our QPTAS useless against many problems such as Max 2-SAT and Max-Cut because the satisfiable instances of those problems are trivial. If we can extend our QPTAS to work on unsatisfiable instances as well, then we may be able to produce interesting results for those problems. Note, however, that, with similar argument to the preceding question, QPTAS for low-value instances likely does not exist. Instead, the case of unsatisfiable instances where [2, 3, 4] are successful is when they look for an additive error guarantee instead of a multiplicative one. Currently, it is unclear whether our technique can achieve such results.
- *Can one arrive at a similar or even better algorithm using SDP hierarchies?* SDP hierarchies have been very useful in finding approximation algorithms for combinatorial optimization problems. A natural question to ask is whether one can apply SDP hierarchies to get similar results to ours. For example, can the $O(i)$-level of the Lasserre hierarchy produce an approximation algorithm with ratio $O(q^{1/i})$ for dense Max 2-CSP? If so, then this may also be an interesting direction to pursue an algorithm with guarantee additive error discussed previously.

References

1 S. Aaronson, R. Impagliazzo, and D. Moshkovitz. AM with multiple Merlins. In *Computational Complexity (CCC), 2014 IEEE 29th Conference on*, pages 44–55, June 2014.

2 N. Alon, W. F. de la Vega, R. Kannan, and M. Karpinski. Random sampling and approximation of max-CSPs. *J. Comput. Syst. Sci.*, 67(2):212–243, September 2003.

3 S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, STOC'95, pages 284–293, New York, NY, USA, 1995. ACM.

4 B. Barak, M. Hardt, T. Holenstein, and D. Steurer. Subsampling mathematical relaxations and average-case complexity. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'11, pages 512–531. SIAM, 2011.

5 B. Barak, A. Rao, R. Raz, R. Rosen, and R. Shaltiel. Strong parallel repetition theorem for free projection games. In *Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX/RANDOM'09, pages 352–365, Berlin, Heidelberg, 2009. Springer-Verlag.

6 M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and nonapproximability – towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998.

7 A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-densities: An $O(n^{1/4})$ approximation for densest k-subgraph. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC'10, pages 201–210, New York, NY, USA, 2010. ACM.

8 F. G.S.L. Brandao and A. W. Harrow. Quantum de finetti theorems under local measurements with applications. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC'13, pages 861–870, New York, NY, USA, 2013. ACM.

9 M. Braverman, Y. K. Ko, and O. Weinstein. Approximating the best nash equilibrium in $n^{o(\log n)}$-time breaks the exponential time hypothesis. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'15, pages 970–982. SIAM, 2015.

10 M. Charikar, M. Hajiaghayi, and H. Karloff. Improved approximation algorithms for label cover problems. In *ESA*, pages 23–34, 2009.

11 U. Feige, D. Peleg, and G. Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

12 J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

13 S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'04, pages 136–145, Washington, DC, USA, 2004. IEEE Computer Society.

14 P. Manurangsi and D. Moshkovitz. Improved approximation algorithms for projection games. In *Algorithms – ESA 2013*, volume 8125 of *Lecture Notes in Computer Science*, pages 683–694. Springer Berlin Heidelberg, 2013.

15 D. Moshkovitz. The projection games conjecture and the NP-hardness of $\ln n$-approximating set-cover. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques – 15th International Workshop, APPROX 2012*, volume 7408, pages 276–287, 2012.

16 R. Raz. A parallel repetition theorem. In *SIAM Journal on Computing*, volume 27, pages 763–803, 1998.

17 R. Shaltiel. Derandomized parallel repetition theorems for free games. *Comput. Complex.*, 22(3):565–594, September 2013.

18 A. Suzuki and T. Tokuyama. Dense subgraph problems with output-density conditions. In Xiaotie Deng and Ding-Zhu Du, editors, *Algorithms and Computation*, volume 3827 of *Lecture Notes in Computer Science*, pages 266–276. Springer Berlin Heidelberg, 2005.

The Container Selection Problem

Viswanath Nagarajan[1], Kanthi K. Sarpatwar[*2], Baruch Schieber[2], Hadas Shachnai[†3], and Joel L. Wolf[2]

1 **University of Michigan, Ann Arbor, MI, USA**
viswa@umich.edu

2 **IBM T.J. Watson Research Center, Yorktown Heights, NY, USA**
sarpatwa@us.ibm.com, sbar@us.ibm.com, jlwolf@us.ibm.com

3 **Technion, Haifa, Israel**
hadas@cs.technion.ac.il

Abstract

We introduce and study a network resource management problem that is a special case of *non-metric k-median*, naturally arising in cross platform scheduling and cloud computing. In the continuous d-dimensional *container selection problem*, we are given a set $\mathcal{C} \subset \mathbb{R}^d$ of input points, for some $d \geq 2$, and a budget k. An input point p can be assigned to a "container point" c only if c dominates p in every dimension. The assignment cost is then equal to the ℓ_1-norm of the container point. The goal is to find k container points in $\mathbb{R}^d$, such that the total assignment cost for all input points is minimized. The discrete variant of the problem has one key distinction, namely, the container points must be chosen from a given set $\mathscr{F}$ of points.

For the continuous version, we obtain a *polynomial time approximation scheme* for any fixed dimension $d \geq 2$. On the negative side, we show that the problem is NP-hard for any $d \geq 3$. We further show that the discrete version is significantly harder, as it is NP-hard to approximate without violating the budget k in any dimension $d \geq 3$. Thus, we focus on obtaining bi-approximation algorithms. For $d = 2$, the bi-approximation guarantee is $(1+\epsilon, 3)$, i.e., for any $\epsilon > 0$, our scheme outputs a solution of size $3k$ and cost at most $(1+\epsilon)$ times the optimum. For fixed $d > 2$, we present a $(1+\epsilon, O(\frac{1}{\epsilon}\log k))$ bi-approximation algorithm.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases non-metric k-median, geometric hitting set, approximation algorithms, cloud computing, cross platform scheduling

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.416

1 Introduction

This paper introduces and studies the *container selection* problem, a special case of the non-metric k-median problem [12]. This network resource management problem naturally occurs in virtualized distributed computer environments, the goal being to maximize resource utilization. This environment may consist, e.g., of a private cloud [13], or a collection of in-house, physical computer processors employing a cluster manager such as Mesos [9] or YARN [16].

We describe and motivate the container selection problem as follows. The input points correspond to *tasks*, each of which can be described in terms of multiple resource requirements.

* Work done when the author was a student at the University of Maryland (College Park) and was supported by NSF grant CCF-1217890.

† This work was partly carried out during a visit to DIMACS supported by the National Science Foundation under grant number CCF-1144502.

These dimensions typically include both CPU and memory, sometimes also network and I/O bandwidth. The tasks are then placed and executed in *virtual containers*, and of course each task must “fit” into its assigned container. For a variety of reasons, including ease of selection, maintenance and testing, it is important to create only a modest number k of container sizes. Amazon's EC2 cloud offering [2], for example, allows its customers to choose from $k = 13$ standard “instance types”. The goal is to select k container sizes so that the aggregate resource usage (when each task is assigned its “smallest” dominating container) is minimized. We use the (normalized) sum of resources as the aggregate resource usage of a container. In these applications, the container sizes are usually determined in advance, before the actual tasks arrive: so suitably massaged historical task data is used as input. We refer the reader to [17] for more details.

Formally, an instance of the *continuous container selection* problem consists of a set of input points $\mathcal{C}$ in a d-dimensional space $\mathbb{R}^d$, and a budget k. We say that a point $c(c_1, c_2, \ldots, c_d)$ *dominates* (or, *contains*) a point $p(x_1, x_2, \ldots, x_d)$ if $x_i \leq c_i$, for all $i \in [d]$. The cost of assigning any input point p to a container point $c(c_1, c_2, \ldots, c_d)$ is the ℓ_1-norm of the container point, i.e, $c_1 + c_2 + \ldots + c_d$, if c dominates p; else, the assignment cost is ∞. The goal is to choose a set $S \subseteq \mathbb{R}^d$ of k container points, such that each input point is assigned to a container point in S, and the total assignment cost is minimized. In the *discrete* version of the problem, we have a further restriction that $S \subseteq \mathscr{F}$, where $\mathscr{F} \subseteq \mathbb{R}^d$ is an arbitrary, but finite, subset of points in the space. This problem variant is motivated by the fact that each container must itself “fit” into at least one physical processing node, or by the fact that certain resources (memory, for instance) are only allocated in fixed increments.

Related work. Clustering problems such as k-median, k-center, and k-means have received considerable attention in recent decades [10, 11, 3] (and references therein). Below, we only discuss the highlights directly relevant to our work. Our problem is a special case of the *non-metric k-median* problem. It also bears some similarity to the *Euclidean k-median* problem under the ℓ_1-norm metric. However, this similarity cannot be leveraged due to the “non-metric" characteristics of our problem. There is a $(1 + \epsilon, (1 + \frac{1}{\epsilon}) \ln n)$ bi-approximation algorithm for non-metric k-median [12], which finds a solution whose cost is within a $(1 + \epsilon)$ factor of optimal, for any constant $\epsilon > 0$, while using at most $k(1 + \frac{1}{\epsilon}) \ln n$ centers. The paper [12] also shows, using a reduction from the set cover problem, that these guarantees are the best one can hope for. On the other hand, the metric variant of the k-median problem is known to have small constant-factor approximation algorithms, with no violation of k. The best known ratio $2.611 + \epsilon$ is due to [6]. For the Euclidean k-median problem, which is a special case of metric k-median, there is a *polynomial time approximation scheme (PTAS)* [4].

Ackermann et al. [1] obtain PTAS for the non-metric k-median problem assuming that the following property holds: the corresponding 1-median problem can be approximated within a $1 + \epsilon$ factor by choosing a constant size sample and computing the optimal 1-median of such a sample. However, we note that the container selection problem does not satisfy this property. Indeed, consider a simple 1-dimensional instance with $n - 1$ points close to origin, and one point far away from origin. Clearly, with high probability, any constant size sample will, not contain the point far away from origin. An optimal 1-median of such a sample would in turn be infeasible for the original instance.

Our contribution. As noted above, the container selection problem is a special case of non-metric k-median, which is inapproximable unless we violate k significantly [12]. However, our problem still has sufficient geometric structure. This structure allows us to obtain near optimal algorithms that, in the case of continuous container selection, do not violate k, and in the discrete case, violate k mildly. In particular, we show that:

- the *continuous container selection problem* admits a PTAS, for any fixed d.
  On the negative side, we show that the problem is NP-hard for $d \geq 3$.
- the *discrete* variant (for $d \geq 3$) is NP-hard to approximate within *any* guarantee if the budget k is not violated. On a positive note, we obtain constant factor bi-approximation algorithms for this variant. For any constant $\epsilon > 0$, the guarantees are $(1+\epsilon, 3)$, for $d = 2$, and $(1+\epsilon, O(\frac{d}{\epsilon}\log dk))$, for any $d \geq 3$. The latter result is an improvement over the bi-approximation that follows from non-metric k-median [12] as long as $k = o(\log^{O(1)} n)$, $d = o(\frac{\log n}{\log\log n})$.

Techniques and outline. Our PTAS for the continuous problem (Section 2) relies on showing the existence of a near-optimal solution, where every container point lies on one among a constant number of rays through the origin. Ensuring this structure costs us a $1+\epsilon$ factor in the approximation ratio. The algorithm itself is then a dynamic program which optimally solves such a "restricted" container selection problem. A seemingly simpler approach is to use the near-optimal structure, where every container point lies on a grid with $O(\log n)$ geometrically spaced values in each dimension; however, this is not directly useful, as we do not know an exact algorithm for the resulting sub-problem.

The flexibility of using container points in the continuous space is essential – not just for our algorithm, but for any approach: we show the discrete version is NP-hard to approximate to any factor when $d \geq 3$. The reduction (Section 4) is from a restricted version of planar vertex cover [7], and in fact shows that even testing feasibility is NP-hard. We also reduce the discrete container selection problem to the continuous version (not approximation preserving), which proves its NP-hardness when $d \geq 3$.

We obtain two different algorithms for the discrete container selection problem, both of which provide bi-approximation guarantees. The first algorithm (Section 3.1) is specialized to dimension two and is a $(1+\epsilon, 3)$-approximation. The main idea here is a partitioning of $\mathbb{R}^2_+$ into $O(\log n)$ "cells", where all points in a cell have roughly the same ℓ_1-norm, thus allowing to decouple "local assignments" within a single cell, and "distant assignments" from one cell to another. This partitioning uses the definition of rays from the algorithm for the continuous problem. (Using a more standard partitioning yields $O(\log^2 n)$ cells which is too large for a polynomial-time algorithm.) The algorithm then uses enumeration to handle distant assignments and a dynamic-program for the local assignments. This decoupling necessitates the violation in the bound k.

The second algorithm for the discrete version (Section 3.2) works for any dimension d and yields a $(1+\epsilon, O(\frac{d}{\epsilon}\log dk))$-approximation. This is based on the natural linear programming relaxation used even for the non-metric k-median problem [12]. However, we obtain a sharper guarantee in the violation of k, using the geometry specific to our setting. In particular, we show an LP-based reduction to hitting-set instances having VC-dimension $O(d)$. Then our algorithm uses the well-known result of [8, 5] for such hitting-set instances. We note that a constant bi-approximation algorithm for $d = 2$ also follows from this approach, using a known $O(\frac{1}{\epsilon})$-size ϵ-net construction for "pseudo-disks" [14]. However, the constant obtained here is much larger than our direct approach.

▶ Remark. There is also a *quasi-polynomial* time approximation scheme (no violation of the bound k) for the discrete container selection problem in dimension $d = 2$. This is based on a different dynamic program (details deferred to the full version). However, this approach does not lead to any non-trivial approximation ratio in polynomial time. We leave open the possibility of a polynomial-time $O(1)$-approximation algorithm for this problem ($d = 2$).

Notation. For integers $a < b$, we use $[b] := \{1, 2, \cdots b\}$ and $[a, b] := \{a, a+1, \ldots, b\}$. All co-ordinates of input points are assumed to be non-negative. A point $c(c_1, c_2, \ldots, c_d) \in \mathbb{R}^d$ *dominates* (or, *contains*) another $p(x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ if, for all $i \in [d]$, $x_i \leq c_i$. By $p \prec c$, we mean c dominates p. Two points p_1 and p_2 are called *incomparable* if $p_2 \not\prec p_1$ and $p_1 \not\prec p_2$. The ℓ_1-norm of a point $c(c_1, c_2, \ldots, c_d)$ is denoted by $\|c\|$, i.e., $\|c\| = c_1 + c_2 \ldots + c_d$. For a subset of container points, S, we denote the total cost of assignment by $\mathsf{cost}(S)$. The cartesian product of two sets A and B is denoted by $A \times B$.

2 The Continuous Container Selection Problem

In this section, we describe a polynomial time approximation scheme for the continuous container selection problem. We start with a formal definition.

▶ **Definition 1** (continuous container selection)**.** In an instance of the problem, we are given a set of input points $\mathcal{C}$ in $\mathbb{R}^d_+$ and a budget k. The goal is to find a subset S of k container points in $\mathbb{R}^d_+$, such that the following cost is minimized.

$$\underset{\substack{S \subseteq \mathbb{R}^d \\ |S| \leq k}}{\mathsf{Min}} \sum_{p \in \mathcal{C}} \underset{\substack{c \in S \\ p \prec c}}{\mathsf{Min}} \|c\|$$

We describe the algorithm for $d = 2$ in Section 2.1 and subsequently, in Section 2.2 we extend this to dimension $d > 2$.

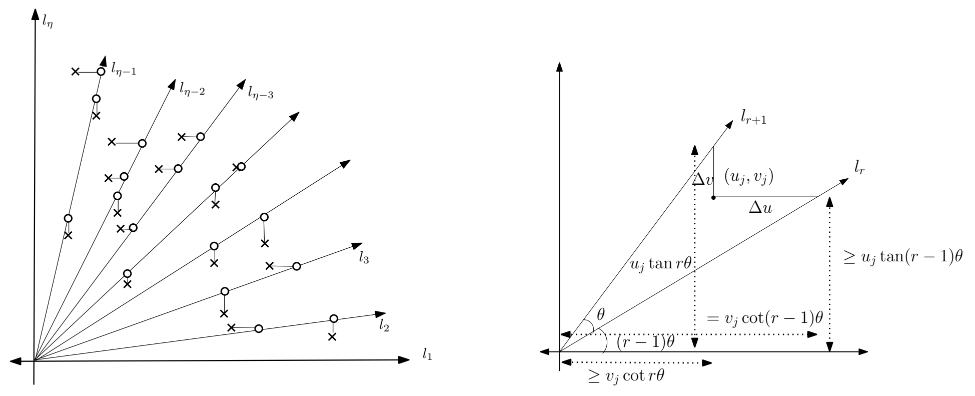
2.1 The two dimensional container selection problem

We denote the set of input points by $\mathcal{C} = \{p_i(x_i, y_i) : i \in [n]\}$. Let S_{opt} denote an optimal set of k container points. Let $X = \{x_i : i \in [n]\}$ and $Y = \{y_i : i \in [n]\}$. It is an easy observation that $S_{opt} \subseteq X \times Y$. We call $X \times Y$ the set of potential container points and denote it by $\mathscr{F} = \{c_j(u_j, v_j) : j \in [m]\}$, where $m \leq n^2$.

Algorithm outline. Given an instance of the problem, we transform it into an easier instance where all the chosen container points must lie on a certain family of rays. The number of rays in this family will be bounded by a constant that depends on ϵ, where $1 + \epsilon$ is the desired approximation ratio. Subsequently, we show that the restricted problem can be solved in polynomial time using a dynamic program.

Transformation of container points. Fix a constant $\theta \approx \frac{\epsilon}{2} \in (0, \frac{\pi}{4}]$, such that $\eta = \frac{\pi}{2\theta}$ is an integer. Define the following lines $l_r \equiv y \cos{(r-1)\theta} - x \sin{(r-1)\theta} = 0$, for $r \in [\eta + 1]$. We define the following transformation of any point $c_j(u_j, v_j) \in \mathscr{F}$ to construct the set of potential container points $\mathscr{F}^T$. If c_j lies on the line l_r, for some $r \in [\eta]$, then $c_j^T = c_j$. Otherwise, c_j is contained in the region bounded by the lines l_r and l_{r+1}, for some $r \leq \eta$. Now define two points $c_j^u(u_j + \Delta u, v_j)$ and $c_j^v(u_j, v_j + \Delta v)$, such that c_j^u is on l_r and c_j^v is on l_{r+1}. Now, the transformed point can be defined as follows:

$$c_j^T = \begin{cases} c_j^u, & \text{if } \Delta u \leq \Delta v \\ c_j^v, & \text{otherwise} \end{cases}$$



(a) Transformation of the container points **(b)** Computing the error in transformation

Figure 1 Continuous container selection problem.

Figure 1a illustrates this transformation. We emphasize that this transformation is only performed on the potential container points $\mathscr{F}$. The input points $\mathcal{C}$ themselves are unchanged. Under this transformation the optimal solution is preserved within an approximation factor of $(1+\epsilon)$.

▶ **Lemma 2.** *For instance $\mathcal{I} = (\mathcal{C}, k)$, let $S_{opt} = \{o_1, o_2, \ldots, o_k\}$ be an optimal solution. Further, let $S^T_{opt} = \{o^T_1, o^T_2, \ldots, o^T_k\} \subseteq \mathscr{F}^T$ be the set of transformed points corresponding to S_{opt}. Then, S^T_{opt} is a feasible solution to $\mathcal{I}$ and* $\mathsf{cost}(S^T_{opt}) \leq (1+\epsilon)\mathsf{cost}(S_{opt})$.

Proof. Recall that $\eta = \frac{\pi}{2\theta}$ and $\theta \approx \frac{\epsilon}{2}$. The feasibility of S^T_{opt} follows from the observation that if a point $p_i \in \mathcal{C}$ is dominated by a container $o_i \in S_{opt}$, it is also dominated by the point o^T_i. We now argue that $\mathsf{cost}(S^T_{opt}) \leq (1+\epsilon)\mathsf{cost}(S_{opt})$. It suffices to show that for every point $o_j = (u_j, v_j)$, $u^T_j + v^T_j \leq (1+\epsilon)(u_j + v_j)$, where $o^T_j = (u^T_j, v^T_j)$. The claim holds trivially in the case where o_j lies on a line l_r, for $r \in [1, 2, \ldots, \eta+1]$. Hence, assume that o_j lies in the region bounded by the two lines l_r and l_{r+1}, where $r \in [1, 2, \ldots, \eta]$. Further, let $o^u_j = (u_j + \Delta u, v_j)$ and $o^v_j = (u_j, v_j + \Delta v)$, be the points on lines l_r and l_{r+1} respectively. By geometry (refer to Figure 1b), we have the following equations:

$$\Delta u \leq v_j \left(\frac{\cos(r-1)\theta}{\sin(r-1)\theta} - \frac{\cos r\theta}{\sin r\theta} \right) = v_j \frac{\sin\theta}{\sin r\theta \sin(r-1)\theta} \tag{1}$$

$$\Delta v \leq u_j \left(\frac{\sin r\theta}{\cos r\theta} - \frac{\sin(r-1)\theta}{\cos(r-1)\theta} \right) = u_j \frac{\sin\theta}{\cos r\theta \cos(r-1)\theta} \tag{2}$$

Let $\Delta = \min(\Delta u, \Delta v)$. From Equations 1 and 2, we have,

$$(u_j + v_j)\sin\theta \geq \Delta(\sin r\theta \sin(r-1)\theta + \cos r\theta \cos(r-1)\theta) \;=\; \Delta\cos\theta.$$

$$\text{So } \Delta \leq (u_j + v_j)\tan\theta \leq (u_j + v_j)(2\theta) = (u_j + v_j)\epsilon. \tag{3}$$

Now, the claim follows from Equation 3 and the fact that $u^T_j + v^T_j = (u_j + v_j) + \Delta$. ◀

In Section 2.3, we show that the following restricted problem can be solved in polynomial time (by dynamic programming), for any fixed dimension $d \geq 2$.

▶ **Definition 3** (restricted container selection). For a constant $\eta \geq 0$, let $L_d = \{l_1, l_2, \ldots, l_\eta\}$ be a given family of η rays in $\mathbb{R}^d_+$. The input is a set of points $\mathcal{C} \subseteq \mathbb{R}^d_+$, a set of potential container points $\mathscr{F}$ that lie on the lines in L_d and a budget k. The goal is to find a subset $S \subseteq \mathscr{F}$ with $|S| \leq k$ such that $\mathsf{cost}(S)$ is minimized.

By Lemma 2, the 2D continuous container selection problem reduces to this restricted problem, at a $(1+\epsilon)$-factor loss. So we obtain a PTAS for the 2D continuous container selection problem.

2.2 Continuous container selection in dimension $d > 2$

We now consider the container selection problem in higher, but fixed, dimensions. Formally, an instance, $\mathcal{I} = (\mathcal{C}, k)$, of the d-dimensional container selection problem consists of a set of input points, $\mathcal{C} = \{p_i(x_1^i, x_2^i, \ldots, x_d^i) : i \in [n]\}$ and a budget k.

Potential container points. For each dimension $j \in [d]$, we define $X_j = \{x_j^i : i \in [n]\}$, as the set of j^{th} coordinates of all input points. An easy observation is that any container point chosen by any optimal solution must belong to $\mathscr{F} = X_1 \times X_2 \times \ldots \times X_d = \{c_i(u_1^i, u_2^i, \ldots, u_d^i) : i \in [m]\}$ where, $m \le n^d$.

Algorithm outline. As in the two dimensional case, the main idea is a reduction to the following restricted problem. An instance is $\mathcal{I} = (\mathcal{C}, k, L_d)$ where $\mathcal{C}$ is a set of input points in $\mathbb{R}^d$, k is an integer and L_d is a family of rays in $\mathbb{R}_+^d$ with $|L_d| = O_d(1)$. The goal is to choose k container points that lie on the rays in L_d, such that the total assignment cost of $\mathcal{C}$ is minimized.

Transformation of container points. Fix a constant $\theta \approx \frac{\epsilon}{2} \in (0, \frac{\pi}{4}]$, such that $\eta = \frac{\pi}{2\theta}$ is an integer. In order to construct L_d, we use the recursive procedure described in Algorithm 1. Let $\bar{u}_i$ denote the i^{th} unit vector ($i \le d$), i.e., $\bar{u}_i$ is a 0-1 vector with value 1 at the i^{th} coordinate and 0 elsewhere. Starting from the family L_2 of rays in two dimensions (using the transformation in Section 2), we add one dimension at a time and construct the corresponding families for higher dimensions. In the recursive step, we start with the family L_{r-1} and observe that each of these rays will induce a 2-D plane in r-dimensions. Then, we use the two dimensional construction to handle the extra dimension. Observe that $|L_d| \le (\pi/\theta)^d = O(1)$ for any fixed θ and d.

Algorithm 1 Construction of the family of lines in r-dimensions: L_r

```
1: let $\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_r$ be the unit vectors along the axis lines
2: if $r = 2$ then return equiangular rays in $\mathbb{R}_+^2$ from Section 2 (see also Figure 1)
3: construct the family $L_{r-1}$ in $\mathbb{R}_+^{r-1}$ recursively.
4: initiate: $L_r \leftarrow \emptyset$
5: for all $\ell \in L_{r-1}$ do
6:     let $\bar{\ell}$ be the unit vector along the line $\ell$
7:     consider the (two dimensional) plane $\Pi_\ell$ formed by the vectors $\bar{u}_r$ and $\bar{\ell}$
8:     let $Q_\ell$ be the family of rays obtained by applying the 2D transformation in Section 2
       to the plane $\Pi_\ell$
9:     $L_r \leftarrow L_r \cup Q_\ell$
10: end for
11: return $L_r$
```

Algorithm 2 describes a recursive procedure to transform a point $c(u_1, u_2, \ldots, u_d) \in \mathscr{F}$ to a point c^T that lies on some line in L_d. The idea is as follows: for any $r \ge 3$, first recursively transform the point $c_{r-1}(u_1, u_2, \ldots, u_{r-1}) \in \mathbb{R}^{r-1}$ into a point $c_{r-1}^T(u_1', u_2', \ldots, u_{r-1}')$ that lies on some line $\ell \in L_{r-1}$. Now, consider the point $c'_r(u'_1, u'_2, \ldots, u'_{r-1}, u_r)$, where u_r is the r^{th} coordinate of the original point c. The point c'_r lies on the 2D plane spanned by $\bar{\ell}$, the unit vector along the line ℓ, and $\bar{u}_r$. Using the 2D transformation we move c'_r to a point c^T_r that lies on some line in L_r.

Algorithm 2 The transformation of $c_r = (u_1, u_2, \ldots u_r)$ onto L_r, $r \le d$

```
1: if $r = 2$ then use the 2D transformation from the Section 2 (see also Figure 1)
2: $c_{r-1} \leftarrow (u_1, u_2, \ldots, u_{r-1})$
3: recursively transform $c_{r-1}$ into a point on some line $\ell$ in $L_{r-1}$ and compute the trans-
   formed point $c^T_{r-1} = (u'_1, u'_2, \ldots, u'_{r-1})$
4: $c'_r \leftarrow (u'_1, u'_2, \ldots, u'_{r-1}, u_r)$, which lies on the plane $\Pi_\ell$ spanned by $\bar{u_r}$ and $\bar{\ell}$
5: let $Q_\ell$ denote the lines on plane $\Pi_\ell$ from Algorithm 1 step 8.
6: use the 2D transformation (Section 2) on plane $\Pi_\ell$ to move $c'_r$ onto a line in $Q_\ell$ and
   obtain $c^T_r = (u^T_1, u^T_2, \ldots, u^T_{r-1}, u^T_r)$
7: return $c^T_r$
```

▶ **Lemma 4.** *For any $\theta = \frac{\epsilon}{2} \in (0, \frac{1}{2d-2}]$ and point $c(u_1, u_2, \ldots, u_d) \in \mathscr{F}$, applying Algorithm 2, we obtain $c^T = (u^T_1, u^T_2, \ldots, u^T_d)$ where $c \prec c^T$ and:*

$$\|c^T\| \le (1 + 2(d-1)\epsilon)\|c\|.$$

Proof. It is straightforward to see $c \prec c^T$. Using induction we will show that

$$\|c^T_r\| \le (1+\epsilon)^{r-1}\|c_r\|$$

The base case $r = 2$ follows from Lemma 2. Now consider $r \ge 3$ and assume the statement for $r - 1$. In Algorithm 2, c^T_r is obtained by transforming the point c'_r in the 2D plane Π_ℓ. Note that c'_r has coordinates $\sqrt{(u'_1)^2 + (u'_2)^2 + \ldots + (u'_{r-1})^2}$ and u_r in plane Π_ℓ. Hence, as shown in Lemma 2, we can obtain the following:

$$\begin{aligned} u^T_1 + u^T_2 + \ldots + u^T_{r-1} + u^T_r &\le (1+\epsilon)(\sqrt{(u'_1)^2 + (u'_2)^2 + \ldots + (u'_{r-1})^2} + u_r) \\ &\le (1+\epsilon)(u'_1 + u'_2 + \ldots + u'_{r-1} + u_r) \end{aligned} \tag{4}$$

By the inductive hypothesis, $u'_1 + u'_2 + \ldots + u'_{r-1} = \|c^T_{r-1}\| \le (1+\epsilon)^{r-2}\|c_{r-1}\|$, i.e.

$$u'_1 + u'_2 + \ldots + u'_{r-1} \le (1+\epsilon)^{r-2}(u_1 + u_2 + \ldots + u_{r-1}) \tag{5}$$

Using Equations 4, 5, we have

$$\begin{aligned} u^T_1 + u^T_2 + \ldots + u^T_{r-1} + u^T_r &\le (1+\epsilon)((u'_1 + u'_2 + \ldots + u'_{r-1}) + u_r) \\ &\le (1+\epsilon)((1+\epsilon)^{r-2}(u_1 + u_2 + \ldots + u_{r-1}) + u_r) \\ &\le (1+\epsilon)^{r-1}(u_1 + u_2 + \ldots + u_r) \end{aligned}$$

Now since $(d-1)\epsilon \le 1$, using $r = d$ above, $\|c^T\| \le (1+\epsilon)^{d-1}\|c\| \le (1 + (2d-2)\epsilon) \cdot \|c\|$. ◀

For any $\epsilon' > 0$, setting $\epsilon = \frac{\epsilon'}{2(d-1)}$, we can restrict the loss to a $(1+\epsilon')$ factor. Thus, we have reduced the original instance to a restricted instance, where the potential container points lie on a family with a constant number of lines. Using the exact algorithm for this problem (Section 2.3) we obtain:

▶ **Theorem 5.** *There is a PTAS for continuous container selection in fixed dimension d.*

2.3 Algorithm for restricted container selection

Here we provide an exact algorithm for the restricted container selection problem (Definition 3). We need the following notion of a profile of a given subset of container points.

Profile of a subset. For a given line l_i and $S \subseteq \mathscr{F}$, let $c_i \in S$ be the container point on l_i with maximum ℓ_1-norm; if there is no such point then c_i is set to the origin. We define the *profile* of S, denoted by $\Pi(S)$, as the ordered tuple $(c_1, c_2, \ldots, c_\eta)$. The *feasible* region of a profile $\Pi(S) = (c_1, c_2, \ldots, c_\eta)$, denoted by $\mathsf{feas}(\Pi(S))$, is the set of those input points that are dominated by at least one of the points c_i, $i \in [\eta]$. We slightly abuse this notation and refer to the tuple itself as a profile, without any mention of S.

▶ Observation 6. The number of distinct profiles is at most $\left(\frac{|\mathscr{F}|}{\eta}\right)^\eta$.

Proof. Let n_i be the number of potential container points on the line l_i. The total number of distinct profiles is simply the number of ways of choosing the tuple $(c_1, c_2, \ldots, c_\eta)$, which is equal to $n_1 n_2 \ldots n_\eta \leq \left(\frac{\sum_{i=1}^{\eta} n_i}{\eta}\right)^\eta = \left(\frac{|\mathscr{F}|}{\eta}\right)^\eta$. ◀

For a given profile $\Pi = (c_1, c_2, \ldots, c_\eta)$, let c_m denote the profile point with maximum ℓ_1-norm, i.e., $c_m = \underset{c_i}{\arg\max} \|c_i\|$. Further, let $c'_m \prec c_m$ be some potential container point such that both the points are on the line l_m; if c'_m does not exist we set it to the origin. We define the *child profile* of Π corresponding to c'_m, denoted by $\mathsf{chld}(\Pi, c'_m)$, as the profile $(c_1, c_2, \ldots, c_{m-1}, c'_m, \ldots, c_\eta)$. A profile tuple could have multiple child profiles. The following observation is immediate from the definition of a child profile.

▶ Observation 7. Any profile tuple Π has at most $|\mathscr{F}|$ child profile tuples.

The DP variable. For every possible profile tuple $\Pi = (c_1, c_2, \ldots, c_\eta)$ and all budgets $k' \leq k$, define the dynamic program variable, $\mathscr{M}(\Pi, k')$ as the cost of an optimal solution $S \subseteq \mathsf{feas}(\Pi) \cap \mathscr{F}$, to assign all the input points in $\mathsf{feas}(\Pi)$, such that $|S| \leq k'$, and $c_i \in S$, for $i \in [\eta]$. The following lemma allows us to set up the dynamic program recurrence.

▶ **Lemma 8.** *Let $\Pi = (c_1, c_2, \ldots, c_\eta)$ be a profile with c_m as the point with maximum ℓ_1-norm. For a given child profile $\mathsf{chld}(\Pi, c'_m)$ of Π, let $n(c'_m) = |\mathsf{feas}(\Pi) \setminus \mathsf{feas}(\mathsf{chld}(\Pi, c'_m))|$. Then, for any $k' \geq 1$, the following holds.*

$$\mathscr{M}(\Pi, k') = \underset{c'_m}{\mathit{Min}} \left(\mathscr{M}(\mathsf{chld}(\Pi, c'_m), k' - 1) + n(c'_m)\|c_m\|\right)$$

Proof. We denote the optimal solution corresponding to the variable $\mathscr{M}(\Pi, k')$ by $S(\Pi, k')$. Firstly, note that, for any c'_m, the solution $S(\mathsf{chld}(\Pi, c'_m), k' - 1) \cup \{c_m\}$ is a feasible candidate for the computation of $\mathscr{M}(\Pi, k')$. Hence, we have

$$\mathscr{M}(\Pi, k') \leq \underset{c'_m}{\mathsf{Min}} \left(\mathscr{M}(\mathsf{chld}(\Pi, c'_m), k' - 1) + n(c'_m)\|c_m\|\right) \tag{6}$$

Let l_m be the ray containing the point c_m. Further, let $q_0 = (0^d), q_1, \ldots, q_{j-1}, q_j = p_i$ be the container points, on l_m and in $S(\Pi, k)$, in the increasing order of ℓ_1-norm. Now, we set $q' = q_{j-1}$ and prove that the child profile corresponding to q' satisfies the following equation:

$$\mathscr{M}(\Pi, k') = \mathscr{M}(\mathsf{chld}(\Pi, q'), k' - 1) + n(q')\|c_m\|$$

To this end, we first observe that, without loss of generality, no point in $\mathsf{feas}\,(\mathsf{chld}(\Pi, q'))$ is assigned to c_m. Indeed, this follows from the fact that c_m is the container point with maximum cost and therefore, any point in the above feasible region can be assigned to some container point on the profile $\mathsf{chld}(\Pi, q')$ without increasing the solution cost. Further, any point in $\mathsf{feas}(\Pi) \setminus \mathsf{feas}(\mathsf{chld}(\Pi, q'))$ must be assigned to c_m, since it is the only potential container point that dominates these points. Now,

$$\begin{aligned}\mathscr{M}(\Pi, k') &= \mathscr{M}(\mathsf{chld}(\Pi, q'), k' - 1) + n(q')\|c_m\| \\ &\geq \underset{c'_m}{\mathsf{Min}}\ (\mathscr{M}(\mathsf{chld}(\Pi, c'_m), k' - 1) + n(c'_m)\|c_m\|) \end{aligned} \tag{7}$$

From Equations 6 and 7, we have our lemma. ◀

Algorithm 3 describes the dynamic program.

Algorithm 3 Dynamic program for the restricted container selection problem

```
Input: Family of lines $L_d = \{l_1, l_2 \ldots, l_\eta\}$, input points $\mathcal{C}$, potential container points set $\mathscr{F}$
    on $L_d$ and a budget $k$
 1: for all profile tuples $\Pi$ (w.r.t $L_d$) and integers $k' \leq k$ do
 2:   if $k' = 0$ then
 3:     if $\Pi = ((0^d), (0^d), \ldots, (0^d))$ then
 4:       $\mathscr{M}(\Pi, k') = 0$
 5:     else
 6:       $\mathscr{M}(\Pi, k') = \infty$
 7:     end if
 8:   else
 9:     let $c_m$ be the container point with maximum $\ell_1$-norm in $\Pi$
10:     for all $c'_m \prec c_m$ such that both $c_m$ and $c'_m$ lie on the same line $l_m$ do
11:       $n(c'_m) \leftarrow |\mathsf{feas}(\Pi) \setminus \mathsf{feas}(\mathsf{chld}(\Pi, c'_m))|$
12:       $f(c'_m) \leftarrow (\mathscr{M}(\mathsf{chld}(\Pi, c'_m), k' - 1) + n(c'_m)\|c_m\|)$
13:     end for
14:     $\mathscr{M}(\Pi, k') \leftarrow \underset{c'_m}{\mathsf{Min}}\, f(c'_m)$
15:   end if
16: end for
17: return profile $\Pi$ with least cost $\mathscr{M}(\Pi, k)$ such that $\mathcal{C} = \mathsf{feas}(\Pi)$.
```

3 The Discrete Container Selection Problem

In this section, we consider the discrete version of the container selection problem. We start with the problem definition.

▶ **Definition 9** (discrete container selection)**.** In an instance of the problem, $\mathcal{I} = (\mathcal{C}, \mathscr{F}, k)$, we are given a set of input points $\mathcal{C} \subset \mathbb{R}^d_+$, a set of potential container points $\mathscr{F} \subset \mathbb{R}^d_+$ and a budget k. The goal is to find a subset of container points $S \subseteq \mathscr{F}$, such that $|S| \leq k$ and the total assignment cost of all the input points, $\mathsf{cost}(S)$ is minimized.

This problem is considerably harder than the continuous version, as we show that there is no true approximation algorithm for this problem, unless $P = NP$, for $d \geq 3$. Hence, we look for bi-approximation algorithms, defined as follows. An (α, β) bi-approximation algorithm obtains a solution S, such that $|S| \leq \beta \cdot k$ and $\mathsf{cost}(S) \leq \alpha \cdot \mathsf{cost}(S_{opt})$.

▶ **Theorem 10** (two-dimensions). *For $d = 2$, and any constant $\epsilon > 0$, there is a $(1+\epsilon, 3)$-bi-approximation algorithm for the discrete container selection problem.*

▶ **Theorem 11** (higher-dimensions). *For $d > 2$ and $\epsilon > 0$, there is a $(1+\epsilon,\ O(\frac{d}{\epsilon}\log dk))$-bi-approximation algorithm for the discrete container selection problem.*

3.1 Two dimensional discrete container selection problem

Algorithm outline. The first step is to partition the plane into a logarithmic number of "cells", such that the ℓ_1-norms of points in a particular cell are approximately uniform. One standard way of doing this, where we create a two-dimensional grid with logarithmic number of lines in each dimension, fails because such a process would yield $\Omega(\log^2 n)$ cells. Our approach uses the *rays partitioning* idea. Given such a partitioning, we "guess" the "good" cells that have any container points belonging to a fixed optimal solution. For each one of these good cells, we then pick two representative container points. These points are chosen such that if, in the optimal solution, an input point i outside a cell e is assigned to a container point inside e, at least one of the representative points in e dominates i. This enables us to make "local decisions" for each cell independently. We then solve this localized instance, using k more container points. Hence, in total we use $3k$ container points.

The algorithm. Choose $\delta = \frac{\epsilon}{11}$ such that $\frac{\pi}{4\delta} = \eta$ is an integer. We first use a simple scaling argument to bound the maximum to minimum ratio of ℓ_1-norms by $O(n)$. We guess the maximum norm container point p_{max} that is used in some fixed optimal solution (there are only $|\mathscr{F}|$ guesses) and delete all larger points from $\mathscr{F}$. Let p_{min} be the point in $\mathcal{C} \cup \mathscr{F}$ with minimum positive norm. We increase the x-coordinates of all the input points and the container points by $\frac{\delta}{n}\|p_{max}\|$ and then divide all the co-ordinates of all points by $\|p_{min}\|$.

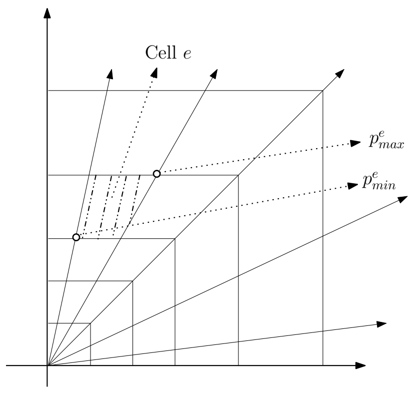


Figure 2 Description of the cells.

▶ Observation 12. Let S_{opt} and S'_{opt} be the optimal solutions of a given instance before and after scaling respectively. $\|p_{min}\|\mathsf{cost}(S'_{opt}) \leq \mathsf{cost}(S_{opt})(1+\delta)$

Proof. Since all the points are increased and scaled uniformly, the feasibility is maintained. Further, we note that $\mathsf{cost}(S_{opt}) \geq \|p_{max}\|$ since our guess $p_{max} \in S_{opt}$. If the cost of assignment of any input point is C in the original instance, the new cost is equal to $(C + \frac{\delta}{n}\|p_{max}\|)/\|p_{min}\|$ and the lemma follows. ◀

From now on, we assume that all the points are scaled as above and therefore $\|p_{min}\| = 1$ and $\|p_{max}\| \leq \frac{n}{\delta}$. Let $t = \log_{1+\delta}\|p_{max}\|$ and define the following families of rays.

$$\mathcal{L}_1 = \{x\sin(r\delta) - y\cos(r\delta) = 0 : r \in [0,\eta)\} \quad \mathcal{L}_3 = \{y = (1+\delta)^i : i \in [0,t]\}$$

$$\mathcal{L}_2 = \{x\sin(r\delta) - y\cos(r\delta) = 0 : r \in [\eta,2\eta]\} \quad \mathcal{L}_4 = \{x = (1+\delta)^i : i \in [0,t]\}$$

Cells. We define the notion of cell as exemplified in the Figure 2. A *cell* is a quadrilateral formed with the following bounding lines: either, two consecutive lines in $\mathcal{L}_1$ and two consecutive lines in $\mathcal{L}_4$, or, two consecutive lines in $\mathcal{L}_2$ and two consecutive lines in $\mathcal{L}_3$. The number of cells formed is at most $(2\eta+1)t = O(\log n)$

▶ **Lemma 13.** *For a given cell e, let p^e_{min} and p^e_{max} be the points of minimum and maximum cost, respectively. Then,*

$$\|p^e_{max}\| \leq (1+\epsilon)(\|p^e_{min}\|)$$

Proof. Without loss of generality, let e be formed by lines $y = (1+\delta)^i$, $y = (1+\delta)^{i+1}$, $x\sin\theta - y\cos\theta = 0$ and $x\sin(\theta+\delta) - y\cos(\theta+\delta) = 0$, where $\theta \geq \frac{\pi}{4}$. Clearly, as shown in Figure 2, we have

$$p^e_{min} = ((1+\delta)^i \cot(\theta+\delta), (1+\delta)^i)$$
$$p^e_{max} = ((1+\delta)^{i+1} \cot\theta, (1+\delta)^{i+1})$$

$$\begin{aligned}
\frac{\|p^e_{max}\|}{\|p^e_{min}\|} &= \frac{(1+\delta)^{i+1}(1+\cot\theta)}{(1+\delta)^i(1+\cot(\theta+\delta))} = (1+\delta)\frac{(\sin\theta+\cos\theta)\sin(\theta+\delta)}{(\sin(\theta+\delta)+\cos(\theta+\delta))\sin\theta} \\
&= (1+\delta)\frac{\sin\theta\sin(\theta+\delta)+\cos\theta\sin(\theta+\delta)}{\sin(\theta+\delta)\sin\theta+\cos(\theta+\delta)\sin\theta} \\
&= (1+\delta)\left(1+\frac{\cos\theta\sin(\theta+\delta)-\cos(\theta+\delta)\sin\theta}{\sin(\theta+\delta)\sin\theta+\cos(\theta+\delta)\sin\theta}\right) \\
&= (1+\delta)\left(1+\frac{\sin\delta}{\sin(\theta+\delta)\sin\theta+\cos(\theta+\delta)\sin\theta}\right) \leq (1+\delta)\left(1+\frac{\sin\delta}{\sin^2\theta}\right) \\
&\leq (1+\delta)(1+2\delta) = (1+3\delta+2(\delta)^2) \leq (1+\epsilon)
\end{aligned}$$

We note that the second last inequality follows from the fact that $\sin^2\theta \geq \sin^2\frac{\pi}{4} \geq \frac{1}{2}$. ◀

Representative points. For a given optimal solution, a cell is good if at least one container point is chosen from it (we break the ties between two cells sharing an edge arbitrarily). Since, there are $O(\log n)$ cells, there are a polynomial number of good-bad classifications. Therefore, we can try out all possible configurations and assume that we know which cells are good. For each good cell e, let p^e_x be the container point with maximum x-coordinate and p^e_y the one with maximum y-coordinate. We define the set of representative points, $\mathcal{R} = \{\, p^e_x,\ p^e_y : \forall\ e \text{ good cell} \,\}$. Clearly $|\mathcal{R}| \leq 2k$. We will show (in Lemma 15) that any input point that is not assigned to a "local container" (one in the same cell) in the optimal solution, can be re-assigned to some point of $\mathcal{R}$ at approximately the same cost.

Localized container selection problem. In an instance of the *localized container selection problem*, $(\mathcal{C}, \mathscr{F}_1, \mathscr{F}_2, k)$, we are given a set of input points $\mathcal{C}$, a set of potential container points $\mathscr{F}_1$, a set of pre-chosen container points $\mathscr{F}_2$ and a budget k. Moreover, for each cell e, the points in $\mathscr{F}_1 \cap e$ are all incomparable to each other. For a cell e, let $\Delta^e_{max} = \underset{p\in\mathscr{F}_1\cap e}{\mathsf{Max}} \|p\|$, be the maximum ℓ_1-norm of any container point in e. The cost of assignment of any input point to any point, in $\mathscr{F}_1 \cap e$, is uniform and equal to Δ^e_{max}. The cost of assignment of an input point to a container point $c \in \mathscr{F}_2$ is $\|c\|$. Further, any input point p in the cell e, can only be assigned to:

- a container point $c \in \mathscr{F}_2$ such that $p \prec c$, or
- a container point $c \in \mathscr{F}_1$ such that c belongs to e and $p \prec c$.

Given an instance of the discrete container selection problem, $\mathcal{I} = (\mathcal{C}, \mathscr{F}, k)$, we construct the following instance of the *localized container selection problem*, $\mathcal{I}' = (\mathcal{C}, \mathscr{F}_1, \mathscr{F}_2, k)$.

▶ Construction 14. The input point set $\mathcal{C}$, remains the same and $\mathscr{F}_2$ is the set of representative points, i.e., $\mathscr{F}_2 = \mathcal{R}$. $\mathscr{F}_1$ is constructed as follows: starting with $\mathscr{F}_1 = \mathscr{F} \setminus \mathcal{R}$, while there are two points p and p' in $\mathscr{F}_1$ that belong to same cell e and $p \prec p'$, delete p from $\mathscr{F}_1$.

▶ **Lemma 15.** *For a given instance of the discrete container selection problem, $\mathcal{I} = (\mathcal{C}, \mathscr{F}, k)$, with the optimal solution cost OPT, the corresponding localized container selection instance $\mathcal{I}' = (\mathcal{C}, \mathscr{F}_1, \mathscr{F}_2, k)$ has an optimal cost of at most $(1+\epsilon)OPT$.*

Proof. Suppose S is an optimal solution for the instance $\mathcal{I}$. We iteratively construct a solution, S', for the instance $\mathcal{I}'$. Initiating $S' = \phi$, we add exactly one container point for every container point $c \in S$ in the following way: let c belong to a cell e. If $c \in \mathscr{F}_1$, then we add c to S'; otherwise, we add some $c' \in \mathscr{F}_1 \cap e$, such that $c \prec c'$, which must exist by Construction 14. Clearly $|S'| \le |S| \le k$. We show that S' is a feasible solution, with a cost at most $(1+\epsilon)OPT$, for the instance $\mathcal{I}'$.

Consider an input point p that is assigned to some container point $c \in S$, in the optimal solution for $\mathcal{I}$. Suppose, firstly, that c and p are contained in the same cell e. By the construction of S', there must be some $c' \in S' \cap e$ (possibly $c = c'$) such that $c \prec c'$ and we can assign p to c'. Further, note that since p and c' belong to the same cell this is a valid "local" assignment and by Lemma 13, the cost of assignment equals $\Delta^e_{max} \le \|c\|(1+\epsilon)$.

Subsequently, assume that p belongs to a cell e_1 and c belongs to a cell e_2, such that $e_1 \ne e_2$. We show that p can be assigned to one of the two representative points of e_2, namely $p_x^{e_2}$ or $p_y^{e_2}$. Recall that $p_x^{e_2}$ (resp. $p_y^{e_2}$) is a container point in e_2 with maximum x-coordinate (resp. y-coordinate). We first claim that there must exist a separating line $y = mx + C$ with slope $m \ge 0$, such that e_1 and e_2 lie on the opposite sides of this line (they could share a boundary along this line). We overload notation and allow $m = \infty$ in which the line is $x + C = 0$. So when $m = 0$ the line ($y = C$) is parallel to the x-axis and when $m = \infty$ the line ($x = -C$) is parallel to the y-axis.

Observe that by our construction, all the boundary lines have non-negative slopes. Therefore, if e_1 and e_2 share a boundary line segment, this will be our separating line. Suppose, on the other hand, that they do not share a boundary line segment and therefore are disjoint. If e_1 and e_2 are on the opposite sides of the line $y = x$, this will be our separating line. So, we assume that both the cells are on the same side of $y = x$, without loss of generality say above $y = x$. Then both these cells must be bounded by lines from the families $\mathcal{L}_2$ and $\mathcal{L}_3$. Let the lines bounding e_1 and e_2, respectively be, $B_1 = \{y = (1+\delta)^i,\ y = (1+\delta)^{i+1},\ x\sin\theta - y\cos\theta = 0,\ x\sin(\theta+\delta) - y\cos(\theta+\delta) = 0\}$ and $B_2 = \{y = (1+\delta)^j,\ y = (1+\delta)^{j+1},\ x\sin\theta' - y\cos\theta' = 0,\ x\sin(\theta'+\delta) - y\cos(\theta'+\delta) = 0\}$. Now, if $i = j$, then for the cells not to intersect, we must have $\theta \ge \theta' + \delta$ or $\theta' \ge \theta + \delta$. Without loss of generality, let $\theta \ge \theta' + \delta$. In this case, clearly the separating line is $x\sin\theta - y\cos\theta = 0$. In the case, where $i > j$ (resp. $i < j$), $y = (1+\delta)^j$ (resp. $y = (1+\delta)^i$) is a separating line.

We consider two different cases based on the value of m and prove that p can be assigned to some representative point in e_2.

Case 1: $m \in \{0, \infty\}$. The separating line between e_1 and e_2 is axis parallel, say $x = a$, without loss of generality. Since $p \prec c$, we have that the x-co-ordinates of all points in e_1 are less than a and x-coordinates of all points in e_2 are more than a. Hence, clearly the point with maximum y-coordinate in e_2, namely $p_y^{e_2}$ must dominate p.

Case 2: $m > 0$ and finite. Let the separating line be $y = mx + C$. There are two further cases here. First assume that p lies below the $y = mx + C$ and c lies above it. Letting $p = (x_1, y_1)$, $c = (x_2, y_2)$ and $p_x^{e_2} = (x_3, y_3)$, we have $y_1 \le mx_1 + C$ and $y_2 \ge mx_2 + C$ and $y_3 \ge mx_3 + C$. By definition, $x_1 \le x_2 \le x_3$ and we focus on showing that $y_1 \le y_3$. Indeed we have $y_1 \le mx_1 + C \le mx_2 + C \le mx_3 + C \le y_3$. Thus, $p \prec p_x^{e_2}$. Next, we assume that p lies above $y = mx + C$ and c lies below it. Letting $p = (x_1, y_1)$, $c = (x_2, y_2)$ and $p_y^{e_2} = (x_3, y_3)$, we have $y_1 \geq mx_1 + C$, $y_2 \leq mx_2 + C$ and $y_3 \leq mx_3 + C$. By definition, $y_1 \leq y_2 \leq y_3$. Further, $x_1 \leq y_1/m - C/m \leq y_2/m - C/m \leq y_3/m - C/m \leq x_3$. Hence, $p \prec p_y^{e_2}$. Therefore, we have shown that if p is assigned to c, we can assign it to a representative point, c_r, that lies in the same cell as c. From Lemma 13, this implies that our cost of assignment is $\|c_r\| \leq (1+\epsilon)\|c\|$. ◀

We now describe a dynamic program based poly-time algorithm to solve the *localized container selection problem.* This completes the proof of Theorem 10.

Algorithm for localized container selection. We define the dynamic program variable, $\mathcal{M}(e, k_e)$, for a given cell e, as the optimal cost of assigning all input points in e, to $k_e \leq k$ newly chosen container points in e, along with the set $\mathcal{R}$ of representative container points. We note that this variable can be computed in polynomial time using ideas in [15]. For completeness, we describe a simple algorithm to compute this variable for every e and $k_e \leq k$.

We recall that by the problem definition, all the container points in e are incomparable and have the same cost, C. Let $c_1(x_1, y_1), c_2(x_2, y_2), \ldots, c_l(x_l, y_l)$ be the ordering of the container points in e, in the descending order of the y_i. That is $y_1 \geq y_2 \geq \ldots \geq y_l$ and $x_1 \leq x_2 \leq \ldots \leq x_l$. For a given index $i \in [l]$ and integer $k_i \leq k_e$, we define the variable $\mathcal{N}(i, k_i, j)$ as the optimal cost of assigning every input point, (x, y), in e, such that $y > y_{i+1}$, by choosing k_i container points with index $\leq i$, with $j \leq i$ being the highest index container point chosen (that is c_j is chosen and none of $c_{j+1}, \ldots, c_i$ are chosen). The following recurrence computes the variable $\mathcal{N}(i, k_i, j)$. Let n_i be the number of input points contained by c_i, whose y-co-ordinates are $> y_{i+1}$. If c_i is chosen,

$$\mathcal{N}(i, k_i, i) = \underset{j<i}{\mathsf{Min}}\ \mathcal{N}(i-1, k_i-1, j) + n_i C$$

Now, if c_i is not chosen and c_j is the highest index container point chosen, with $j \leq i$, we assign the input points contained in c_i with x-coordinate $> x_j$ and y-coordinate $> y_{i+1}$ to the nearest representative container point (if no such point exists, then the cost of assignment is ∞). Further, we assign those, so far, unassigned input points with y-co-ordinate $> y_{i+1}$ and x-co-ordinate $\leq x_j$ to c_j. Let C_i denote the total cost of assignment of all these input points. We have

$$\mathcal{N}(i, k_i, j) = \mathcal{N}(i-1, k_i, j) + C_i$$

We can compute $\mathcal{M}$, using the following equation: $\mathcal{M}(e, k_e) = \underset{j \leq l}{\mathsf{Min}}\ \mathcal{N}(l, k_e, j)$ Let there be μ cells in total. We order them arbitrarily as $e_1, e_2 \ldots e_\mu$. We define the variable $\mathcal{D}(i, k_i)$ as the total cost of assigning all the input points in the cells e_j, for $j \in [i]$, while choosing k_i new container points from these cells and using the representative set $\mathcal{R}$. The following simple recurrence defines the dynamic program.

$$\mathcal{D}(i, k_i) = \underset{\ell \leq k_i}{\mathsf{Min}}\ \mathcal{D}(i-1, k_i - \ell) + \mathcal{M}(e_i, \ell)$$

The optimal solution has a cost $\mathcal{D}(\mu, k)$.

▶ Remark. This approach does not extend directly even to dimension $d = 3$. There are issues in both main steps of the algorithm (1) we do not know a similar construction with $O(\log n)$ cells, and (2) the localized container selection problem also appears hard. In Section 3.2 we obtain an algorithm for the discrete container selection problem in $d > 2$ dimensions, using a linear programming relaxation and prove Theorem 11.

3.2 Discrete container selection in higher dimension

We now consider the discrete container selection problem in any dimension $d > 2$. Recall that $\mathcal{C}$ denotes the input points and $\mathscr{F}$ the potential container points. We prove Theorem 11. Our algorithm is based on the linear programming relaxation in the adjacent figure.

$$\begin{array}{lll} \text{Min} & \sum_{i\in\mathscr{F}} \|i\| \sum_{j\in\mathcal{C}} y_{ij} & \\ s.t. & y_{ij} \le x_i, & \forall i \in \mathscr{F}, j \in \mathcal{C}, \\ & y_{ij} = 0, & \forall j \not\prec i, \\ & \sum_{i\in\mathscr{F}} y_{ij} \ge 1, & \forall j \in \mathcal{C}, \\ & \sum_{i\in\mathscr{F}} x_i \le k, & \\ & x, y \ge 0. & \end{array}$$

When the x and y variables are restricted to lie in $\{0,1\}$ note that we obtain an exact formulation. This LP relaxation is similar to the one for (non-metric) facility location [12]. Indeed, our problem is a special case of non-metric k-median, for which the result of [12] implies a $\left(1+\epsilon, O(\frac{1}{\epsilon}\log n)\right)$-bicriteria approximation algorithm. Our result (Theorem 11) is an improvement for fixed dimensions since $k \le n$.

The first step in our algorithm is to solve the LP. Let (x, y) denote an optimal LP solution. The second step performs a filtering of the y variables, as in [12]. Let $C_j^* = \sum_{i\in\mathscr{F}} \|i\| \cdot y_{ij}$ denote the contribution of input point $j \in \mathcal{C}$ to the optimal LP objective. Define:

$$\overline{y}_{ij} = \begin{cases} (1+\frac{1}{\epsilon})y_{ij} & \text{if } \|i\| \le (1+\epsilon)C_j^* \\ 0 & \text{otherwise.} \end{cases}$$

Also define $\overline{x}_i = (1+\frac{1}{\epsilon})x_i$ for all $i \in \mathscr{F}$, and $\overline{C}_j = (1+\epsilon)C_j^*$ for $j \in \mathcal{C}$.

▶ Claim 16. For each $j \in \mathcal{C}$, $\sum_{i\in\mathscr{F}} \overline{y}_{ij} \ge 1$. For each $j \in \mathcal{C}$ and $i \in \mathscr{F}$, $\overline{y}_{ij} \le \overline{x}_i$.

Proof. Fix any $j \in \mathcal{C}$ and let $F_j = \{i \in \mathscr{F} : \|i\| > (1+\epsilon)C_j^*\}$. By Markov's inequality we have $\sum_{i\in F_j} y_{ij} < \frac{1}{1+\epsilon}$. So $\sum_{i\in\mathscr{F}} \overline{y}_{ij} = (1+\frac{1}{\epsilon}) \sum_{i\in\mathscr{F}\setminus F_j} y_{ij} \ge 1$. ◀

The third step of our algorithm formulates a geometric hitting-set problem with VC-dimension d. For each input point $j \in \mathcal{C}$, define a polytope $P_j \subseteq \mathbb{R}^d$ given by

$$P_j = \{v \in \mathbb{R}^d : j \prec v \text{ and } \|v\| \le \overline{C}_j\} = \left\{ v \in \mathbb{R}^d : v_r \ge j_r \, \forall r \in [d], \sum_{r=1}^{d} v_r \le \overline{C}_j \right\}.$$

Note that each P_j is described by $d+1$ *parallel inequalities* of the form:

$$\{-e_r^t v \le -j_r\}_{r=1}^d \cup \{e^t v \le \overline{C}_j\}.$$

Above e_r denotes the r^{th} coordinate unit vector and $e = (1, 1, \ldots, 1)$.

▶ Claim 17. For each $j \in \mathcal{C}$, $\sum_{i\in\mathscr{F}\cap P_j} \overline{x}_i \ge 1$.

Proof. This follows directly from Claim 16 since $\overline{y}_{ij} = 0$ for all $j \in \mathcal{C}$ and $i \notin P_j$. ◀

VC dimension bound. We use the following fact about the VC-dimension of a range space $(\mathscr{F}, \mathcal{P})$ where $\mathscr{F}$ is a finite set of points in $\mathbb{R}^d$ and $\mathcal{P}$ consists of all positive scaling and translations of a fixed polytope $Q \subseteq \mathbb{R}^d$ with $q \ge d$ facets.

▶ **Lemma 18.** *The VC-dimension of $(\mathscr{F}, \mathcal{P})$ is at most q.*

Proof. This may be a known result; in any case we give a short proof here. Let polytope $Q = \{x \in \mathbb{R}^d : \alpha_r^t x \le \beta_r, \forall r \in [q]\}$ where each $\alpha_r \in \mathbb{R}^d$ and $\beta_r \in \mathbb{R}$.

The VC-dimension is the size of the largest subset $A \subseteq \mathscr{F}$ such that $\{A \cap P : P \in \mathcal{P}\} = 2^A$. Consider any such set A. Suppose (for contradiction) that $|A| > q$, then we will show a subset $A' \subseteq A$ such that there is no $P \in \mathcal{P}$ with $A \cap P = A'$. This would prove the claim.

For each constraint $r \in [q]$ let $a_r \in A$ denote a point that maximizes $\{\alpha_r^t x : x \in A\}$. Set $A' = \{a_r\}_{r=1}^q$. Note that there is some $a' \in A \setminus A'$ since $|A| > q$ and $|A'| \le q$; moreover, by the choice of a_rs, we have $\alpha_r^t a' \le \alpha_r^t a_r$ for all $r \in [q]$.

Suppose $P \in \mathcal{P}$ is any polytope that contains all points in A'. Note that $P = \{x \in \mathbb{R}^d : \alpha_r^t x \le \gamma_r, \forall r \in [q]\}$ for some $\{\gamma_r \in \mathbb{R}\}_{r=1}^q$ since it is a scaled translation of the fixed polytope Q. Since $a_r \in P$ for each $r \in [q]$, we have $\gamma_r \ge \alpha_r^t a_r \ge \alpha_r^t a'$. This means that $a' \in P$ as well. Hence there is no set $P \in \mathcal{P}$ with $P \cap A = A'$. ◀

Applying Lemma 18 we obtain $(\mathscr{F}, \{P_j : j \in \mathcal{C}\})$ has VC-dimension at most $d+1$. Moreover, by Claim 17 the hitting set instance $(\mathscr{F}, \{P_j : j \in \mathcal{C}\})$ has a fractional hitting set $\{\overline{x}_i : i \in \mathscr{F}\}$ of size $(1 + \frac{1}{\epsilon})k$. Thus we can use the following well-known result:

▶ **Theorem 19** ([8, 5])**.** *Given any hitting set instance on a set-system with VC-dimension d and a fractional hitting set of size k, there is a polynomial time algorithm to compute an integral hitting set of size $O(d \log(dk)) \cdot k$.*

This completes the proof of Theorem 11.

▶ Remark. We can also use this LP-based approach to obtain a constant-factor bicriteria approximation for the discrete container selection problem in $\mathbb{R}^2$. This is based on the ϵ-net result for "pseudo-disks" in $\mathbb{R}^2$ [14] and the observation that in dimension two the above set-system $(\mathscr{F}, \{P_j : j \in \mathcal{C}\})$ is a collection of pseudo-disks. However, the constant factor obtained via this approach is much worse than the direct approach in Section 3.1.

4 Hardness Results

In this section, we provide hardness results for the continuous and discrete container selection problems in dimension $d = 3$. All hardness results discussed here are strongly NP-hard. The reductions are based on the planar degree 3 vertex cover problem. The following restriction of this problem is also known to be NP-hard [7].

▶ **Definition 20** (Plane Degree 3 Vertex Cover (PVC))**.** The input is a bound k and a plane drawing of a degree 3 planar graph $G = (V, E)$ with girth at least 4, where the Euclidean distance between any pair $u, v \in V$ of vertices is exactly one if $(u, v) \in E$ and at least $\sqrt{3}$ if $(u, v) \notin E$. The decision problem is to determine whether G has a vertex cover of size at most k.

We first show that the following auxiliary problem is NP-hard.

▶ **Definition 21** (Δ-hitting problem)**.** The input is a bound k, a set V of points in the plane where each pairwise distance is at least one and a set $\{\Delta_e\}_{e \in E}$ of (possibly intersecting) equilateral triangles with side $s := \frac{2}{\sqrt{3}}$ that are all translates of each other. The goal is to find a subset $T \subseteq V$ with $|T| \le k$ such that $T \cap \Delta_e \neq \emptyset$ for all $e \in E$.

▶ **Theorem 22.** *The Δ-hitting problem is NP-hard.*

Proof. We reduce the NP-hard PVC problem to the Δ-hitting problem (refer to Figure 3). An instance of PVC consists of a plane drawing of graph $G = (V, E)$ and bound k. We construct an instance of the Δ-hitting problem as follows. The set of points is V and the bound is k. Note that the the distance between each pair of points is at least one, by Definition 20. For each edge $e = (u, v) \in E$ we can find (in polynomial time) an equilateral triangle Δ_e with side $s = \frac{2}{\sqrt{3}}$ such that $V \cap \Delta_e = \{u, v\}$. To see this, first note that we can easily find $\Delta_e \ni u, v$ as $d(u, v) = 1$. Since the diameter of Δ_e is $\frac{2}{\sqrt{3}} < \sqrt{3}$ the vertices $V \cap \Delta_e$ form a clique in G, and as G has girth 4 we must have $|V \cap \Delta_e| = 2$. The set of triangles in the Δ-hitting problem is $\{\Delta_e\}_{e\in E}$. Moreover, we can ensure that the triangles $\{\Delta_e\}_{e\in E}$ are all translates of some canonical triangle. It is now clear that the Δ-hitting problem is a yes-instance if and only if the PVC instance has a vertex cover of size at most k. ◀

▶ **Theorem 23.** *The 3-dimensional discrete container selection problem is NP-hard.*

Proof. We reduce the Δ-hitting problem to this problem (refer to Figure 4). Consider an instance as described in Definition 21. We construct an instance of the discrete problem in $\mathbb{R}^3$ as follows. Set $A = 2|V|$ and let Π denote the plane $x+y+z = A$. We place the points V and triangles $\{\Delta_e\}_{e\in E}$ of the Δ-hitting instance on plane Π oriented so that every triangle Δ_e is parallel[1] to the triangle $\{(A, 0, 0), (0, A, 0), (0, 0, A)\}$. We can ensure that all points in V are in the positive orthant since A is large. The potential container points are V. Observe that for each triangle Δ_e there is a unique point $p_e \in \mathbb{R}^3$ such that $\Delta_e = \Pi \cap \{x \in \mathbb{R}^3 : p_e \prec x\}$. The set of input points is $\{p_e\}_{e\in E}$. The bound k is same as for the Δ-hitting problem.

It is easy to see that the discrete container selection instance has a feasible solution with k containers if and only if the Δ-hitting instance is a yes-instance. ◀

We immediately have the following corollary of the Theorem 23, which stems from the fact that it is NP-hard to even test feasibility of the discrete container selection problem.

▶ **Corollary 24.** *It is NP-hard to approximate the 3-dimensional discrete container selection problem within any approximation guarantee.*

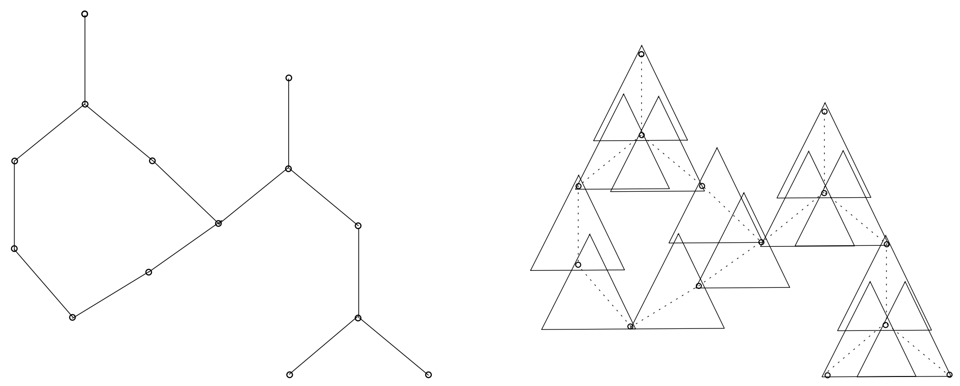
▶ **Theorem 25.** *The* 3*-dimensional continuous container selection problem is NP-hard.*

Proof. We reduce a special variant of the discrete container selection problem whose instances are defined as in Theorem 23. Let $\mathcal{I}_1 = (\mathcal{C}, \mathscr{F}, k)$ denote an instance of the discrete container selection problem g from Theorem 23 where $\mathcal{C}$ are the input points and $\mathscr{F}$ denotes the potential container points. Note that all points of $\mathscr{F}$ lie on the plane $x + y + z = A$, and the distance between every pair of points in $\mathscr{F}$ is at least one. Observe that the latter property implies that the points in $\mathscr{F}$ are incomparable.

We construct an instance $\mathcal{I}_2 = (\mathcal{C}', k')$, of the continuous problem in the following way. Fix parameter $\delta < \frac{1}{2}$. For every point $c \in \mathscr{F}$ we define another point $\hat{c} := c + \delta(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$; note that $\|\hat{c}\| = \|c\| + \delta$ and $\hat{c}$ dominates c but no other point in $\mathscr{F} \setminus \{c\}$. Let $\hat{\mathscr{F}} = \{\hat{c} : c \in \mathscr{F}\}$. Observe that this is well-defined: since the distance between every pair of points in $\mathscr{F}$ is at least one, any point dominating more than one point of $\mathscr{F}$ costs at least $A + 1$.

Now, the set $\mathcal{C}'$ of input points is constructed as follows. Let $M_1 \gg |\mathcal{C}|A$ and $M_2 \gg 2(|\mathcal{C}|A + |\mathscr{F}|M_1)$ be two sufficiently large integers. For each $c \in \mathscr{F}$, we create M_1 input points at c and M_2 input points at $\hat{c}$, which are added to $\mathcal{C}'$. Finally we also add the points $\mathcal{C}$ to $\mathcal{C}'$. The bound $k' := k + |\mathscr{F}|$. We claim that $\mathcal{I}_1$ is feasible if and only if $\mathcal{I}_2$ has a solution of cost at most $T := |\mathcal{C}|A + |\mathscr{F}|(M_1 + M_2)(A + \delta) - kM_1\delta$.

[1] Two triangles Δ_1 and Δ_2 are parallel if and only if their corresponding sides are mutually parallel



(a) Plane degree 3 vertex cover instance **(b)** Δ-hitting set instance

Figure 3 Reduction of a PVC instance to a Δ-hitting set instance:For every edge in Figure 3a, we construct an equilateral triangle, in Figure 3b, that contains the incident vertices of the edge and no other vertex. All such triangles are translates of each other. A vertex cover in the former instance is a Δ-hitting set in the latter and vice versa.

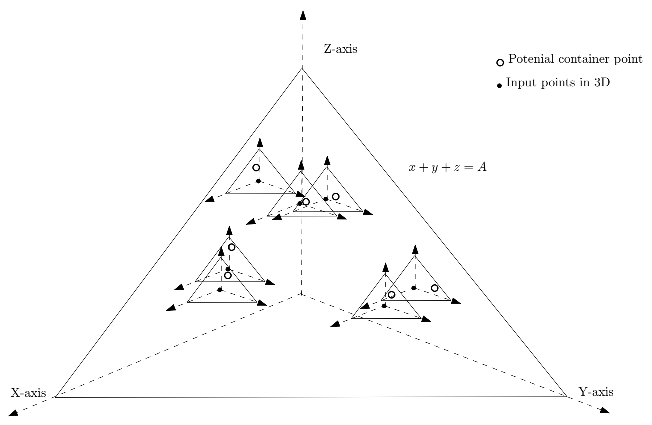


Figure 4 Reduction of the Δ-hitting set problem to the discrete container selection problem (in 3D). We consider special instances of the latter where all the potential container points are assumed to be on the plane $\Pi \equiv x + y + z = A$ and all input points lie below Π. Notice that the projections of input points onto Π form equilateral triangles as shown. Any feasible solution for the container selection problem is a Δ-hitting set in the resulting instance and vice versa.

Forward direction. Let $S = \{c_1, c_2, \ldots, c_k\}$ be the set of container points chosen by a feasible solution of $\mathcal{I}_1$. Consider the set $S' = S \cup \hat{\mathscr{F}}$. Observe that S' is a feasible solution for the instance $\mathcal{I}_2$. We now compute the assignment cost of this solution.

- The assignment cost for each point in $\mathscr{C}$ is A (it is covered by S).
- The input points at locations of S have assignment cost A (there are kM_1 such points).
- The remaining $(|\mathscr{F}| - k)M_1 + |\mathscr{F}|M_2$ input points have assignment cost $A + \delta$ each.

Therefore the total cost of this solution is exactly T.

Backward direction. Let S' with $|S'| = k + |\mathscr{F}|$ be a feasible solution to $\mathcal{I}_2$ of cost at most T. We first argue that $\hat{\mathscr{F}} \subseteq S'$. Indeed, assume that it is not true. Observe that, in this case, the input points at $\hat{\mathscr{F}}$ should be dominated by $< |\mathscr{F}|$ container points. So some container point $s \in S'$ should dominate input points at two distinct locations $\hat{c_i}$ and $\hat{c_j}$. Note that $|s| \geq A + 1$ since $c_i, c_j \prec s$ (using the distance one separation between points of $\mathscr{F}$). Hence any such solution has assignment cost at least $AM_1|\mathscr{F}| + (A+\delta)M_2|\mathscr{F}| + (1-\delta)M_2 > T$ using the definition of M_2. We now assume $\hat{\mathscr{F}} \subseteq S'$. Next we show that each of the remaining k container points in S' dominates at most one point of $\mathscr{F}$. If $s \in S'$ dominates two distinct locations c_i and c_j, its cost $|s| \geq A + 1$ as noted above. However, any input point can be assigned to one of the container points in $\hat{\mathscr{F}}$ at cost $A + \delta < A + 1$, which makes point s redundant.

Now we show that each of the k container points $S' \setminus \hat{\mathscr{F}}$ dominates some point of $\mathscr{F}$. If not, consider a container point $s' \in S'$ that does not dominate any $\mathscr{F}$ point. Let $f \in \mathscr{F}$ be some point which is not dominated by any $S' \setminus \hat{\mathscr{F}}$; note that this must exist since each $S' \setminus \hat{\mathscr{F}}$ dominates at most one $\mathscr{F}$-point and $|S' \setminus \hat{\mathscr{F}}| = k \leq |\mathscr{F}|$. Suppose we modify the solution by removing s' and adding f: the increase in cost is at most $|\mathcal{C}|(A+\delta) + M_1 A - M_1(A+\delta) < 0$ by the definition of M_1. Thus, $\hat{\mathscr{F}} \subseteq S'$ and $S'' = S' \setminus \hat{\mathscr{F}} \subseteq \mathscr{F}$. We now claim that S'' dominates every point of $\mathcal{C}$. For a contradiction, suppose there is some point of $\mathcal{C}$ that is not dominated by S'' : then this point has assignment cost $A + \delta$. Every other points of $\mathcal{C}$ has assignment cost at least A. The assignment cost of points at $\hat{\mathscr{F}} \cup \mathscr{F}$ is $|\mathscr{F}|(M_1 + M_2)(A + \delta) - kM_1\delta$. So the total assignment cost is at least $T + \delta$, a contradiction. Hence S'' is a feasible solution for $\mathcal{I}_1$. ◀

References

1 Marcel R. Ackermann, Johannes Blömer, and Christian Sohler. Clustering for metric and nonmetric distance measures. *ACM Transactions on Algorithms (TALG)*, 6(4):59, 2010.

2 Amazon EC2. In `http://aws.amazon.com/ec2/`.

3 Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. In *IPCO*, pages 52–63, 2014.

4 Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *STOC*, pages 106–113, 1998.

5 Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.

6 Jaroslaw Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An Improved Approximation for k-median, and Positive Correlation in Budgeted Optimization. In *SODA*, 2015.

7 Tomás Feder and Daniel H. Greene. Optimal algorithms for approximate clustering. In *STOC*, pages 434–444, 1988.

8 David Haussler and Emo Welzl. ϵ-Nets and Simplex Range Queries. *Discrete &Computational Geometry*, 2:127–151, 1987.

9 Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony Joseph, Scott Shenker, and Ion Stoica. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. In *NSDI*, 2011.

10 A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3), September 1999.

11 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *STOC*, pages 901–910, 2013.

12 Jyh-Han Lin and Jeffrey Scott Vitter. epsilon-approximations with minimum packing constraint violation (extended abstract). In *STOC*, pages 771–782, 1992.

13 Private cloud. In `http://wikipedia.org/wiki/Cloud_computing#Private_cloud`.

14 Evangelia Pyrga and Saurabh Ray. New existence proofs epsilon-nets. In *SOCG*, pages 199–207, 2008.

15 Baruch Schieber. Computing a Minimum Weight k-Link Path in Graphs with the Concave Monge Property. *Journal of Algorithms*, 29(2):204–222, 1998.

16 V. Vavilapalli, A. Murthy, C. Douglis, A. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwiele. Apache Hadoop YARN: Yet Another Resource Negotiator. In *SoCC*, 2013.

17 Joel Wolf, Zubair Nabi, Viswanath Nagarajan, Robert Saccone, Rohit Wagle, Kirsten Hildrum, Edward Ping, and Kanthi Sarpatwar. The X-Flex Cross-Platform Scheduler: Who's The Fairest Of Them All? In *Middleware*, 2014.

Tight Bounds for Graph Problems in Insertion Streams*

Xiaoming Sun[1] **and David P. Woodruff**[2]

1 **Institute of Computing Technology, CAS, China**
sunxiaoming@ict.ac.cn
2 **IBM Almaden, USA**
dpwoodru@us.ibm.com

Abstract

Despite the large amount of work on solving graph problems in the data stream model, there do not exist tight space bounds for almost any of them, even in a stream with only edge insertions. For example, for testing connectivity, the upper bound is $O(n \log n)$ bits, while the lower bound is only $\Omega(n)$ bits. We remedy this situation by providing the first tight $\Omega(n \log n)$ space lower bounds for randomized algorithms which succeed with constant probability in a stream of edge insertions for a number of graph problems. Our lower bounds apply to testing bipartiteness, connectivity, cycle-freeness, whether a graph is Eulerian, planarity, H-minor freeness, finding a minimum spanning tree of a connected graph, and testing if the diameter of a sparse graph is constant. We also give the first $\Omega(nk \log n)$ space lower bounds for deterministic algorithms for k-edge connectivity and k-vertex connectivity; these are optimal in light of known deterministic upper bounds (for k-vertex connectivity we also need to allow edge duplications, which known upper bounds allow). Finally, we give an $\Omega(n \log^2 n)$ lower bound for randomized algorithms approximating the minimum cut up to a constant factor with constant probability in a graph with integer weights between 1 and n, presented as a stream of insertions and deletions to its edges. This lower bound also holds for cut sparsifiers, and gives the first separation of maintaining a sparsifier in the data stream model versus the offline model.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases communication complexity, data streams, graphs, space complexity

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.435

1 Introduction

In a data stream one sees a sequence of elements $a_1, \ldots, a_m$ one by one and one would like to evaluate certain functions of the stream. There are example data streams which come from internet search logs, network traffic, sensor networks, and scientific data streams. The elements a_i may be numbers, points, edges in a graph, etc. Due to the sheer size of the sequence, very stringent requirements are imposed on a data stream algorithm. For instance, it is often assumed that the algorithm can only make one, or a small number, of passes over the stream. Moreover, the algorithm is assumed to have very limited memory, which in particular makes storing the stream in its entirety infeasible. We refer the reader to the surveys [4, 29] for a more thorough introdution to this area.

* Xiaoming Sun was supported in part by the National Natural Science Foundation of China Grant 61170062, 61222202, 61433014 and the China National Program for support of Top-notch Young Professionals.

In this paper we focus on the case when the elements a_i are edges of an underlying graph G. That is, we insert edges into the graph one at a time, and would like to compute a function of G. Graphs arise is many applications to model relationships bewteen basic entities, such as links between webpages or network flows between destinations. These graphs are often massive and running classical algorithms on such graphs has proven quite challenging. This has motivated the recent work on processing graphs in the data stream model. In the case when only edges are inserted into the stream (as opposed to also being deleted), we have algorithms for a number of problems, including testing connectivity, finding minimum spanning trees, computing cut and spectral sparsifiers, counting subgraphs, finding matchings, and many other problems. We refer the reader to the survey by McGregor [28] for an overview of these results.

It is known for many graph problems that there is a space lower bound of $\Omega(n)$ [16, 17]. The graph streaming model is therefore sometimes identified with the "semi-streaming" model, which allows the streaming algorithm to use $n \cdot \text{polylog}(n)$ bits of space. Note that this is still a substantial improvement over the naïve algorithm of storing the graph, which may take $\Omega(n^2)$ bits of space. Despite the fact that we have $n \cdot \text{polylog}(n)$ space upper bounds and $\Omega(n)$ space lower bounds for a number of graph problems in the data stream model, we are not aware of a single natural problem for which we have asymptotically tight bounds. Could it be that simple upper bounds, such as the $O(n \log n)$ bit upper bound for testing if a graph is connected by maintaining a spanning forest in the stream, can be improved using more clever hashing techniques to represent the edges, perhaps chosen adaptively as the stream is presented? Such a scheme could potentially allow us to avoid spending $O(\log n)$ bits to remember each edge in the spanning forest.

Dowling and Wilson [14] show that the deterministic communication complexity of connectivity is $\Omega(n \log n)$ bits, see also [31] for a discussion. Via standard connections to data streams (see Section 1.3 below), this implies any deterministic streaming algorithm requires $\Omega(n \log n)$ bits. However, to the best of our knowledge, there was no lower bound for randomized 1-way protocols known, prior to this work, stronger than $\Omega(n)$ bits.

Graph problems in a data stream should be contrasted to a number of other areas in streaming for which tight asymptotic space bounds are known, such as estimating frequency moments [2, 7, 18, 22, 23, 27], empirical entropy [8, 11, 20, 21, 22], numerical linear algebra [10], and compressed sensing [3, 30]. The goal of this paper is to remedy this situation.

1.1 Our Results

Throughout this paper we will restrict our attention to 1-pass algorithms and focus on their space complexity. Our focus is on the model in which edges are only allowed to be inserted into the graph, i.e., the "insertion model", rather than also being allowed to be deleted, which is referred to as the "turnstile model". Since we prove lower bounds, this only makes our bounds stronger. We will, however, show how to use our techniques to obtain stronger lower bounds in the turnstile model for approximating the minimum cut of a graph.

Our results are summarized in Table 1. We provide the first tight space lower bounds for a number of graph problems in a stream of edge insertions. In particular, for randomized algorithms which succeed with constant probability, we show an $\Omega(n \log n)$ lower bound for testing if a graph is connected, testing if a graph with $O(n)$ edges has diameter at most 5 or diameter ∞, testing if a graph is Eulerian, testing if a graph is bipartite, testing if a graph is cycle-free, finding a minimum spanning tree in a graph that is promised to be connected, testing if a graph is planar, and testing whether a graph contains a fixed graph H as a minor. No lower bounds better than $\Omega(n)$ were known for any of these problems. Many of the upper

Table 1 Summary of our results. All lower bounds are new and given by this work. In the comments we say "UB stores graph" for those problems for which there is no graph with more than $C \cdot n$ edges, for a constant $C > 0$, satisfying the property. For such problems it suffices to store all edges in the graph and abort if more than $C \cdot n$ edges are inserted, yielding an $O(n \log n)$ bit upper bound. All upper bounds, with the exception of Eulerian-testing, either come from previous work or "UB stores graph" applies to the problem. For Eulerian-testing the upper bound maintains a spanning forest and the parities of node degrees, using that a graph is Eulerian iff it is connected and all node degrees are even.

Problem	Lower Bound	Upper Bound	Comments
Connectivity	$\Omega(n \log n)$	$O(n \log n)$ [28]	–
Diameter in sparse graphs	$\Omega(n \log n)$	$O(n \log n)$	UB stores graph
Eulerian-testing	$\Omega(n \log n)$	$O(n \log n)$	UB: spanning forest, degrees
Bipartiteness	$\Omega(n \log n)$	$O(n \log n)$ [28]	–
Cycle-freeness	$\Omega(n \log n)$	$O(n \log n)$	UB stores graph
MST in connected graphs	$\Omega(n \log n)$	$O(n \log n)$ [28]	–
Planarity	$\Omega(n \log n)$	$O(n \log n)$	UB stores graph
H-Minor Free	$\Omega(n \log n)$	$O(n \log n)$	UB stores graph
k-Edge Connectivity	$\Omega(kn \log n)$	$O(kn \log n)$ [13]	Deterministic bounds
k-Vertex Connectivity w/edge duplications	$\Omega(kn \log n)$	$O(kn \log n)$ [15]	Deterministic bounds
$O(1)$-Approximate Minimum Cut	$\Omega(n \log^2 n)$	$O(n \log^4 n)$ [1, 24]	Bounds in the turnstile model

bounds follow simply by storing all edges in the graph and aborting if the number of edges is too large; see Table 1 for details.

Next, we turn to k-edge connectivity, which is equivalent to testing if the minimum cut of the graph is at least k. There is a deterministic space upper bound of $O(kn \log n)$ bits [13, 28]. We show a matching $\Omega(kn \log n)$ bit lower bound for deterministic algorithms. For randomized algorithms our space bound is a weaker $\Omega(kn)$, and closing the $\log n$ factor gap for deterministic and randomized k-edge connectivity algorithms remains an important open question. For k-vertex connectivity, there is a deterministic space upper bound of $O(kn \log n)$ bits due to [15]. We are able to prove a matching $\Omega(kn \log n)$ bit lower bound for deterministic algorithms, but require that multiple edges are allowed, i.e., our hard instance is a multi-graph for this problem. We notice, however, that the upper bound of [15] also holds for multi-graphs. Our lower bound becomes a weaker $\Omega(kn)$ in the case of randomized algorithms, and closing the $\log n$ factor gap for randomized algorithms for k-vertex connectivity and/or removing the edge duplication assumption is an important open problem.

Finally, we illustrate the power of our technique by proving an $\Omega(n \log^2 n)$ lower bound for approximating the minimum cut up to a constant factor of a graph with integer weights between 1 and n, in the turnstile model. The same lower bound holds for cut sparsifiers (since they can be used to approximate the minimum cut), and gives the first separation of maintaining a sparsifier in the data stream model versus the offline model. Indeed, in the offline model, by a result of Batson et al. [6] it is possible to build a cut sparsifier (in fact, a stronger notion of a spectral sparsifier) of a graph using only $O(n)$ reweighted edges of the input graph, with $O(\log n)$ bits to specify each edge and its weight. Our $\Omega(n \log^2 n)$ bit lower bound shows it is fundamentally impossible to implement the algorithm of [6] in a dynamic stream. For general integer edge weights between 1 and W, our lower bound is $\Omega(n \log n \log W)$.

1.2 Our Techniques

Our results come from identifying a new two-player one-way communication problem which generalizes the well-studied Index problem [26], to a problem Perm which is more suitable for proving graph lower bounds. Despite the simplicity of the Perm problem, we are able to apply it to the wide array of problems above. In this problem, Alice is given a permutation σ on $[n] \overset{\text{def}}{=} \{1, 2, \dots, n\}$, which she represents in a slightly redundant way as an $n \log n$-length bitstring $\sigma(1), \dots, \sigma(n)$ formed by concatenating the images of $1, 2, \dots, n$ under σ. We call this the *redundant encoding* of σ. Bob is interested in obtaining the i-th bit in the redundant encoding of σ. We show that if Alice sends a single message to Bob, then for Bob to succeed with constant probability, Alice's message needs to be $\Omega(n \log n)$ bits long. In other words, the randomized 1-way communication complexity $R^{1-way}(\mathsf{Perm}) = \Omega(n \log n)$. We generalize this to the case where Alice has r permutations $\sigma^1, \dots, \sigma^r$ each on $[n]$, while Bob now has an index $i \in [n]$, an index $k \in [r]$, as well as Alice's permutations $\sigma^{k+1}, \dots, \sigma^r$, and Bob is interested in the i-th bit in the redundant encoding of σ^k. We call this problem r-AugmentedPerm and show $R^{1-way}(r\text{-}\mathsf{AugmentedPerm}) = \Omega(rn \log n)$.

After identifying Perm and r-AugmentedPerm as the right problems to study, the proofs of their respective lower bounds follow standard information-theoretic arguments used to prove lower bounds for Index and direct sum theorems in streaming [5, 9], with small modifications to account for the redundancy. The second part of our proofs is reducing graph problems to these communication problems. The core idea of our lower bounds is to identify a permutation σ as a random matching on a bipartite graph with n vertices in the left part L and n vertices in the right part R. This is Alice's input graph G in many of our reductions. Alice runs the streaming algorithm on G, sends the state to Bob, who then inserts edges into G in a problem-specific way. As Bob is interested in learning a bit of the redundant encoding of Alice's permutation, this corresponds to a bit j of the unique neighbor in R of a vertex $u \in L$. We therefore create gadgets which group all vertices in R into two groups, based on the value of their j-th bit, and connect these groups to vertices in L in different ways depending on the particular problem.

1.3 Preliminaries

Let $f : X \times Y \to \{0, 1\}$ be a Boolean function, where X and Y are two arbitrary sets. In the *one-way* communication model Alice receives an input $x \in X$ and Bob receives an input $y \in Y$. Alice is only allowed to send one message to Bob and no message is allowed to be sent from Bob to Alice. The goal is for Bob to compute $f(x, y)$. The communication cost is measured by the number of bits Alice sends in the worst case. Denote by $D^{1-way}(f)$ the minimum communication cost over all deterministic one-way protocols for f. For a randomized protocol P, we say P has error probability at most ϵ if $\Pr(P(x, y) = f(x, y)) \geq 1 - \epsilon$ for all inputs x and y. The randomness here is only over the private coin tosses of Alice and Bob. The one-way (bounded-error) randomized communication complexity of f, denote by $R^{1-way}(f)$, is the minimum communication cost over all randomized one-way protocols for f with error probability at most $1/3$.

Communication lower bounds on $D^{1-way}(f)$ and $R^{1-way}(f)$ provide lower bounds on the memory required of deterministic and randomized data stream algorithms, respectively, via a standard reduction. Indeed, Alice creates a stream σ_x from her input x, and runs the streaming algorithm on σ_x, passing the state of the algorithm to Bob. Bob creates a stream σ_y from his input y, and continues the execution of the streaming algorithm on σ_y. If the output of the streaming algorithm on the concatenated stream $\sigma_x \circ \sigma_y$ can be used to solve the problem f, either deterministically or with constant error probability, then the space complexity of the streaming algorithm must be at least $D^{1-way}(f)$ or $R^{1-way}(f)$, respectively.

We also need a few concepts and notation from information theory. We refer the reader to [12] for a more comprehensive introduction. We give a short primer on the standard properties we use in Appendix A.

2 Permutation Problems

We consider the following communication problem Perm which will be used in our reductions. In this problem Alice is given a permutation σ of $[n]$, represented as an ordered list $\sigma(1), \sigma(2), \ldots, \sigma(n)$. This list has $n \log n$ bits. Bob is given an index $i \in [n \log n]$ and would like to know the i-th bit of σ. This problem is similar to the well-studied Index problem in randomized 1-way communication complexity, but it is slightly different in that $\sigma(1), \ldots, \sigma(n)$ is a redundant encoding of a permutation.

▶ **Lemma 1.** $R^{1-way}(\mathsf{Perm}) = \Omega(n \log n)$.

Proof. Let us place the uniform distribution on strings σ. Let $M(\sigma)$ be Alice's message to Bob, which is a random variable depending on the randomness of σ and the private random coin tosses of Alice. Then $R^{1-way}(\mathsf{Perm}) \geq H(M(\sigma)) \geq I(M(\sigma); \sigma)$, so it suffices to lower bound $I(M(\sigma); \sigma)$. We write σ_j to denote the j-th bit in the list $\sigma(1), \ldots, \sigma(n)$, where $j \in \{1, 2, \ldots, n \log n\}$.

By the chain rule,

$$\begin{aligned} I(M(\sigma); \sigma) &= \sum_{j=1}^{n \log n} I(M(\sigma); \sigma_j \mid \sigma_{<j}) \\ &= \sum_j (H(\sigma_j \mid \sigma_{<j}) - H(\sigma_j \mid M(\sigma), \sigma_{<j})) \\ &\geq \sum_j H(\sigma_j \mid \sigma_{<j}) - \sum_j H(\sigma_j \mid M(\sigma)) \\ &= H(\sigma) - \sum_j H(\sigma_j \mid M(\sigma)). \end{aligned}$$

Using Stirling's approximation, $H(\sigma) = \log n! = n \log(n/e) + O(\log n)$. Now consider $H(\sigma_j \mid M(\sigma))$. Since M is randomized protocol which succeeds on every pair of inputs (σ, i) with probability at least 9/10, and M does not depend on j, it follows that from $M(\sigma)$ Bob can predict σ_i for any given i with probability at least 9/10. By Fano's inequality, for each j this implies $H(\sigma_j \mid M(\sigma)) \leq H(1/10)$. Hence,

$$I(M(\sigma); \sigma) \geq n \log(n/e) - H(1/10) n \log n \geq (1 - H(1/10)) n \log n - O(n).$$

This completes the proof. ◀

We also define the problem r-AugmentedPerm, used in our reductions. In this problem, Alice is given r permutations $\sigma^1, \ldots, \sigma^r$, where each σ^j is represented as a list of $n \log n$ bits. Bob is given an index $i \in [n \log n]$, an index $k \in [r]$, and is given $\sigma^{k+1}, \sigma^{k+2}, \ldots, \sigma^r$. Bob's goal is to output σ_i^k, which is the i-th bit of σ^k.

▶ **Lemma 2.** $R^{1-way}(r\text{-}\mathsf{AugmentedPerm}) = \Omega(rn \log n)$.

Proof. As in the proof of Lemma 1, we place the uniform distribution on strings σ^j, for each $j \in [r]$, and the σ^j are independent. Let $M(\sigma^1, \ldots, \sigma^r)$ be Alice's message to Bob, which is a random variable depending on the randomness of $\sigma^1, \ldots, \sigma^r$ and her private random coin tosses. Then $R^{1-way}(r\text{-}\mathsf{AugmentedPerm}) \geq H(M(\sigma^1, \ldots, \sigma^r)) \geq I(M(\sigma^1, \ldots, \sigma^r); \sigma^1, \ldots, \sigma^r)$, so it suffices to lower bound $I(M(\sigma^1, \ldots, \sigma^r); \sigma^1, \ldots, \sigma^r)$.

By the chain rule,

$$I(M(\sigma^1, \ldots, \sigma^r); \sigma^1, \ldots, \sigma^r) \quad = \quad \sum_{k=1}^{r} I(M(\sigma^1, \ldots, \sigma^r); \sigma^k \mid \sigma^{k+1}, \ldots, \sigma^r) \tag{1}$$

We claim that for each $k \in [r]$,

$$I(M(\sigma^1, \ldots, \sigma^r); \sigma^k \mid \sigma^{k+1}, \ldots, \sigma^r) = \Omega(n \log n).$$

To see this, consider any fixing of the random variables $\sigma^{k+1}, \ldots, \sigma^r$, and let Π be a randomized protocol which succeeds on every input to $\mathsf{AugmentedPerm}$ with probability at least $9/10$, over its random coin tosses. Then, given an input (σ, i) to the Perm problem, Alice and Bob can use Π as follows. Alice hardwires the fixed values of $\sigma^{k+1}, \ldots, \sigma^r$. Alice also sets $\sigma^k = \sigma$. Finally, she randomly and independently samples uniform permutations $\sigma^1, \ldots, \sigma^{k-1}$. Bob, given i as the input to Perm, also holds the input k and has the hardwired values of $\sigma^{k+1}, \ldots, \sigma^r$. Alice and Bob run Π on these inputs to $\mathsf{AugmentedPerm}$, and output whatever Π outputs. By correctness of Π, it follows that this is a correct 1-way protocol for the Perm problem with probability at least $9/10$. Hence, as argued in the proof of Lemma 1, $I(M'(\sigma); \sigma) = \Omega(n \log n)$, where M' is Alice's resulting message function in the created protocol for Perm. By construction,

$$I(M(\sigma^1, \ldots, \sigma^r); \sigma^k \mid \sigma^{k+1}, \ldots, \sigma^r) = I(M'(\sigma); \sigma) = \Omega(n \log n),$$

as claimed. Plugging into (1), it follows that

$$I(M(\sigma^1, \ldots, \sigma^r); \sigma^1, \ldots, \sigma^r) = \Omega(rn \log n),$$

which completes the proof. ◀

3 Lower Bounds for Graph Problems

3.1 Connectivity

We start with an $\Omega(n \log n)$ bit lower bound for the randomized one-way communication of the graph connectivity problem, denoted Conn. In this problem, Alice has a subset E_A of edges of an undirected graph G on a set V of n vertices, while Bob has a disjoint subset E_B of the edges of G. Alice sends a single randomized message $M(E_A)$ to Bob, who should decide if the graph $(V, E_A \cup E_B)$ is connected. Bob should succeed with probability at least $9/10$. We let $R^{1-way}(\mathsf{Conn})$ denote the minimum, over all correct protocol for Conn with probability at least $9/10$, of the maximum length message sent, over all inputs and random coin tosses.

▶ **Theorem 3.** $R^{1-way}(\mathsf{Conn}) = \Omega(n \log n)$.

Proof. We perform a reduction from Perm on instances of size $n/2$. Alice, given a permutation σ, creates a perfect matching from $[n/2]$ to $[n/2]$ where the i-th left vertex connects to the $\sigma(i)$-th right vertex. Alice's edgeset E_A consists of the edges in this perfect matching. Let L and R denote the two parts of the vertex set V, each of size $n/2$.

Suppose Bob has the input i to Perm. This corresponds to the ℓ-th bit in $\sigma(j)$ for some $j \in [n/2]$ and $\ell \in [\log(n/2)]$. Bob creates his input edgeset to Conn from i as follows. Let $S \subset R$ denote the subset of vertices whose ℓ-th bit is equal to 0. Bob's input edgeset E_B consists of a spanning tree on the vertices in $(L \setminus \{j\}) \cup S$. We can ensure the edges of the spanning tree are disjoint from E_A by including a new vertex w, and including edges from all vertices in $(L \setminus \{j\}) \cup S$ to w.

Observe that since the vertices in $L \setminus \{j\}$ are connected, it follows that since we placed a perfect matching from L to R, that any vertex u is connected to any other vertex except possibly to j or $\sigma(j)$. Now, if the $\sigma(j)$-th right vertex has its ℓ-th bit equal to 0, then $\sigma(j)$ is connected to S, and hence to $L \setminus \{j\}$. It follows that the graph is connected. On the other hand, if the $\sigma(j)$-th right vertex has its ℓ-th bit equal to 1, then the edge from the j-th left vertex to the $\sigma(j)$-th right vertex is isolated, that is, it is not incident to any other vertices. In this case the graph is disconnected.

Let $M(E_A)$ be Alice's message to Bob in a protocol for Conn. Suppose Bob can decide, from $M(E_A)$ and E_B, if the resulting graph on vertex set $L \cup R$ and edgeset $E_A \cup E_B$ is connected with probability at least 9/10. It follows that Bob can solve Perm with probability at least 9/10, and therefore from Lemma 1, $R^{1-way}(\mathsf{Conn}) = \Omega(n \log n)$. ◀

▶ Remark. The lower bound in Theorem 3 is matched by a simple $O(n \log n)$ bit upper bound in which Alice sends a spanning forest of her edges to Bob.

3.2 Diameter

As a corollary of Conn, we show a lower bound for the following Diameter-k problem on sparse graphs, i.e., graphs with $O(n)$ edges: Given $k \in [n-1]$, Bob wants to decide if the diameter d of $(V, E_A \cup E_B)$ is at most k, or ∞.

▶ **Theorem 4.** *For any $k \geq 4$, $R^{1-way}(\mathsf{diameter}\text{-}k) = \Omega(n \log n)$.*

Proof. In the Conn proof, instead of only putting a spanning tree on the vertices in $(L\setminus\{j\})\cup S$, we also put a clique on the vertices in $L \setminus \{j\}$ and a clique on the vertices in S. It follows that the diameter of $(V, E_A \cup E_B)$ is either $+\infty$ if the graph is disconnected, or 4 if the graph is connected. Therefore, the Diameter-k problem is as hard as Conn. ◀

▶ Remark. For sparse graphs the upper bound is just to store the entire graph with $O(n)$ edges.

3.3 Eulerian-Testing

In this part we show a lower bound for the Eulerian problem: Bob wants to decide if $(V, E_A \cup E_B)$ is an Eulerian graph.

▶ **Theorem 5.** $R^{1-way}(\mathsf{Eulerian}) = \Omega(n \log n)$.

Proof. In the Conn proof, call the graph $G_1 = (L, R, E)$, make another copy of the graph $G_2 = (L', R', E')$, i.e., in G_2 the edges are the same as in G_1. Alice and Bob also add the following edges to E_A and E_B: if there is an edge (u, v) in G_1, add edges (u, v') and (u', v). Let $V = L \cup R \cup L' \cup R'$. It is easy to check that the degree of every vertex is twice its degree in G_1, thus is an even number. Therefore, the resulting graph is Eulerian if and only if it is connected, but this is equivalent to the connectivity of G_1. Hence, $R^{1-way}(\mathsf{Eulerian}) = \Omega(n \log n)$. ◀

▶ Remark. An upper bound is to maintain a spanning forest to test connectivity, as well as to maintain the parities of all node degrees. Then one uses that a graph is Eulerian if and only if it is connected and all node degrees are even.

3.4 Bipartiteness

We now give a lower bound for the the Bipartite problem. In this problem Alice has a subset E_A of edges of an undirected graph G on a set V of n vertices, while Bob has a disjoint subset E_B of the edges of G. Alice sends a single randomized message $M(E_A)$ to Bob, who should decide if the graph $(V, E_A \cup E_B)$ is bipartite.

▶ **Theorem 6.** $R^{1-way}(\mathsf{Bipartite}) = \Omega(n \log n)$.

Proof. We again reduce from Perm on instances of size $n/2$. Alice, given a permutation σ, creates a perfect matching from $[n/2]$ to $[n/2]$ where the i-th left vertex connects to the $\sigma(i)$-th right vertex. Alice's edgeset E_A consists of the edges in this perfect matching. Let L and R denote the two parts of the vertex set V, each of size $n/2$.

Suppose Bob has the input i to Perm. This corresponds to the ℓ-th bit in $\sigma(j)$ for some $j \in [n/2]$ and $\ell \in [\log(n/2)]$. Bob creates his input edgeset to Bipartite from i as follows. We create a new node w (so the input graph has $n+1$ nodes) and Bob includes an edge in E_B from the j-th vertex in L, denoted v, to w. Bob also includes all edges in E_B from w to any vertex in R whose ℓ-th bit is equal to 0.

Since E_A is a perfect matching, it is bipartite. Further, all edges in E_B are incident to w, and therefore G is bipartite if and only if there is no odd cycle which contains w. If we remove the edge $\{v, w\}$ then the graph is acyclic, and so any cycle must contain $\{v, w\}$, and hence also $\{v, \sigma(v)\}$, and hence also $\{\sigma(v), w\}$. It follows that an odd cycle exists iff $\{\sigma(v), w\}$ is in E_B, that is, iff the ℓ-th bit of $\sigma(j)$ is equal to 0.

It follows that Bob can solve Perm with probability at least 9/10, and therefore from Lemma 1, $R^{1-way}(\mathsf{Bipartite}) = \Omega(n \log n)$. ◀

▶ Remark. There is an upper bound of $O(n \log n)$ bits for bipartiteness; see section 3.1 of [28]. It is stated as a streaming algorithm which immediately gives rise to a 1-way communication protocol.

3.5 Cycle-free

As a corollary of Bipartite, we show a lower bound for the following Cycle-free problem: Bob wants to decide if there is a cycle in $(V, E_A \cup E_B)$.

▶ **Theorem 7.** $R^{1-way}(\mathsf{cycle\text{-}free}) = \Omega(n \log n)$.

Proof. In the Bipartite proof, if the ℓ-th bit of $\sigma(j)$ is 0, then there is a cycle between j, $\sigma(j)$ and w. If the ℓ-th bit of $\sigma(j)$ is 0, then there is no cycle. Therefore, $R^{1-way}(\mathsf{cycle\text{-}free}) = \Omega(n \log n)$. ◀

▶ Remark. There is an upper bound of $O(n \log n)$ bits by storing the first $n-1$ edges of G. If G has more than $n-1$ edges, it necessarily contains a cycle. If it has fewer, one can test whether it contains a cycle.

3.6 Minimum Spanning Tree

We now present an application to the minimum spanning tree (MST) of a connected graph. In the MST problem, Alice has a subset E_A of edges of an undirected graph G on a set V of n vertices, while Bob has a disjoint subset E_B of the edges of $G = (V, E_A, \cup E_B)$, and the players are promised that G is a connected graph. Alice sends a single randomized message $M(E_A)$ to Bob, who should output a spanning tree of G. Note that in the case that G is unweighted, all such spanning trees are minimal.

▶ **Theorem 8.** $R^{1-way}(\mathsf{MST}) = \Omega(n \log n)$.

Proof. We again reduce from Perm on instances of size $n/2$. Alice, given a permutation σ, creates a perfect matching from $[n/2]$ to $[n/2]$ where the i-th left vertex connects to the $\sigma(i)$-th right vertex. Alice's edgeset E_A consists of the edges in this perfect matching. Let L and R denote the two parts of the vertex set V, each of size $n/2$.

Bob's edgeset E_B is just a line connecting the vertices in L. Observe that $G = (V, E_A \cup E_B)$ is connected and has $n-1$ edges, and therefore is itself the only spanning tree of G. Therefore, Bob can reconstruct G. Hence, the players can solve the Perm problem with probability at least $9/10$ given a protocol for MST which succeeds with probability at least $9/10$, and therefore and therefore from Lemma 1, $R^{1-way}(\mathsf{MST}) = \Omega(n \log n)$. ◀

▶ Remark. There is an $O(n \log n)$ bits of space upper bound for MST for integer weights bounded by poly(n), see section 2.1 of [28].

3.7 k-Edge Connectivity

▶ **Theorem 9.** $D^{1-way}(k\text{-Edge Connectivity}) = \Omega(nk \log n)$.

Proof. Consider a bipartite graph with parts L and R. L is partitioned into $k/2$ blocks L_i each of n/k vertices. Similarly R is partitioned into $k/2$ blocks R_i each of n/k vertices. For each pair (L_i, R_j) containing a left block and a right block, we have a random perfect matching between the blocks. Alice has all of these edges. Bob is interested in the t-th bit of the neighbor of vertex a in the block R_b.

We show a $kn \log n$ lower bound for deterministic protocols. Bob guesses each vertex c in R_b to see if it the neighbor of vertex a in R_b; since the protocol is deterministic it does not err, and so he will figure out the correct neighbor and thus reconstruct the graph as follows. Suppose c is the current candidate. Bob adds edges connecting vertices in the set $(L \setminus a) \cup (R \setminus c)$ to make the graph on these vertices k-edge-connected. This can be done without edge duplications by introducing a clique of k new vertices, and connecting all vertices in $(L \setminus a) \cup (R \setminus c)$ to each of these k new vertices. Bob also adds a set W of $O(k)$ additional vertices and places a k-connected graph H on vertex set $\{a, c\} \cup W$. The resulting graph is k-edge-connected iff there is an edge $\{a, c\}$; if there is such an edge, then by deleting $k/2 - 1$ neighbors of a and $k/2 - 1$ neighbors of c, one deletes in total $k - 2$ edges and causes H to be disconnected from the rest of the graph. On the other hand if there is no such edge, then at least k edges need to be deleted.

Hence, Bob reconstructs the input graph, which by construction has $\Omega(nk \log n)$ bits of entropy, since there are $(k/2)^2$ random perfect matchings, so the logarithm of the number of possible graphs is $\log_2(((n/k)!)^{k^2/4})$, which gives the desired $\Omega(nk \log n)$ bits of entropy lower bound. This implies $D^{1-way}(k\text{-Edge Connectivity}) = \Omega(nk \log n)$. ◀

▶ Remark. There is a deterministic upper bound of $O(kn \log n)$ bits. See Theorem 1 in [13].

3.8 k-Vertex Connecvitiy

▶ **Theorem 10.** $D^{1-way}(k\text{-Vertex Connectivity}) = \Omega(nk \log n)$.

Proof. Consider a bipartite graph with parts L and R. L is partitioned into $k-1$ blocks L_i each of $n/(k-1)$ vertices. Similarly R is partitioned into $k-1$ blocks R_i each of $n/(k-1)$ vertices. For each pair of left block and right block (L_i, R_j), we have a random perfect matching between the blocks. Alice has all of these edges. Bob is interested in the t-th bit of the neighbor of a in the block R_b. Bob guesses each vertex c in R_b to see if c is the neighbor of a in R_b; since the protocol is deterministic, it does not err, so Bob will figure out the correct neighbor as follows. Suppose c is the current candidate.

Bob adds k new vertices and connects every vertex except a to all k new vertices. Bob also puts a clique on the k new vertices. Finally, Bob adds the edge $\{a, c\}$ to the graph.

Then if $\{a, c\}$ existed in the graph before Bob added it, then vertex a still has only $k-1$ neighbors and so the graph is disconnected by deleting these $k-1$ neighbors. On the other hand, if $\{a, c\}$ did not exist in the graph, then vertex a now has k neighbors and the graph is k-vertex connected.

Thus, since the protocol is deterministic, Bob can reconstruct the input graph, which has $\Omega(kn \log n)$ bits of entropy by construction. This shows $D^{1-way}(k\text{-Vertex Connectivity}) = \Omega(nk \log n)$. ◀

▶ Remark. There is a streaming algorithm due to Eppstein et al. [15] (see also [19] for a discussion) which includes a new edge $\{a, b\}$ iff there are no k-vertex disjoint paths connecting a to b among the edges already stored. Correctness follows from Menger's theorem for vertex connectivity. Note that the algorithm is insensitive to edge duplications, and is deterministic. It achieves $O(kn \log n)$ bits of space.

3.9 H-minor-free

Let H be a fixed graph. In the H-minor-free problem Alice has a subset E_A of edges of an undirected graph G on a set V of n vertices and Bob has a subset E_B of the edges of G. Alice sends a single randomized message $M(E_A)$ to Bob, who should decide if the graph $(V, E_A \cup E_B)$ is H-minor-free. Bob should succeed with probability at least 9/10.

▶ **Theorem 11.** *For any fixed graph H with minimum degree at least 2, $R^{1-way}(H\text{-minor-free}) = \Omega(n \log n)$.*

Proof. We again reduce from Perm on instances of size $n/2$. Alice, given a permutation σ, creates a perfect matching from $[n/2]$ to $[n/2]$ where the i-th left vertex connects to the $\sigma(i)$-th right vertex. Alice's edgeset E_A consists of the edges in this perfect matching. Let L and R denote the two parts of the vertex set V, each of size $n/2$.

Suppose Bob has the input i to Perm. This corresponds to the ℓ-th bit in $\sigma(j)$ for some $j \in [n/2]$ and $\ell \in [\log(n/2)]$. Bob creates his input to H-minor-free from i and H as follows. Suppose H has $r+1$ vertices $h_0, h_1, \dots, h_r$. Since $\delta(H) \geq 2$, w.l.o.g., we assume there are two edges $\{h_0, h_1\}$ and $\{h_0, h_2\}$ in $E(H)$. Bob creates r new vertices $p_1, \dots, p_r$ and puts a copy of $H \setminus \{h_0, h_1\}$ between j and $p_1, \dots, p_r$ with the mapping $h_0 \to j$ and $h_i \to p_i$ $(i = 1, \dots, r)$, i.e., j, $p_1, \dots, p_r$, is an isomorphism to H except for the one edge (j, p_1). Let $V = L \cup R \cup \{p_1, \dots, p_r\}$ and $S \subset R$ denote the subset of vertices whose ℓ-th bit is equal to 1. Bob also includes all edges in E_B from p_1 to all vertices in S.

Now we claim that there is an H-minor in $(V, E_A \cup E_B)$ iff the ℓ-th bit of $\sigma(j)$ is 1. Indeed, if the ℓ-th bit of $\sigma(j)$ is 1, then there is a edge between $\sigma(j)$ and p_1 in E_B. We can contract the edges $\{j, \sigma(j)\}$ and $\{\sigma(j), p_1\}$ and we obtain a copy of H. Hence H is a minor of $E_A \cup E_B$.

For the case that the ℓ-th bit of $\sigma(j)$ is 0, then $\sigma(j) \notin S$ and j is not adjacent to any vertex in S. Note that we can delete all isolated matching edges since $\delta(H) \geq 2$. Also since j is not adjacent to a vertex in S and H has minimum degree at least 2, we can contract all edges incident to S, and then contract all nodes in S to p_1. These operations preseve the property of having an H-minor since the minimum degree of H is at least 2. We can also contract $\sigma(j) \in R$ to j since $\deg(\sigma(j)) = 1$ and $\delta(H) \geq 2$. At this point we are left with vertices $p_1, ..., p_r$, and j, with edgeset exactly equal to that of H except we are missing the edge (p_1, j). This implies H is not a minor. ◀

▶ Remark. Kostochka [25] shows that an H-minor-free graph has at most $O(n|H|\sqrt{\log|H|})$ edges. Storing all these edges can be done using $O(n \log n)$ bits. So our lower bound is tight.

As a corollary, we show that the Planar problem in which Bob want to decide if $(V, E_A \cup E_B)$ is planar also has a lower bound of $\Omega(n \log n)$ bits.

▶ **Corollary 12.** $R^{1-way}(\mathsf{Planar}) = \Omega(n \log n)$.

Proof. Consider $H = K_5$ in the previous proof. The graph $(V, E_A \cup E_B)$ is either contracted to a K_5, or a K_5 with one missing edge, according to the ℓ-th bit of $\sigma(j)$. Notice that the former one is non-planar and the latter is planar. Therefore, Planar is as hard as Perm. ◀

▶ Remark. There is an $O(n \log n)$ bit upper bound for Planar, simply store up to $3n$ edges and use that any graph with more than $3n$ edges cannot be planar.

3.10 Approximate Min-Cut

In this section we show an $\Omega(n \log^2 n)$ lower bound for 1-way protocols which provide a constant-factor approximation with constant probability to the minimum cut value of a graph with integer edge weights between 1 and n. Our lower bound also implies an $\Omega(n \log^2 n)$ bit lower bound for $O(1)$-approximate cut sparsifiers of such graphs, as such sparsifiers can be used to approximate the minimum cut value. We let c-approx Min-Cut denote the problem of approximating the minimum cut up to a factor of $c > 1$.

▶ **Theorem 13.** *Suppose a graph has edges with weights in the set* $\{1, 2, \dots, W\}$*, where* W *is at most* $2^{\gamma n}$ *for a sufficiently small constant* $\gamma > 0$*. Then for any constant* $c > 1$*,* $R^{1-way}(c\text{-approx Min-Cut}) = \Omega(n \log n \log W)$*. In particular, if* $W = n$*, then* $R^{1-way}(c\text{-approx Min-Cut}) = \Omega(n \log^2 n)$*.*

Proof. We can reduce the r-AugmentedPerm problem to c-approx Min-Cut, which we abbreviate as the Min-Cut problem in the remainder of the proof. Let $\alpha = 2c + 1$ and $r = \log_\alpha W$. Suppose Alice is given r random permutations $\sigma^1, \dots, \sigma^r$ of size $n/2$. As in the construction in the proof of Conn, Alice creates r perfect matchings from $\sigma^1, \dots, \sigma^r$ as her input to the Min-Cut problem. All edge weights in the i-th instance are equal to α^i $(i = 1, \dots, r)$. The largest weight is $\alpha^r = W$. In expectation there will be $O(r^2)$ duplicate edges when we overlay the matchings. Alice can send the identities of all the duplicate edges together with which instances they occur in to Bob, and not include these in her graph. This only requires $O(\log^2 W(\log n + \log logW))$ additional communication from Alice to Bob, using our choice of r. This is negligible given the upper bound on W in the theorem statement.

Suppose in the r-AugmentedPerm problem Bob is given an index $i \in [n \log n]$, and index $k \in [r]$, and $\sigma^{k+1}, \dots, \sigma^r$. For the Min-Cut problem, Bob will delete all the edges in the matchings corresponding to $\sigma^{k+1}, \dots, \sigma^n$. Bob, depending on which instances he deletes from r-AugmentedPerm, can decide which of the duplicate edges to put back in Alice's graph. As in the Conn problem, Bob also adds a spanning tree to the vertices $(L \setminus \{j\}) \cup S$ in the matching corresponding to σ_k.

Now if $\sigma_i^k = 0$, then the graph is connected. Hence the minimum cut is at least α^k. On the other hand, if $\sigma_i^k = 1$, then the k-th instance is disconnected. In the instances corresponding to $\sigma^1, \dots, \sigma^{k-1}$, all the vertices have degree one, so if we cut $\{j, \sigma^k(j)\}$ from other vertices, the total weight of this cut is at most $2(\alpha + \alpha^2 + \cdots + \alpha^{k-1}) < \frac{2 \cdot \alpha^k}{\alpha - 1} = \frac{\alpha^k}{c}$. Therefore, if Bob can c-approximate the total weight of the min-cut, then he can distinguish the case $\sigma_i^k = 0$ from $\sigma_i^k = 1$, i.e., he can solve r-AugmentedPerm. ◀

Acknowledgements. We thank Andrew McGregor for helpful discussions regarding this work.

References

1 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14, 2012.

2 Alexandr Andoni, Huy L. Nguyên, Yury Polyanskiy, and Yihong Wu. Tight lower bound for linear sketches of moments. In *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 25–32, 2013.

3 Khanh Do Ba, Piotr Indyk, Eric Price, and David P. Woodruff. Lower bounds for sparse recovery. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1190–1197, 2010.

4 Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, pages 1–16, 2002.

5 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.

6 Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM J. Comput.*, 41(6):1704–1721, 2012.

7 Vladimir Braverman, Jonathan Katzman, Charles Seidell, and Gregory Vorsanger. An optimal algorithm for large frequency moments using o(n^(1-2/k)) bits. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 531–544, 2014.

8 Amit Chakrabarti, Graham Cormode, and Andrew McGregor. A near-optimal algorithm for estimating the entropy of a stream. *ACM Transactions on Algorithms*, 6(3), 2010.

9 Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, and Andrew Chi-Chih Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 270–278, 2001.

10 Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 205–214, 2009.

11 Peter Clifford and Ioana Cosma. A simple sketching algorithm for entropy estimation over streaming data. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 – May 1, 2013*, pages 196–206, 2013.

12 T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley and Sons, Inc., 1991.

13 Michael S. Crouch, Andrew McGregor, and Daniel Stubbs. Dynamic graphs in the sliding-window model. In *Algorithms – ESA 2013 – 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 337–348, 2013.

14 T.A. Dowling and R.M. Wilson. Whitney number inequalities for geometric lattices. *Proc. Amer. Math. Soc.*, 47:504–512, 1975.

15 David Eppstein, Zvi Galil, Giuseppe F. Italiano, and Amnon Nissenzweig. Sparsification – a technique for speeding up dynamic graph algorithms. *J. ACM*, 44(5):669–696, 1997.

16 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. In *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, pages 531–543, 2004.

17 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the streaming model: the value of space. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 745–754, 2005.

18 Sumit Ganguly. Polynomial estimators for high frequency moments. *CoRR*, abs/1104.4552, 2011.

19 Sudipto Guha, Andrew McGregor, and D. Tench. Vertex and hyperedge connectivity in dynamic graph streams. In *PODS*, 2015.

20 Nicholas J. A. Harvey, Jelani Nelson, and Krzysztof Onak. Sketching and streaming entropy via approximation theory. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 489–498, 2008.

21 T. S. Jayram and David P. Woodruff. Optimal bounds for johnson-lindenstrauss transforms and streaming problems with subconstant error. *ACM Transactions on Algorithms*, 9(3):26, 2013.

22 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1161–1178, 2010.

23 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 41–52, 2010.

24 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 561–570, 2014.

25 Alexandr Kostochka. The minimum hadwiger number for graphs with a given mean degree of vertices. *Metody Diskret. Analiz.*, 38:37–58, 1982.

26 Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 596–605, 1995.

27 Yi Li and David P. Woodruff. A tight lower bound for high frequency moment estimation with small error. In *Approximation, Randomization, and Combinatorial Optimization.*

Algorithms and Techniques – 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings, pages 623–638, 2013.

28 Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014.

29 S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

30 Eric Price and David P. Woodruff. (1 + eps)-approximate sparse recovery. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 295–304, 2011.

31 Ran Raz and Boris Spieker. On the "log rank"-conjecture in communication complexity. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 168–176, 1993.

A Information Theory Facts

For a discrete random variable X with possible values $\{x_1, x_2, \ldots, x_n\}$, the Shannon entropy of X is defined as $H(X) = -\sum_{i=1}^{n} \Pr(X = x_i) \log_2 \Pr(X = x_i)$. Let $H_b(p) = -p \log_2 p - (1 - p) \log_2(1 - p)$ denote the binary entropy function when $p \in (0, 1)$. For two random variables X and Y with possible values $\{x_1, x_2, \ldots, x_n\}$ and $\{y_1, y_2, \ldots, y_m\}$, respectively, the conditional entropy of X given Y is defined as $H(X \mid Y) = \sum_{i,j} \Pr(X = x_i, Y = y_j) \log_2 \frac{\Pr(Y=y_j)}{\Pr(X=x_i,Y=y_j)}$. Let $I(X; Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$ denote the mutual information between two random variables X, Y. Let $I(X; Y \mid Z)$ denote the mutual information between two random variables X, Y conditioned on Z, i.e., $I(X; Y \mid Z) = H(X \mid Z) - H(X \mid Y, Z)$.

The following summarizes several basic properties of entropy and mutual information.

▶ **Proposition 14.** *Let X, Y, Z, W be random variables.*

1. *If X takes value in $\{1, 2, \ldots, m\}$, then $H(X) \in [0, \log m]$.*
2. *$H(X) \geq H(X \mid Y)$ and $I(X; Y) = H(X) - H(X \mid Y) \geq 0$.*
3. *If X and Z are independent, then we have $I(X; Y \mid Z) \geq I(X; Y)$. Similarly, if X, Z are independent given W, then $I(X; Y \mid Z, W) \geq I(X; Y \mid W)$.*
4. *(Chain rule of mutual information) $I(X, Y; Z) = I(X; Z) + I(Y; Z \mid X)$.*
   More generally, for any random variables $X_1, X_2, \ldots, X_n, Y$,
   $I(X_1, \ldots, X_n; Y) = \sum_{i=1}^{n} I(X_i; Y \mid X_1, \ldots, X_{i-1})$.
   Thus, $I(X, Y; Z \mid W) \geq I(X; Z \mid W)$.
5. *(Fano's inequality) Let X be a random variable chosen from domain $\mathcal{X}$ according to distribution μ_X, and Y be a random variable chosen from domain $\mathcal{Y}$ according to distribution μ_Y. For any reconstruction function $g : \mathcal{Y} \to \mathcal{X}$ with error δ_g,*

$$H_b(\delta_g) + \delta_g \log(|\mathcal{X}| - 1) \geq H(X \mid Y).$$

A Chasm Between Identity and Equivalence Testing with Conditional Queries*

Jayadev Acharya[1], **Clément L. Canonne**[2], **and Gautam Kamath**[1]

1 **Massachusetts Institute of Technology**
32 Vassar Street, Cambridge MA, USA
`{jayadev,g}@csail.mit.edu`

2 **Columbia University**
500 W 120th Street, New York NY, USA
`ccanonne@cs.columbia.edu`

Abstract

A recent model for property testing of probability distributions [11, 9] enables tremendous savings in the sample complexity of testing algorithms, by allowing them to condition the sampling on subsets of the domain.

In particular, Canonne, Ron, and Servedio [9] showed that, in this setting, testing identity of an unknown distribution D (i.e., whether $D = D^*$ for an explicitly known D^*) can be done with a *constant* number of samples, independent of the support size n – in contrast to the required $\sqrt{n}$ in the standard sampling model. However, it was unclear whether the same held for the case of testing equivalence, where *both* distributions are unknown. Indeed, while Canonne, Ron, and Servedio [9] established a polylog(n)-query upper bound for equivalence testing, very recently brought down to $\tilde{O}(\log\log n)$ by Falahatgar et al. [13], whether a dependence on the domain size n is necessary was still open, and explicitly posed by Fischer at the Bertinoro Workshop on Sublinear Algorithms [14]. In this work, we answer the question in the positive, showing that any testing algorithm for equivalence must make $\Omega\left(\sqrt{\log\log n}\right)$ queries in the conditional sampling model. Interestingly, this demonstrates an intrinsic qualitative gap between identity and equivalence testing, absent in the standard sampling model (where both problems have sampling complexity $n^{\Theta(1)}$).

Turning to another question, we investigate the complexity of support size estimation. We provide a doubly-logarithmic upper bound for the adaptive version of this problem, generalizing work of Ron and Tsur [22] to our weaker model. We also establish a logarithmic lower bound for the non-adaptive version of this problem. This latter result carries on to the related problem of non-adaptive uniformity testing, an exponential improvement over previous results that resolves an open question of Chakraborty, Fischer, Goldhirsh, and Matsliah [11].

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.3 Probability and Statistics

Keywords and phrases property testing, probability distributions, conditional samples

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.449

* The full version of this paper can be found at [1].

1 Introduction

'No, Virginia, there is no constant-query tester.'

Understanding properties and characteristics of an unknown probability distribution is a fundamental problem in statistics, and one that has been thoroughly studied. However, it is only since the seminal work of Goldreich and Ron [15] and Batu et al. [5] that the problem has been considered through the lens of theoretical computer science, more particularly in the setting of *property testing*.

Over the following decade, a flurry of subsequent work explored and delved into this new area, resulting in a better and often complete understanding of a number of questions in distributional property testing (see e.g. [15, 4, 6, 20, 24, 2, 7, 23, 17, 3, 12, 29] or [8] for a survey). In many cases, these culminated in provably sample-optimal algorithms. However, the standard setting of distribution testing, where one only obtains independent samples from an unknown distribution D, does not encompass *all* scenarios one may encounter. In recent years, stronger models have thus been proposed to capture more specific situations [16, 11, 9, 18, 10]: among these is the *conditional oracle model* [11, 9] which will be the focus of our work. In this setting, the testing algorithms are given the ability to sample from conditional distributions: that is, to specify a subset S of the domain and obtain samples from D_S, the distribution induced by D on S (the formal definition of the model can be found in Definition 2.1). In particular, the hope is that allowing algorithms to have stronger interactions with the unknown underlying distributions might significantly reduce the number of samples they need, thereby sidestepping the strong lower bounds that hold in the standard sampling model.

1.1 Background and previous work

We focus in this paper on proving lower bounds for testing two extremely natural properties of distributions, namely *equivalence testing* ("are these two datasets identically distributed?") and *support size estimation* ("how many different outcomes can actually be observed?"). Along the way, we use some of the techniques we develop to obtain an upper bound on the query complexity of the latter. We state below the informal definition of these two problems, along with closely related ones (uniformity and identity testing). Hereafter, "oracle access" to a distribution D over $[n] = \{1, \dots, n\}$ means access to samples generated independently from D.

Uniformity testing: granted oracle access to D, decide whether $D = \mathcal{U}$ (the uniform distribution on $[n]$) or is far from it;

Identity testing: granted oracle access to D and the full description of a fixed D^*, decide whether $D = D^*$ or is far from it;

Equivalence (closeness) testing: granted independent oracle accesses to D_1, D_2 (both unknown), decide whether $D_1 = D_2$ or D_1, D_2 are far from each other.

Support size estimation: granted oracle access to D, output an estimate of the size of the support[1] $\operatorname{supp}(D) = \{\, x : D(x) > 0 \,\}$, accurate within a multiplicative factor.

It is not difficult to see that each of the first three problems generalizes the previous, and is therefore at least as hard. All of these tasks are known to require sample complexity $n^{\Omega(1)}$

[1] For this problem, it is typically assumed that all points in the support have probability mass at least $\Omega(1)/n$, as without such guarantee it becomes impossible to give any non-trivial estimate (consider for instance a distribution D such that $D(i) \propto 1/2^{in}$).

in the standard sampling model (SAMP); yet, as prior work [11, 9] shows, their complexity decreases tremendously when one allows the stronger type of access to the distribution(s) provided by a conditional sampling oracle (COND). For the problems of uniformity testing and identity testing, the sample complexity even becomes a constant provided the testing algorithm is allowed to be *adaptive* (i.e. when the next queries it makes can depend on the samples it previously obtained).

Testing uniformity and identity

Given the complete description of a distribution D^* over $[n]$, a parameter $\varepsilon > 0$, and oracle access to a distribution D, identity testing asks to distinguish the case $D_1 = D^*$ from where their total variation distance $\mathrm{d_{TV}}(D, D^*)$ is at least ε. This is a generalization of uniformity testing, where D^* is taken to be the uniform distribution over $[n]$. The complexity of these tasks is well-understood in the sampling model; in particular, it is known that for both uniformity and identity testing $\Theta\big(\sqrt{n}/\varepsilon^2\big)$ samples are necessary and sufficient (see [15, 5, 20, 29] for the tight bounds on these problems).

The uniformity testing problem emphasizes the additional flexibility granted by conditional sampling: as Canonne, Ron, and Servedio [9] showed, in this setting only $\tilde{O}\big(1/\varepsilon^2\big)$ adaptive queries now suffice (and this is optimal, up to logarithmic factors). They further prove that identity testing has constant sample complexity as well, namely $\tilde{O}\big(1/\varepsilon^4\big)$ – very recently improved to a near-optimal $\tilde{O}\big(1/\varepsilon^2\big)$ by Falahatgar et al. [13]. The power of the COND model is evident from the fact that a task requiring polynomially many samples in the standard model can now be achieved with a number of samples *independent of the domain size n*.

Focusing on the case of non-adaptive algorithms, Chakraborty et al. [11] describe a $\mathrm{poly}(\log n, 1/\varepsilon)$-query tester for uniformity, showing that even without the full power of conditional queries one can still get an exponential improvement over the standard sampling setting. They also obtain an $\Omega(\log\log n)$ lower bound for this problem, and leave open the possibility of improving this lower bound up to a logarithmic dependence. The present work answers this question, establishing that any non-adaptive uniformity tester must perform $\Omega(\log n)$ conditional queries.

Testing equivalence

A natural generalization of these two testing problems is the question of *equivalence testing*, defined as follows. Given oracle access to two unknown distributions D_1 and D_2 over $[n]$ and a parameter $\varepsilon > 0$, equivalence testing asks to distinguish between the cases $D_1 = D_2$ and $\mathrm{d_{TV}}(D_1, D_2) > \varepsilon$. This problem has been extensively studied over the past decade, and its sample complexity is now known to be $\Theta(\max(n^{2/3}/\varepsilon^{4/3}, \sqrt{n}/\varepsilon^2))$ in the sampling model [5, 30, 12].

In the COND setting, Canonne, Ron, and Servedio showed that equivalence testing is possible with only $\mathrm{poly}(\log n, 1/\varepsilon)$ queries. Concurrent to our work, Falahatgar et al. [13] brought this upper bound down to $\tilde{O}\big((\log\log n)/\varepsilon^5\big)$, a *doubly exponential* improvement over the $n^{\Omega(1)}$ samples needed in the standard sampling model. However, these results still left open the possibility of a constant query complexity: given that both uniformity and identity testing admit constant-query testers, it is natural to wonder where equivalence testing lies[2].

This question was explicitly posed by Fischer at the Bertinoro Workshop on Sublinear Algorithms 2014 [14]: in this paper, we make decisive progress in answering it, ruling out

[2] It is worth noting that an $\Omega(\log^c n)$ lower bound was known for equivalence testing in a weaker version of the conditional oracle, PAIRCOND (where the tester's queries are restricted to being either $[n]$ or subsets of size 2 [9]).

the possibility of any constant-query tester for equivalence. Along with the upper bound of Falahatgar et al. [13], our results essentially settle the dependence on the domain size, showing that $(\log\log n)^{\Theta(1)}$ samples are both necessary and sufficient.

Support size estimation

Finally, the question of approximating the support size of a distribution has been considered by Raskhodnikova et al. [21], where it was shown that obtaining *additive* estimates requires sample complexity almost linear in n. Subsequent work by Valiant and Valiant [28, 26] settles the question, establishing that an $n/\log n$ dependence is both necessary and sufficient. Note that the proof of their lower bound translates to multiplicative approximations as well, as they rely on the hardness of distinguishing a distribution with support $s \le n$ from a distribution with support $s + \varepsilon n \ge (1+\varepsilon)s$. To the best of our knowledge, the question of getting a multiplicative-factor estimate of the support size of a distribution given conditional sampling access has not been previously considered. We provide upper and lower bounds for both the adaptive and non-adaptive versions of this problem.

1.2 Our results

In this work, we make significant progress in each of the problems introduced in the previous section, yielding a better understanding of their intrinsic query complexities. We prove four results pertaining to the sample complexity of equivalence testing, support size estimation, and uniformity testing in the COND framework. Our main result on the sample complexity of equivalence testing is presented in this extended abstract, the details of the other results are available in the full version of this paper [1].

Our main result considers the sample complexity of testing equivalence with adaptive queries under the COND model, resolving in the negative the question of whether constant-query complexity was achievable [14]. More precisely, we prove the following theorem:

▶ **Theorem 1.1** (Testing Equivalence). *Any* adaptive *algorithm which, given* COND *access to unknown distributions D_1, D_2 on $[n]$, distinguishes with probability at least $2/3$ between* (a) $D_1 = D_2$ *and* (b) $d_{TV}(D_1, D_2) \ge \frac{1}{4}$, *must have query complexity* $\Omega\big(\sqrt{\log\log n}\big)$.

Combined with the recent $\tilde{O}(\log\log n)$ upper bound of Falahatgar et al. [13], this almost settles the sample complexity of this question. Furthermore, as the related task of identity

Table 1 Summary of results. Note that the lower bound (†) can also be easily derived from our lower bound on testing equivalence.

Problem	COND model	Standard model
Are D_1, D_2 (both unknown) equivalent? (adaptive)	$\tilde{O}\left(\frac{\log\log n}{\varepsilon^5}\right)$ [13] $\Omega\big(\sqrt{\log\log n}\big)$ [this work]	$\Theta\left(\max\left(\frac{n^{2/3}}{\varepsilon^{4/3}}, \frac{n^{1/2}}{\varepsilon^2}\right)\right)$ [12]
What is the support size of D? (adaptive)	$\tilde{O}\left(\frac{\log\log n}{\varepsilon^3}\right)$ [this work] $\Omega\big(\sqrt{\log\log n}\big)$ [11] (†)	$\Theta\left(\frac{n}{\log n}\right)$ [26]
What is the support size of D? (non-adaptive)	$O(\mathrm{poly}(\log n, 1/\varepsilon))$ [this work] $\Omega(\log n)$ [this work]	
Is D uniform over the domain? (non-adaptive)	$\tilde{O}\left(\frac{\log^5 n}{\varepsilon^6}\right)$ [11] $\Omega(\log n)$ [this work]	$\Theta\left(\frac{\sqrt{n}}{\varepsilon^2}\right)$ [15, 5, 20]

testing *can* be performed with a constant number of queries in the conditional sampling model, this demonstrates an intriguing and intrinsic difference between the two problems. Our result can also be interpreted as showing a fundamental distinction from the usual sampling model, where both identity and equivalence testing have polynomial sample complexity.

Next, we establish a logarithmic lower bound on *non-adaptive* support size estimation, for any factor larger than a fixed constant. This improves on the result of Chakraborty et al. [11], which gave a doubly logarithmic lower bound for constant factor support-size estimation.

▶ **Theorem 1.2** (Non-Adaptive Support Size Estimation). *Any non-adaptive algorithm which, given* COND *access to an unknown distribution D on $[n]$, estimates the size of its support up to a factor γ must have query complexity $\Omega\left(\frac{\log n}{\log \gamma}\right)$, for any $\gamma \geq \sqrt{2}$.*

Moreover, the approach used to prove this theorem also implies an analogous lower bound on *non-adaptive* uniformity testing in the conditional model, answering a conjecture of Chakraborty et al. [11]:

▶ **Theorem 1.3** (Non-Adaptive Uniformity Testing). *Any non-adaptive algorithm which, given* COND *access to an unknown distribution D on $[n]$, distinguishes with probability at least $2/3$ between* (a) $D = \mathcal{U}$ *and* (b) $\mathrm{d_{TV}}(D, \mathcal{U}) \geq \frac{1}{4}$, *must have query complexity $\Omega(\log n)$.*

We note that these results complement polylog(n)-query upper bounds, the former of which we sketch in the full version of this paper, and the latter obtained by Chakraborty et al. [11]. This shows that both of these problems have query complexity $\log^{\Theta(1)} n$ in the non-adaptive case.

Finally, we also show an upper bound for *adaptive* support size estimation. Specifically, we provide a $\tilde{O}(\log\log n)$-query algorithm for support size estimation. This shows that the question becomes *double exponentially* easier when conditional samples are allowed.

▶ **Theorem 1.4** (Adaptive Support Size Estimation). *Let $\tau > 0$ be any constant. There exists an adaptive algorithm which, given* COND *access to an unknown distribution D on $[n]$ which has minimum non-zero probability τ/n and accuracy parameter ε makes $\tilde{O}\big((\log\log n)/\varepsilon^3\big)$ queries to the oracle and outputs a value $\tilde{\omega}$ such that the following holds. With probability at least $2/3$, $\tilde{\omega} \in [\frac{1}{1+\varepsilon} \cdot \omega, (1+\varepsilon) \cdot \omega]$, where $\omega = |\mathrm{supp}(D)|$.*

1.2.1 Relation to the Ron-Tsur model

Recent work of Ron and Tsur [22] studies a model which is slightly stronger than ours. In their setting, the algorithm still performs queries consisting of a subset of the domain. However, the algorithm is also given the promise that the distribution is uniform on a subset of the domain, and whenever a query set contains 0 probability mass the oracle explicitly indicates this is the case. Their paper provides a number of results for support size estimation in this model.

We point out two connections between our work and theirs. First, our $\Omega(\log n)$ lower bound for non-adaptive support size estimation (Theorem 1.2) leads to the same lower bound for the problem in the model of Ron and Tsur. Although lower bounds in the conditional sampling setting do not apply directly to theirs, we note that our construction and analysis still carry over. This provides a nearly tight answer to this question, which was left unanswered in their paper. Also, our $\tilde{O}(\log\log n)$-query algorithm for adaptive support size estimation (Theorem 1.4) can be seen as generalizing their result to the weaker conditional sampling model (most significantly, when we are not given the promise that the distribution be uniform).

1.3 Techniques and proof ideas

We now provide an overview of the techniques and arguments used to prove our results.

Lower bound on adaptive equivalence testing

In order to prove our main $\omega(1)$ lower bound on the query complexity of testing equivalence in the conditional sampling model, we have to deal with one main conceptual issue: *adaptivity*. While the standard sampling model does not, by definition, allow any choice on what the next query to the oracle should be, this is no longer the case for COND algorithms. Quantifying the power that this grants an algorithm makes things much more difficult. To handle this point, we follow the approach of Chakraborty et al. [11] and focus on a restricted class of algorithms they introduce, called "core adaptive testers" (see Section 2.2 for a formal definition). They show that this class of testers is equivalent to general algorithms for the purpose of testing a broad class of properties, namely those which are invariant to any permutation of the domain. Using this characterization, it remains for us to show that none of these structurally much simpler core testers can distinguish whether they are given conditional access to (a) a pair of random identical distributions (D_1, D_1), or (b) two distributions (D_1, D_2) drawn according to a similar process, which are far apart.

At a high level, our lower bound works by designing instances where the property can be tested if and only if the support size is known to the algorithm. Our construction randomizes the support size by embedding the instance into a polynomially larger domain. Since the algorithm is only allowed a small number of queries, Yao's Principle allows us to argue that, with high probability, a deterministic algorithm is unable to "guess" the support size. This separates queries into several cases. First, in a sense we make precise, it is somehow "predictable" whether or not a query will return an element we have previously observed. If we do, it is similarly predictable *which* element the query will return. On the other hand, if we observe a fresh element, the query set is either "too small" or "too large." In the former case, the query will entirely miss the support, and the sampling process is identical for both types of instance. In the latter case, the query will hit a large portion of the support, and the amount of information gleamed from a single sample is minimal.

At a lower level, this process itself is reminiscent of the lower bound construction of Canonne, Ron, and Servedio [9] on testing identity (with a PAIRCOND oracle), with one pivotal twist. As in their work, both D_1 and D_2 are uniform within each of $\omega(1)$ "buckets" whose size grows exponentially and are grouped into "bucket-pairs." Then, D_2 is obtained from D_1 by internally redistributing the probability mass of each pair of buckets, so that the total mass of each pair is preserved but each particular bucket has mass going up or down by a constant factor (see Section 3.1 for details of the construction). However, we now add a final step, where in both D_1 and D_2 the resulting distribution's support is *scaled by a random factor*, effectively reducing it to a (randomly) negligible fraction of the domain. Intuitively, this last modification has the role of "blinding" the testing algorithm: we argue that unless its queries are on sets whose size somehow match (in a sense formalized in Section 3.2) this random size of the support, the sequences of samples it will obtain under D_1 and D_2 are almost identically distributed. The above discussion crucially hides many significant aspects and technical difficulties which we address in Section 3. Moreover, we observe that the lower bound we obtain seems to be optimal with regard to our proofs techniques (specifically, to the decision tree approach), and not an artifact of our lower bound instances. Namely, there appear to be conceptual barriers to strengthening our result, which would require new ideas.

Lower bound on non-adaptive support size estimation

Turning to the (non-adaptive) lower bound of Theorem 1.2, we define two families of distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, where an instance is either a draw (D_1, D_2) from $\mathcal{D}_1 \times \mathcal{D}_2$, or simply (D_1, D_1). Any distribution in $\mathcal{D}_2$ has support size γ times that of its corresponding distribution in $\mathcal{D}_1$. Yet, we argue that no non-adaptive *deterministic* tester making too few queries can distinguish between these two cases, as the tuple of samples it will obtain from D_1 or (the corresponding) D_2 is almost identically distributed (where the randomness is over the choice of the instance itself). To show this last point, we analyze separately the case of "small" queries (conditioning on sets which turn out to be much smaller than the actual support size, and thus with high probability will not even intersect it) and the "big" ones (where the query set A is so big in front of the support size S that a uniform sample from $A \cap S$ is essentially indistinguishable from a uniform sample from A). We conclude the proof by invoking Yao's Principle, carrying the lower bound back to the setting of non-adaptive *randomized* testers.

Interestingly, this argument essentially gives us Theorem 1.3 "for free:" indeed, the big-query-set case above is handled by proving that the distribution of samples returned on those queries is indistinguishable, both for $\mathcal{D}_1$ and $\mathcal{D}_2$, from samples obtained from the *actual* uniform distribution. Considering again the small-query-set case separately, this allows us to argue that a random distribution from (say) $\mathcal{D}_1$ is indistinguishable from uniform.

Upper bound on support size estimation

Our algorithm for estimating the support size to a constant factor (Theorem 1.4) is simple in spirit, and follows a guess-and-check strategy. In more detail, it first obtains a "reference point" *outside* the support, to check whether subsequent samples it may consider belong to the support. Then, it attempts to find a *rough upper bound* on the size of the support, of the form 2^{2^j} (so that only $\log\log n$ many options have to be considered); by using its reference point to check if a uniform random subset of this size contains, as it should, at least one point from the support. Once such an upper bound has been obtained using this double-exponential strategy, a refined bound is then obtained via a binary search on the new range of values for the exponent, $\{2^{j-1}, \dots, 2^j\}$. Not surprisingly, our algorithm draws on similar ideas as in [22, 25], with some additional machinery to supplement the differences in the models. Interestingly, as a side-effect, this upper bound shows our analysis of Theorem 1.1 to be tight up to a quadratic dependence. Indeed, the lower bound construction we consider (see Section 3.1) can be easily "defeated" if an estimate of the support size is known, and therefore cannot yield better than a $\Omega(\log\log n)$ lower bound. Similarly, this also shows that the adaptive lower bound for support size estimation of Chakraborty et al. [11] is also tight up to a quadratic dependence.

Organization

The rest of the paper describes details and proofs of the results mentioned in the above discussion. In Section 2, we introduce the necessary definitions and some of the tools we shall use. Section 3 covers our main result on adaptive equivalence testing, Theorem 1.1. Further details on our other results are available in the full version of this paper.

2 Preliminaries

2.1 Notation and sampling models

All throughout this paper, we denote by $[n]$ the set $\{1, \dots, n\}$, and by log the logarithm in base 2. A probability distribution over a (countable) domain $[n]$ is a non-negative function $D\colon [n] \to [0,1]$ such that $\sum_{x\in[n]} D(x) = 1$. We denote by $\mathcal{U}(S)$ the uniform distribution on a set S. Given a distribution D over $[n]$ and a set $S \subseteq [n]$, we write $D(S)$ for the total probability mass $\sum_{x\in S} D(x)$ assigned to S by D. Finally, for $S \subseteq [n]$ such that $D(S) > 0$, we denote by D_S the conditional distribution of D restricted to S, that is $D_S(x) = \frac{D(x)}{D(S)}$ for $x \in S$ and $D_S(x) = 0$ otherwise.

As is usual in distribution testing, in this work the distance between two distributions D_1, D_2 on $[n]$ will be the *total variation distance*:

$$\mathrm{d}_{\mathrm{TV}}(D_1, D_2) \overset{\text{def}}{=} \frac{1}{2}\|D_1 - D_2\|_1 = \frac{1}{2}\sum_{x\in[n]} |D_1(i) - D_2(i)| = \max_{S\subseteq[n]}(D_1(S) - D_2(S)) \tag{1}$$

which takes value in $[0,1]$.

In this work, we focus on the setting of *conditional access* to the distribution, as introduced and studied in [11, 9]. We reproduce below the corresponding definition of a conditional oracle, henceforth referred to as COND:

▶ **Definition 2.1** (Conditional access model)**.** Fix a distribution D over $[n]$. A COND *oracle for D*, denoted COND_D, is defined as follows: the oracle takes as input a *query set $S \subseteq [n]$*, chosen by the algorithm, that has $D(S) > 0$. The oracle returns an element $i \in S$, where the probability that element i is returned is $D_S(i) = D(i)/D(S)$, independently of all previous calls to the oracle.

Note that as described above the behavior of $\mathsf{COND}_D(S)$ is undefined if $D(S) = 0$, i.e., the set S has zero probability under D. Various definitional choices could be made to deal with this. These choice do not do not make significant difference in most situations, as most (adaptive) algorithms can always include in their next queries a sample previously obtained; while our lower bounds can be thought of as putting exponentially small probability mass of elements outside the support. For this reason, and for convenience, we shall hereafter assume, following Chakraborty et al., that the oracle returns in this case a sample uniformly distributed in S.

Finally, recall that a *property* $\mathcal{P}$ of distributions over $[n]$ is a set consisting of all distributions that have the property. The distance from D to a property $\mathcal{P}$, denoted $\mathrm{d}_{\mathrm{TV}}(D, \mathcal{P})$, is then defined as $\inf_{D'\in\mathcal{P}} \mathrm{d}_{\mathrm{TV}}(D, \mathcal{P})$. We use the standard definition of testing algorithms for properties of distributions over $[n]$, tailored for the setting of conditional access to an unknown distribution:

▶ **Definition 2.2** (Property tester)**.** Let $\mathcal{P}$ be a property of distributions over $[n]$. A *t-query* COND *testing algorithm for $\mathcal{P}$* is a randomized algorithm $\mathcal{T}$ which takes as input n, $\varepsilon \in (0,1]$, as well as access to COND_D. After making at most $t(\varepsilon, n)$ calls to the oracle, $\mathcal{T}$ either outputs ACCEPT or REJECT, such that the following holds:

[2] Recall that a non-adaptive tester is an algorithm whose queries do not depend on the answers obtained from previous ones, but only on its internal randomness. Equivalently, it is a tester that can commit "upfront" to all the queries it will make to the oracle.

- if $D \in \mathcal{P}$, $\mathcal{T}$ outputs ACCEPT with probability at least $2/3$;
- if $\mathrm{d_{TV}}(D, \mathcal{P}) \geq \varepsilon$, $\mathcal{T}$ outputs REJECT with probability at least $2/3$.

We observe that the above definitions can be straightforwardly extended to the more general setting of *pairs* of distributions, where given independent access to two oracles COND_{D_1}, COND_{D_2} the goal is to test whether (D_1, D_2) satisfies a property (now a set of pairs of distributions). This will be the case in Section 3, where we will consider equivalence testing, that is the property $\mathcal{P}_{\mathrm{eq}} = \{\ (D_1, D_2)\ :\ D_1 = D_2\ \}$.

2.2 Adaptive Core Testers

In order to deal with adaptivity in our lower bounds, we will use ideas introduced by Chakraborty et al. [11]. These ideas, for the case of *label-invariant* properties[3] allow one to narrow down the range of possible testers and focus on a restricted class of such algorithms called *adaptive core testers*. These core testers do not have access to the full information of the samples they draw, but instead only get to see the relations (inclusions, equalities) between the queries they make and the samples they get. Yet, Chakraborty et al. [11] show that any tester for a label-invariant property can be converted into a core tester with same query complexity; thus, it is enough to prove lower bounds against this – seemingly – weaker class of algorithms.

We here rephrase the definitions of a core tester and the view they have of the interaction with the oracle (the *configuration* of the samples), tailored to our setting.

▶ **Definition 2.3** (Atoms and partitions). Given a family $\mathcal{A} = (A_1, \dots, A_t) \subseteq [n]^t$, the *atoms* generated by $\mathcal{A}$ are the (at most) 2^t distinct sets of the form $\bigcap_{r=1}^t C_r$, where $C_r \in \{A_r, [n] \setminus A_r\}$. The family of all such atoms, denoted $\mathrm{At}(\mathcal{A})$, is the *partition* generated by $\mathcal{A}$.

This definition essentially captures "all sets (besides the A_i's) about which something can be learnt from querying the oracle on the sets of $\mathcal{A}$." Now, given such a sequence of queries $\mathcal{A} = (A_1, \dots, A_t)$ and pairs of samples $\mathbf{s} = ((s_1^{(1)}, s_1^{(2)}), \dots, (s_t^{(1)}, s_t^{(2)})) \in A_1^2 \times \cdots \times A_t^2$, we would like to summarize "all the label-invariant information available to an algorithm that obtains $((s_1^{(1)}, s_1^{(2)}), \dots, (s_t^{(1)}, s_t^{(2)}))$ upon querying $A_1, \dots, A_t$ for D_1 and D_2." This calls for the following definition:

▶ **Definition 2.4** (t-configuration). Given $\mathcal{A} = (A_1, \dots, A_t)$ and $\mathbf{s} = ((s_j^{(1)}, s_j^{(2)}))_{1 \leq j \leq t}$ as above, the *t-configuration of* $\mathbf{s}$ consists of the $6t^2$ bits indicating, for all $1 \leq i, j \leq t$, whether

- $s_i^{(k)} = s_j^{(\ell)}$, for $k, \ell \in \{1, 2\}$; and (relations between samples)
- $s_i^{(k)} \in A_j$, for $k \in \{1, 2\}$. (relations between samples and query sets)

In other terms, it summarizes which is the unique atom $S_i \in \mathrm{At}(\mathcal{A})$ that contains $s_i^{(k)}$, and what collisions between samples have been observed.

As aforementioned, the key idea is to argue that, without loss of generality, one can restrict one's attention to algorithms that only have access to t-configurations, and generate their queries in a specific (albeit adaptive) fashion:

▶ **Definition 2.5** (Core adaptive tester). A *core adaptive distribution tester* for pairs of distributions is an algorithm $\mathcal{T}$ that acts as follows.

[3] Recall that a property is label-invariant (or *symmetric*) if it is closed under relabeling of the elements of the support. More precisely, a property of distributions (resp. pairs of distributions) $\mathcal{P}$ is label-invariant if for any distribution $D \in \mathcal{P}$ (resp. $(D_1, D_2) \in \mathcal{P}$) and permutation σ of $[n]$, one has $D \circ \sigma \in \mathcal{P}$ (resp. $(D_1 \circ \sigma, D_2 \circ \sigma) \in \mathcal{P}$).

- In the i-th phase, based only on its own internal randomness and the configuration of the previous queries $A_1, \dots, A_{i-1}$ and samples obtained $(s_1^{(1)}, s_1^{(2)}), \dots, (s_{i-1}^{(1)}, s_{i-1}^{(2)})$ – whose labels it does not actually know, $\mathcal{T}$ provides:
  - a number k_i^A for each $A \in \mathrm{At}(A_1, \dots, A_{i-1})$, between 0 and $\left| A \setminus \{s_j^{(1)}, s_j^{(2)}\}_{1\leq j\leq i-1} \right|$ ("how many *fresh, not-already-seen* elements of each particular atom A should be included in the next query")
  - sets $K_i^{(1)}, K_i^{(2)} \subseteq \{1, \dots, i-1\}$ ("which of the samples $s_1^{(k)}, \dots, s(k)_{i-1}$ (whose label is unknown to the tester, but referred to by the index of the query it got them) will be included in the next query").
- based on these specifications, the next query A_i is drawn (but not revealed to $\mathcal{T}$) by
  - drawing uniformly at random a set Λ_i in

$$\left\{ \Lambda \subseteq [n] \setminus \{s_j^{(1)}, s_j^{(2)}\}_{1\leq j\leq i-1} \ : \ \forall A \in \mathrm{At}(A_1, \dots, A_{i-1}), \, |\Lambda \cap A| = k_i^A \right\} .$$

  That is, among all sets, containing only "fresh elements," whose intersection with each atom contains as many elements as $\mathcal{T}$ requires.
  - adding the selected previous samples to this set:

$$\Gamma_i \stackrel{\text{def}}{=} \left\{ s_j^{(1)} \ : \ j \in K_i^{(1)} \right\} \cup \left\{ s_j^{(2)} \ : \ j \in K_i^{(2)} \right\} \ ; \quad A_i \stackrel{\text{def}}{=} \Lambda_i \cup \Gamma_i \ .$$

  This results in a set A_i, not fully known to $\mathcal{T}$ besides the samples it already got and decided to query again; in which the *labels* of the fresh elements are unknown, but the *proportions* of elements belonging to each atom are known.
- samples $s_i^{(1)} \sim (D_1)_{A_i}$ and $s_i^{(2)} \sim (D_2)_{A_i}$ are drawn (but not disclosed to $\mathcal{T}$). This defines the i-configuration of $A_1, \dots, A_i$ and $(s_1^{(1)}, s_1^{(2)}), \dots, (s_i^{(1)}, s_i^{(2)})$, which is revealed to $\mathcal{T}$. Put differently, the algorithm only learns (i) to which of the A_ℓ's the new sample belongs, and (ii) if it is one of the previous samples, in which stage(s) and for which of D_1, D_2 it has already seen it.

After $t = t(\varepsilon, n)$ such stages, $\mathcal{T}$ outputs either ACCEPT or REJECT, based only on the configuration of $A_1, \dots, A_t$ and $(s_1^{(1)}, s_1^{(2)}), \dots, (s_t^{(1)}, s_t^{(2)})$ (which is all the information it ever had access to).

Note that in particular, $\mathcal{T}$ does not know the labels of samples it got, nor the actual queries it makes: it knows all about their sizes and sizes of their intersections, but not the actual "identity" of the elements they contain.

2.3 On the use of Yao's Principle in our lower bounds

We recall Yao's Principle (e.g., see Chapter 2.2 of [19]), a technique which is ubiquitous in the analysis of randomized algorithms. Consider a set S of instances of some problem: what this principle states is that the worst-case expected cost of a randomized algorithm on instances in S is lower-bounded by the expected cost of the best deterministic algorithm on an instance drawn randomly from S.

As an example, we apply it in a standard way for the proofs of Theorems 1.2 and 1.3: instead of considering a randomized algorithm working on a fixed instance, we instead analyze a *deterministic* algorithm working on a *random* instance. (We note that, importantly, the randomness in the samples returned by the COND oracle is "external" to this argument, and these samples behave identically in an application of Yao's Principle.)

On the other hand, our application for the proof of Theorem 1.1 in Section 3 is slightly different, due to our use of adaptive core testers. Once again, we focus on deterministic algorithms working on random instances, and the randomness in the samples is external and therefore unaffected by Yao's Principle. However, we stress that the randomness in the choice of the set Λ_i is also external to the argument, and therefore unaffected – similar to the randomness in the samples, the algorithm has no control here. Another way of thinking about this randomness is via another step in the distribution over instances: after an instance (which is a pair of distributions) is randomly chosen, we permute the labels on the elements of the distribution's domain uniformly at random. We note that since the property in question is label-invariant, this does not affect its value. We can then use the model as stated in Section 2.2 for ease of analysis, observing that this can be considered an application of the principle of deferred decisions (as in Chapter 3.5 of [19]).

3 A Lower Bound for Equivalence Testing

We prove our main lower bound on the sample complexity of testing equivalence between unknown distributions. We construct two priors $\mathcal{Y}$ and $\mathcal{N}$ over *pairs* of distributions (D_1, D_2) over $[n]$. $\mathcal{Y}$ is a distribution over pairs of distributions of the form (D, D), namely the case when the distributions are identical. Similarly, $\mathcal{N}$ is a distribution over (D_1, D_2) with $d_{\mathrm{TV}}(D_1, D_2) \geq \frac{1}{4}$. We then show that no algorithm $\mathcal{T}$ making $O(\sqrt{\log\log n})$ queries to $\mathsf{COND}^{D_1}, \mathsf{COND}^{D_2}$ can distinguish between a draw from $\mathcal{Y}$ and $\mathcal{N}$ with constant probability (over the choice of (D_1, D_2), the randomness in the samples it obtains, and its internal randomness). We describe the construction of $\mathcal{Y}$ and $\mathcal{N}$ in Section 3.1, and provide a detailed analysis in Section 3.2. (Due to space constraints, some of the proofs are deferred to the full version of the paper.)

3.1 Construction

We now summarize how a pair of distribution is constructed under $\mathcal{Y}$ and $\mathcal{N}$. (Each specific step will be described in more detail in the subsequent paragraphs.)

1. **Effective Support**
   a. Pick k_b from the set $\{0, 1, \ldots, \frac{1}{2}\log n\}$ at random.
   b. Let $b = 2^{k_b}$ and $m \overset{\text{def}}{=} b \cdot n^{1/4}$.
2. **Buckets**
   a. ρ and r are chosen with $\sum_{i=1}^{2r} \rho^i = n^{1/4}$.
   b. Divide $\{1, \ldots, m\}$ into intervals $B_1, \ldots, B_{2r}$ with $|B_i| = b \cdot \rho^i$.
3. **Distributions**
   a. Assign probability mass $\frac{1}{2r}$ uniformly over B_i to generate distribution D_1.
   b. **(i)** Let $\pi_1, \ldots, \pi_r$ be independent $0/1$ with $\Pr(\pi_i = 0) = \frac{1}{2}$.
   (ii) If $\pi_i = 0$, assign probability mass $\frac{1}{4r}$ and $\frac{3}{4r}$ over B_{2i-1} and B_{2i} respectively, else $\frac{3}{4r}$ and $\frac{1}{4r}$ respectively. This generates a distribution D_2.
3. **Support relabeling**
   a. Pick a permutation $\sigma \in S_n$ of the *total* support n.
   b. Relabel the symbols of D_1 and D_2 according to σ.
4. **Output:** Generate (D_1, D_1) for $\mathcal{Y}$, and (D_1, D_2) otherwise.

We now describe the various steps of the construction in greater detail.

Effective support. Both D_1 and D_2, albeit distributions on $[n]$, will have (common) *sparse* support. The support size is taken to be $m \overset{\text{def}}{=} b \cdot n^{1/4}$. Note that, from the above definition, m is chosen uniformly at random from products of $n^{1/4}$ with powers of 2, resulting in values in $[n^{1/4}, n^{3/4}]$.

In this step b will act as a random scaling factor. The objective of this random scaling is to induce uncertainty in the algorithm's knowledge of the true support size of the distributions, and to prevent it from leveraging this information to test equivalence. In fact one can verify that the class of distributions induced for a single value of b, namely all distributions have the same value of m, then one can distinguish the $\mathcal{Y}$ and $\mathcal{N}$ cases with only $O(1)$ conditional queries.

Buckets. Our construction is inspired by the lower bound of Canonne, Ron, and Servedio [9, Theorem 8] for the more restrictive PAIRCOND access model. We partition the support in $2r$ consecutive intervals (henceforth referred to as *buckets*) $B_1, \ldots, B_{2r}$, where the size of the i-th bucket is $b\rho^i$. We note that r and ρ will be chosen such that $\sum_{i=1}^{2r} b\rho^i = bn^{1/4}$, i.e., the buckets fill the effective support.

Distributions. We output a pair of distributions (D_1, D_2). Each distribution that we construct is uniform within any particular bucket B_i. In particular, the first distribution assigns the same mass $\frac{1}{2r}$ to each bucket. Therefore, points within B_i have the same probability mass $\frac{1}{(2rb\rho^i)}$. For the $\mathcal{Y}$ case, the second distribution is identical to the first. For the $\mathcal{N}$ case, we pair buckets in r consecutive *bucket-pairs* $\Pi_1, \ldots, \Pi_r$, with $\Pi_i = B_{2i-1} \cup B_{2i}$. For the second distribution D_2, we consider the same buckets as D_1, but repartition the mass $1/r$ *within* each Π_i. More precisely, in each pair, one of the buckets gets now total probability mass $\frac{1}{4r}$ while the other gets $\frac{3}{4r}$ (so that the probability of every point is either decreased by a factor $\frac{1}{2}$ or increased by $\frac{3}{2}$). The choice of which goes up and which goes down is done uniformly and independently at random for each bucket-pair determined by the random choices of π_i's.

Random relabeling. The final step of the construction randomly relabels the symbols, namely is a random injective map from $[m]$ to $[n]$. This is done to ensure that no information about the individual symbol labels can be used by the algorithm for testing. For example, without this the algorithm can consider a few symbols from the first bucket and distinguish the $\mathcal{Y}$ and $\mathcal{N}$ cases. As mentioned in Section 2.3, for ease of analysis, the randomness in the choice of the permutation is, in some sense, deferred to the randomness in the choice of Λ_i during the algorithm's execution.

Summary. A no-instance (D_1, D_2) is thus defined by the following parameters: the support size m, the vector $(\pi_1, \ldots, \pi_m) \in \{0,1\}^r$ (which only impacts D_2), and the final permutation σ of the domain. A yes-instance (D_1, D_1) follows an identical process, however, π has no influence on the final outcome. See Figure 1 for an illustration of such a (D_1, D_2) when σ is the identity permutation and thus the distribution is supported over the first m natural numbers.

Values for ρ and r. By setting $r = \frac{\log n}{8 \log \rho} + O(1)$, we have as desired $\sum_{i=1}^{2r} |B_i| = m$ and there is a factor $(1 + o(1))n^{1/4}$ between the height of the first bucket B_1 and the one of the last, B_{2r}. It remains to choose the parameter ρ itself; we shall take it to be $2^{\sqrt{\log n}}$, resulting in $r = \frac{1}{8}\sqrt{\log n} + O(1)$. (Note that for the sake of the exposition, we ignore technical details such as the rounding of parameters, e.g. bucket sizes; these can be easily taken care of at the price of cumbersome case analyses, and do not bring much to the argument.)

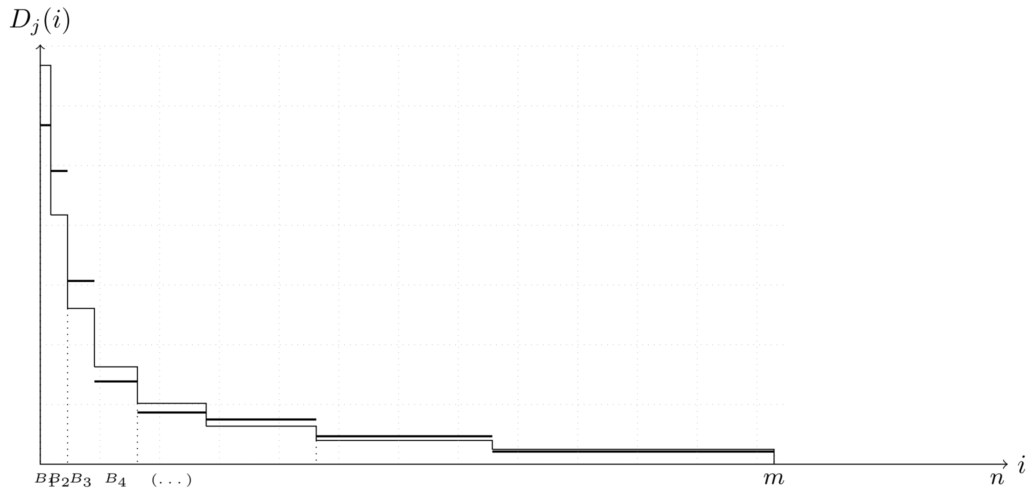


Figure 1 A no-instance (D_1, D_2) (before permutation).

3.2 Analysis

We now prove our main lower bound, by analyzing the behavior of core adaptive testers (as per Definition 2.5) on the families $\mathcal{Y}$ and $\mathcal{N}$ from the previous section. In Section 3.2.1, we argue that, with high probability, the sizes of the queries performed by the algorithm satisfy some specific properties. Conditioned upon this event, in Section 3.2.2, we show that the algorithm will get similar information from each query, whether it is running on a yes-instance or a no-instance.

Before moving to the heart of the argument, we state the following fact, straightforward from the construction of our no-instances:

▶ **Fact 3.1.** *For any (D_1, D_2) drawn from $\mathcal{N}$, one has $\mathrm{d_{TV}}(D_1, D_2) = 1/4$.*

Moreover, as allowing more queries can only increase the probability of success, we hereafter focus on a core adaptive tester that performs exactly $q = \frac{1}{10}\sqrt{\log\log n}$ (adaptive) queries; and will show that it can only distinguish between yes- and no-instances with probability $o(1)$.

3.2.1 Banning "bad queries"

As mentioned in Section 3.1, the draw of a yes- or no-instance involves a random scaling of the size of the support of the distributions, meant to "blind" the testing algorithm. Recall that a testing algorithm is specified by a decision tree, which at step i, specifies how many unseen elements from each atom to include in the query ($\{k_i^A\}$) and which previously seen elements to include in the query (sets $K_i^{(1)}, K_i^{(2)}$, as defined in Section 2.2), where the algorithm's choice depends on the observed configuration at that time. Note that, using Yao's Principle (as discussed in Section 2.3), these choices are deterministic for a given configuration – in particular, we can think of all $\{k_i^A\}$ and $K_i^{(1)}, K_i^{(2)}$ in the decision tree as being fixed. In this section, we show that all k_i^A values satisfy with high probability some particular conditions with respect to the choice of distribution, where the randomness is over the choice of the support size. Due to space constraints, all proofs from this section are deferred to the full version.

First, we recall an observation from [11], though we modify it slightly to apply to configurations on pairs of distributions and we apply a slightly tighter analysis. This essentially limits the number of states an algorithm could be in by a function of how many queries it makes.

▶ **Proposition 3.2.** *The number of nodes in a decision tree corresponding to a q-sample algorithm is at most* 2^{6q^2+1}.

For the sake of the argument, we will introduce a few notions applying to the *sizes* of query sets: namely, the notions of a number being *small*, *large*, or *stable*, and of a vector being *incomparable*. Roughly speaking, a number is small if a uniformly random set of this size does not, in expectation, hit the largest bucket B_{2r}. On the other hand, it is large if we expect such a set to intersect many bucket-pairs (i.e., a significant fraction of the support). The definition of stable numbers is slightly more quantitative: a number β is stable if a random set of size β, for each bucket B_i, either completely misses B_i or intersects it in a number of points very close to the expected number (in this case, we say the set *concentrates* over B_i). Finally, a vector of values (β_j) is incomparable if the union of random sets $S_1, \ldots, S_m$ of sizes $\beta_1, \ldots, \beta_m$ contains (with high probability) an amount of mass $D\left(\bigcup_j S_j\right)$ which is either much smaller or much larger than the probability $D(s)$ of any single element s.
We formalize these concepts in the definitions below. To motivate them, it will be useful to bear in mind that, from the construction described in Section 3.1, the expected intersection of a uniform random set of size β with a bucket B_i is of size $\beta b\rho^i/n$; while the expected probability mass from B_i it contains (under either D_1 or D_2) is $\beta/(2rn)$.

▶ **Definition 3.3.** Let q be an integer, and let $\varphi = \Theta(q^{5/2})$. A number β is said to be *small* if $\beta < \frac{n}{b\rho^{2r}}$; it is *large* (with relation to some integer q) if $\beta \geq \frac{n}{b\rho^{2r-2\varphi}}$.

Note that the latter condition equivalently means that, in expectation, a set of large size will intersect at least $\varphi + 1$ bucket-pairs (as it hits an expected $2\varphi + 1$ buckets, since $\beta |B_{2r-2\varphi}| / n \geq 1$). From the above definitions we get that, with high probability, a random set of any fixed size will in expectation either hit many or no buckets:

▶ **Proposition 3.4.** *A number is either small or large with probability* $1 - O\left(\frac{\varphi \log \rho}{\log n}\right)$.

The next definition characterizes the sizes of query sets for which the expected intersection with any bucket is either close to 0 (less than $1/\alpha$, for some threshold α), or very big (more than α). (It will be helpful to keep in mind that we will eventually use this definition with $\alpha = \mathrm{poly}(q)$.)

▶ **Definition 3.5.** A number β is said to be *α-stable* (for $\alpha \geq 1$) if, for each $j \in [2r]$, $\beta \notin \left[\frac{n}{\alpha b\rho^j}, \frac{\alpha n}{b\rho^j}\right]$. A vector of numbers is said to be α-stable if all numbers it contains are α-stable.

▶ **Proposition 3.6.** *A number is α-stable with probability* $1 - O\left(\frac{r \log \alpha}{\log n}\right)$.

The following definition characterizes the sizes of query sets which have a probability mass far from the probability mass of any individual element. (For the sake of building intuition, the reader may replace ν in the following by the parameter b of the distribution.)

▶ **Definition 3.7.** A vector of numbers $(\beta_1, \ldots, \beta_\ell)$ is said to be *(α, τ)-incomparable with respect to ν* (for $\tau \geq 1$) if the two following conditions hold.
- $(\beta_1, \ldots, \beta_\ell)$ is α-stable.
- Let Δ_j be the minimum $\Delta \in \{0, \ldots, 2r\}$ such that $\frac{\beta_j \nu \rho^{2r-\Delta}}{n} \leq \frac{1}{\alpha}$, or $2r$ if no such Δ exists. For all $i \in [2r]$, $\frac{1}{2rn} \sum_{j=1}^{\ell} \beta_j \Delta_j \notin \left[\frac{1}{\tau 2r\nu\rho^i}, \frac{\tau}{2r\nu\rho^i}\right]$.

Recall from the definition of α-stability of a number that a random set of this size either has essentially no intersection with a bucket or "concentrates over it" (i.e., with high probability, the probability mass contained in the intersection with this bucket is very close to the expected value). The above definition roughly captures the following. For any j, Δ_j is the number of buckets that will concentrate over a random set of size β_j. The last condition asks that the total probability mass from D_1 (or D_2) enclosed in the union of m random sets of size $\beta_1, \dots, \beta_\ell$ be a multiplicative factor of τ from the individual probability weight $\frac{1}{2rb\rho^i}$ of a single element from any of the $2r$ buckets.

▶ **Proposition 3.8.** *Given that a vector of numbers of length ℓ is α-stable, it is (α, q^2)-incomparable with respect to b with probability at least $1 - O\left(\frac{r \log q}{\log n}\right)$.*

We put these together to obtain the following lemma:

▶ **Lemma 3.9.** *With probability at least $1 - O\left(\frac{2^{6q^2+q}(r\log\alpha + \varphi\log\rho) + 2^{6q^2}(r\log q)}{\log n}\right)$, the following holds for the decision tree corresponding to a q-query algorithm:*

- *the size of each atom is α-stable and either large or small;*
- *the size of each atom, after excluding elements we have previously observed,[4] is α-stable and either large or small;*
- *for each i, the vector $(k_i^A)_{A \in \mathrm{At}(A_1,\dots,A_i)}$ is (α, q^2)-incomparable (with respect to b).*

Proof. From Proposition 3.2, there are at most 2^{6q^2+1} tree nodes, each of which contains one vector $(k_i^A)_A$, and at most 2^q atom sizes. The first point follows from Propositions 3.4 and 3.6 and applying the union bound over all $2^{6q^2+1} \cdot 2 \cdot 2^q$ sizes, where we note the additional factor of 2 comes from either including or excluding the old elements. The latter point follows from Proposition 3.8 and applying the union bound over all 2^{6q^2+1} (k_i^A) vectors. ◀

3.2.2 Key lemma: bounding the variation distance between decision trees

In this section, we prove a key lemma on the variation distance between the distribution on leaves of any decision tree, when given access to either an instance from $\mathcal{Y}$ or $\mathcal{N}$. This lemma will in turn directly yield Theorem 1.1. Hereafter, we set the parameters α (the threshold for stability), φ (the parameter for smallness and largeness) and γ (an accuracy parameter for how well things concentrate over their expected value) as follows:[5] $\alpha \overset{\text{def}}{=} q^7$, $\varphi \overset{\text{def}}{=} q^{5/2}$ and $\gamma \overset{\text{def}}{=} 1/\varphi = q^{-5/2}$. (Recall further that $q = \frac{1}{10}\sqrt{\log\log n}$.)

▶ **Lemma 3.10.** *Conditioned on the events of Lemma 3.9, consider the distribution over leaves of any decision tree corresponding to a q-query adaptive algorithm when the algorithm is given a* yes*-instance, and when it is given a* no*-instance. These two distributions have total variation distance $o(1)$.*

[4] More precisely, we mean to say that for each $i \leq q$, for every atom A defined by the partition of $(A_1, \dots, A_i)$, the values k_i^A and $|A \setminus \{s_1^{(1)}, s_1^{(2)}, \dots, s_{i-1}^{(1)}, s_{i-1}^{(2)}\}| - k_i^A$ are α-stable and either large or small;

[5] This choice of parameters is not completely arbitrary: combined with the setting of q, r and ρ, they ensure a total bound $o(1)$ on variation distance and probability of "bad events" as well as a (relative) simplicity and symmetry in the relevant quantities.

Proof. This proof is by induction. We will have three inductive hypotheses, $\boldsymbol{E_1}(t), \boldsymbol{E_2}(t)$, and $\boldsymbol{E_3}(t)$. Assuming all three hold for all $t < i$, we prove $\boldsymbol{E_1}(i)$. Additionally assuming $\boldsymbol{E_1}(i)$, we prove $\boldsymbol{E_2}(i)$ and $\boldsymbol{E_3}(i)$.

Roughly, the first inductive hypothesis states that the query sets behave similarly to as if we had picked a random set of that size. It also implies that whether or not we get an element we have seen before is "obvious" based on past observances and the size of the query we perform. The second states that we never observe two distinct elements from the same bucket-pair. The third states that the next sample is distributed similarly in either a yes-instance or a no-instance. Note that this distribution includes both features which our algorithm can observe (i.e., the atom which the sample belongs to and if it collides with a previously seen sample), as well as those which it can not (i.e., which bucket-pair the observed sample belongs to). It is necessary to show the latter, since the bucket-pair a sample belongs to may determine the outcome of future queries.

More precisely, the three inductive hypotheses are as follows:

- $\boldsymbol{E_1}(i)$: In either a yes-instance or a no-instance, the following occurs: For an atom S in the partition generated by $A_1, \ldots, A_i$, let $S' = S \setminus \{s_1^{(1)}, s_1^{(2)}, \ldots, s_{i-1}^{(1)}, s_{i-1}^{(2)}\}$. For every such S', let $\ell^{S'}$ be the largest index $\ell \in \{0, \ldots, 2r\}$ such that $\frac{|S'|b\rho^\ell}{n} \le \frac{1}{\alpha}$, or 0 if no such ℓ exists. We claim that $\ell^{S'} \in \{0, \ldots, 2r - \varphi - 2\} \cup \{2r\}$, and say S' is small if $\ell^{S'} = 2r$ and large otherwise. Additionally:
  - for $j \le \ell^{S'}$, $|S' \cap B_j| = 0$;
  - for $j > \ell^{S'}$, $|S' \cap B_j|$ lies in $[1 - i\gamma, 1 + i\gamma] \frac{|S'|b\rho^j}{n}$.

  Furthermore, let p_1 and p_2 be the probability mass contained in Λ_i and Γ_i, respectively. Then $\frac{p_1}{p_1+p_2} \le O\!\left(\frac{1}{q^2}\right)$ or $\frac{p_2}{p_1+p_2} \le O\!\left(\frac{1}{q^2}\right)$ (that is, either almost all the probability mass comes from elements which we have not yet observed, or almost all of it comes from previously seen ones).
- $\boldsymbol{E_2}(i)$: No two elements from the set $\{s_1^{(1)}, s_1^{(2)}, \ldots, s_i^{(1)}, s_i^{(2)}\}$ belong to the same bucket-pair.
- $\boldsymbol{E_3}(i)$: Let T_i^{yes} be the random variable representing the atoms and bucket-pairs[6] containing $(s_i^{(1)}, s_i^{(2)})$, as well as which of the previous samples they intersect with, when the i-th query is performed on a yes-instance, and define T_i^{no} similarly for no-instances. Then $\mathrm{d_{TV}}(T_i^{\mathsf{yes}}, T_i^{\mathsf{no}}) \le O\!\left(\frac{1}{q^2} + \frac{1}{\rho} + \gamma + \frac{1}{\varphi}\right) = o(1)$.

We state the lemmata, whose proofs are deferred to the full version of this paper:

▶ **Lemma 3.11.** *Assuming that $\boldsymbol{E_1}(t), \boldsymbol{E_2}(t), \boldsymbol{E_3}(t)$ hold for all $1 \le t \le i-1$, then $\boldsymbol{E_1}(i)$ holds with probability at least $1 - O\!\left(2^i \exp\left(-\frac{2\gamma^2\alpha}{3}\right)\right) = 1 - 2^{i - \Omega(q^2)}$.*

▶ **Lemma 3.12.** *Assuming that $\boldsymbol{E_1}(t), \boldsymbol{E_2}(t), \boldsymbol{E_3}(t)$ hold for all $1 \le t \le i-1$ and additionally $\boldsymbol{E_1}(i)$, then $\boldsymbol{E_2}(i)$ holds with probability at least $1 - O\!\left(\frac{i}{\varphi}\right)$.*

▶ **Lemma 3.13.** *Assuming that $\boldsymbol{E_1}(t), \boldsymbol{E_2}(t), \boldsymbol{E_3}(t)$ hold for all $1 \le t \le i-1$ and additionally $\boldsymbol{E_1}(i)$, then $\boldsymbol{E_3}(i)$ holds.*

Let T^{yes} be the random variable representing the q-configuration and the bucket-pairs containing each of the observed samples in a yes-instance, and define T^{no} similarly for a no-instance. We note that this random variable determines which leaf of the decision tree

[6] If a sample $s_i^{(k)}$ does not belong to any bucket (if the corresponding i-th query did not intersect the support), it is marked in T_i^{yes} with a "dummy label" to indicate so.

we reach. By a union bound over all q queries of the algorithm, a coupling argument, and the triangle inequality, the above lemmata imply that the total variation distance between T^{yes} and T^{no} will be $O\Big(2^q \exp\left(-\frac{2\gamma^2\alpha}{3}\right) + \frac{q^2}{\varphi} + \frac{1}{q} + \frac{q}{\rho} + q\gamma + \frac{q}{\varphi}\Big) = o(1)$ (from our choice of α, γ, φ), concluding the proof of Lemma 3.10. ◀

With this lemma in hand, the proof of the main theorem is straightforward:

Proof of Theorem 1.1. Conditioned on Lemma 3.9, Lemma 3.10 implies that the distribution over the leaves in a yes-instance vs. a no-instance is $o(1)$. Since an algorithm's choice to accept or reject depends deterministically on which leaf is reached, this bounds the difference between the conditional probability of reaching a leaf which accepts. Since Lemma 3.9 occurs with probability $1 - o(1)$, the difference between the unconditional probabilities is also $o(1)$. ◀

Acknowledgments. Clément Canonne would like to thank Dana Ron and Rocco Servedio for the many helpful discussions and remarks that influenced the lower bound construction of Section 3.

References

1 Jayadev Acharya, Clément L. Canonne, and Gautam Kamath. A chasm between identity and equivalence testing with conditional queries. *ArXiV*, abs/1411.7346, April 2015.
2 Jayadev Acharya, Hirakendu Das, Ashkan Jafarpour, Alon Orlitsky, and Shengjun Pan. Competitive closeness testing. In *Proceedings of 24th COLT*, pages 47–68, 2011.
3 Jayadev Acharya, Hirakendu Das, Ashkan Jafarpour, Alon Orlitsky, Shengjun Pan, and Ananda Theertha Suresh. Competitive classification and closeness testing. In *Proceedings of 25th COLT*, pages 1–18, 2012.
4 Tuğkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *Proceedings of FOCS*, pages 442–451, 2001.
5 Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing closeness of discrete distributions. *Journal of the ACM*, 60(1):1–25, 2013.
6 Tuğkan Batu, Ravi Kumar, and Ronitt Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *Proceedings of STOC*, pages 381–390, New York, NY, USA, 2004. ACM.
7 Arnab Bhattacharyya, Eldar Fischer, Ronitt Rubinfeld, and Paul Valiant. Testing monotonicity of distributions over general partial orders. In *Proceedings of ITCS*, pages 239–252, 2011.
8 Clément L. Canonne. A Survey on Distribution Testing: your data is Big, but is it Blue? *Electronic Colloquium on Computational Complexity (ECCC)*, TR15-063, April 2015.
9 Clément L. Canonne, Dana Ron, and Rocco A. Servedio. Testing probability distributions using conditional samples. *SIAM Journal on Computing*, 44(3), 2015.
10 Clément L. Canonne and Ronitt Rubinfeld. Testing probability distributions underlying aggregated data. In *Proceedings of ICALP*, pages 283–295, 2014.
11 Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. In *Proceedings of ITCS*, pages 561–580, New York, NY, USA, 2013. ACM.
12 Siu-On Chan, Ilias Diakonikolas, Gregory Valiant, and Paul Valiant. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of SODA*, pages 1193–1203. Society for Industrial and Applied Mathematics (SIAM), 2014.

13 Moein Falahatgar, Ashkan Jafarpour, Alon Orlitsky, Venkatadheeraj Pichapathi, and Ananda Theertha Suresh. Faster algorithms for testing under conditional sampling. In *Proceedings of 28th COLT*, 2015.

14 Eldar Fischer. List of Open Problems in Sublinear Algorithms: Problem 66. `http://sublinear.info/66`. Suggested by Fischer at Bertinoro Workshop on Sublinear Algorithms 2014.

15 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, TR00-020, March 2000.

16 Sudipto Guha, Andrew McGregor, and Suresh Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. In *Proceedings of SODA*, pages 733–742. Society for Industrial and Applied Mathematics (SIAM), 2006.

17 Piotr Indyk, Reut Levi, and Ronitt Rubinfeld. Approximating and Testing k-Histogram Distributions in Sub-linear Time. In *Proceedings of PODS*, pages 15–22, 2012.

18 Reut Levi, Dana Ron, and Ronitt Rubinfeld. Testing properties of collections of distributions. *Theory of Computing*, 9(8):295–347, 2013.

19 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1995.

20 Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008.

21 Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam Smith. Strong lower bounds for approximating distributions support size and the distinct elements problem. *SIAM Journal on Computing*, 39(3):813–842, 2009.

22 Dana Ron and Gilad Tsur. The power of an example: Hidden set size approximation using group queries and conditional sampling. *ArXiV*, abs/1404.5568, 2014.

23 Ronitt Rubinfeld. Taming Big Probability Distributions. *XRDS*, 19(1):24–28, September 2012.

24 Ronitt Rubinfeld and Rocco A. Servedio. Testing monotone high-dimensional distributions. *Random Structures and Algorithms*, 34(1):24–44, January 2009.

25 L. Stockmeyer. On approximation algorithms for #P. *SIAM Journal on Computing*, 14(4):849–861, 1985.

26 Gregory Valiant and Paul Valiant. A CLT and tight lower bounds for estimating entropy. *Electronic Colloquium on Computational Complexity (ECCC)*, TR10-179, 2010.

27 Gregory Valiant and Paul Valiant. Estimating the unseen: A sublinear-sample canonical estimator of distributions. *Electronic Colloquium on Computational Complexity (ECCC)*, TR10-180, 2010.

28 Gregory Valiant and Paul Valiant. The power of linear estimators. In *Proceedings of FOCS*, pages 403–412, October 2011. See also [26] and [27].

29 Gregory Valiant and Paul Valiant. An automatic inequality prover and instance optimal identity testing. In *Proceedings of FOCS*, 2014.

30 Paul Valiant. Testing symmetric properties of distributions. *SIAM Journal on Computing*, 40(6):1927–1968, 2011.

Harnessing the Bethe Free Energy[*].

Victor Bapst and Amin Coja-Oghlan

Mathematics Institute, Goethe University
10 Robert Mayer St, Frankfurt 60325, Germany
`{bapst, acoghlan}@math.uni-frankfurt.de`

Abstract

Gibbs measures induced by random factor graphs play a prominent role in computer science, combinatorics and physics. A key problem is to calculate the typical value of the partition function. According to the "replica symmetric cavity method", a heuristic that rests on non-rigorous considerations from statistical mechanics, in many cases this problem can be tackled by way of maximising a functional called the "Bethe free energy". In this paper we prove that the Bethe free energy upper-bounds the partition function in a broad class of models. Additionally, we provide a sufficient condition for this upper bound to be tight.

1998 ACM Subject Classification G.2.1 Combinatorics, G.3 Probability and Statistics, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Belief Propagation, free energy, Gibbs measure, partition function

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.467

1 Introduction

Many problems in combinatorics, computer science and physics can be cast along the following lines [2, 23]. There are a (large) number of *variables*, each of them ranging over a finite domain Ω. The variables interact through *constraints* that each bind a few variables. Every constraint comes with a "weight function" that either encourages or discourages certain value combinations of the incident variables. The interactions can be described naturally by a *factor graph*, whose vertices are the variables and the constraints. A constraint is adjacent to the variables that it binds. The weight of an *assignment* σ that maps each variable to a value from Ω is the product of all the weights of the constraints. The obvious questions is: how many assignments of a specific total weight exist?

In this paper we are concerned with models where the factor graph is random. An excellent example is the *random k-SAT model:* there are n Boolean variables $x_1, \ldots, x_n$ and m clauses $a_1, \ldots, a_m$. Each clause binds k variables, which are chosen independently and uniformly from $x_1, \ldots, x_n$, and discourages them from taking one of the 2^k possible value combinations. This value combination is chosen uniformly and independently for each clause. The key quantity associated with the random k-SAT instance $\mathbf{\Phi}$ is its *partition function*, defined as

$$Z_{\beta,\mathbf{\Phi}} = \sum_{\sigma\in\{0,1\}^n} \prod_{i=1}^{m} \exp(-\beta \mathbf{1}\{\sigma \text{ violates } a_i\}) \qquad (\beta > 0). \tag{1}$$

[*] The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 278857–PTCC

In words, we sum the weights of all 2^n possible truth assignments σ. Each σ incurs a "penalty factor" of $\exp(-\beta)$ for every violated clause. It is not difficult to see that the random variable $Z_{\beta,\mathbf{\Phi}}$ incorporates key characteristics of the model. For instance, the maximum number of clauses that can be satisfied simultaneously equals

$$m + \lim_{\beta \to \infty} \frac{\partial}{\partial \beta} \ln Z_{\beta,\mathbf{\Phi}}.$$

Apart from random k-SAT, there are a host of other models of a similar nature. Prominent examples include the random graph colouring problem, LDPC codes or the so-called "mean-field" models of statistical mechanics [23].

Over the past decade the *second moment method* has emerged as the principal tool for the analysis of such models [1, 2, 15]. Its "vanilla" version works as follows. If the partition function Z of the model satisfies the bound $\mathbb{E}[Z^2] \leq O(\mathbb{E}[Z]^2)$ in the limit as the number n of variables tends to infinity, then $n^{-1}\ln(Z/\mathbb{E}[Z])$ converges to 0 in probability. Since $\mathbb{E}[Z]$ is normally easy to compute, we thus obtain the exponential order of Z. In fact, by calculating $\mathbb{E}[Z^2]/\mathbb{E}[Z]^2$ accurately enough it is sometimes possible to infer the limiting distribution of Z [20].

However, in many examples the use of the second moment method is precluded by large deviations phenomena. The random k-SAT model with $m = \lceil \alpha n \rceil$ clauses is a case in point as $n^{-1}\ln(Z_{\beta,\mathbf{\Phi}}/\mathbb{E}[Z_{\beta,\mathbf{\Phi}}])$ does *not* converge to 0 as $n \to \infty$ for any $\alpha, \beta > 0$. The reason is that the first moment $\mathbb{E}[Z_{\beta,\mathbf{\Phi}}]$ is driven up by a "lottery effect": there are a tiny minority of formulas with an abundance of "good" assignments [1, 3, 4]. Of course, this implies that $\mathbb{E}[Z_{\beta,\mathbf{\Phi}}^2] \geq \exp(\Omega(n))\mathbb{E}[Z_{\beta,\mathbf{\Phi}}]^2$. Thus, the second moment method fails rather spectacularly.

The obvious remedy is to condition such lottery effects away. That is, we ought to condition on an event $\mathcal{U}$ that pins down those parameters of the model whose large deviations drive $\mathbb{E}[Z]$ up. But even if we manage to identify the relevant parameters, the necessary conditioning on $\mathcal{U}$ may be so complicated as to render a second moment computation at best unpleasant and at worst infeasible. Indeed, the recent history of the random k-SAT problem illustrates how conditioning turns a second moment computation into a formidable task [7, 12].

A completely different but non-rigorous method for calculating Z, the *replica symmetric cavity method*, has been suggested on the basis of ideas from statistical physics [23]. According to the cavity method, under certain assumptions the asymptotic value of $n^{-1}\ln Z$ can be calculated by maximising a functional called the *Bethe free energy*. Furthermore, the physics recipe for solving this maximisation problem is to iterate a message passing algorithm called *Belief Propagation* on the factor graph until convergence. This recipe is somewhat plausible due to the (rigorous) fact that the stationary points of the Bethe free energy are in one-to-one correspondence with the Belief Propagation fixed points [32]. However, in general there are several fixed points and non-trivial insights are necessary to steer Belief Propagation toward the "correct" one. Even worse, in general the maximum value of the Bethe free energy may or may not approximate $n^{-1}\ln Z$ well.[1]

The purpose of this paper is to provide a rigorous foundation for the idea of using Belief Propagation to calculate the free energy. We establish two main results. First, that under mild assumptions the maximum of the Bethe free energy provides an upper bound on the typical value of $n^{-1}\ln Z$ on a random factor graph (Theorem 3). The proof of this is based

[1] The quantity $n^{-1}\ln Z$ is called the *free energy* of the factor graph. We do not use this term to avoid confusion with the *Bethe free energy*.

on a physics-enhanced version of the classical "first moment method". Along the way we derive several general results on Gibbs distributions that should be of independent interest (e.g., Theorem 6). Second, we propose a corresponding refined "second moment method" (Theorem 14). More specifically, we prove that if the maximum of the Bethe free energy on a certain auxiliary model is upper-bounded by a term that corresponds to the square of the first moment and if certain additional (reasonable) assumptions hold, then the free energy converges in probability to the value predicted by the cavity method.

2 Related work

Belief Propagation has been re-discovered several times in varying degrees of generality [5, 17, 29]. On finite acyclic factor graphs Belief Propagation has a unique fixed point and the corresponding Bethe free energy equals $n^{-1}\ln Z$. (e.g., [23, Chapter 14]). To what extent this is true in the presence of cycles is a long-standing problem.

The results of the present paper are most relevant in cases where the local structure of the factor graph is not perfectly "uniform". For instance, we are going to be interested in the case that different variable nodes may have different degrees. More subtly, different variable nodes may have different marginals under the Gibbs distribution that the factor graph induces, see (3) below. The case of uniform models is conceptually simpler and has been treated before [9]. In fact, in the uniform case the computation of $n^{-1}\ln Z$ can essentially be transformed into the problem of maximising the Bethe free energy of a "tensorised" model on the d-regular tree [9, 13, 11]. This fact has played a key role in recent work on the hardness of counting problems [16, 26, 30]. Although we use a similar tensor construction in our second moment argument as well (cf. Proposition 13), non-uniformity makes matters far more intricate, as witnessed by recent work on random k-SAT [7, 12]. Thus, the main point of the present work is to establish a connection between the Bethe free energy and $|V|^{-1}\ln Z$ even in the non-uniform case.

That said, if the model enjoys certain spatial mixing properties (such as "Gibbs uniqueness"), then the Bethe free energy is known to yield the correct value of $n^{-1}\ln Z$ even in the non-uniform case [10, 25]. However, the necessary spatial mixing properties are quite strong and they cease to be satisfied, e.g., in the random k-SAT model from (1) for large β for clause/variable ratios as low as about $\ln k/k$ [25]. By comparison, the k-SAT threshold is about $2^k\ln 2$ [12].

The "interpolation method" provides a different approach to calculating or at least upper-bounding $n^{-1}\ln Z$ [14, 19]. In particular, the upper bound comes in a variational form [28]. For example, this can be used to obtain a tight upper bound on the k-SAT threshold [12]. Generally speaking, the interpolation method is great if it works, but it comes with certain (convexity-type) assumptions that are not always satisfied. Furthermore, it seems difficult to use the interpolation method directly to carry out a second moment argument in order to lower-bound the partition function. By contrast, Theorems 3 and 14 do not require such assumptions.

The physicists' cavity method comes in several instalments; for a detailed discussion we refer to [23]. In this paper we are chiefly concerned with the simplest, "replica symmetric" variant. This version does not always provide the correct value of $n^{-1}\ln Z$ [8]. It seems that one reason for this is that models such as random k-SAT undergo a so-called "condensation phase transition" [21]. The more complex "1-step replica symmetry breaking (1RSB)" version of the cavity method [24] is expected to yield the correct value of $n^{-1}\ln Z$ some way beyond condensation. However, another phase transition called *full replica symmetry breaking* seems to spell doom on even the 1RSB cavity method (see [23] for details). In summary, we do not hope for an unconditional result that vindicates either the replica symmetric or the 1RSB version of the cavity method.

3 Random factor graphs

In this section we explain the class of models that we deal with. Throughout, $\Delta > 0$ is an integer, Ω, Θ are finite sets and $\Psi = \{\psi_1, \dots, \psi_l\}$ is a finite set of maps $\psi_i : \Omega^{h_i} \to (0, \infty)$, where $1 \le h_i \le \Delta$. The following abstract definition encompasses a multitude of concrete examples.

▶ **Definition 1.** A $(\Delta, \Omega, \Psi, \Theta)$-*model* $\mathcal{M} = (V, F, d, t, \psi)$ consists of

M1 a countable set V of ***variable nodes***,

M2 a countable set F of ***constraint nodes***,

M3 a map $d : V \cup F \to [\Delta]$ such that $\sum_{x \in V} d(x) = \sum_{a \in F} d(a)$,

M4 a map $t : C_V \cup C_F \to \Theta$, where we let

$$C_V = \bigcup_{x \in V} \{x\} \times [d(x)], \qquad\qquad C_F = \bigcup_{a \in F} \{a\} \times [d(a)],$$

such that $\left|t^{-1}(\theta) \cap C_V\right| = \left|t^{-1}(\theta) \cap C_F\right|$ for each $\theta \in \Theta$,

M5 a map $F \to \Psi$, $a \mapsto \psi_a$ such that $\psi_a : \Omega^{d(a)} \to (0, \infty)$ for all $a \in F$.

The ***size*** of the model is defined as $\#\mathcal{M} = |V|$. Furthermore, a $\mathcal{M}$***-factor graph*** is a bijection

$$G : C_V \to C_F, \ (x, i) \mapsto G_{x,i} \qquad \text{such that} \qquad t(G_{x,i}) = t(x, i) \text{ for all } (x, i) \in C_V.$$

Of course, the equalities in **M3** and **M4** require that either both quantities are infinite or both are finite, in which case they have to coincide.

The semantics is that the map d prescribes the degree of each variable and constraint node (i.e., their number of neighbours in any $\mathcal{M}$-factor graph). Just like in the "configuration model" of graphs with a given degree sequence we create $d(v)$ "clones" of each node v. The sets C_V, C_F contain the clones of the variable and constraint nodes, respectively. Additionally, the map t assigns each clone a "type" from the set Θ. Moreover, each constraint node a comes with a "weight function" ψ_a from the set Ψ.

Like in the "configuration model" a $\mathcal{M}$-factor graph is a type-preserving matching G of the variable and constraint clones. Let $\mathcal{G}(\mathcal{M})$ be the set of all $\mathcal{M}$-factor graphs and let $\boldsymbol{G}(\mathcal{M})$ denote a uniformly random sample from $\mathcal{G}(\mathcal{M})$. We usually think of $G \in \mathcal{G}(\mathcal{M})$ as the (multi-)graph obtained by contracting the clones of each node. Clearly, this yields a bipartite graph with $|V|$ variable nodes and $|F|$ constraint nodes. For a node $x \in V$ we denote by $\partial_G x$ the set of neighbours of x in this multi-graph, i.e., the set of all $a \in F$ such that there exist $i \in [d(x)]$, $j \in [d(a)]$ such that $G_{x,i} = (a, j)$. Analogously, for $a \in F$ and $j \in [d(a)]$ we write $\partial_G(a, j) = x$ if there is $i \in [d(x)]$ such that $G_{x,i} = (a, j)$. Moreover, $\partial_G a = \{\partial_G(a, j) : j \in [d(a)]\}$. Finally, we denote the inverse image of a clone $(a, j) \in C_F$ under the bijection G simply by $G_{a,j}$.

A $\mathcal{M}$***-assignment*** is a map $\sigma : V \to \Omega$. Let $\mathcal{C}_{\mathcal{M}}$ be the set of all $\mathcal{M}$-assignments. Further, define the ***partition function*** of $G \in \mathcal{G}(\mathcal{M})$ as

$$Z_G = \sum_{\sigma \in \mathcal{C}_{\mathcal{M}}} \prod_{a \in F} \psi_a\big(\sigma(\partial_G(a, 1)), \dots, \sigma(\partial_G(a, d(a)))\big). \tag{2}$$

It is closely intertwined with the ***Gibbs distribution*** of G, which is the distribution on $\mathcal{C}_{\mathcal{M}}$ defined by

$$\mu_G(\sigma) = Z_G^{-1} \prod_{a \in F} \psi_a\big(\sigma(\partial_G(a,1)), \ldots, \sigma(\partial_G(a,d(a)))\big). \tag{3}$$

Our key object of study is the random variable $|V|^{-1} \ln Z_{\boldsymbol{G}(\mathcal{M})}$.

▶ **Example 2** (The random k-SAT model)**.** Let $\Omega = \{0,1\}$. Given some $\beta \geq 0$ let Ψ contain the 2^k weight functions

$$\psi^{(\tau)} : \Omega^k \to (0,\infty), \qquad \sigma \mapsto \exp(-\beta \mathbf{1}\{\sigma = \tau\}) \qquad \text{for } \tau \in \Omega^k.$$

Let $\Delta > 0$ be a positive integer and let $\Theta = \{\star\}$. We obtain a $(\Delta, \Omega, \Psi, \Theta)$-model $\mathcal{M}_{\mathrm{SAT}}$ by letting $V = \{x_1, \ldots, x_n\}$ and $F = \{a_1, \ldots, a_m\}$. Pick any degree sequence $d : V \to [\Delta]$ such that $\sum_{x \in V} d(x) = km$ and let $d(a) = k$ for all $a \in F$. Further, pick some $\psi_a \in \Psi$ for each $a \in F$, thereby prescribing a "sign pattern" for each "clause" a. Finally, let $t : C_V \cup C_F \to \Theta$ be the trivial (constant) map. Then $\boldsymbol{G}(\mathcal{M}_{\mathrm{SAT}})$ corresponds to choosing a random k-SAT formula with the given degree sequence and sign patterns. Moreover, $n^{-1} \ln Z_{\boldsymbol{G}(\mathcal{M}_{\mathrm{SAT}})}$ accounts for weighted truth assignments (cf. (1)) [18].

In Example 2 we did not actually use the types in a non-trivial way. They could be used to prescribe not merely the degree of each variable but also how many times each Boolean variable appears positively or negatively.

While Definition 1 encompasses a many problems of interest, there are two restrictions. They arise because we are going to be interested in *sequences* $(\mathcal{M}_n)_n$ of $(\Delta, \Omega, \Psi, \Theta)$-models of sizes $\#\mathcal{M}_n = n$. That is, the size of the model tends to infinity while $\Delta, \Omega, \Psi, \Theta$ remain fixed. In effect, the maximum degree remains bounded as $n \to \infty$. This is not quite the case in, e.g., the "standard" random k-SAT model where clauses are chosen uniformly and independently and where consequently the variable degrees are asymptotically Poisson. However, in such examples the free energy can by means of standards arguments be approximated arbitrarily well by truncating the degrees at a large enough Δ.

The second restriction is that the weight functions $\psi \in \Psi$ are assumed to be strictly positive. This condition precludes *hard* constraints such as "no single clause must be violated". Although most of our proofs extend to the case of hard constraints, we chose to exclude them from the general statement of the results for the sake of clarity. For instance, the positivity assumption ensures that $Z_G > 0$ for all $G \in \mathcal{G}(\mathcal{M}_n)$ and hence that the random variable $n^{-1} \ln Z_{\boldsymbol{G}(\mathcal{M}_n)}$ has a finite mean. Furthermore, the case of hard constraints can be handled by introducing an "inverse temperature" parameter $\beta > 0$ like in Example 2 and ultimately taking the limit $\beta \to \infty$ (cf. [25]), although some additional work is needed.

In Section 4 we will prove that the "Bethe free energy" provides an upper bound on $|V|^{-1} \ln Z_{\boldsymbol{G}(\mathcal{M})}$. Further, in Section 5 we are going to provide a sufficient condition under which this upper bound is asymptotically tight.

Preliminaries

Throughout the paper we always let $\Delta \geq 1$ be an integer, Ω, Θ finite sets, and Ψ a finite set of functions as in Definition 1. We let $\#\psi$ be the arity of $\psi \in \Psi$, i.e., $\psi : \Omega^{\#\psi} \to (0,\infty)$.

For a finite set $\mathcal{X} \neq \emptyset$ we denote by $\mathcal{P}(\mathcal{X})$ the set of probability measures on $\mathcal{X}$, which we identify with the $|\mathcal{X}|$-simplex. For $\mu \in \mathcal{P}(\mathcal{X})$ we denote by $H(\mu) = -\sum_{x \in \mathcal{X}} \mu(x) \ln \mu(x)$

the entropy of μ (with the convention $0 \ln 0 = 0$). Further, if $\mu, \nu : \mathcal{X} \to [0, \infty)$ are such that $\nu(x) > 0$ only if $\mu(x) > 0$, then

$$D(\nu \| \mu) = \sum_{x \in \mathcal{X}} \nu(x) \ln(\nu(x)/\mu(x))$$

signifies the Kullback-Leibler divergence. Moreover, for integers $k > 0$, $j \in [k]$ and $\mu \in \mathcal{P}(\mathcal{X}^k)$ we let $\mu_{\downarrow j} \in \mathcal{P}(\mathcal{X})$ be the marginal of the jth component.

For $\mu \in \mathcal{P}(\mathcal{X})$ we write $\boldsymbol{\sigma}_\mu$ for an random element of $\mathcal{X}$ chosen according to μ. Where μ is apparent from the context we drop the index. Further, if $X : \mathcal{X} \to \mathbb{R}$ is a random variable we write $\langle X \rangle_\mu = \sum_{\sigma \in \mathcal{X}} X(\sigma)\mu(\sigma)$ for the expectation of X with respect to μ. For the sake of brevity we normally write $\langle \cdot \rangle_G$ instead of $\langle \cdot \rangle_{\mu_G}$ for $G \in \mathcal{G}(\mathcal{M})$.

Further, if S is a subset of the set V of variable nodes of $\mathcal{M}$, $\sigma : V \to \Omega$ and $\omega \in \Omega$ we write

$$\sigma[\omega|S] = \frac{1}{|S|} \sum_{x \in S} \mathbf{1}\{\sigma(x) = \omega\}.$$

Thus, $\sigma[\cdot|S] \in \mathcal{P}(\Omega)$ is the empirical distribution of σ on S. Analogously, if $G \in \mathcal{G}(\mathcal{M})$ is a factor graph and $A \neq \emptyset$ is a set of factor nodes such that all $a \in A$ have degree $d(a) = l$ for some $l > 0$, then we let

$$\sigma[\omega_1, \ldots, \omega_l|A] = \frac{1}{|A|} \sum_{a \in A} \prod_{j=1}^{l} \mathbf{1}\{\sigma(\partial_G(a, j)) = \omega_j\}.$$

Thus, $\sigma[\cdot|A] \in \mathcal{P}(\Omega^l)$ is the joint empirical distribution of the value combinations induced by σ on $a \in A$.

4 The upper bound

Let $\mathcal{M} = (V, F, d, t, (\psi_a)_{a \in F})$ be a $(\Delta, \Omega, \Psi, \Theta)$-model of finite size $n = |V|$. Let $\boldsymbol{G} = \boldsymbol{G}(\mathcal{M})$ for brevity.

4.1 The Bethe free energy

The aim in this section is to show that the "Bethe free energy", a concept that hails from the cavity method, provides an upper bound on the partition function. To formulate the result we need the following definition [23, Chapter 14]. Let $G \in \mathcal{G}(\mathcal{M})$. A ***marginal sequence*** of G is a family $\nu = (\nu_x, \nu_a)_{x \in V, a \in F}$ such that $\nu_x \in \mathcal{P}(\Omega)$ for each $x \in V$, $\nu_a \in \mathcal{P}(\Omega^{d(a)})$ for each $a \in F$ and if $G_{x,i} = (a, j)$ entails that $\nu_x = \nu_{a \downarrow j}$. Thus, if a variable x occurs in the jth position of a constraint a, then the jth marginal of ν_a coincides with ν_x. The ***Bethe free energy***[2] of (G, ν) is

$$\mathcal{B}_{\mathcal{M}}(G, \nu) = -\frac{1}{n}\left[\sum_{a \in F} D(\nu_a \| \psi_a) + \sum_{x \in V} (d(x) - 1) H(\nu_x)\right].$$

Additionally, the ***Bethe free energy of*** G is

$$\mathcal{B}_{\mathcal{M}}(G) = \max\{\mathcal{B}_{\mathcal{M}}(G, \nu) : \nu \text{ is a marginal sequence of } G\}.$$

[2] For a detailed derivation of the Bethe free energy in the context of the cavity method see [23, Chapter 14].

▶ **Theorem 3.** *For any $\Delta, \Omega, \Psi, \Theta$ and any $\varepsilon > 0$ there exists $n_0 > 0$ such that the following is true. Suppose that $\mathcal{M}$ is a finite $(\Delta, \Omega, \Psi, \Theta)$-model of size $n > n_0$. Moreover, let $\emptyset \neq \mathcal{U} \subset \mathcal{G}(\mathcal{M})$ be an event. Then*

$$n^{-1} \ln \mathbb{E}[Z_{\boldsymbol{G}} \mathbf{1}\{\boldsymbol{G} \in \mathcal{U}\}] \leq \max\{\mathcal{B}_{\mathcal{M}}(G) : G \in \mathcal{U}\} + \varepsilon.$$

Thus, there exists a number n_0 that depends *only* on the basic parameters $\Delta, \Omega, \Psi, \Theta$ and the desired accuracy ε such that for any model of size $n \geq n_0$ the Bethe free upper bounds on the expectation of $Z_{\boldsymbol{G}}$ on $\mathcal{U}$. The following corollary provides a handy way to apply Theorem 3.

▶ **Corollary 4.** *Let $(\mathcal{M}_n)_n$ be a sequence of $(\Delta, \Omega, \Psi, \Theta)$-models such that $\#\mathcal{M}_n = n$. Assume that $b > 0$ is such that the event $\mathcal{U}_n = \{\mathcal{B}_{\mathcal{M}_n}(\boldsymbol{G}(\mathcal{M}_n)) \leq b\}$ satisfies $\lim_{n\to\infty} \mathbb{P}[\mathcal{U}_n] = 1$. Then*

$$\limsup_{n\to\infty} n^{-1} \ln \mathbb{E}\left[Z_{\boldsymbol{G}(\mathcal{M}_n)} | \mathcal{U}_n\right] \leq b.$$

By Markov's and Jensen's inequalities, the bound $\limsup_{n\to\infty} n^{-1} \ln \mathbb{E}\left[Z_{\boldsymbol{G}(\mathcal{M}_n)} | \mathcal{U}_n\right] \leq b$ entails that

$$\lim_{\varepsilon \searrow 0} \lim_{n\to\infty} \mathbb{P}\left[n^{-1} \ln Z_{\boldsymbol{G}(\mathcal{M}_n)} \leq b + \varepsilon\right] = 1.$$

In other words, if the Bethe free energy is bounded by b with high probability, then $n^{-1} \ln Z_{\boldsymbol{G}(\mathcal{M}_n)} \leq b + o(1)$ with high probability.

The proof of Theorem 3 contains several concepts that we deem to be of independent interest. The most important one is that of a "state". More specifically, we prove Theorem 3 by showing that the lion's share of $\mathbb{E}[Z_{\boldsymbol{G}} \mathbf{1}\{\boldsymbol{G} \in \mathcal{U}\}]$ comes from a set Γ of factor graph/assignment pairs (G, σ) such that certain key parameters of all pairs $(G, \sigma) \in \Gamma$ approximately coincide. For instance, for (almost) any ψ and value combination $\omega = (\omega_1, \ldots, \omega_{\#\psi})$, about the same number of constraint nodes a with $\psi_a = \psi$ display the value combination ω. Theorem 3 will follow because the contribution of any single state s to $\mathbb{E}[Z_{\boldsymbol{G}} \mathbf{1}\{\boldsymbol{G} \in \mathcal{U}\}]$ can be cast as the Bethe free energy of a marginal sequence induced by s. We proceed with the precise definition of states.

4.2 States

For an integer $N \geq 1$ we write

$$\Psi[N] = \{(\psi, h_1, \ldots, h_{\#\psi}) : h_1, \ldots, h_{\#\psi} \in [N]\}.$$

▶ **Definition 5.** A $\mathcal{M}$-***state*** of size $N \geq 1$ consists of

ST1 a map $s : V \to [N]$ such that $s(x) = s(y)$ only if $d(x) = d(y)$ and $t(x,i) = t(y,i)$ for all $i \in d(x)$,

ST2 a probability distribution $\bar{s} = (\bar{s}_{\psi,h})_{\psi,h}$ on $\Psi[N]$,

ST3 a set $\hat{s} \subset \Psi[N]$,

ST4 a sequence $(\tilde{s}_h)_{h\in[N]}$ of probability distributions on Ω,

ST5 for any $(\psi, h) \in \Psi[N]$ a distribution $\tilde{s}_{\psi,h} \in \mathcal{P}(\Omega^{\#\psi})$ such that $\tilde{s}_{\psi,h\downarrow j} = \tilde{s}_{h_j}$ for all $j \in [\#\psi]$.

Normally we denote an $\mathcal{M}$-state simply by s and we write $\#s$ for its size. Moreover, let

$$V_h^s = s^{-1}(h) \quad \text{for } h \in [\#s], \qquad V_{\psi,h}^s = \prod_{j\in[\#\psi]} V_{h_j}^s \quad \text{for } (\psi, h) \in \Psi[N].$$

In addition, if $G \in \mathcal{G}(\mathcal{M})$ and $(\psi, h) \in \Psi[N]$ we let

$$\partial_{G,s}(\psi, h) = \left\{a \in F : \psi_a = \psi, \partial_G(a) \in V^s_{\psi,h}\right\}.$$

Thus, a state induces a partition $V^s_1, \ldots, V^s_N$ of the set of variable nodes. Condition **ST1** ensures that this partition respects the degrees and the types. Let us call $G \in \mathcal{G}(\mathcal{M})$ ***ε-compatible*** with s for some $\varepsilon > 0$ ("$G \models_\varepsilon s$") if

$$\sum_{(\psi,h)\in\Psi[N]} \left|\frac{|\partial_{G,s}(\psi, h)|}{|F|} - \bar{s}_{\psi,h}\right| < \varepsilon, \qquad \sum_{(\psi,h)\in\Psi[N]} \mathbf{1}\left\{(\psi, h) \in \hat{s}\right\} \bar{s}_{\psi,h} < \varepsilon.$$

Thus, for any (ψ, h) there are about $\bar{s}_{\psi,h}|F|$ constraint nodes a with $\psi_a = \psi$ that join variable nodes from the classes $V^s_{h_1}, \ldots, V^s_{h_{\#\psi}}$. And no more than an ε fraction of all constraint nodes belong to the "rogue classes" $(\psi, h) \in \hat{s}$.

Further, suppose that $G \models_\varepsilon s$ and $\sigma \in \mathcal{C}_\mathcal{M}$. We say that (G, σ) is ***ε-judicious*** with respect to s (in symbols: $(G, \sigma) \models_\varepsilon s$) if
J1 for all $h \in [N]$ we have $\|\tilde{s}_h - \sigma[\,\cdot\,|V^s_h]\|_{\mathrm{TV}} < \varepsilon$,
J2 for all $(\psi, h) \in \Psi[N] \setminus \hat{s}$ such that $\partial_{G,s}(\psi, h) \neq \emptyset$ we have $\|\tilde{s}_{\psi,h} - \sigma[\,\cdot\,|\partial_{G,s}(\psi, h)]\|_{\mathrm{TV}} < \varepsilon$.
Hence, the empirical distributions $\sigma[\,\cdot\,|V^s_h]$ do not deviate by more than ε from $\tilde{s}_h$. Similarly, for a "non-rogue" (ψ, h) the empirical distribution $\sigma[\,\cdot\,|\partial_{G,s}(\psi, h)]$ of the ψ-factors that connect variables in $V^s_{\psi,h}$ is within ε of $\tilde{s}_{\psi,h}$. The following theorem provides the key fact about states. It should be of interest in its own right.

▶ **Theorem 6.** *For any $\Delta, \Omega, \Psi, \Theta$ and any $\varepsilon > 0$ there exists $\eta > 0$ such that the following is true. Let $\mathcal{M}$ be a finite $(\Delta, \Omega, \Psi, \Theta)$-model of size $\#\mathcal{M} \geq 1/\eta$ and let $G \in \mathcal{G}(\mathcal{M})$. Then there exists a $\mathcal{M}$-state s of size $\#s \leq 1/\eta$ such that $G \models_\varepsilon s$ and $\langle \mathbf{1}\{(G, \boldsymbol{\sigma}) \models_\varepsilon s\}\rangle_G \geq \eta$.*

Crucially, the number η promised by Theorem 6 depends on ε and the basic parameters $\Delta, \Omega, \Psi, \Theta$ only. It is independent of the model and its size. Hence, for *any* large enough $\mathcal{M}$ and *any* $G \in \mathcal{G}(\mathcal{M})$ there is a single "dominant state" s that captures a constant fraction of the mass of the Gibbs distribution μ_G.

Theorem 6 sits well with the *replica symmetry breaking* picture drafted by the cavity method. According to this prediction, there are three possible shapes that the Gibbs distribution can take. Roughly speaking, in the case of *replica symmetry* the joint distribution of any two variable nodes that are far apart (say, at distance at least $\ln \ln n$) in the factor graph is close to a product distribution. The state corresponding to this scenario simply partitions the variable nodes according to their Gibbs marginals. In the second scenario, called *1-step replica symmetry breaking*, the Gibbs distribution is mixture of a bounded number of distributions, i.e.,

$$\left\|\mu_G - \sum_{i=1}^{K} w_i \mu_{G,i}\right\|_{\mathrm{TV}} < \varepsilon \qquad \text{where } (w_1, \ldots, w_K) \in \mathcal{P}([K]), \quad \mu_{G,1}, \ldots, \mu_{G,K} \in \mathcal{P}(\mathcal{C}_\mathcal{M}).$$

Each $\mu_{G,i}$ corresponds to a "cluster" of assignments and is such that the joint distribution of far apart variables factorises. In this case, we obtain a state by partitioning the variables according to their $\mu_{G,i}$-marginals for some i with $w_i \geq \eta$. In the third case, called *full replica symmetry breaking*, the $\mu_{G,i}$ themselves are mixtures of distributions $\mu_{G,i,j}$. Further, each of the $\mu_{G,i,j}$ decomposes into clusters etc., yielding an infinite cascade. A dominant state would truncate the cascade after a finite number of steps (depending on ε) and home in on one of the sub-clusters.

The key concept behind the proof of Theorem 6 is the following.

▶ **Definition 7.** Let Ω be a finite set, let $\varepsilon > 0$, let n be an integer and let μ be a probability measure on Ω^n. A partition $\boldsymbol{V} = (V_1, \dots, V_N)$ of $[n]$ is called ***ε-homogeneous with respect to*** μ if there is a set $J \subset [N]$ such that $\sum_{i \in [N] \setminus J} |V_i| < \varepsilon n$ and such that for all $j \in J$ the following is true.

For any subset $S \subset V_j$ of size $|S| \ge \varepsilon |V_j|$ we have $\left\langle \|\boldsymbol{\sigma}[\,\cdot\,|S] - \boldsymbol{\sigma}[\,\cdot\,|V_j]\|_{\mathrm{TV}} \right\rangle_\mu < \varepsilon$.

If $\boldsymbol{V} = (V_1, \dots, V_k)$, then we call $\#\boldsymbol{V} = k$ the ***size*** of $\boldsymbol{V}$. Furthermore, a partition $\boldsymbol{W} = (W_1, \dots, W_l)$ ***refines*** another partition $\boldsymbol{V} = (V_1, \dots, V_k)$ if for each $i \in [l]$ there is $j \in [k]$ such that $W_i \subset V_j$. The proof of Theorem 6 builds upon

▶ **Theorem 8.** *Let Ω be a finite set. For any $\varepsilon > 0$ there exists $N = N(\varepsilon, \Omega)$ such that for $n > N$ and any probability measure μ on Ω^n the following is true. Let $\boldsymbol{V}_0$ be a partition of $[n]$ such that $\#\boldsymbol{V}_0 \le 1/\varepsilon$. Then $\boldsymbol{V}_0$ has a refinement $\boldsymbol{V}$ of size $\#\boldsymbol{V} \le N$ that is ε-homogeneous with respect to μ.*

Theorem 8 and its proof are inspired by the proof of Szemerédi's regularity lemma [31].

Theorem 6 produces a "dominant state" for each individual factor graph. In combination with a compactness argument this entails that a single state suffices to approximate $\frac{1}{n} \ln \mathbb{E}[Z_{\boldsymbol{G}} \mathbf{1}\{\boldsymbol{G} \in \mathcal{U}\}]$ for a given event $\mathcal{U}$.

▶ **Corollary 9.** *For any $\varepsilon > 0$ and any $\Delta, \Omega, \Psi, \Theta$ there exist $\gamma > 0, n_0 > 0$ such that the following is true. Suppose that $\mathcal{M}$ is a finite $(\Delta, \Omega, \Psi, \Theta)$-model of size $\#\mathcal{M} \ge n_0$ and that $\emptyset \ne \mathcal{U} \subset \mathcal{G}(\mathcal{M})$. Then there exist a $\mathcal{M}$-state s and $G_0 \in \mathcal{U}$ such that $G_0 \models_\gamma s$ and*

$$n^{-1} \ln \mathbb{E}[Z_{\boldsymbol{G}} \mathbf{1}\{\boldsymbol{G} \in \mathcal{U}\}] \le \varepsilon + n^{-1} \ln \mathbb{E}\left[Z_{\boldsymbol{G}} \left\langle (\boldsymbol{G}, \boldsymbol{\sigma}) \models_\gamma s \right\rangle_{\boldsymbol{G}} \middle| \boldsymbol{G} \models_\gamma s \right].$$

Finally, it is not difficult to derive Theorem 3. Indeed,

$$n^{-1} \ln \mathbb{E}\left[Z_{\boldsymbol{G}} \left\langle (\boldsymbol{G}, \boldsymbol{\sigma}) \models_\gamma s \right\rangle_{\boldsymbol{G}} \middle| \boldsymbol{G} \models_\gamma s \right]$$

can be cast as the Bethe free energy of the marginal sequence induced by s: let $\nu_x = \tilde{s}_{s(x)}$ for $x \in V$ and $\nu_a = \tilde{s}_{\psi,h}$ for all $a \in \partial_{G_0,s}(\psi, h)$. But how we can get a handle on the Bethe free energy $\mathcal{B}_{\mathcal{M}}(G)$?

4.3 Belief Propagation

The Bethe free energy of a given factor graph G can be calculated by analysing the Belief Propagation message passing algorithm. Belief Propagation can be viewed as an operator acting on the ***message space*** $\mathrm{Mes}_{\mathcal{M}}(G)$ of G, which we define as the set of all maps $\hat{\nu} : C_V \cup C_F \to \mathcal{P}(\Omega)$, $(v, j) \mapsto \hat{\nu}_{v,j}$. The ***Belief Propagation operator*** $\mathrm{BP} : \mathrm{Mes}_{\mathcal{M}}(G) \to \mathrm{Mes}_{\mathcal{M}}(G)$ maps $\hat{\nu} \in \mathrm{Mes}_{\mathcal{M}}(G)$ to $\tilde{\nu} = \mathrm{BP}(\hat{\nu})$ defined by[3]

$$\tilde{\nu}_{x,i}(\omega_i) \propto \prod_{h \in [d(x)] \setminus \{i\}} \hat{\nu}_{G_{x,h}}(\omega_i) \tag{4}$$

for $(x, i) \in C_V, \omega_i \in \Omega$ and

$$\tilde{\nu}_{a,j}(\omega_j) \propto \sum_{(\omega_h)_{h \in [d(a)] \setminus \{j\}}} \psi_a(\omega_1, \dots, \omega_{d(a)}) \prod_{h \in [d(a)] \setminus \{j\}} \hat{\nu}_{G_{a,h}}(\omega_h) \tag{5}$$

[3] As per common practice, we use the $\propto$ symbol to define probability distributions on a finite set $\mathcal{X}$ as follows. If $f : \mathcal{X} \to [0, \infty)$, then $p \propto f$ means that $p(\omega) = f(\omega) / \sum_{x \in \mathcal{X}} f(x)$ unless $\sum_{x \in \mathcal{X}} f(x) = 0$, in which case p is the uniform distribution.

for $(a,j) \in C_F, \omega_1, \ldots, \omega_{d(a)} \in \Omega$. Let $\mathrm{Fix}_{\mathcal{M}}(G)$ be the set of all Belief Propagation fixed points, i.e., all $\hat{\nu} \in \mathrm{Mes}_{\mathcal{M}}(G)$ such that $\mathrm{BP}(\hat{\nu}) = \hat{\nu}$. Any point $\hat{\nu} \in \mathrm{Fix}_{\mathcal{M}}(G)$ gives rise to a marginal sequence, namely

$$\hat{\nu}_x(\omega) \propto \prod_{h \in [d(x)]} \hat{\nu}_{G_{x,h}}(\omega) \tag{6}$$

for $x \in V, \omega \in \Omega$ and

$$\hat{\nu}_a(\omega_1, \ldots, \omega_{d(a)}) \propto \psi_a(\omega_1, \ldots, \omega_{d(a)}) \prod_{h \in [d(a)]} \hat{\nu}_{G_{a,h}}(\omega_h) \tag{7}$$

for $a \in F, \omega_1, \ldots, \omega_{d(a)} \in \Omega$.

▶ **Proposition 10.** *We have* $\mathcal{B}_{\mathcal{M}}(G) = \max\left\{\mathcal{B}_{\mathcal{M}}(G, \hat{\nu}) : \hat{\nu} \in \mathrm{Fix}_{\mathcal{M}}(G)\right\}$.

Proof. The set M of marginal sequences is compact. Because the functions $\psi \in \Psi$ are strictly positive and as the derivative of the entropy diverges as ν approaches the boundary ∂M, $\mathcal{B}_{\mathcal{M}}(G, \cdot\,)$ does not attain its global maximum on ∂M. Furthermore, for any stationary point $\nu \in M$ of the Bethe free energy $\mathcal{B}_{\mathcal{M}}(G, \cdot\,)$ there exists $\hat{\nu} \in \mathrm{Fix}_{\mathcal{M}}(G)$ such that $\mathcal{B}_{\mathcal{M}}(G, \nu) = \mathcal{B}_{\mathcal{M}}(G, \hat{\nu})$ [23, Proposition 14.6]. ◀

Theorem 3 shows that the Bethe free energy provides an upper bound on $n^{-1} \ln Z_{\boldsymbol{G}(\mathcal{M})}$. Furthermore, Proposition 10 reduces the problem of calculating the Bethe free energy to the task of determining the "dominant fixed point" of Belief Propagation, i.e., the task of analysing an algorithm on a random graph.

5 The lower bound

In this section we consider a sequence $(\mathcal{M}(n) = (V_n, F_n, d_n, t_n, (\psi_{n,a}))_n$ *of* $(\Delta, \Omega, \Psi, \Theta)$*-models such that* $\#\mathcal{M}(n) = n$. *Let* $\boldsymbol{G}(n) = \boldsymbol{G}(\mathcal{M}(n))$ *and* $\mathcal{G}(n) = \mathcal{G}(\mathcal{M}(n))$.

5.1 A Bethe-enhanced second moment method

The cavity method provides a "recipe" for calculating a number ϕ such that $(n^{-1} \ln Z_{\boldsymbol{G}(n)})_n$ is deemed to converge to ϕ in probability. This number is determined by applying Belief Propagation and the Bethe free energy to the "limit" of the typical local structure of $\boldsymbol{G}(n)$ as $n \to \infty$. The aim in this section is to develop a version of the second moment method that allows us to prove such a claim rigorously. But first we need to formalise the "limiting local structure". To this end we adapt the concept of *local weak convergence* of graph sequences [22, Part 4] to our current setup, which can be viewed as a generalisation of the one from [10].

▶ **Definition 11.** A $(\Delta, \Omega, \Theta, \Psi)$-***template*** consists of a $(\Delta, \Omega, \Psi, \Theta)$-model $\mathcal{M}$, a connected factor graph $H \in \mathcal{G}(\mathcal{M})$ and a ***root*** (r_H, i_H), which is a variable or factor clone. We denote the template by H. Its ***size*** is $\#H = \#\mathcal{M}$.

Two templates H, H' with models $\mathcal{M} = (V, F, d, t, (\psi_a), \sigma_*)$, $\mathcal{M}' = (V', F', d', t', (\psi'_a), \sigma'_*)$ are ***isomorphic*** if there exists a bijection $\pi : V \cup F \to V' \cup F'$ such that the following conditions are satisfied.

ISM1 $\pi(x) \in V'$ for all $x \in V$ and $\pi(a) \in F'$ for all $a \in F$,

ISM2 if $r_H = (x_H, i_H)$ and $r_{H'} = (x_{H'}, i_{H'})$, then $\pi(x_H) = x_{H'}$ and $i_H = i_{H'}$,

ISM3 $d(v) = d'(\pi(v))$, $\sigma_*(v) = \sigma'_*(\pi(v))$ for all $v \in V \cup F$ and $t(v,i) = t'(\pi(v,i))$ for all $(v,i) \in C_V \cup C_F$,

ISM4 $\psi_a = \psi_{\pi(a)}$ for all $a \in F$,

ISM5 for all $(v,i) \in C_V$ we have $H_{v,i} = (a,j)$ iff $H'_{\pi(v),i} = (\pi(a), j)$.

We denote the isomorphism class of a template H by $[H]$. Let $\mathfrak{G} = \mathfrak{G}(\Delta, \Omega, \Theta, \Psi)$ be the set of all isomorphism classes. Further, let $\mathfrak{T} \subset \mathfrak{G}$ be the set of all isomorphism classes of acyclic templates. For each $[H] \in \mathfrak{G}$ and $\ell \geq 1$ let $\partial^\ell[H]$ be the isomorphism class of the template obtained by removing all vertices at a distance greater than ℓ from the root if the root is a variable clone and $\ell + 1$ if the root is a factor node. We endow $\mathfrak{G}$ with the coarsest topology that makes all the functions $\Gamma \in \mathfrak{G} \mapsto \mathbf{1}\{\partial^\ell[\Gamma] = \partial^\ell[\Gamma_0]\} \in \{0,1\}$ for $\ell \geq 1, \Gamma_0 \in \mathfrak{G}$ continuous. Moreover, the space $\mathcal{P}(\mathfrak{G})$ of probability measures on $\mathfrak{G}$ carries the weak topology. So does the space $\mathcal{P}^2(\mathfrak{G})$ of probability measures on $\mathcal{P}(\mathfrak{G})$. For $\Gamma \in \mathfrak{G}$ and $\lambda \in \mathcal{P}(\mathfrak{G})$ we write $\delta_\Gamma \in \mathcal{P}(\mathfrak{G})$ and $\delta_\lambda \in \mathcal{P}^2(\mathfrak{G})$ for the Dirac measure that puts mass one on Γ resp. λ.

For a factor graph $G \in \mathcal{G}(n)$ and a clone (v,i) we write $[G, v, i]$ for the isomorphism class of the connected component of (v,i) in G rooted at (v,i). Each $G \in \mathcal{G}(n)$ gives rise to the empirical distribution

$$\lambda_G = \frac{1}{|C_{V_n}| + |C_{F_n}|} \sum_{(v,i) \in C_{V_n} \cup C_{F_n}} \delta_{[G,v,i]} \in \mathcal{P}(\mathfrak{G}).$$

Let $\Lambda_n = \mathbb{E}[\delta_{\lambda_{\boldsymbol{G}(n)}}] \in \mathcal{P}^2(\mathfrak{G})$. We say that $(\mathcal{M}(n))_n$ ***converges locally*** to $\vartheta \in \mathcal{P}(\mathfrak{T})$ if $\lim_{n\to\infty} \Lambda_n = \delta_\vartheta$.

Additionally, to exclude some pathological cases we need the following assumption. Let us call a factor graph G ***ℓ-acyclic*** if it does not contain a cycle of length at most ℓ. We say that the sequence $(\mathcal{M}(n))_n$ of models ***has high girth*** if for any $\ell > 0$ we have

$$\liminf_{n\to\infty} \mathbb{P}\left[\boldsymbol{G}(n) \text{ is } \ell\text{-acyclic}\right] > 0.$$

The key prediction of the "replica symmetric cavity method" can be cast as follows: $(n^{-1} \ln Z_{\boldsymbol{G}(n)})_n$ converges in probability to the Bethe free energy of a "Belief Propagation fixed point" on the (possibly infinite) trees in the support of ϑ [23]. To formalise this, let $\boldsymbol{T}_\vartheta \in \mathfrak{T}$ be a sample from $\vartheta \in \mathcal{P}(\mathfrak{T})$. Further, let $\mathcal{V}$ be the event that the root of $\boldsymbol{T}_\vartheta$ is a variable clone and let $\mathcal{F}$ be the event that the root is a constraint clone. For $T \in \mathfrak{T}$ let d_T denote the degree of the root of T. Moreover, for $j \in [d_T]$ let $T \uparrow j \in \mathfrak{T}$ denote the tree pending on the jth child of the root of T. Finally, if the root is the clone of a constraint node we let ψ_T be its associated function.

▶ **Definition 12.** A measurable map $p : \mathfrak{T} \to \mathcal{P}(\Omega)$, $T \mapsto p_T$ is called a ***ϑ-Belief Propagation fixed point*** if the following conditions are satisfied ϑ-almost surely.

1. if the root of T is a variable clone (x,i), then

$$p_T(\omega) \propto \prod_{j \in [d_T] \setminus \{i\}} p_{T\uparrow j}(\omega).$$

2. if the root of T is a factor clone (a,i) with associated factor $\psi \in \Psi$, then

$$p_T(\omega_i) \propto \sum_{(\omega_j)_{j\in[d_T]\setminus\{i\}}} \psi(\omega_1, \ldots, \omega_{d_T}) \prod_{j \in [d_T]\setminus\{i\}} p_{T\uparrow j}(\omega_j).$$

Further, we need to define the Bethe free energy of a ϑ-Belief Propagation fixed point p. To this end, we turn p into a map that assigns each tree a "marginal". More precisely, we let

$$\hat{p}_T(\omega) \propto \prod_{j\in[d_T]} p_{T\uparrow j}(\omega) \qquad \text{if } T\in\mathcal{V},\ \omega\in\Omega,$$

$$\hat{p}_T(\omega_1,\ldots,\omega_{\#\psi_T}) \propto \psi_T(\omega_1,\ldots,\omega_{\#\psi_T}) \prod_{j\in[\#\psi_T]} p_{T\uparrow j}(\omega_j) \quad \text{if } T\in\mathcal{F},\ \omega_1,\ldots,\omega_{\#\psi_T}\in\Omega.$$

The ***Bethe free energy*** of p with respect to ϑ is

$$\mathcal{B}_\vartheta(p) = \left(\mathbb{E}\left[d_{\boldsymbol{T}_\vartheta}^{-1}(1-d_{\boldsymbol{T}_\vartheta})H(\hat{p}_{\boldsymbol{T}_\vartheta})|\mathcal{V}\right] - \mathbb{E}\left[d_{\boldsymbol{T}_\vartheta}^{-1}D\left(\hat{p}_{\boldsymbol{T}_\vartheta}\|\psi_{\boldsymbol{T}_\vartheta}(\sigma)\right)|\mathcal{F}\right]\right)\mathbb{E}[d_{\boldsymbol{T}_\vartheta}|\mathcal{V}]$$

Finally, to obtain a sufficient condition for the convergence $n^{-1}\ln Z_{\boldsymbol{G}(n)} \to \mathcal{B}_\vartheta(p)$ we are going to apply Theorem 3 to upper-bound the second moment of $Z_{\boldsymbol{G}(n)}$. The necessary construction, reminscent of those used in [9, 13, 11, 16, 26, 30], is as follows.

▶ **Proposition 13.** *For any $\varepsilon>0$ there exists $\eta>0$ such that the following is true. Suppose that $\mathcal{M}$ is a $(\Delta,\Omega,\Psi,\Theta)$-model of size $n=\#\mathcal{M}\geq 1/\eta$. There exists a finite set of functions $\Psi^\otimes$ and a $(\Delta,\Omega\times\Omega,\Psi^\otimes,\Theta)$-model $\mathcal{M}^\otimes$ with the following properties.*

(i) *There is a bijection $\mathcal{G}(\mathcal{M})\to\mathcal{G}(\mathcal{M}^\otimes)$, $G\mapsto G^\otimes$.*

(ii) *Let $\mathcal{U}\subset\mathcal{G}(\mathcal{M})$ be an event such that $\mathbb{P}[\boldsymbol{G}\in\mathcal{U}]>\varepsilon$. Then*

$$n^{-1}\ln\mathbb{E}[Z_{\boldsymbol{G}}^2|\mathcal{U}] \leq \max\left\{\mathcal{B}_{\mathcal{M}^\otimes}(G^\otimes) : G\in\mathcal{U}\right\}+\varepsilon.$$

Proof. Let $\Omega^\otimes=\Omega\times\Omega$ and denote $(\omega,\omega')\in\Omega^\otimes$ by $\omega\otimes\omega'$. For $\psi\in\Psi$ let

$$\psi^\otimes : (\Omega^\otimes)^{\#\psi}\to(0,\infty), \qquad (\omega_1\otimes\omega_1',\ldots,\omega_{\#\psi}\otimes\omega_{\#\psi}')\mapsto\psi(\omega_1,\ldots,\omega_{\#\psi})\cdot\psi(\omega_1',\ldots,\omega_{\#\psi}').$$

Then $\mathcal{M}^\otimes=(V,F,d,t,(\psi_a^\otimes)_{a\in F})$ satisfies the requirements. ◀

▶ **Theorem 14.** *Suppose that $(\mathcal{M}(n))_{n\geq1}$ has high girth and converges locally to $\vartheta\in\mathcal{P}(\mathfrak{T})$. Furthermore, assume that there is a ϑ-Belief Propagation fixed point p such that for any $\varepsilon>0$ we have*

$$\lim_{n\to\infty}\mathbb{P}\left[\mathcal{B}_{\mathcal{M}^\otimes(n)}(\boldsymbol{G}^\otimes(n))\leq 2\mathcal{B}_\vartheta(p)+\varepsilon\right]=1 \quad \text{and} \tag{8}$$

$$\lim_{\ell\to\infty}\lim_{n\to\infty}\frac{1}{n}\mathbb{E}\left[\ln\frac{\mathbb{E}[Z_{\boldsymbol{G}(n)}\mathbf{1}\{\mathcal{B}_{\mathcal{M}^\otimes(n)}(\boldsymbol{G}^\otimes(n))\leq 2\mathcal{B}_\vartheta(p)+\varepsilon\}|\mathcal{T}_\ell]}{\mathbb{E}[Z_{\boldsymbol{G}(n)}|\mathcal{T}_\ell]}\right]=0. \tag{9}$$

Then $\frac{1}{n}\ln Z_{\boldsymbol{G}(n)}$ converges to $\mathcal{B}_\vartheta(p)$ in probability.

For a given ϑ the construction or, at least, identification of the ϑ-Belief Propagation fixed point p in Theorem 14 is similar to the computations done in the physics literature. However, to apply Theorem 14 it will generally be necessary to perform these calculations more thoroughly, e.g., by means of the contraction method [27]. Further, to verify condition (8) we need to study the Bethe free energy of the models $\mathcal{M}_n^\otimes$, which will typically be done by way of analysing Belief Propagation on the random factor graph $\boldsymbol{G}^\otimes(n)$. This task may be far from trivial, but at least it is a well-defined combinatorial problem.

Finally, (9) provides that given that the local structure up to depth ℓ is "typical", conditioning on the event that the second moment Bethe free energy is bounded by $2\mathcal{B}_\vartheta(p)+\varepsilon$ does not cause a substantial drop in the first moment. This is a technical condition that can be verified by studying an auxiliary probability space, namely a variant of the "planted model" with a given local structure. Technically, this task can be tackled via a generalised "configuration model" as put forward in [6].

References

1 Dimitris Achlioptas and Cristopher Moore. Random k-sat: Two moments suffice to cross a sharp threshold. *SIAM J. Comput.*, 36(3):740–762, September 2006.

2 Dimitris Achlioptas, Assaf Naor, and Yuval Peres. Rigorous location of phase transitions in hard optimization problems. *Nature*, 435(7043):759–764, 06 2005.

3 Dimitris Achlioptas, Assaf Naor, and Yuval Peres. On the maximum satisfiability of random formulas. *J. ACM*, 54(2), April 2007.

4 Dimitris Achlioptas and Yuval Peres. The threshold for random k-sat is $2k(\ln 2 - o(k))$. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC'03, pages 223–231, New York, NY, USA, 2003. ACM.

5 H. A. Bethe. Statistical theory of superlattices. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 150(871):552–575, 1935.

6 Charles Bordenave and Pietro Caputo. Large deviations of empirical neighborhood distribution in sparse random graphs. *Probability Theory and Related Fields*, pages 1–73, 2014.

7 Amin Coja-Oghlan and Konstantinos Panagiotou. The asymptotic k-SAT threshold. *arXiv:1310.2728*, 2014.

8 Amin Coja-Oghlan and Lenka Zdeborová. The condensation transition in random hypergraph 2-coloring. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 241–250, 2012.

9 Amir Dembo, Andrea Montanari, Allan Sly, and Nike Sun. The replica symmetric solution for potts models on d-regular graphs. *Communications in Mathematical Physics*, 327(2):551–575, 2014.

10 Amir Dembo, Andrea Montanari, and Nike Sun. Factor models on locally tree-like graphs. *Ann. Probab.*, 41(6):4162–4213, 11 2013.

11 Jian Ding, Allan Sly, and Nike Sun. Maximum independent sets on random regular graphs. *arXiv:1310.4787*, 2013.

12 Jian Ding, Allan Sly, and Nike Sun. Proof of the satisfiability conjecture for large k. *arXiv:1411.0650*, 2014.

13 Jian Ding, Allan Sly, and Nike Sun. Satisfiability threshold for random regular NAE-SAT. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC'14, pages 814–822, New York, NY, USA, 2014. ACM.

14 Silvio Franz and Michele Leone. Replica bounds for optimization problems and diluted spin systems. *J. Stat. Phys.*, 111(3-4):535–564, 2003.

15 Alan Frieze and Nicholas C. Wormald. Random k-sat: A tight threshold for moderately growing k. In *Proceedings of the Fifth International Symposium on Theory and Applications of Satisfiability Testing*, pages 1–6, 2002.

16 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability for antiferromagnetic spin systems in the tree non-uniqueness region. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC'14, pages 823–831, New York, NY, USA, 2014. ACM.

17 R.G. Gallager. *Low-density Parity-check Codes.* M.I.T. Press research monographs. M.I.T. Press, 1963.

18 H.O. Georgii. *Gibbs Measures and Phase Transitions.* De Gruyter studies in mathematics. De Gruyter, 2011.

19 Francesco Guerra. Broken replica symmetry bounds in the mean field spin glass model. *Communications in Mathematical Physics*, 233(1):1–12, 2003.

20 Svante Janson. Random regular graphs: Asymptotic distributions and contiguity. *Combinatorics, Probability and Computing*, 4:369–405, 12 1995.

21 Florent Krzakala, Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *P. Natl. Acad. Sci. USA*, 104(25):10318–10323, 2007.

22 L. Lovász. *Large Networks and Graph Limits.* American Mathematical Society colloquium publications. American Mathematical Society, 2012.

23 M. Mézard and A. Montanari. *Information, Physics and Computation.* Oxford University Press, 2009.

24 M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297:812–815, 2002.

25 Andrea Montanari and Devavrat Shah. Counting good truth assignments of random k-sat formulae. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'07, pages 1255–1264, Philadelphia, PA, USA, 2007.

26 Elchanan Mossel, Dror Weitz, and Nicholas Wormald. On the hardness of sampling independent sets beyond the tree threshold. *Probability Theory and Related Fields*, 143(3-4):401–439, 2009.

27 Ralph Neininger and Ludger Rüschendorf. A general limit theorem for recursive algorithms and combinatorial structures. *Ann. Appl. Probab.*, 14(1):378–418, 02 2004.

28 Dmitry Panchenko and Michel Talagrand. Bounds for diluted mean-fields spin glass models. *Probab. Theory Relat. Fields*, 130(3):319–336, 2004.

29 Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

30 Allan Sly and Nike Sun. The computational hardness of counting in two-spin models on d-regular graphs. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 0:361–369, 2012.

31 Endre Szemerédi. Regular partitions of graphs. In *Colloq. Internat. CNRS*, volume 260, pages 399–401, 1978.

32 Jonathan S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 51(7):2282–2312, July 2005.

Internal Compression of Protocols to Entropy[*]

Balthazar Bauer[1], Shay Moran[2], and Amir Yehudayoff[3]

1 **Département d'Infomatique, ENS Lyon**
Lyon, France
balthazarbauer@aol.com

2 **Departments of Computer Science, Technion-IIT**
Haifa, Israel, and
Max Planck Institute for Informatics
Saarbrücken, Germany
shaymrn@cs.technion.ac.il

3 **Departments of Mathematics, Technion-IIT**
Haifa, Israel
amir.yehudayoff@gmail.com

Abstract

We study internal compression of communication protocols to their internal entropy, which is the entropy of the transcript from the players' perspective. We provide two internal compression schemes with error. One of a protocol of Feige et al. for finding the first difference between two strings. The second and main one is an internal compression with error $\varepsilon > 0$ of a protocol with internal entropy H^{int} and communication complexity C to a protocol with communication at most order $(H^{int}/\varepsilon)^2 \log(\log(C))$.

This immediately implies a similar compression to the internal information of public-coin protocols, which provides an exponential improvement over previously known public-coin compressions in the dependence on C. It further shows that in a recent protocol of Ganor, Kol and Raz, it is impossible to move the private randomness to be public without an exponential cost. To the best of our knowledge, No such example was previously known.

1998 ACM Subject Classification H.1.1 Coding and Information Theory: Data Compaction and Compression

Keywords and phrases Communication complexity, Information complexity, Compression, Simulation, Entropy

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.481

1 Introduction

The problem of compressing information and communication is fundamental and useful. This paper studies one shot compression of interactive communication (as opposed to amortized compression).

The basic scenario, the transmission problem, was studied by Fano and Shannon [12] and by Huffman [17]. In it, Alice wishes to transmit to Bob a message $u \in U$ with u that is distributed according to a known distribution μ over U. They proved that the above transmission can be optimally compressed in the sense that Alice may send u to Bob using roughly $\log(1/\mu(u))$ many bits on average, and conversely if Alice sends fewer

[*] This work was supported by the Taub foundation, by ISF, and by BSF.

than $\log(1/\mu(u))$ bits on average then information is lost. In the transmission problem, the information flow is one-way, only Alice talks.

How about more complex communication protocols in which both sides are allowed to talk? The standard model for interactive communication was introduced by Yao [28]. Interactive communication, not surprisingly, allows for more efficient conversations than one-way ones. For example, the following lemma (which we also use later on) demonstrates the power of interaction (and of public randomness) in handling a variant of the transmission problem in which only Bob knows the distribution μ over U.

▶ **Lemma 1.1.** *Let U be a finite set, and $0 < \varepsilon < 1/2$. Assume Alice knows some $u_a \in U$ and that Bob knows a distribution μ on U which Alice does not know. Using public randomness, Alice and Bob can communicate at most $2\log(1/\mu(u_a)) + \log(1/\varepsilon) + 5$ bits, after which Bob outputs u_b so that $u_a = u_b$ with probability at least $1 - \varepsilon$.*

This lemma describes a one shot protocol (i.e. for a single instance) that enables transmission when Bob has some prior knowledge on Alice's input. A stronger version of this lemma was proved in [5] and also in [6], but since this lemma is sufficient for us and its proof is simpler than that of [5, 6] we provide its proof in Section A.3. A related result for the case when there is also an underlying distribution on Alice's input is the Slepian-Wolf theorem [26] which solves an amortized version of this problem. It is also related to the transmission problem considered by Harsha et al. [16] who studied the case that Alice knows μ and Bob wishes to sample from it.

Continuing recent works which we survey below, the main question we study is compression of interactive communication protocols. Compression of protocols, on a high level, means to simulate a given protocol π by a more efficient protocol σ in the sense that the communication complexity of σ is roughly the "information content" of π. It was shown to be strongly related to direct sum and product questions in randomized communication complexity [5, 2, 7].

We describe new compression schemes, and also provide a preliminary discussion of concepts and basic facts related to compression.

1.1 A preliminary discussion

In this section we provide intuitive definitions of important concepts. See Section 2 for formal definitions.

1.1.1 Computation and simulation

There is a distinction between external computation and internal computation [2, 7]. A protocol externally computes a function f if an external observer can deduce the value of f from the transcript, and a protocol internally computes f if the value of f may be privately obtained by Alice and Bob but not necessarily by an external observer (who only sees the transcript of the protocol but not the inputs). Note that for $f : X \times Y \to Z$ the difference between internal and external computation of f can be at most $\log|Z|$. Indeed, every protocol that internally computes f can be transformed to a protocol that externally computes f by adding one more message in which one of the parties sends the value of the function. Therefore, this distinction is only meaningful for large Z.

It is interesting that for deterministic protocols these two seemingly different notions coincide, so the strength of internal computation is evident only in randomized or distributional settings (the proof is given in Section A.1).

▶ **Proposition 1.2.** *Let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$. If π is a deterministic protocol that internally computes f then it also externally computes f.*

External and internal computations induce the corresponding types of simulations. Here we provide an intuitive meaning of the notion of simulation. In Section 2 we provide formal definitions, and discuss them in more detail. A protocol σ externally simulates the protocol π if an external observer who has access only to the public data (i.e. transcript and public randomness) of σ can deduce from it the public data of π. The protocol σ internally simulates π if each of Alice and Bob can obtain their private data of π from their private data of σ (i.e. transcript and private randomness).

As an example which illustrates the difference between internal and external simulation, consider the simple case when (x, y) are jointly distributed so that $x = y$, Alice knows x, Bob knows y and π is the protocol in which Alice sends x to Bob. In this case, it is clear that the empty protocol internally simulates π but every external simulation of π must in general use many bits. This example also demonstrates that Proposition 1.2 does not hold for promise problems, when the inputs are guaranteed to come from a non rectangular set.

1.1.2 Compression

To define compression, we should ask ourselves what is the "benchmark quantity" that we should strive to compress to. Shannon's source coding theorem [25] states that in the transmission problem (i.e., one-way communication), the entropy of the message is equal to the amortized communication of sending many independent messages. Braverman and Rao [5] analogously showed that the internal information (defined below) is equal to the amortized cost of making several independent conversations. Entropy and internal information are therefore two reasonable choices for "benchmark quantities". Below we survey several additional options.

1.1.3 Information complexities

The most studied measures in the context of protocol compression are information complexities. For every communication protocol π and every distribution μ on inputs, two versions of information have been defined: The internal information [1, 2] denoted $I^{int}_{\mu}(\pi)$ and the external one [9] denoted $I^{ext}_{\mu}(\pi)$ (see Section 2 for formal definitions). The intuitive semantic of internal information is the amount of information the communication transcript reveals to Alice and Bob about the inputs, and the intuitive semantic of external information is the amount of information the communication transcript reveals to an external observer about the inputs. It always holds that the internal information is at most the external one, which is at most the average communication complexity $\mathsf{CC}^{avg}_{\mu}(\pi)$ (see e.g. [2, 18]).

The following claim shows that information provides a lower bound for errorless simulations. This generalizes the basic fact that entropy provides a lower bound for errorless transmission. This claim seems to be known but we could not find an explicit reference to it so we provide a proof in Section A.2 (the special case of deterministic external simulation was proved in [23]).

▶ **Claim 1.3.** *Let π be a general protocol with input distribution μ.*
- *If σ simulates π externally without error then $\mathsf{CC}^{avg}_{\mu}(\sigma) \geq I^{ext}_{\mu}(\pi)$.*
- *If σ simulates π internally without error then $\mathsf{CC}^{avg}_{\mu}(\sigma) \geq I^{int}_{\mu}(\pi)$.*

Although $I^{int}_{\mu}(\pi) \leq I^{ext}_{\mu}(\pi)$, the second bullet in the claim above does not follow from the first, since not every internal simulation is an external simulation.

In the other direction, [2] provided two different compression schemes for general protocols. An external compression with error that uses roughly $I_\mu^{ext}(\pi)\log(CC(\pi))$ bits, and an internal compression with error that uses roughly $\sqrt{I_\mu^{int}(\pi)\cdot CC(\pi)}$ bits. A second internal compression with error that uses at most roughly $2^{I_\mu^{int}(\pi)}$ bits, regardless of $CC(\pi)$, appears in [3]. Later on, [8, 24] showed that the internal compression from [2] applied to public-coin protocols yields a much better compression with only order $I_\mu^{int}(\pi)\log(CC(\pi))$ bits. We discuss connections of these works to ours below.

1.1.4 Entropy complexities

We consider two additional complexity measures for compression:

The first one, which was studied in [11], is the external entropy $H_\mu^{ext}(\pi)$. Its semantic is how many bits are required for describing the transcript of π to an external observer. The second measure we consider is the internal entropy $H_\mu^{int}(\pi)$. Its semantic is the number of bits required in order to describe the transcript to Alice plus the number of bits required to describe the transcript to Bob (see Section 2 for formal definitions).

Some connections between the information measures and the entropy measures are provided in the following claim.

▶ **Claim 1.4.** *Let π be a protocol with input distribution μ. Then,*

$$H_\mu^{ext}(\pi) \geq I_\mu^{ext}(\pi) \quad \textit{and} \quad H_\mu^{int}(\pi) \geq I_\mu^{int}(\pi).$$

Moreover, if π does not have private randomness then

$$H_\mu^{ext}(\pi) = I_\mu^{ext}(\pi) \quad \textit{and} \quad H_\mu^{int}(\pi) = I_\mu^{int}(\pi).$$

As mentioned, in the case of one-way deterministic protocols, the external entropy fully captures the compression problem. The above claim combined with Claim 1.3 implies that, more generally, for public-coin protocols entropy provides a lower bound on errorless simulation. Interestingly, the authors of [11] proved that this lower bound is essentially tight. They gave an optimal external compression of general protocols[1]

▶ **Theorem 1.5** ([11]). *Every protocol π can be externally simulated without error by a protocol σ so that $\mathsf{CC}_\mu^{avg}(\sigma) \leq O(H_\mu^{ext}(\pi))$.*

1.1.5 With or without error

Another important distinction is between exact simulation and simulation with error.

A meaningful example already appears in the transmission problem, when there is a distribution μ on inputs x and Alice sends a (prefix free) encoding of x to Bob. Any exact solution to this problem requires expected communication of at least $H(\mu)$. However, if μ is highly concentrated on a point but with probability ε it is uniform on the remaining elements, an empty protocol simulates μ with ε error while the entropy is potentially huge. So entropy and information are not in general lower bounds for simulation with error, and the lower bounds from Claim 1.3 do not hold for simulation with error.

In the other direction, we have seen that entropy (or information) provides a lower bound on errorless simualtion. We shall see below that this lower bound is not tight, that is, there are protocols with small entropy that can not be efficiently simulated without error.

[1] They only considered deterministic protocols but their arguments can be generalized to general protocols.

1.2 Internal compression

1.2.1 Impossibility of errorless compression

Theorem 1.5 above provides errorless compression to external entropy. The main compression question is, however, whether a protocol can be internally simulated with communication that is close to its internal information. Motivation for studying this questions comes from direct sum and product questions in randomized communication complexity [5, 2, 7].

How about an errorless internal compression to internal entropy? It is long known that internal compression to internal entropy is not always possible [14, 22, 20, 3]. In [22], for example, Orlitsky studied zero-error compression of the transmission protocol in which Alice sends her input x to Bob who knows y, and constructed distributions on (x, y) such that every errorless internal simulation of the transmission protocol must communicate at least $H(x)$ bits, which is strictly larger than the internal entropy $H(x|y)$. Later on, Naor, Orlitskty and Shor [20] strengthened it to the amortized setting. A concrete statement (that can be proved e.g. using ideas from [3, 18]) is that for every n, there is a one round deterministic protocol π and input distribution μ so that $H^{int}_{\mu}(\pi) \leq 1$ and $CC(\pi) \leq n$ but if σ is an errorless internal simulation of π then $\mathsf{CC}^{avg}_{\mu}(\sigma) \geq n - 2$.

Our internal compression scheme and the ones from [2, 3] must therefore introduce errors.

1.2.2 Finding the first difference

Before stating our general compression scheme, we demonstrate its ideas by an internal compression of the *finding the first difference* problem, which lies at the heart of the internal compression schemes of [2, 8, 24]. Feige et al. [13] gave an optimal randomized protocol for this problem in terms of communication complexity (Viola [27] proved a matching lower bound).

▶ **Theorem 1.6** ([13]). *There is a public-coin protocol that on inputs $x, y \in \{0,1\}^n$ externally outputs the smallest index i in which x, y differ (or outputs "equal" if $x = y$) with probability at least $1 - \varepsilon$. The communication complexity of this protocol is at most $O(\log(n/\varepsilon))$.*

The protocol of Feige et al. externally solves the problem. The following Theorem provides an internal solution for this problem, which is more efficient when the internal information is small (the protocol is presented in Section 3).

▶ **Theorem 1.7.** *Let μ be a distribution on $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, and let $\varepsilon > 0$. Denote by $i = i(x, y)$ the smallest index in which x, y differ (or $i =$ "equal" if $x = y$). Denote $h^{int} = H(i|x) + H(i|y)$. There is a public-coin protocol and an event $\mathcal{E} \subset \{0,1\}^n \times \{0,1\}^n$ with probability $\mu(\mathcal{E}) < \varepsilon$ so that for all $(x, y) \notin \mathcal{E}$, the communication complexity of the protocol on input (x, y) is at most*[2]

$$O\left(\log\left(\frac{1}{\mu(i|x) \cdot \mu(i|y)}\right) \log\left(\log(n) h^{int}/\varepsilon\right)\right),$$

and it internally computes i with probability at least $1 - \varepsilon$. The overall communication complexity with error ε is at most

$$O\left(\frac{h^{int}}{\varepsilon} \log\left(\log(n) h^{int}/\varepsilon\right)\right).$$

2 Here and below, for simplicity of notation, we write $\mu(i|y)$ to denote $\mu(\{(x, y) : i(x, y) = i\}|\{y\})$.

We state the theorem in this form since it hints at the core of its proof. To understand it better, it may be helpful to observe

$$H(i|x) + H(i|y) = I(i;y|x) + I(i;x|y) = \mathbb{E}_\mu \log\left(\frac{1}{\mu(i|x)\cdot\mu(i|y)}\right).$$

This protocol gives an improvement over that of [13] when the internal entropy is small. It highlights the importance of internal computation and may help to understand the more general compression below. It may also be useful in future internal compression schemes.

1.2.3 Main compression

We finally state our internal compression scheme (see Section 4 for its description). As mentioned above, such a compression must have positive error, even for one round protocols.

▶ **Theorem 1.8.** *Let μ be a distribution on $\mathcal{X}\times\mathcal{Y}$ and let $\varepsilon > 0$. Let π be a protocol with inputs from μ. Then, there is a public-coin protocol σ with communication complexity*

$$CC(\sigma) \le O\left(\frac{\left(H_\mu^{int}(\pi)\right)^2}{\varepsilon^2}\cdot\log(\log(CC(\pi)))\right)$$

that internally simulates π with error ε.

As noted earlier, if π is a protocol that uses no private randomness then the internal entropy of π is equal to the internal information of π. So, for public-coin protocols, Theorem 1.8 gives an internal compression in terms of internal information, which exponentially improves [8, 24] in terms of the dependence on $CC(\pi)$. It, therefore, also concerns the power of private randomness in saving information, which we now discuss.

1.2.4 Transferring private to public randomness

Every private-coin protocol can be simulated by a public-coin protocol with the same *communication* complexity. Conversely, Newman [21] proved that for communication complexity public randomness may be efficiently replaced by private one, when dealing with computation of relations (it however does not yield a simulation of public-coin protocols by private-coin protocols). In the information complexity context the situation is opposite, every public-coin protocol can be simulated by a private-coin protocol with the same *information* complexity. The authors of [8, 4] showed that for information complexity private randomness may be efficiently simulated by public one when the number of rounds is bounded. If any private-coin protocol could be simulated by a private-coin one without increasing the information and communication complexities, then to compress general protocol it would suffice to compress public-coin protocols.

Our compression shows limitations on moving private randomness to being public. A recent work of Ganor, Kol and Raz [15] shows that for every large enough $k \in \mathbb{N}$ there is a distribution μ and a private-coin protocol π_0 with internal information $O(k)$ so that every protocol that internally simulates π_0 with small error must communicate at least 2^k bits. This marks the first known separation between information and communication complexities. The protocol π_0 has communication complexity $O(k\cdot 2^{4^k})$ so that $\log(\log(CC(\pi_0)) = O(k)$. Together with our compression scheme, this means that there is no way to simulate π_0 using only public randomness without a cost; for example, every public-coin internal simulation of π_0 with near-optimal information complexity of $O(k)$ must communicate at least $2^{2^{2^{\Omega(k)}}}$ bits.

1.2.5 Discussion of the proof of Theorem 1.8

Compression to internal entropy, as mentioned above, must be done in an internal fashion. Namely, an observer of the conversation (who does not know the inputs nor the private randomness) should not be able to make much sense of it.

The only two compression schemes with this property that were previously known are from [2, 3]. The scheme from [3] is not efficient in terms of information complexity so we do not discuss it in detail here. In the scheme from [2] the players privately sample a candidate transcript, and they communicate to fix errors. After an error is located, the candidate transcript is modified until converging to the correct transcript. The errors are fixed using the protocol of Feige et al. for finding the first difference, and each error fixing costs about $\log(CC(\pi))$ bits.

The main problem in analyzing their protocol is bounding the number of errors in terms of the internal information. They are able to do so but the cost is quite high and the overall upper bound on the number of errors they show is order[3] $\sqrt{CC(\pi)I_\mu^{int}(\pi)}$. The authors of [8, 24] showed that for a deterministic protocol π, the expected number of errors in this scheme decreases[4] to roughly $I^{int}(\pi)$ which sums up to total communication of order $I^{int}(\pi)\log(CC(\pi))$ bits.

It is natural to consider a slight variation of this scheme in which the errors are fixed using our protocol from Theorem 1.7, instead of the protocol of Feige et al. However, it is not clear that this modified scheme yields the desired result. On a high level, this is because it may be the case that the additional information that is revealed from correcting the mistakes is large, and we do not know how to bound it by the internal information of the simulated protocol.

Our approach is different and starts with the compression of deterministic protocols to external entropy of [11]. The main idea there is that a deterministic protocol induces a distribution on the leaves of the protocol tree, and that there is always a vertex u in the tree with probability mass roughly 1/2 (Lemma 2.1 below). Both players know u and they can check if the rectangle[5] it defines contains x and y with 2 bits of communication. It can easily be shown that by doing so they (roughly) learn one bit of information. This yields an optimal but external compression (an observer knows u as well).

In the internal case, there is no single node that is good for both players. Alice knows a node v_a and Bob a node v_b, which are in general arbitrary nodes in the protocol tree. The crux of our protocol is an efficient way for Alice and Bob to learn enough about v_a, v_b so that at least one of them obtains one bit of information. We show that using Theorem 1.6 one of them, say Alice, can identify a good vertex u to focus on (roughly, u is somewhere in between v_a, v_b). Using Lemma 1.1 Alice then tries to internally transmit u to Bob. If this transmission succeeds, then, with high probability, Bob learns one bit of information, and if this transmission fails then, with high probability, Alice learns one bit of information. The transmission is indeed internal in that an external observer does not in general learn u even when Bob does. The full protocol appears in Section 4.

[3] On a high level this cost occurs for the following reason: if we denote by $h(p)$ the entropy of a random bit with bias $p \in [0,1]$, then $h(\frac{1}{2}+\delta) - h(\frac{1}{2})$ is of order δ^2. The second power of δ yields the square root $CC(\pi)$ in the analysis.

[4] The improvement comes from that $h(\delta) - h(0)$ is of order δ.

[5] The set of inputs that reach u is a rectangle, that is, it is of the form $\mathcal{X}' \times \mathcal{Y}' \subset \mathcal{X} \times \mathcal{Y}$.

2 Definitions and preliminaries

Logarithms in this text are to the base two. We provide the basic definitions needed for this text. For background and more details on information theory see the book [10] and on communication complexity see the book [19].

2.1 Information theory

The entropy of a random variable X taking values in U is defined as

$$H(X) = \sum_{u \in U} \Pr[X = u] \log(1/\Pr[X = u]).$$

The entropy of X conditioned on Y is defined as $H(X|Y) = H(X,Y) - H(Y)$. The mutual information between X, Y conditioned on Z is defined as $I(X;Y|Z) = H(X|Z) - H(X|Y,Z)$.

2.2 Protocols

A deterministic communication protocol π with inputs from $\mathcal{X} \times \mathcal{Y}$ is a rooted directed binary tree with the following structure. Edges are directed from root to leaves. Each internal node in the protocol is owned by either Alice or Bob. For every $x \in \mathcal{X}$, each internal node v owned by Alice is associated with an edge $e_{v,x}$ from v to one of the children of v. Similarly, for every $y \in \mathcal{Y}$, each internal node v owned by Bob is associated with an edge $e_{v,y}$. On input x, y, a protocol π is executed by starting at the root and following the unique path defined by $e_{v,x}, e_{v,y}$ until reaching a leaf. We denote by $T_\pi = T_\pi(x,y)$ the leaf reached, which we also call the transcript of π with input (x,y). The length of a transcript, denoted $|T_\pi|$, is the depth of the corresponding leaf.

In a public-coin protocol, Alice and Bob also have access to public randomness r that they both know. In a private-coin protocol, Alice has access to a random string r_a, and Bob has access to a random string r_b. A general protocol is a protocol which uses both public and private coins. The four random variables $(x,y), r, r_a, r_b$ are always assumed independent.

The communication complexity of a deterministic π, denoted by $\mathsf{CC}(\pi)$, is the maximum length of a transcript. For general protocols, $\mathsf{CC}(\pi)$ is defined as the maximum communication complexity over all randomness as well (i.e. over x, y, r, r_a, r_b), and $\mathsf{CC}_\mu^{avg}(\pi)$ is the expected length of a transcript over all randomness.

2.3 Computation

A deterministic protocol π externally computes a function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ if there is a map M so that $f(x,y) = M(T_\pi(x,y))$ for all x, y. A deterministic protocol π internally computes a function f if there are two maps M_a, M_b so that $M_a(x, T_\pi(x,y)) = M_b(y, T_\pi(x,y)) = f(x,y)$ for all x, y. In the randomized setting, M may depends on r; M_a may depend on r, r_a; M_b may depend on r, r_b; and the equalities should hold with probability at least $1 - \varepsilon$ over the distribution of r, r_a, r_b for all x, y. In the distributional setting, the probability is taken over x, y as well.

2.4 Information and entropy of protocols

For a distribution μ on the inputs, define

$$I_\mu^{int}(\pi) = I(T_\pi; X|Y, R, R_b) + I(T_\pi; Y|X, R, R_a)$$

and

$$I_\mu^{ext}(\pi) = I(T_\pi; X, Y|R).$$

Similarly, define

$$H_\mu^{int}(\pi) = H(T_\pi|Y, R, R_b) + H(T_\pi|X, R, R_a)$$

and

$$H_\mu^{ext}(\pi) = H(T_\pi|R).$$

Note that each of these measures also induce a corresponding complexity measure for functions/relations in the standard way.

2.5 Simulation

Let π, σ be protocols, let μ be a distribution on the input space $\mathcal{X} \times \mathcal{Y}$ and let $\varepsilon \geq 0$.

Our goal is defining when σ simulates π with error ε in the distributional setting[6]. Namely, probabilities are taken over all randomness of inputs as well as private and public coins.

The randomness in σ is s, s_a, s_b, and the randomness in π is r, r_a, r_b. We say that σ externally simulates π with error ε if there exists a function $M = M(T_\sigma, s)$ so that the distribution of $(x, y, (T_\pi, r))$ is ε-close in L_1 distance to the distribution of $(x, y, M(T_\sigma, s))$.

We say that σ internally simulates π with error ε if there exist functions $M_a = M_a(T_\sigma, x, s_a, s)$ and $M_b = M_b(T_\sigma, y, s_b, s)$ so that the distribution of $(x, y, (T_\pi, r, r_a))$ is ε-close in L_1 distance to the distribution of (x, y, M_a), and the distribution of $(x, y, (T_\pi, r, r_b))$ is ε-close in L_1 distance to the distribution of (x, y, M_b).

The simulation we present in the proof of Theorem 1.8 is in fact of a stronger form. In the beginning of σ, Alice and Bob interpret the public randomness as a pair $s = (r, s')$ and their private randomness as $s_a = (r_a, s'_a)$ and $s_b = (r_b, s'_b)$. They think of r, r_a, r_b as the fixed randomness of π, and communicate in order to privately compute the fixed transcript $T_\pi = T_\pi(x, y, r, r_a, r_b)$, with error probability (over the remaining randomness s', s'_a, s'_b) of at most ε.

This stronger type of simulation is sometimes too strong to be useful, as the following example demonstrates. Consider a protocol in which x, r_a are uniform in $\{0,1\}^n$, and Alice just sends $x + r_a \in \{0,1\}^n$ to Bob. The transcript of this protocol is just a random noise, and its external information is indeed zero. It can, indeed, be externally simulated without error by a protocol with zero communication; interpret s as a uniform element in $\{0,1\}^n$ and set $M(\emptyset, s) = (s, \emptyset)$. However, every strong simulation of this protocol (as the one in Theorem 1.8 mentioned above) must communicate many bits. Indeed, the transcript of a strong simulation must reveal the value of $x + r_a$ to Bob, with high probability. This stronger type of simulation corresponds to internal entropy rather than internal information. In the example above, the internal information is 0 but the internal entropy is n.

2.6 Balanced nodes in trees

We use the following well known lemma (see e.g. [19]).

▶ **Lemma 2.1.** *Let μ be a probability measure on the leaves of a rooted binary tree. The distribution μ may be extended to a function on all nodes in the tree by setting $\mu(v)$ to be the μ-probability that a leaf that is a successor of v is chosen. Then, there exists a node u such that either u is a leaf and $\mu(u) \geq 2/3$, or $1/3 \leq \mu(u) \leq 2/3$.*

[6] There is also a natural variant of this definition in the randomized setting but it is not relevant for this text.

3 Finding the first difference

Proof of Theorem 1.7. Denote by $\mathcal{E}$ the (event) set of inputs (x, y) so that

$$\mu(i|x) \cdot \mu(i|y) < 2^{-2h^{int}/\varepsilon}.$$

By Markov's inequality,

$$\mu(\mathcal{E}) < \varepsilon/2.$$

For inputs in $\mathcal{E}$, the protocol may fail. For the rest of the proof, fix $(x, y) \notin \mathcal{E}$ and set $i = i(x, y)$.

The protocol proceeds in iterations indexed by $t \in \mathbb{N}$. For every t, Alice knows a distribution α_t on $[n] \cup \{\text{“}equal''\}$ and Bob a distribution β_t on $[n] \cup \{\text{“}equal''\}$ where we use the order $1 < 2 < \ldots < n < \text{“}equal''$. It may help to think of the distributions α_t and β_t as representing Alice and Bob's opinions for what is the first difference, given what they have learned upto iteration t. They start with

$$\alpha_0(j) = \Pr_\mu[i = j|x] \text{ and } \beta_0(j) = \Pr_\mu[i = j|y], \text{ for all } j.$$

Iteration t starts with Alice knowing α_t and Bob knowing β_t, and ends with an update of these distributions to $\alpha_{t+1}, \beta_{t+1}$. There are $O(h^{int}/\varepsilon)$ iterations, and the probability of failure in each iteration is at most δ for $\delta = c\varepsilon^2/h^{int}$ for a small constant $c > 0$. The union bound implies that the overall error is at most ε.

The goal of every iteration is, given α_t, β_t, to construct with probability at least $1 - O(\delta)$ distributions $\alpha_{t+1}, \beta_{t+1}$ so that (if they did not stop)

$$\alpha_{t+1}(i) \geq \alpha_t(i) \ , \quad \beta_{t+1}(i) \geq \beta_t(i)$$

and

$$\alpha_{t+1}(i) \cdot \beta_{t+1}(i) \geq \frac{3}{2} \cdot \alpha_t(i) \cdot \beta_t(i).$$

This immediately implies that the number of iterations is at most $O\left(\log\left(\frac{1}{\mu(i|x)\cdot\mu(i|y)}\right)\right) = O(h^{int}/\varepsilon)$ since we conditioned on not $\mathcal{E}$ and since α_t, β_t are probability distributions so their maximum value is at most 1.

The protocol uses the following subroutine we call *check*(j) with error δ. It gets as input $j \in [n] \cup \{\text{“}equal''\}$ and with communication $O(\log(1/\delta))$ it externally outputs “yes” if $j = i$ and “no” if $j \neq i$. This subroutine just uses public randomness[7] to check if $x_{<j} = y_{<j}$ and $x_j \neq y_j$ for $j \in [n]$ or if $x = y$ for $j = \text{“}equal''$.

Iteration t is performed as follows:

1. Let d_a be the maximum integer so that $\alpha_t(\{1, 2, \ldots, d_a - 1\}) < 2/3$ and let d_b be the maximum integer so that $\beta_t(\{1, 2, \ldots, d_b - 1\}) < 2/3$. Alice knows d_a and Bob d_b. Using the protocol from Theorem 1.6, with communication $O(\log(\log(n)/\delta))$ the players find[8] d that is between d_a, d_b with error at most δ.
2. If $\alpha_t(d) > 1/3$ then the players check(d) with error δ. If the answer is “yes” then they stop and output d.
   If the answer is “no” then they update α_t, β_t to $\alpha_{t+1}, \beta_{t+1}$ by conditioning on the event $([n] \cup \text{“}equal'') \setminus \{d\}$ and continue to the next iteration.

[7] For example, using the standard randomized protocol for equality [19].
[8] If we represent d_a, d_b as binary strings of length order $\log(n)$ then to find d it suffices to find the first index in which d_a, d_b differ.

3. If $\beta_t(d) > 1/3$ then the players check(d) with error δ. If the answer is "yes" then they stop and output d. If the answer is "no" then they update α_t, β_t to $\alpha_{t+1}, \beta_{t+1}$ by conditioning on the event $([n] \cup$ "$equal''$$) \setminus \{d\}$ and continue to the next iteration.
4. The players check using public randomness with error δ if $x_{<d} = y_{<d}$.
If the answer is "yes" then they update α_t, β_t to $\alpha_{t+1}, \beta_{t+1}$ by conditioning on the event $\{d, d+1, \ldots, n\} \cup \{$"$equal''$$\}$ and continue to the next iteration.
If the answer is "no" then they update α_t, β_t to $\alpha_{t+1}, \beta_{t+1}$ by conditioning on the event $\{1, 2, \ldots, d-1\}$ and continue to the next iteration.

We analyse the correctness step by step assuming that no error occurred (we have already bounded the probability of error):

1. The players found d that is between d_a, d_b.
2. If $\alpha_t(d) > 1/3$ and the players output d then indeed the output is correct. If $\alpha_t(d) > 1/3$ and the players do not output d then $d \neq i$ which means that
$$\alpha_{t+1}(i) = \frac{\alpha_t(i)}{1 - \alpha_t(d)} > \frac{\alpha_t(i)}{2/3}$$
and $\beta_{t+1}(i) \geq \beta_t(i)$.
3. As in previous case.
4. If the players reached here then $\alpha_t(d), \beta_t(d) \leq 1/3$. Assume without loss of generality that $d_a \leq d_b$. The proof in the other case is similar.
If $x_{<d} = y_{<d}$ then $i \geq d \geq d_a$. This implies that $\beta_{t+1}(i) \geq \beta_t(i)$. By choice,
$$\alpha_t(\{d, d+1, \ldots, n\}) = \alpha_t(d) + 1 - \alpha_t(\{1, \ldots, d\}) \leq \frac{1}{3} + \frac{1}{3} \leq \frac{2}{3},$$
which implies $\alpha_{t+1}(i) \geq 3\alpha_t(i)/2$.
If $x_{<d} \neq y_{<d}$ then $i < d \leq d_b$. This implies that $\alpha_{t+1}(i) \geq \alpha_t(i)$. By choice,
$$\beta_t(\{1, 2, \ldots, d-1\}) \leq \frac{2}{3},$$
which implies $\beta_{t+1}(i) \geq 3\beta_t(i)/2$. ◀

4 Internal compression

Proof of Theorem 1.8. Let x, y be the inputs to π, let r be the public randomness, and let r_a, r_b be the private randomness. The first observation is that
$$H^{int} = H^{int}_\mu(\pi) = \mathbb{E}_{x,y,r,r_a,r_b} \log\left(\frac{1}{\mu(T_\pi | x, r, r_a) \cdot \mu(T_\pi | y, r, r_b)}\right),$$
where here $T_\pi = T_\pi(x, y, r, r_a, r_b)$. Denote by $\mathcal{E}$ the event (i.e. set of (x, y, r, r_a, r_b)) that
$$\mu(T_\pi | x, r, r_a) \cdot \mu(T_\pi | y, r, r_b) < 2^{-2H^{int}/\varepsilon}.$$
By Markov's inequality,
$$\Pr(\mathcal{E}) < \varepsilon/2.$$
When $\mathcal{E}$ occurs, the protocol σ may fail. For the rest of the proof, fix $(x, y, r, r_a, r_b) \notin \mathcal{E}$ and set $T_\pi = T_\pi(x, y, r, r_a, r_b)$.

The protocol σ proceeds in iterations indexed by $t \in \mathbb{N}$. The starting point of every iteration is a distribution α_t on leaves of π that Alice knows and a distribution β_t on the leaves of π that Bob knows. These distributions reflect the current perspective of the players after the communication so far. The first distributions are

$$\alpha_0(v) = \Pr[v|x, r, r_a] \text{ and } \beta_0(v) = \Pr[v|y, r, r_b]$$

for all leaves v of the protocol tree (the probability in α_0 for example is over Bob's randomness). The goal of every iteration is to construct with probability at least $1-\delta$ distributions $\alpha_{t+1}, \beta_{t+1}$ so that

$$\alpha_{t+1}(T_\pi) \geq \alpha_t(T_\pi) \ , \quad \beta_{t+1}(T_\pi) \geq \beta_t(T_\pi)$$

and

$$\alpha_{t+1}(T_\pi) \cdot \beta_{t+1}(T_\pi) \geq \frac{3}{2} \cdot \alpha_t(T_\pi) \cdot \beta_t(T_\pi).$$

The number of iterations is set to be at most

$$O(\log(2^{2H^{int}/\varepsilon})) = O(H^{int}/\varepsilon),$$

and the communication complexity of each iteration is at most

$$O\left(\log\left(\frac{\log(CC(\pi))}{\delta}\right) + \frac{H^{int}}{\varepsilon} + \log(1/\delta)\right).$$

Thus, setting $\delta = c\varepsilon^2/H^{int}$ for some small constant $c > 0$, the union bound implies the overall bound on the error.

Here is how iteration t is performed:

1. Alice finds a vertex v_a promised by Lemma 2.1 with α_t, and Bob finds v_b promised by Lemma 2.1 with β_t. Denote $d_a = depth(v_a)$ and $d_b = depth(v_b)$.
2. Using the protocol from Lemma 1.6, with communication $O(\log(\log(CC(\pi))/\delta))$ the players find[9] d that is between d_a, d_b with error $\delta/2$.
3. If $d_a \geq d_b$, the players do the following: Let u be the ancestor of v_a at depth d and let U be the set of nodes of depth d of π . Using the protocol from Lemma 1.1 Alice sends u to Bob. They use this protocol with error parameter $\delta/2$, where Alice's input is u and Bob's input is the distribution β_t induced on U.
   If this stage takes more than $O((H^{int}/\varepsilon) + \log(1/\delta))$ bits, then the players abort.
   At the end of this stage, either Bob thinks[10] he knows u as well or they have aborted.
   - If Bob thinks he knows u there are two options:
     If u is a leaf then the players stop and internally output u.
     Otherwise, the players set $\alpha_{t+1} = \alpha_t$ and β_{t+1} to be the distribution β_t conditioned on passing through u.
   - Otherwise, the players aborted and they set $\beta_{t+1} = \beta_t$ and α_{t+1} to be the distribution α_t conditioned on not passing through u.
4. When $d_a < d_b$, the players exchange roles.

We now analyse the performance in iteration t. For this, we assume that no error occurred. That is, that the protocols from Theorem 1.6 and Lemma 1.1 gave the desired result (this happens with probability at least $1 - \delta$). The analysis follows the outline of the protocol:

[9] Represent d_a, d_b as binary strings of length roughly $\log(CC)$.
[10] There is some small probability that Bob holds some $u' \neq u$ but he still thinks he knows u.

1. Lemma 2.1 says that there are always such nodes v_a, v_b.
2. the players find d that is between d_a, d_b.
3. We distinguish between two cases:

   Bob thinks he knows u: This means that $\beta_t(u) > 0$ and so (y, r_b) is in the rectangle defined by u. Thus, $((x, r_a), (y, r_b))$ is in the rectangle defined[11] by u, which implies that T_π is a successor of u.
   If u is a leaf then indeed $T_\pi = u$.
   Otherwise, there are two cases:
   The first is when v_b is an ancestor of u. In this case, v_b is not a leaf, $\beta_t(v_b) \geq \beta_t(u)$ and
   $$\beta_{t+1}(T_\pi) = \frac{\beta_t(T_\pi)}{\beta_t(u)} \geq \frac{\beta_t(T_\pi)}{\beta_t(v_b)} \geq \frac{\beta_t(T_\pi)}{2/3}.$$
   The second is when v_b is not an ancestor of u. In this case, $\beta_t(u) \leq 1 - \beta_t(v_b) \leq 2/3$ and
   $$\beta_{t+1}(T_\pi) = \frac{\beta_t(T_\pi)}{\beta_t(u)} \geq \frac{\beta_t(T_\pi)}{2/3}.$$

   Bob does not think he knows u: Since we assumed $\mathcal{E}$ does not occur, if u is an ancestor of T_π then
   $$\beta_t(u) \geq \beta_t(T_\pi) \geq \beta_0(T_\pi) \geq 2^{-2H^{int}/\varepsilon}.$$
   Since the players aborted (we ignore possibility of error), this means that u is not an ancestor of T_π. Since u is an ancestor of v_a, $\alpha_t(u) \geq \alpha_t(v_a) \geq 1/3$. Thus, by choice,
   $$\alpha_{t+1}(T_\pi) = \frac{\alpha_t(T_\pi)}{1 - \alpha_t(u)} \geq \frac{\alpha_t(T_\pi)}{1 - \alpha_t(v_a)} \geq \frac{\alpha_t(T_\pi)}{2/3}.$$

4. When $d_a < d_b$, the proof is similar. ◀

Acknowledgments. We thank Anup Rao for helpful conversations. We also thank Makrand Sinha and anonymous referees for comments on an earlier version of this text.

References

1 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.

2 Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. *SIAM J. Comput.*, 42(3):1327–1363, 2013.

3 Mark Braverman. Interactive information complexity. In *STOC*, pages 505–524, 2012.

4 Mark Braverman and Ankit Garg. Public vs private coin in bounded-round information. In *ICALP (1)*, pages 502–513, 2014.

5 Mark Braverman and Anup Rao. Information equals amortized communication. In *FOCS*, pages 748–757, 2011.

6 Mark Braverman, Anup Rao, Omri Weinstein, and Amir Yehudayoff. Direct product via round-preserving compression. In *ICALP (1)*, pages 232–243, 2013.

[11] For any fixed public string r, the set of all $(x, r_a), (y, r_b)$ for which $T_\pi(x, r_a, y, r_b)$ is a descendent of u with a positive probability is a rectangle.

7 Mark Braverman, Anup Rao, Omri Weinstein, and Amir Yehudayoff. Direct products in communication complexity. In *FOCS*, pages 746–755, 2013.

8 Joshua Brody, Harry Buhrman, Michal Koucký, Bruno Loff, Florian Speelman, and Nikolay K. Vereshchagin. Towards a reverse newman's theorem in interactive information complexity. In *IEEE Conference on Computational Complexity*, pages 24–33, 2013.

9 Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, , and Andrew Yao. Informational complexity and the direct sum problem for simultaneous message complexity. *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 270–278, 2001.

10 Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley Interscience, 2006.

11 Martin Dietzfelbinger and Henning Wunderlich. A characterization of average case communication complexity. *Inf. Process. Lett.*, 101(6):245–249, 2007.

12 R. M. Fano. The transmission of information. Technical Report 65, Research Laboratory for Electronics, MIT, Cambridge, MA, USA, 1949.

13 Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994.

14 Abbas El Gamal and Alon Orlitsky. Interactive data compression. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 0:100–108, 1984.

15 Anat Ganor, Gillat Kol, and Ran Raz. Exponential separation of information and communication. *Electronic Colloquium on Computational Complexity*, 2014.

16 Prahladh Harsha, Rahul Jain, David A. McAllester, and Jaikumar Radhakrishnan. The communication complexity of correlation. *IEEE Transactions on Information Theory*, 56(1):438–449, 2010.

17 David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, September 1952.

18 Gillat Kol, Shay Moran, Amir Shpilka, and Amir Yehudayoff. Direct sum fails for zero error average communication. In *ITCS*, pages 517–522, 2014.

19 Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, 1997.

20 Alon Orlitsky Moni Naor and Peter Shor. Three results on interactive communication. *Information Theory, IEEE Transactions on*, 39(5):1608–1615, Sep 1993.

21 Ilan Newman. Private vs. common random bits in communication complexity. *Inf. Process. Lett.*, 39(2):67–71, 1991.

22 Alon Orlitsky. Average-case interactive communication. *Information Theory, IEEE Transactions on*, 38(5):1534–1547, Sep 1992.

23 Alon Orlitsky and Abbas El Gamal. Average and randomized communication complexity. *IEEE Transactions on Information Theory*, 36(1):3–16, 1990.

24 Denis Pankratov. *Direct sum questions in classical communication complexity.* PhD thesis, University of Chicago, 2012.

25 C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

26 David Slepian and Rahul Jack K. Wolf. Noiseless coding of correlate information sources. *IEEE Transactions on Information Theory*, 19(4), July 1973.

27 Emanuele Viola. The communication complexity of addition. In *SODA*, pages 632–651, 2013.

28 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *STOC*, pages 209–213, 1979.

A Appendix

A.1 Internal deterministic computation is also external

Proof of Proposition 1.2. There are maps M_a, M_b so that for all x, y,

$$M_a(x, T_\pi(x,y)) = M_b(y, T_\pi(x,y)) = f(x,y).$$

Fix some rectangle $\rho = \{(x,y) : T_\pi(x,y) = T_\pi(x_0,y_0)\}$. For every $(x,y) \in \rho$, we know $M_a(x,\rho) = f(x,y) = f(x,y_0)$, and similarly $M_b(y_0,\rho) = f(x,y_0) = f(x_0,y_0)$. Therefore, f is constant on ρ and we can define $M(\rho) = f(x_0,y_0)$. ◀

A.2 Information lower bounds errorless simulation

Proof of Claim 1.3. The external case: Let σ be a protocol that externally simulated π without error. By definition of simulation, there exists a function M so that for all (x,y) so that $\mu(x,y) > 0$, it holds that $p_\sigma = p_\pi$, where p_σ is the distribution of $M(T_\sigma, s)$ and p_π is that of (T_π, r). Thus,

$$\begin{aligned}
\mathsf{CC}_\mu^{avg}(\sigma) &\geq I_\mu^{ext}(\sigma) && \text{(see e.g. [18])}\\
&= I(T_\sigma; X, Y|S)\\
&= I(T_\sigma, S; X, Y) && (S \text{ is independent of } (X,Y))\\
&\geq I(M(T_\sigma, S); X, Y) && \text{(data processing inequality)}\\
&= I(T_\pi, R; X, Y) && \text{(errorless simulation)}\\
&= I(T_\pi; X, Y|R) && (R \text{ is independent of } (X,Y))\\
&= I_\mu^{ext}(\pi).
\end{aligned}$$

The internal case: similarly to the external case,

$$\begin{aligned}
\mathsf{CC}_\mu^{avg}(\sigma) &\geq I_\mu^{ext}(\sigma)\\
&\geq I_\mu^{int}(\sigma)\\
&= I(T_\sigma, S, S_b; X|Y) + I(T_\sigma, S, S_a; Y|X)\\
&\geq I(M_b(T_\sigma, Y, S, S_b); X|Y) + I(M_a(T_\sigma, X, S, S_a); Y|X)\\
&= I(T_\pi, R, R_b; X|Y) + I(T_\pi, R, R_a; Y|X)\\
&= I_\mu^{int}(\pi).
\end{aligned}$$

◀

A.3 Transmission

Proof of Lemma 1.1. The players interpret the public randomness as boolean random hash functions on U. The protocol proceeds in iterations indexed by $t \in \mathbb{N}$. In iteration $t = 0$, the following is performed:

1. Alice sends $k = \lceil \log(1/\varepsilon) \rceil + 2$ hash values of u_a to Bob.
2. Bob computes the set
$$S_0 = \{u \in U : \mu(u) \in (1/2, 1]\}.$$
He compares every element of S_0 to the k hash values he received. He deletes every $s \in S_0$ that does not agree with at least one of these k hash values. Denote by S_0' the set S_0 after this deletion.

If S'_0 is empty, he sends a "0" to Alice.
If S'_0 is not empty, he sets u_b as an arbitrary element S'_0, and sends "1" to Alice, and the players stop.

For every $t = 1, 2, \ldots$, the following is performed (until the players stop):

1. Alice sends 2 new hash values of u_a to Bob.
2. Bob computes the set
$$S_t = \{u \in U : \mu(u) \in (2^{-t-1}, 2^{-t}]\}.$$
He compares every element of S_t to the $k + 2t$ hash values he received so far. He deletes every $s \in S_t$ that does not agree with at least one of these hash values. Denote by S'_t the set S_t after this deletion.
If S'_t is empty, he sends a "0" to Alice.
If S'_t is not empty, he sets u_b as an arbitrary element in S'_t, and sends "1" to Alice, and the players stop.

We now analyse the protocol. Let t_0 be so that $u_a \in S_{t_0}$. First, the protocol stops after at most $t_0 \leq \log(1/\mu(u_a)) + 1$ iterations, because u_a agrees with all hash values sent. Second, for every t, by the union bound,

$$\Pr[S'_t \neq \{u_a\} \cap S_t] \leq 2^{-(k+2t)} 2^{t+1} \leq 2^{-\log(1/\varepsilon)-t-1} = \frac{\varepsilon}{2^{t+1}}.$$

Thus, by the union bound, the probability that either there is some $t < t_0$ for which $S'_t \neq \emptyset$ or $S'_{t_0} \neq \{u_a\}$ is at most $\sum_{t=0}^{\infty} \varepsilon/2^{t+1} \leq \varepsilon$. ◀

On Fortification of Projection Games

Amey Bhangale*[1], Ramprasad Saptharishi†[2], Girish Varma‡[3], and Rakesh Venkat§[3]

1 Rutgers University
New Brunswick, USA
amey.bhangale@rutgers.edu

2 Tel Aviv University
Tel Aviv, Israel
ramprasad@cmi.ac.in

3 Tata Institute of Fundamental Research
Mumbai, India
{girishrv,rakesh}@tifr.res.in

Abstract

A recent result of Moshkovitz [10] presented an ingenious method to provide a completely elementary proof of the *Parallel Repetition Theorem* for certain projection games via a construction called *fortification*. However, the construction used in [10] to fortify arbitrary label cover instances using an arbitrary extractor is insufficient to prove parallel repetition. In this paper, we provide a fix by using a stronger graph that we call *fortifiers*. Fortifiers are graphs that have both ℓ_1 and ℓ_2 guarantees on induced distributions from large subsets.

We then show that an expander with sufficient spectral gap, or a bi-regular extractor with stronger parameters (the latter is also the construction used in an independent update [11] of [10] with an alternate argument), is a good fortifier. We also show that using a fortifier (in particular ℓ_2 guarantees) is necessary for obtaining the robustness required for fortification.

1998 ACM Subject Classification F.2.2 Computations on discrete structures

Keywords and phrases Parallel Repetition, Fortification

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.497

1 Introduction

Label-cover and general two-prover games

A label cover instance is specified by a bipartite graph $G = ((X, Y), E)$, a pair of alphabets Σ_X and Σ_Y and a set of constraints $\psi_e : \Sigma_X \to \Sigma_Y$ on each edge $e \in E$. The goal is to label the vertices of X and Y using labels from Σ_X and Σ_Y so as to satisfy as many constraints are possible.

This problem is often viewed as a two-prover game. The verifier picks an edge (x, y) at random and sends x to the first prover and y to the second prover. They are to return a label of the vertex that they received, and the verifier accepts if the labels they returned are consistent with the constraint $\psi_{(x,y)}$. The value of this game G, denoted by $\mathrm{val}(G)$, is given

* Research supported by the NSF grant CCF-1253886.
† The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.
‡ Supported by the Google India Fellowship in Algorithms.
§ Research supported in part by ISF-UGC grant 1399/4.

by the acceptance probability of the verifier maximized over all possible strategies of the provers. These are also called *projection games* as the constraints are functions from Σ_X to Σ_Y. They are called *general games* if the constraint on each edge is an arbitrary relation $\psi_{(x,y)} \subseteq \Sigma_X \times \Sigma_Y$.

These two notions are equivalent in the sense that val(G) is exactly equal to the maximum fraction of constraints that can be satisfied by any labelling.

This problem is central to the PCP Theorem [2, 1] and almost all inapproximability results that stem from it. The (Strong) PCP Theorem can be rephrased as stating that for every $\varepsilon > 0$, it is NP-hard to distinguish whether a given label cover instance has $\text{val}(G) = 1$ or $\text{val}(G) < \varepsilon$. An important step is a way to transform instances with $\text{val}(G) < 1 - \varepsilon$ to instances G' with $\text{val}(G') < \varepsilon$. This is usually achieved via the *Parallel Repetition Theorem.*

Parallel Repetition

The k-fold repetition of a game G, denoted by G^k, is the following natural definition – the verifier picks k edges $(x_1, y_1), \cdots, (x_k, y_k)$ from E uniformly and independently, sends $(x_1, \ldots, x_k)$ and $(y_1, \ldots, y_k)$ to the provers respectively, and accepts if the labels returned by them are consistent on each of the k edges.

If $\text{val}(G) = 1$ to start with then $\text{val}(G^k)$ still remains 1. How does $\text{val}(G^k)$ decay with k if $\text{val}(G) < 1$? Turns out even this simple operation of repeating a game in parallel has a counter-intuitive effect on the value of the game. It is easy to see that $\text{val}(G^k) \geq \text{val}(G)^k$ as provers can use a same strategy as in G to answer each query (x_i, y_i). The first surprise is $\text{val}(G^k)$ is *not* $\text{val}(G)^k$, but sometimes can be *much larger* than $\text{val}(G)^k$. Fortnow [8] presented a game G for which $\text{val}(G^2) > \text{val}(G)^2$, Feige [6] improved this by giving an example of game G with $\text{val}(G) < 1$ but $\text{val}(G^2) = \text{val}(G)$. Indeed, there are known examples [15] of projection games where $\text{val}(G) = (1 - \varepsilon)$ but $\text{val}(G^k) \geq \left(1 - \varepsilon\sqrt{k}\right)$ for a large range of k.

The first non trivial upper bound on $\text{val}(G^k)$ was proven by Verbitsky [17] who showed that if $\text{val}(G) < 1$ then the value $\text{val}(G^k)$ must go to zero as k goes to infinity. It is indeed true that $\text{val}(G^k)$ decays exponentially with k (if $\text{val}(G) < 1$). This breakthrough was first proved by Raz [14], and has subsequently seen various simplifications and improvements in parameters [9, 13, 5, 4]. The following statements are due to Holenstein [9], Dinur and Steurer [5] respectively.

▶ **Theorem 1.1** (Parallel repetition theorem for general games). *Suppose G is a projection game such that* $\text{val}(G) \leq 1 - \varepsilon$ *and let* $|\Sigma_X|\,|\Sigma_Y| \leq s$. *Then, for any* $k \geq 0$*,*

$$\text{val}(G^k) \quad \leq \quad \left(1 - \varepsilon^3/2\right)^{\Omega(k/\log s)}.$$

▶ **Theorem 1.2** (Parallel repetition theorem for projection games). *Suppose G is a projection game such that* $\text{val}(G) \leq \rho$. *Then, for any* $k \geq 0$*,*

$$\text{val}(G^k) \quad \leq \quad \left(\frac{2\sqrt{\rho}}{1+\rho}\right)^{k/2}.$$

Although a lot of these results are substantial simplifications of earlier proofs, they continue to be involved and delicate. Arguably, one might still hesitate to call them *elementary* proofs.

Recently, Moshkovitz [10] came up with an ingenious method to prove a parallel repetition theorem for certain projection games by slightly modifying the underlying game via a process that she called *fortification*. The method of fortification suggested in [10] was a rather mild change to the underlying game and proving parallel repetition for such *fortified projection games* was sufficient for most applications. The advantage of fortification was that parallel repetition theorem for fortified games had a simple, elementary and elegant proof as seen in [10].

1.1 Fortified games

Fortified games will be described more formally in Section 2, but we give a very rough overview here. Moshkovitz showed that there is an easy way to bound the value of repeated game if we knew that the game was *robust on large rectangles*. We shall first need the notion of *symmetrized projection games*.

Symmetrized Projection games. Given a projection game G on $((X,Y),E)$, the symmetrized game G_{sym} is a game on the (multi)graph $((X,X),E')$ such that, there is an edge $(x,x') \in E'$, for every $y \in Y$ with $(x,y),(x',y) \in E$, with the constraint $\pi_{(x,y)}(\sigma_x) = \pi_{(x',y)}(\sigma_{x'})$.

For projection games, it would be more convenient to work with the above symmetrized version for reasons that shall be explained shortly. It is not hard to see that $\text{val}(G)$ and $\text{val}(G_{\text{sym}})$ are within a quadratic factor of each other. Thus for projection games, we shall work with the game G_{sym} instead of the original game G.

▶ **Definition 1.3** ((δ,ε)-robust games)**.** Let G be a two-prover game on $((X,X),E)$. For any pair of sets $S,T \subseteq X$, let $G_{S\times T}$ be the game where the verifier chooses his random query $(x,x') \in E$ conditioned on the event that $x \in S$ and $x' \in T$.

G is said to be (δ,ε)-*robust* if for every $S,T \subseteq X$ with $|S|,|T| \geq \delta|X|$, we have that

$$\text{val}(G_{S\times T}) \quad \leq \quad \text{val}(G) + \varepsilon.$$

▶ **Theorem 1.4** (Parallel repetition for robust projection games [10])**.** *Let G be a projection game on a bi-regular bipartite graph $((X,Y),E)$ with alphabets Σ_X and Σ_Y. For any positive integer k, if $\varepsilon_1,\varepsilon_2,\delta > 0$ are parameters such that $2\delta|\Sigma_Y|^{k-1} \leq \varepsilon_1$ and G_{sym} is (δ,ε_2)-robust, then*[1]

$$\text{val}(G_{sym}^k) \quad \leq \quad (\text{val}(G_{sym}) + \varepsilon_2)^k + k\varepsilon_1.$$

Not all projection games are robust on large rectangles, but Moshkovitz suggested a neat way of slightly modifying a projection game and making it robust. This process was called *fortification*.

On a high level, for any two-prover game, the verifier chooses to verify a constraint corresponding to an edge (x,y) but is instead going to sample several other dummy vertices and give the provers two sets of D vertices $\{x_1,\dots,x_D\}$ and $\{y_1,\dots,y_D\}$ such that $x = x_i$ and $y = y_j$ for some i and j. The provers are expected to return labels of all D vertices sent to them but the verifier checks consistency on just the edge (x,y). This is very similar to the "confuse/match" perspective of Feige and Kilian [7].

[1] The following is the corrected statement from [11].

To derandomize this construction, Moshkovitz [10] uses a pseudo-random bipartite graph where given a vertex w, the provers are expected to return labels of all its neighbours (Definition 2.1). The most natural candidate of such a pseudo-random graph is an (δ, ε)-extractor, as we really want to ensure that conditioned on "large enough events" S and T, the underlying distribution on the constraints does not change much. This makes a lot of intuitive sense, since on choosing a random element of S and then a random neighbour, the extractor property guarantee that the induced distribution on vertices in X is ε-close to uniform. Thus, it is natural to expect that conditioning on the events S and T should not change the underlying distribution on the constraints by more than $O(\varepsilon)$. This was the rough argument in [10], which unfortunately turns out to be false. We elaborate on this in Section 3.2 and Appendix A.

A recent updated version [11] of [10] provides an different argument for the fortification lemma using a stronger extractor. We discuss this at the end of Section 1.2.

1.2 Our contributions

We present a fix to the approach of [10], by describing a way to transform any given game instance G into a robust instance G^* with the same value following the framework of [10] but using a different graph for concatenation, and a different analysis.

We first describe a concrete counter-example to the original argument of [10] in Section 3.2, that shows concatenating (Definition 2.1) with an arbitrary (δ, ε)-extractor is insufficient. In fact, as we show in Appendix B, *concatenating* (Definition 2.1) with *any* left-regular graph with left-degree by $o(1/\varepsilon\delta)$ fails to make arbitrary instances (δ, ε)-robust. We instead use bipartite graphs called *fortifiers*, defined below.

▶ **Definition 1.5** (Fortifiers). A bipartite graph $H = ((W, X), E_H)$ is an $(\delta, \varepsilon_1, \varepsilon_2)$-*fortifier* if for any set $S \subseteq W$ such that $|S| \geq \delta|W|$, if π is the probability distribution on X induced by picking a uniformly random element w from S, and a uniformly random neighbor x of w, then

$$|\pi - \mathbf{u}|_1 \ \leq \ \varepsilon_1 \quad \text{and} \quad \|\pi - \mathbf{u}\|^2 \ \leq \ \frac{\varepsilon_2}{|X|}.$$

Notice that a fortifier is an extractor, with the additional condition that the ℓ_2-distance of π from the uniform distribution is small. This is what enables us to show that *concatenation* (Definition 2.1) with a fortifier produces a robust instance.

▶ **Theorem 1.6** (Fortifiers imply robustness). *Suppose G is a two-prover projection game on a bi-regular graph $((X, Y), E)$. Then, for any $\varepsilon, \delta > 0$, if $H = ((W, X), E_H)$ is a $(\delta, \varepsilon, \varepsilon)$-fortifier, then the symmetrized concatenated game $G^* = (H \circ G)_{sym}$ is $(\delta, O(\varepsilon))$-robust.*

In particular, bipartite spectral expanders are good fortifiers, as Lemma 2.8 shows. This gives us our main result which follows from Lemma 2.8 and Theorem 1.6:

▶ **Corollary 1.7.** *Let G be a two-prover projection game on a bi-regular graph $((X, Y), E)$. For any $\varepsilon, \delta > 0$, if $H = ((X, X), E_H)$ is a symmetric bipartite graph that is a λ-expander (Definition 2.3) with $\lambda < \varepsilon\sqrt{\delta}$ then the symmetrized concatenated game $G^* = (H \circ G)_{sym}$ is $(\delta, 4\varepsilon)$-robust.*

As one would expect, the condition on the fortifier can be relaxed if the underlying graph of G_{sym} is a spectral-expander. We prove the following theorem. Theorem 1.6 follows from this theorem by setting $\lambda_0 = 1$.

▶ **Theorem 1.8.** *Let G be a two-prover projection game on bi-regular graph $((X, Y), E)$ where G_{sym} is a λ_0-expander. Then for any $\varepsilon, \delta > 0$, if $H = ((W, X), E_H)$ is a $(\delta, \varepsilon, (\varepsilon/\lambda_0))$-fortifier, then the symmetrized concatenated game $G^* = (H \circ G)_{sym}$ is $(\delta, O(\varepsilon))$-robust.*

One could ask if the definition of a fortifier is too strong, or if a weaker object would suffice. We argue in Section 3.1 that if we proceed through concatenation, fortifiers are indeed necessary to make a game robust.

Bipartite Ramanujan graphs of degree $\Theta(1/\varepsilon^2\delta)$ have $\lambda < \varepsilon\sqrt{\delta}$ and are therefore good fortifiers. In Appendix B, we show that this is almost optimal by proving a lower bound of $\Omega(1/\varepsilon\delta)$ on the left-degree of any graph that can achieve (δ, ε)-robustness. This shows that our construction of using expanders to achieve robustness is almost optimal, in terms of the degree of the fortifier graph. Note that the degree of the fortifier is important as the alphabet size of the concatenated game is the alphabet size of the original game raised to the degree. There are known explicit constructions of bi-regular (δ, ε)-extractors with left-degree $\mathrm{poly}(1/\varepsilon)\mathrm{poly}\log(1/\delta)$. But the lower bound in Section 3.1 shows that (δ, ε)-extractors are not fortifiers if $\delta \ll \varepsilon$, which is usually the relevant setting (see Theorem 1.4).

Independently, the author of [10] came up with a different argument to obtain robustness of projection games by using a $(\delta, \varepsilon\delta)$-extractor. This is described in an updated version [11] present on the author's homepage.

It is also seen from Theorem 1.8 that bi-regular $(\delta, \varepsilon\delta)$-extractors are indeed $(\delta, \varepsilon, \varepsilon)$-fortifiers as well. Using an expander instead is arguably simpler, and is almost optimal.

▶ Remark. Although this fix provides a proof of a Parallel Repetition Theorem for projection games following the framework of [10], the degree of the fortifier is too large to get the required PCP for proving optimal hardness of the SET-COVER problem that Dinur and Steurer [5] obtained. See [11] for a discussion on this.

Remark about parallel repetition for general games

A fairly straightforward generalization Theorem 1.4 to robust general games on bi-regular graphs is the following.

▶ **Claim 1.9.** *Let G be a general two-prover game on a bi-regular graph $((X, Y), E)$ with alphabets Σ_X and Σ_Y. For any positive integer k, if $\varepsilon, \delta > 0$ are parameters such that $2\delta|\Sigma_X \times \Sigma_Y|^{k-1} \leq \varepsilon$ and G is (δ, ε)-robust, then*

$$\mathrm{val}(G^k) \quad \leq \quad (\mathrm{val}(G) + \varepsilon)^k + k\varepsilon.$$

One could attempt a fortifying any game by using a fortifier on both sides. But the issue with this procedure is that it makes $|\Sigma_X| = \exp(1/\delta)$ and in such scenarios $\delta|\Sigma_X| \gg 1$ making it infeasible to ensure $2\delta|\Sigma_X \times \Sigma_Y|^{k-1} \leq \varepsilon$. Hence, though Lemma 1.9 may be useful in cases where we know that the game G is robust via other means, the technique of fortification via concatenation increases the alphabet size too much for Lemma 1.9 to be applicable.

For the case of projection games, this is not an issue as Theorem 1.4 only requires $2\delta|\Sigma_Y|^{k-1} < \varepsilon$ and concatenating G_{sym} by a fortifier only increases $|\Sigma_X|$ and keeps Σ_Y unchanged. Thus, one can indeed choose ε and δ small enough to give a parallel repetition theorem for a robust version of an arbitrary projection game.

2 Preliminaries

Notation

- For any vector $\mathbf{a}$, let $|\mathbf{a}|_1 := \sum_i |\mathbf{a}_i|$, and $\|\mathbf{a}\| := \sqrt{\sum_i \mathbf{a}_i^2}$ be the ℓ_1 and ℓ_2-norms respectively.
- We shall use $\mathbf{u}_S$ to refer to the uniform distribution on a set S. Normally, the set S would be clear from context and in such case we shall drop the subscript S.
- For any vector $\mathbf{a}$, we shall use $\mathbf{a}^{\|}$ to refer to the component along the direction of $\mathbf{u}$, and $\mathbf{a}^{\perp}$ to refer to the component orthogonal to $\mathbf{u}$.
- We shall assume that the underlying graph for the games is bi-regular.

We define the *concatenation* operation of a two-prover games with a bipartite graph that was alluded to in Section 1.1.

▶ **Definition 2.1** (Concatenation). Given bipartite graphs $G = ((X,Y),E), H = ((W,X),E_H)$ where H is regular with left degree D, the *concatenated graph* $H \circ G = ((W,Y),E')$ is a multigraph such that there is an edge $(w,y) \in E'$, for every pair of edges $(w,x) \in E_H, (x,y) \in E$.

Given a two-prover projection game on a graph $G = ((X,Y),E)$ with a set of constraints ψ, a pair of alphabets Σ_X and Σ_Y, a bipartite graph $H = ((W,X),E_H)$ with left degree D, the *concatenated game* is a game on the multigraph $H \circ G = ((W,Y),E')$ with $\Sigma_W = \Sigma_X^D$. For any edge $(w,y) \in E'$ which corresponds to the pair $(w,x) \in E, (x,y) \in E_H$, the constraint $\pi_{(w,y)}(a) := \pi_{x,y}(a_x)$, where $a \in \Sigma_X^D$ and a_x is the alphabet at the coordinate corresponding to x (assuming some fixed ordering of vertices in X). The distribution over the edges in the multigraph $H \circ G$ is uniform.

▶ Remark. The concatenated game $H \circ G$ is also a projection game. We shall be working with the symmetrized version $G^* = (H \circ G)_{\mathrm{sym}}$ of this game.

▶ **Lemma 2.2** (Concatenation preserves value). *[10] Given any two-prover game on G, and a biregular bipartite graph H:*

$$\mathrm{val}(H \circ G) = \mathrm{val}(G).$$

Expanders, extractors and fortifiers

▶ **Definition 2.3** (Expanders). For a symmetric, stochastic matrix M, define

$$\lambda(M) \stackrel{\text{def}}{=} \max_{\mathbf{v} \perp 1} \frac{\|M\mathbf{v}\|}{\|\mathbf{v}\|}$$

A D-regular graph $H = (X,E)$ is a graph H is a λ-expander, if $\lambda(H) \leq \lambda$, where H is the normalized adjacency matrix of the graph H.

For a symmetric bipartite graph $G = ((X,X),E)$, we say G is a bipartite λ-*expander* if $\lambda(H) \leq \lambda$ where H is the normalized biadjacency matrix of G.

Henceforth, when we refer to a bipartite graph as being a λ-expander, we implicitly mean a *bipartite* λ-expander.

Any expander $H = (X,E_H)$ can be transformed to a natural bipartite expander H' on $X \times X$, by including the edge (x,x') and (x',x) to H' for every $(x,x') \in E_H$. We shall abuse notation and call this graph $H' = ((X,X),E_H)$ although each edge in H occurs "twice" in H'.

▶ **Lemma 2.4** (Explicit expanders [3]). *For every $D > 0$, there exists a fully explicit family of graphs $\{G_i\}$, such that G_i is D-regular and $\lambda(G_i) \le D^{-1/2}(\log D)^{3/2}$.*

▶ **Definition 2.5** (Extractors). A bipartite graph $H = ((X, Y), E)$ is an (δ, ε)-extractor if for every subset $S \subseteq X$ such that $|S| \ge \delta|X|$, if π is the induced probability distribution on Y by taking a random element of S and a random neighbour, then

$$|\pi - \mathbf{u}|_1 \quad \le \quad \varepsilon.$$

▶ **Lemma 2.6** (Explicit Extractors [16]). *There exists explicit (δ, ε)-extractors $G = (X, Y, E)$ such that $|X| = O(|Y|/\delta)$ and each vertex of X has degree $D = O(\exp(\mathrm{poly}(\log\log(1/\delta))) \cdot (1/\varepsilon^2))$.*

Our earlier definition of a fortifier (Definition 1.5) has properties of both an expander and an extractor. Indeed, we can build fortifiers by just taking a product an expander and an extractor.

▶ **Lemma 2.7.** *Let $H_1 = ((V, W), E_1)$ is a bi-regular (δ, ε)-extractor, and let $H_2 = (W, E_2)$ is a regular λ-expander. Denote H_2' to be the bipartite graph $((W, W), E_2)$. Then the concatenated graph $H_1 \circ H_2'$ is an $(\delta, \varepsilon, \lambda^2\varepsilon/\delta)$-fortifier.*

Proof. Let H_2 be the normalized adjacency matrix of graph H_2. Let π_S denotes the probability distribution on W obtained by picking an element of $S \subseteq V$ uniformly and then choosing a random neighbour in H_1. Thus, $H_2\pi_S$ is the probability distribution on W induced by the uniform distribution on S and a random neighbour in $H_1 \circ H_2'$. We want to show for all S such that $|S| \ge \delta|V|$,

$$|H_2\pi_S - \mathbf{u}|_1 \le \varepsilon \quad \text{and} \quad \|H_2\pi_S - \mathbf{u}\|^2 \le \frac{\lambda^2\varepsilon/\delta}{|X|}.$$

The first inequality is obtained as $|H_2\pi_S - \mathbf{u}|_1 = |H_2(\pi_S - \mathbf{u})|_1 \le |\pi_S - \mathbf{u}|_1 \le \varepsilon$, where we use the fact that $|H_2 v|_1 \le |v|_1$ for any v and any normalized adjacency matrix, and $|\pi_S - \mathbf{u}|_1 \le \varepsilon$ follows form the extractor property of H_1.
As for the second inequality, observe that

$$\|\pi_S - \mathbf{u}\|^2 \le \max_{w \in W}(\pi_S(w)) \cdot |\pi_S - \mathbf{u}|_1 \le \varepsilon \cdot \max_{w \in W}(\pi_S(w)).$$

For a bi-regular extractor[2] H_1 of left-degree D, the degree of any $w \in W$ is $(|V| \cdot D/|W|)$ and the number of edges out of S is least $\delta|V| \cdot D$. Hence, $\max_w \pi_S(w) \le 1/(\delta|W|)$, which is achieved if all neighbours of w are in S. Therefore,

$$\begin{aligned} &\|\pi_S - \mathbf{u}\|^2 \le \frac{(\varepsilon/\delta)}{|W|} \\ \implies &\|H_2(\pi_S - \mathbf{u})\|^2 \le \lambda^2 \frac{|W|}{|X|} \|\pi_S - \mathbf{u}\|^2 \le \frac{|W|}{|X|} \cdot \frac{\lambda^2 \cdot (\varepsilon/\delta)}{|W|} \;=\; \frac{\lambda^2 \cdot (\varepsilon/\delta)}{|X|}. \end{aligned}$$ ◀

In particular, any bi-regular (δ, ε)-extractor is a $(\delta, \varepsilon, \varepsilon/\delta)$-fortifier. Hence, if the underlying graph G of the two-prover game is a $\sqrt{\delta}$-expander, then Theorem 1.8 states that merely using an (δ, ε)-extractor as suggested in [10] would be sufficient to make it $(\delta, O(\varepsilon))$-robust.

Also, since any graph is trivially a 1-expander, a bi-regular $(\delta, \varepsilon\delta)$-extractor is also an $(\delta, \varepsilon, \varepsilon)$-fortifier. The following lemma also shows that expanders are also fortifiers with reasonable parameters as well.

[2] The bound on the right-degree guaranteed by bi-regularity is crucial for this claim. Without this, extractors are not sufficient for fortification (Section 3.2).

▶ **Lemma 2.8.** *Let $H = (X, E_H)$ be any λ-expander. Then, for every $\delta > 0$, the bipartite graph $H' = ((X, X), E_H)$ is also a $(\delta, \sqrt{\lambda^2/\delta}, \lambda^2/\delta)$-fortifier. In particular, if $\lambda \le \varepsilon\sqrt{\delta}$, then H' is an $(\delta, \varepsilon, \varepsilon)$-fortifier.*

Proof. Let H be the normalized adjacency matrix of H. Let $S \subseteq W$ such that $|S| \ge \delta|W|$. We have,

$$\|\mathbf{u}_S^{\perp}\|^2 \le \frac{1}{\delta|W|}.$$

Hence, by the expansion property of H,

$$\|H\mathbf{u}_S - \mathbf{u}\|^2 := \|H\mathbf{u}_S^{\perp}\|^2 \le \lambda^2 \cdot \frac{|W|}{|X|} \cdot \|\mathbf{u}_S^{\perp}\|^2 \le \frac{\lambda^2/\delta}{|X|}.$$

$|H\mathbf{u}_S - \mathbf{u}|_1 \le \sqrt{\lambda^2/\delta}$ follows from above and Cauchy-Schwarz inequality. ◀

Although Lemma 2.8 shows that expanders are also fortifiers for reasonable parameters, the construction in Lemma 2.7 is more useful when the underlying graph for the two-prover game is already a good expander. For example, if the underlying graph G was a δ-expander, then Theorem 1.8 suggests that we only require a $(\delta, \varepsilon, \varepsilon/\delta)$-fortifier. Lemma 2.7 implies that an (δ, ε)-extractor is already a $(\delta, \varepsilon, \varepsilon/\delta)$-fortifier and hence is sufficient to make the game robust. The main advantage of this is the degree of δ-expanders must be $\Omega(1/\delta^2)$ whereas we have explicit (δ, ε)-extractors of degree $(1/\varepsilon^2)\exp(\operatorname{poly}\log\log(1/\delta))$ which has a much better dependence in δ. This dependence on δ is crucial for certain applications.

3 Sub-games on large rectangles

Consider a projection game on graph $G = ((X, Y), E)$ which is biregular with degree d. For a biregular bipartite graph $H = ((W, X), E_H)$ with degree d_H, consider the symmertized concatenated game $G^* = (H \circ G)_{\text{sym}} = ((W, W), E')$. Let $S, T \subseteq W$ and μ_S (or μ_T) denote the induced distributions on X obtained by picking a uniformly random element of S (or T) and taking a uniformly random neighbour in H. In the next claim, we give an expression for the distribution of verifier checking the underlying constraint on (x, x') in the subgame $(G^*)_{S\times T}$.

▶ **Claim 3.1.** *For any $x, x' \in X$ such that there are edges $(x, y), (x', y) \in E$,*

$$\pi_{x,x'} = \frac{\mu_S(x)\mu_T(x')}{\sum_{(x,x')\in G_{sym}} \mu_S(x)\mu_T(x')}. \tag{1}$$

Proof. Let $d_{S,x}, d_{T,x'}$ denote the degree of x to S and x' to T respectively in H. Let $N_H(x)$ denote the neighbor set of a vertex x in H. Then,

$$\mu_S(x) = \frac{d_{S,x}}{\sum_{z\in X} d_{S,z}}.$$

The probability $\pi_{x,x'}$ of the verifier in $(G^*)_{S\times T}$ checking a constraint corresponding to a constraint (x, x') in G_{sym}, is proportional to the number of edges (w, w') in the graph G^* such that $w \in S \cap N_H(x)$, and $w' \in T \cap N_H(x')$. Since every such edge in G^* was equally likely, we have:

$$\pi_{x,x'} = \frac{d_{S,x} \cdot d_{T,x'}}{\sum_{(x,x')\in G_{\text{sym}}} d_{S,x} d_{T,x'}} = \frac{\mu_S(x)\mu_T(x')}{\sum_{(x,x')\in G_{\text{sym}}} \mu_S(x)\mu_T(x')}.$$

◀

One way to show that the concatenated game G^* is $(\delta, O(\varepsilon))$-robust would be to show that the above distribution $\pi_{x,x'}$ is $O(\varepsilon)$-close to uniform whenever $|S|, |T|$ have density at least δ because then the distribution on constraints that the verifier is going to check in $G^*_{S\times T}$ is $O(\varepsilon)$ close to the distribution on constraints in G. Hence, up to additive factor of $O(\varepsilon)$ the quantity $\operatorname{val}(G^*_{S\times T})$ is same as $\operatorname{val}(G)$. The main question here what properties should H satisfy so that the above distribution is close to uniform?

3.1 Fortifiers are necessary

To prove that fortifiers are necessary, we shall restrict ourselves to games on graphs $G = ((X,X),E)$. We show that if a bipartite graph $H = ((W,X),E_H)$, makes a game on a particular graph G, $(\delta, O(\varepsilon))$-robust, then H is a good fortifier.

As mentioned earlier, if the graph G had some expansion properties, then the requirements on the graph H to concatenate with can be relaxed. Thus, naturally, the worst case graph G is one that expands the least – a matching.

▶ **Lemma 3.2** (Fortifiers are necessary)**.** *Let $\varepsilon, \delta > 0$ be small constants. Let $H = ((W,X),E_H)$ be a bi-regular graph, and let $G = ((X,X),E)$ be a matching. Suppose that for every subset $S, T \subseteq W$ with $|S|, |T| \geq \delta|W|$, the distribution (defined in Equation (1)) induced by the sub game on $S \times T$ of $G^* := (H \circ G)_{sym}$ on the edges of G is ε-close to uniform. Then, for every $S \subseteq W$ with $|S| \geq \delta|W|$,*

$$|\mu_S - \mathbf{u}|_1 = \varepsilon, \tag{2}$$
$$\|\mu_S - \mathbf{u}\|^2 = \frac{O(\varepsilon)}{|X|}. \tag{3}$$

Proof. It is clear that (2) is necessary as the distribution on constraints in the sub-game $G^*_{S\times W}$ (as defined in (1)) is essentially μ_S (as μ_T in this case is uniform).

As for (3), let us assume that

$$\|\mu_S - \mathbf{u}\|^2 = \frac{c}{|X|}.$$

Taking $T = S$, we obtain that the distribution (defined in Equation (1)) induced by the game $G^*_{S\times S}$ on the edges of G is given by

$$\pi_{x,x} = \frac{\mu_S(x)^2}{\sum_x \mu_S(x)^2} = \left(\frac{|X|}{1+c}\right)\cdot \mu_S(x)^2,$$

where the last equality used the fact that $\|\mu_S\|^2 = \left\|\mu_S^{\perp}\right\|^2 + \|\mathbf{u}\|^2$.

$$\begin{aligned}
\sum_{x\in X}\left|\left(\frac{|X|}{c+1}\right)\cdot\mu_S(x)^2 - \frac{1}{|X|}\right| &= \left(\frac{|X|}{1+c}\right)\cdot\sum_{x\in X}\left|\mu_S(x)^2 - \frac{c+1}{|X|^2}\right| \\
&= \left(\frac{|X|}{1+c}\right)\cdot\sum_{x\in X}\left|\mu_S(x) - \frac{\sqrt{c+1}}{|X|}\right|\cdot\left(\mu_S(x) + \frac{\sqrt{c+1}}{|X|}\right) \\
&\geq \left(\frac{1}{\sqrt{1+c}}\right)\cdot\sum_{x\in X}\left|\mu_S(x) - \frac{\sqrt{c+1}}{|X|}\right| \\
&\geq \left(\frac{1}{\sqrt{1+c}}\right)\cdot\left((\sqrt{1+c} - 1) - \sum_{x\in X}\left|\mu_S(x) - \frac{1}{|X|}\right|\right) \\
&\geq \left(\frac{1}{\sqrt{1+c}}\right)\cdot\left((\sqrt{1+c} - 1) - \varepsilon\right).
\end{aligned}$$

Thus, if the distribution on constraints is ε-close to uniform, then the above lower bound forces $c = O(\varepsilon)$. ◀

3.2 General (non-regular) extractors are insufficient

Suppose $H = ((W, X), E_H)$ is an arbitrary $(\delta, O(\varepsilon))$-extractor and G^* is the symmetrized concatenated game. Consider a possible scenario where there is a subset $S \subseteq W$ with $|S| \geq \delta|W|$ such that μ_S is of the form

$$\mu_S = \left(\varepsilon, \frac{1-\varepsilon}{|X|-1}, \ldots, \frac{1-\varepsilon}{|X|-1}\right).$$

Notice that this is a legitimate distribution that may be obtained from a large subset S as $|\mu_S - \mathbf{u}|_1$ is easily seen to be at most 2ε. However, if $G = ((X, X), E)$ was d-regular with $d = o(|X|)$, then using (1), the probability mass on the edge $(1, 1)$ on the sub-game over $S \times S$ is

$$\pi_{1,1} = \left(\frac{\varepsilon^2}{\varepsilon^2 + O\left(\frac{\varepsilon d}{|X|}\right)}\right) \approx 1.$$

In other words, if such a distribution μ_S can be induced by the extractor, then the provers can achieve value close to 1 in the game $G^*_{S\times S}$ by just labelling the edge $(1, 1)$ correctly. Thus, G^* is not even $(\delta, 0.9)$-robust.

In Appendix A we show that we can adversarially construct a $(\delta, O(\varepsilon))$-extractor, although non-regular, that induces such a skew distribution. In Appendix B we also show that left-regular graphs of left-degree $o(1/\delta\varepsilon)$ are not fortifiers.

4 Robustness from fortifiers

In this section, we show that concatenating a symmetrized two-prover game by fortifier(s) yields a robust game as claimed by Theorem 1.8.

▶ **Lemma 4.1** (Distributions from large rectangles are close to uniform). *Let $G = ((X, X), E)$ be a graph of a symmetrized two-prover game such that $|X| = n$. Let μ_S and μ_T be two probability distributions such that*

$$|\mu_S^{\perp}|_1 \leq \varepsilon_1 \quad \text{and} \quad |\mu_T^{\perp}|_1 \leq \varepsilon_1, \tag{4}$$

$$\|\mu_S^{\perp}\|^2 \leq \left(\frac{\varepsilon_2}{n}\right) \quad \text{and} \quad \|\mu_T^{\perp}\|^2 \leq \left(\frac{\varepsilon_2}{n}\right). \tag{5}$$

If the bipartite graph G is a λ_0-expander then the distribution on edges (x, y) of G given by (1) is $(2\varepsilon_1 + \varepsilon_1^2 + 2\lambda_0 \cdot \varepsilon_2)$-close to uniform.

As described in Section 3, if H is a $(\delta, \varepsilon_1, \varepsilon_2)$-fortifier, then for any set S and T of density at least δ, the distribution on the constraints of $G^*_{S\times T}$ is given by (1). Applying the above lemma for the graph of the symmetrized game yields that the value of the game on any large rectangle can change only by the above bound on the statistical distance. By setting the parameters, Theorem 1.8 follows immediately from Lemma 4.1. Further, Theorem 1.7 also follows from Lemma 4.1 and Lemma 2.8 as any graph is trivially a 1-expander.

The rest of this section would be devoted to the proof of Lemma 4.1. For convenience, we let d be the left-degree (and hence also, right-degree) of the biparite graph G. We shall prove Lemma 4.1 by proving the following two claims.

▶ **Claim 4.2.**

$$\sum_{(x,y)\in G}\left|\frac{\mu_S(x)\mu_T(y)}{\sum\limits_{(x,y)\in G}\mu_S(x)\mu_T(y)} - \frac{\mu_S(x)\mu_T(y)}{d/n}\right| \leq \lambda_0\cdot\varepsilon_2$$

▶ **Claim 4.3.**

$$\sum_{(x,y)\in G}\left|\frac{\mu_S(x)\mu_T(y)}{d/n} - \frac{1}{n\cdot d}\right| \leq 2\varepsilon_1+\varepsilon_1^2+\lambda_0\cdot\varepsilon_2$$

Clearly, Lemma 4.1 follows from Claim 4.2 and Claim 4.3.

Proof of Claim 4.2. Let G also denote the normalized biadjacency matrix of G. Observe that $\sum_{(x,y)\in G}\mu_S(x)\mu_T(y) = d\cdot\langle G\mu_S,\mu_T\rangle$. If we resolve μ_S and μ_T in the direction of the uniform distribution and the orthogonal component, we have

$$\begin{aligned}\langle G\mu_S,\mu_T\rangle &= \langle\mathbf{u},\mathbf{u}\rangle + \langle G\mu_S^\perp,\mu_T^\perp\rangle = \frac{1}{n} + \langle G\mu_S^\perp,\mu_T^\perp\rangle\\ \implies \left|\langle G\mu_S,\mu_T\rangle - \frac{1}{n}\right| &\leq \lambda_0\cdot\left\|\mu_S^\perp\right\|\cdot\left\|\mu_T^\perp\right\|\\ &\leq \left(\frac{\lambda_0\cdot\varepsilon_2}{n}\right). \qquad \text{(using (5))}\end{aligned}$$

Therefore,

$$\begin{aligned}\sum_{(x,y)\in G}\left|\frac{\mu_S(x)\mu_T(y)}{d\,\langle G\mu_S,\mu_T\rangle} - \frac{\mu_S(x)\mu_T(y)}{d/n}\right| &\leq \sum_{(x,y)\in G}\left(\frac{\mu_S(x)\mu_T(y)}{d\,\langle G\mu_S,\mu_T\rangle}\right)\left|1 - \langle G\mu_S,\mu_T\rangle\right|\\ &\leq \lambda_0\cdot\varepsilon_2.\end{aligned}$$ ◀

Proof of Claim 4.3.

$$\sum_{(x,y)\in G}\left|\frac{\mu_S(x)\mu_T(y)}{d/n} - \frac{1}{n\cdot d}\right| = \left(\frac{n}{d}\right)\sum_{(x,y)\in G}\left|\mu_S(x)\mu_T(y) - \frac{1}{n^2}\right|.$$

Since $\mu_S(x) = \frac{1}{n} + \mu_S^\perp(x)$ and $\mu_T(y) = \frac{1}{n}+\mu_T^\perp(y)$,

$$\begin{aligned}\left(\frac{n}{d}\right)\sum_{(x,y)\in G}\left|\mu_S(x)\mu_T(y) - \frac{1}{n^2}\right| &= \left(\frac{n}{d}\right)\sum_{(x,y)\in G}\left|\frac{\mu_S^\perp(x)}{n} + \frac{\mu_T^\perp(y)}{n} + \mu_S^\perp(x)\mu_T^\perp(y)\right|\\ \text{(Using triangle inequality)} \quad &\leq \frac{1}{d}\sum_{(x,y)\in G}\left|\mu_S^\perp(x)\right| + \frac{1}{d}\sum_{(x,y)\in G}\left|\mu_T^\perp(y)\right|\\ &\quad + \left(\frac{n}{d}\right)\sum_{(x,y)\in G}\left|\mu_S^\perp(x)\mu_T^\perp(y)\right|\\ &= \left|\mu_S^\perp\right|_1 + \left|\mu_T^\perp\right|_1 + \left(\frac{n}{d}\right)\sum_{(x,y)\in G}\left|\mu_S^\perp(x)\mu_T^\perp(y)\right|,\end{aligned}$$

where the last equality uses the fact that G is a bi-regular graph. Define $f_S(x)\equiv|\mu_S^\perp(x)|$ is a vector with the entrywise absolute values of $\mu_S^\perp$, and similarly f_T. Then, the RHS above equation reduces to

$$\begin{aligned}
\left|\mu_S^\perp\right|_1 + \left|\mu_T^\perp\right|_1 + \left(\frac{n}{d}\right) \sum_{(x,y)\in G} \left|\mu_S^\perp(x)\mu_T^\perp(y)\right| &= \left|\mu_S^\perp\right|_1 + \left|\mu_T^\perp\right|_1 \\
&\quad + \left(\frac{n}{d}\right) \cdot \sum_{(x,y)\in G} f_S(x) f_T(y) \\
&= \left|\mu_S^\perp\right|_1 + \left|\mu_T^\perp\right|_1 + n \left\langle G f_S, f_T \right\rangle \\
\text{(Using (4))} \qquad &\leq 2\varepsilon_1 + n \cdot \left\langle G f_S, f_T \right\rangle . \qquad (6)
\end{aligned}$$

A simple bound for $n \cdot \langle Gf_S, f_T\rangle$ would $n \left\|G\mu_S^\perp\right\| \left\|\mu_T^\perp\right\|$ by Cauchy-Schwarz inequality. We can use the expansion of G again to estimate this better. Consider the decomposition $f_S = \alpha_1 \cdot \mathbf{u} + f_S^\perp$ and $f_T = \alpha_2 \cdot \mathbf{u} + f_T^\perp$. It follows that $\alpha_1 = |f_S|_1$ and $\alpha_2 = |f_T|_1$, and hence $\alpha_1, \alpha_2 \leq \varepsilon_1$ by (4). Hence,

$$\begin{aligned}
n \cdot \langle Gf_S, f_T\rangle = \alpha_1 \cdot \alpha_2 + n \cdot \left\langle Gf_S^\perp, f_T^\perp \right\rangle &\leq \varepsilon_1^2 + n \left\|Gf_S^\perp\right\| \left\|f_T^\perp\right\| \\
&\leq \varepsilon_1^2 + n \cdot \lambda_0 \cdot \left\|\mu_S^\perp\right\| \cdot \left\|\mu_T^\perp\right\| \\
\text{(Using (5))} \qquad &\leq \varepsilon_1^2 + \lambda_0 \varepsilon_2 .
\end{aligned}$$

Combining this with (6), we get

$$\sum_{(x,y)\in G} \left| \frac{\mu_S(x)\mu_T(y)}{d/n} - \frac{1}{n \cdot d} \right| \leq 2\varepsilon_1 + \varepsilon_1^2 + \lambda_0 \varepsilon_2 . \qquad ◀$$

Acknowledgements. We would like to thank Dana Moshkovitz for several discussions and clarifications regarding the initial counter-example. We would also like to thank Mohammad Bavarian for pointing out that our proof might generalize for general two-prover games, and would like to thank Anup Rao for pointing out subtleties involving parallel repetition for general games. We would like to thank anonymous reviewers for various suggestions to improve the write-up. We also would like to thank Prahladh Harsha, Irit Dinur and Amir Shpilka for many fruitful conversations and comments on the write-up.

References

1 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, May 1998.

2 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, January 1998.

3 Yonatan Bilu and Nathan Linial. Lifts, discrepancy and nearly optimal spectral gap*. *Combinatorica*, 26(5):495–519, 2006.

4 Mark Braverman and Ankit Garg. Small value parallel repetition for general games. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:95, 2014. To appear in STOC 2015.

5 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 – June 03, 2014*, pages 624–633, 2014.

6 Uriel Feige. On the success probability of the two provers in one-round proof systems. In *Structure in Complexity Theory Conference, 1991., Proceedings of the Sixth Annual*, pages 116–123, Jun 1991.

7 Uriel Feige and Joe Kilian. Two prover protocols: low error at affordable rates. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 172–183, 1994.

8 Lance Jeremy Fortnow. *Complexity-theoretic aspects of interactive proof systems.* PhD thesis, Massachusetts Institute of Technology, 1989.

9 Thomas Holenstein. Parallel repetition: Simplification and the no-signaling case. *Theory of Computing*, 5(1):141–172, 2009.

10 Dana Moshkovitz. Parallel repetition from fortification. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 414–423, 2014.

11 Dana Moshkovitz. Parallel repetition from fortification. `http://people.csail.mit.edu/dmoshkov/papers/par-rep/final3.pdf`, 2015.

12 Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM J. Discrete Math.*, 13(1):2–24, 2000.

13 Anup Rao. Parallel repetition in projection games and a concentration bound. *SIAM J. Comput.*, 40(6):1871–1891, 2011.

14 Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998.

15 Ran Raz. A counterexample to strong parallel repetition. *SIAM J. Comput.*, 40(3):771–777, June 2011.

16 Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 3–13, 2000.

17 Oleg Verbitsky. Towards the parallel repetition conjecture. *Theor. Comput. Sci.*, 157(2):277–282, May 1996.

A An explicit extractor that does not provide robustness

Let $H = ((W, X), E_H)$ be any (δ, ε)-extractor. Let us assume that the extractor is left-regular with left-degree D, and let $m = |W|$ and $n = |X|$. For any $x \in X$ and $S \subseteq W$, let $d_S(x)$ denote the degree of x in S. Let us fix one $S \subset W$ such that $|S| = \delta|W|$.

We will transform the graph H so that the distribution induced by the set S looks like the counter-example described in Section 3.2 in the following two steps by altering the edges in the subgraph $S \times X$:

1. First change the degree into X from S to be exactly uniform.
2. Next further change the degrees into X from S to be like the counterexample

Both these operations can be achieved in a monotone fashion: for every $x \in X$, the neighborhood of every vertex is either a superset, or a subset of its neighborhood before each operation.

We will show that moving the edges this way does not perturb the indegree distribution from other large sets by too much, and the resulting graph is a $(\delta, O(\varepsilon))$ extractor as long as the number of edges we relocate is at most $O(\varepsilon\delta \cdot mD)$. This process will preserve the left-regularity of H but would *not* preserve bi-regularity.

First let us move edges (monotonically) from S into X create the uniform distribution on X. When doing this, the degree of each vertex changes by $\Delta_S(x) := |d_S(x) - \frac{\delta mD}{n}|$, where $d_S(x)$ was the old degree. From the extractor property, we know that:

$$\sum_{x \in X} \Delta_S(x) \;=\; \sum_{x \in X} (\delta mD) \left| \frac{d_S(x)}{\sum d_S(x)} - \left(\frac{1}{n}\right) \right| \quad \leq \quad \varepsilon\delta \cdot mD. \tag{7}$$

Every vertex $x \in X$ now has degree d^S_{avg}. Fix some vertex $x_1 \in X$, and relocate from every other $x \neq x_1$ any set of $\varepsilon \cdot d^S_{\text{avg}}$ edges to be incident on x_1. Thus, if $d'_S(x)$ refers to the new degrees, we have $d'_S(x_1)$ is $(1+\varepsilon n)d^S_{\text{avg}}$ where as $d'_S(x)$ is $(1-\varepsilon)d^S_{\text{avg}}$ for every other $x \neq x_1$.

The further change in degrees incurred on any $x \in X$ is $\Delta'_S(x) := \left|d'_S(x) - \frac{\delta m D}{n}\right|$. Since we this process only relocates $O(\varepsilon \cdot d^S_{\text{avg}}|X|)$ edges, we have

$$\sum_{x\in X} \Delta'_S(x) \quad = \quad \sum_{x \in X} \left|d'_S(x) - d^S_{\text{avg}}\right| \quad \leq \quad O(n \cdot \varepsilon \cdot d^S_{\text{avg}}) \quad = \quad O(\varepsilon\delta \cdot mD). \tag{8}$$

Thus, the neighbourhood of any vertex x has changed additively by at most $\Delta_S(x)+\Delta'_S(x)$. Therefore, for any subset $T \subseteq W$ of size at least $\delta|W|$,

$$\begin{aligned}\sum_{x\in X} \left|d'_T(x) - d^T_{\text{avg}}\right| &\leq \sum_{x\in X} \left|d_T(x) - d^T_{\text{avg}}\right| + \sum_{x\in X} \left|d'_T(x) - d_T(x)\right| \\ &\leq \varepsilon|T|D + \sum_{x\in X} (\Delta_S(x) + \Delta'_S(x)) \\ &\leq \varepsilon|T|D + O(\varepsilon\delta \cdot mD) \quad \text{(using (7) and (8))} \\ &\leq O(\varepsilon \cdot |T|D).\end{aligned}$$

Thus, the new graph after relocating edges is still an $(\delta, O(\varepsilon))$-extractor. This extractor, induces a distribution similar to the one described in Section 3.2 and hence cannot provide robustness.

B Lower bounds on degree of fortifiers

In this section, we will show that an attempt to make a game (δ,ε)-robust by concatenating any left-regular graph with left degree D fails if $D \leq o(1/\varepsilon\delta)$.

▶ **Lemma 2.1.** *Let $H = ((W,X), E_H)$ be a left-regular bipartite graph with left-degree $D = 1/(c \cdot \varepsilon\delta)$ for some $c > 0$, and small enough constants ε, δ. Then, there exists a subset $S \subseteq W$ with $|S| \geq \delta|W|$ such that if p was the distribution on X induced by the uniform distribution on S then*

$$\|p - \mathbf{u}\|^2 \geq \frac{\Omega(c\varepsilon)}{|X|}.$$

Proof. Let $d_{\text{avg}} = |W|D/|X|$. Note that at most $|X|/2$ vertices x satisfy $\deg(x) \geq 2d_{\text{avg}}$. Further, if there is a set S of $|X|/4$ vertices x that $\deg(x) < (0.5)d_{\text{avg}}$, then if p is the distribution on X induced by the uniform distribution on W, then $|p-\mathbf{u}|_1 > 1/4$ which implies that $\|p-\mathbf{u}\|_2^2 \geq \frac{1}{4|X|}$ by Cauchy-Schwarz.

Otherwise, there exists $X' \subset X$ such that $|X'| = c\varepsilon\delta^2|X|$ and for each $x \in X'$ we have $(0.5)d_{\text{avg}} < \deg(x) < 2d_{\text{avg}}$. Consider the set S_0 of all neighbours of X'. If $D < 1/(c\varepsilon\delta)$, we have $|S_0| \leq 2c\delta^2\varepsilon \cdot |W|D = 2\delta|W|$ which is a very small fraction of $|W|$ when δ is small enough. Consider an arbitrary set $S_1 \subseteq W$ such that $|S_1| = \delta m$, with $S_1 \cap S_0 = \emptyset$. Let $S_2 = S_0 \cup S_1$. Let π_1, π_2 be the probability distribution on X induced by S_1, S_2 respectively. Note that $|S_2| \leq 3\delta|W|$.

For every $x \in X'$, we know that $\pi_1(x) = 0$ and $\pi_2(x) = \Omega\left(\frac{1}{\delta|X|}\right)$. Therefore,

$$\|\pi_1 - \pi_2\|^2 \geq \Omega\left(\frac{c\delta^2\varepsilon|X|}{\delta^2|X|^2}\right) = \frac{\Omega(c\varepsilon)}{|X|}.$$

Since $\|\pi_1 - \pi_2\| \leq \|\pi_1 - \mathbf{u}\| + \|\pi_2 - \mathbf{u}\|$, we have that one of the sets S_1 or S_2 shows the validity of the lemma ◀

We thus immediately infer the following:

▶ **Corollary 2.2.** *For all small enough $\delta, \varepsilon > 0$, no left-regular graph $H = ((W, X), E_H)$ with left-degree $D = o(1/\varepsilon\delta)$ is an $(\delta, *, \varepsilon)$-fortifier.*

Note that any $(\delta, \varepsilon, \varepsilon)$-fortifier is in particular an (δ, ε)-extractor, and hence we also have that $D = \Omega((1/\varepsilon^2)\log(1/\delta))$ [12]. We also point out that the construction of Lemma 2.8 has left-degree $D = \tilde{O}(1/\varepsilon^2\delta)$. The above essentially shows this construction is almost optimal.

Learning Circuits with Few Negations

Eric Blais[1], **Clément L. Canonne**[2], **Igor C. Oliveira**[2], **Rocco A. Servedio**[2], **and Li-Yang Tan**[2]

1 University of Waterloo
200 University Avenue West, Waterloo ON, Canada
eric.blais@uwaterloo.ca

2 Columbia University
500 W 120th Street, New York NY, USA
{ccanonne,oliveira,rocco,liyang}@cs.columbia.edu

Abstract

Monotone Boolean functions, and the monotone Boolean circuits that compute them, have been intensively studied in complexity theory. In this paper we study the structure of Boolean functions in terms of the minimum number of negations in any circuit computing them, a complexity measure that interpolates between monotone functions and the class of all functions. We study this generalization of monotonicity from the vantage point of learning theory, establishing nearly matching upper and lower bounds on the uniform-distribution learnability of circuits in terms of the number of negations they contain. Our upper bounds are based on a new structural characterization of negation-limited circuits that extends a classical result of A. A. Markov. Our lower bounds, which employ Fourier-analytic tools from hardness amplification, give new results even for circuits with no negations (i.e. monotone functions).

1998 ACM Subject Classification I.2.6 Learning, F.2.2 Nonnumerical Algorithms and Problems, G.1.2 Approximation, G.3 Probability and Statistics

Keywords and phrases Boolean functions, monotonicity, negations, PAC learning

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.512

1 Introduction

A *monotone* Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is one that satisfies $f(x) \le f(y)$ whenever $x \preceq y$, where $\preceq$ denotes the bitwise partial order on $\{0,1\}^n$. The structural and combinatorial properties of monotone Boolean functions have been intensively studied for many decades, see e.g. [12] for an in-depth survey. Many important results in circuit complexity deal with monotone functions, including celebrated lower bounds on monotone circuit size and monotone formula size (see e.g. [22, 23] and numerous subsequent works).

Monotone functions are also of considerable interest in computational learning theory, in particular with respect to the model of learning under the uniform distribution. In an influential paper, Bshouty and Tamon [6] showed that any monotone Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ can be learned from uniform random examples to error ε in time $n^{O(\sqrt{n}/\varepsilon)}$. They also gave a lower bound, showing that no algorithm running in time 2^{cn} for any $c < 1$ can learn arbitrary monotone functions to accuracy $\varepsilon = 1/(\sqrt{n}\log n)$. (Many other works in learning theory such as [3, 11, 5, 1, 26, 20, 21] deal with learning monotone functions from a range of different perspectives and learning models, but we limit our focus in this paper to learning to high accuracy with respect to the uniform distribution.)

18th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'15) /
19th Int'l Workshop on Randomization and Computation (RANDOM'15).
Editors: Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim; pp. 512–527
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1.1 Beyond monotonicity: Inversion complexity, alternations, and Markov's theorem

Given the importance of monotone functions in complexity theory and learning theory, it is natural to consider various generalizations of monotonicity. One such generalization arises from the simple observation that monotone Boolean functions are precisely the functions computed by *monotone Boolean circuits*, i.e. circuits which have only AND and OR gates but no negations. Given this, an obvious generalization of monotonicity is obtained by considering functions computed by Boolean circuits that have a small number of negation gates. The *inversion complexity* of $f\colon \{0,1\}^n \to \{0,1\}$, denoted $I(f)$, is defined to be the minimum number of negation gates in any AND/OR/NOT circuit (with access to constant inputs 0/1) that computes f. We write $\mathcal{C}_t^n$ to denote the class of n-variable Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ that have $I(f) \leq t$.

Another generalization of monotonicity is obtained by starting from an alternate characterization of monotone Boolean functions. A function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone if and only if the value of f "flips" from 0 to 1 at most once as the input x ascends any chain in $\{0,1\}^n$ from 0^n to 1^n. (Recall that a chain of length ℓ is an increasing sequence $(x^1, \ldots, x^\ell)$ of vectors in $\{0,1\}^n$, i.e. for every $j \in \{1, \ldots, \ell-1\}$ we have $x^j \prec x^{j+1}$.) Thus, it is natural to consider a generalization of monotonicity that allows more than one such "flip" to occur. We make this precise with the following notation and terminology: given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ and a chain $X = (x^1, \ldots, x^\ell)$, a position $j \in [\ell-1]$ is said to be *alternating* with respect to f if $f(x^j) \neq f(x^{j+1})$. We write $A(f,X) \subseteq [\ell-1]$ to denote the set of alternating positions in X with respect to f, and we let $a(f,X) = |A(f,X)|$ denote its size. We write $a(f)$ to denote the maximum of $a(f,X)$ taken over all chains X in $\{0,1\}^n$, and we say that $f\colon \{0,1\}^n \to \{0,1\}$ is *k-alternating* if $a(f) \leq k$.

A celebrated result of A. A. Markov from 1957 [14] gives a tight quantitative connection between the inversion and alternation complexities defined above:

▶ **Markov's Theorem.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function which is not identically 0. Then* (i) *if $f(0^n) = 0$, then $I(f) = \lceil \log(a(f)+1) \rceil - 1$; and* (ii) *if $f(0^n) = 1$, then $I(f) = \lceil \log(a(f)+2) \rceil - 1$.*

This robustness motivates the study of circuits which contain few negation gates, and indeed such circuits have been studied in complexity theory. Amano and Maruoka [2] have given bounds on the computational power of such circuits, showing that circuits for the clique function which contain fewer than $\frac{1}{6}\log\log n$ many negation gates must have superpolynomial size. More recently, Rossman [24] proved that there exists an explicit monotone function that cannot be computed by fan-in two circuits of logarithmic depth containing less than $\left(\frac{1}{2}-\varepsilon\right)\log n$ negations. Other works have studied the effect of limiting the number of negation gates in formulas [16, 9], bounded-depth circuits [25, 27], and non-deterministic circuits [17]. Another line of work that has received attention lately is the role of monotonicity and negation complexity in cryptography and related areas [8, 10].

In the present work, we study circuits with few negations from the vantage point of computational learning theory, giving both positive and negative results. We observe that some of the recent works mentioned [10, 9] build on techniques introduced in a preliminary version of this paper.

1.2 Our results

We begin by studying the structural properties of functions that are computed or approximated by circuits with few negation gates. In Section 2 we establish the following extension of Markov's theorem:

▶ **Theorem 1.1.** *Let f be a k-alternating Boolean function. Then f can be expressed as $f(x) = h(m_1(x), \dots, m_k(x))$, where each $m_i(x)$ is monotone and h is either the parity function or its negation. Conversely, any function of this form is k-alternating.*

Theorem 1.1 along with Markov's theorem yields the following characterization of $\mathcal{C}_t^n$:

▶ **Corollary 1.2.** *Every $f \in \mathcal{C}_t^n$ can be expressed as $f = h(m_1, \dots, m_T)$ where h is either PAR_T or its negation, each $m_i \colon \{0,1\}^n \to \{0,1\}$ is monotone, and $T = O(2^t)$.*

A well-known consequence of Markov's theorem is that every Boolean function is exactly computed by a circuit which has only $\log n$ negation gates, and as we shall see an easy argument shows that every Boolean function is 0.01-approximated by a circuit with $\frac{1}{2}\log n + O(1)$ negations. In Section 2 we note that no significant savings are possible over this easy upper bound:

▶ **Theorem 1.3.** *For almost every function $f \colon \{0,1\}^n \to \{0,1\}$, any Boolean circuit C that 0.01-approximates f must contain $\frac{1}{2}\log n - O(1)$ negations.*

We then turn to our main topic of investigation, the uniform-distribution learnability of circuits with few negations. We use our new extension of Markov's theorem, Theorem 1.1, to obtain a generalization of the Fourier-based uniform-distribution learning algorithm of Bshouty and Tamon [6] for monotone circuits:

▶ **Theorem 1.4.** *There is a uniform-distribution learning algorithm which learns any unknown $f \in \mathcal{C}_t^n$ from random examples to error ε in time $n^{O(2^t\sqrt{n}/\varepsilon)}$.*

We observe that many natural functions are indeed computed by circuits with few negations. As an example, consider the property of undirected graphs that is satisfied by an n-vertex graph G if and only if G contains a triangle but does not contain a cycle of size $\log n$. Clearly, this property is non-monotone. However, it is easy to see that it can be represented by a Boolean function $f \colon \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ that is computed by a circuit with a single negation. Our positive result implies that learning such properties does not take much more time than learning monotone properties.[1]

Theorem 1.4 immediately leads to the following question: can an even faster learning algorithm be given for circuits with t negations, or is the running time of Theorem 1.4 essentially the best possible? Interestingly, prior to our work a matching lower bound for Theorem 1.4 was not known even for the special case of monotone functions (corresponding to $t = 0$). As mentioned earlier, Bshouty and Tamon proved that to achieve accuracy $\varepsilon = 1/(\sqrt{n}\log n)$ any learning algorithm needs time $\omega(2^{cn})$ for any $c < 1$ (see Claim 3.13 for a slight sharpening of this statement). For larger values of ε, though, the strongest previous lower bound was due to Blum, Burch and Langford [5]. Their Theorem 10 implies that any membership-query algorithm that learns monotone functions to error $\varepsilon < \frac{1}{2} - c$ (for any $c > 0$) must run in time $2^{\Omega(\sqrt{n})}$ (in fact, must make at least this many membership queries). However, this lower bound does not differentiate between the number of membership queries required to learn to high accuracy versus "moderate" accuracy – say, $\varepsilon = 1/n^{1/10}$ versus $\varepsilon = 1/10$. Thus the following question was unanswered prior to the current paper: what is

[1] In contrast to the robustness we show in the learning setting, there are natural computational problems whose complexity changes drastically with the addition of a single negation gate. For instance, checking if a monotone circuit is non-constant is trivial. Nevertheless, it is possible to prove that the same computational problem for circuits with a single negation gate admits polynomial time algorithms if and only if $\mathsf{P} = \mathsf{NP}$.

the best lower bound that can be given, both as a function of n and ε, on the complexity of learning monotone functions to accuracy ε?

We give a fairly complete answer to this question, providing a lower bound as a function of n, ε and t on the complexity of learning circuits with t negations. Our lower bound essentially matches the upper bound of Theorem 1.4, and is thus simultaneously essentially optimal in all three parameters n, ε and t for a wide range of settings of ε and t. Our lower bound result is the following:

▶ **Theorem 1.5.** *For any $t \leq \frac{1}{28}\log n$ and any $\varepsilon \in [1/n^{1/12}, 1/2 - c]$, $c > 0$, any membership-query algorithm that learns any unknown function $f \in \mathcal{C}_t^n$ to error ε must make $2^{\Omega(2^t\sqrt{n}/\varepsilon)}$ membership queries.*

We note that while our algorithm uses only uniform random examples, our lower bound holds even for the stronger model in which the learning algorithm is allowed to make arbitrary membership queries on points of its choosing.

Theorem 1.5 is proved using tools from the study of hardness amplification. The proof involves a few steps. We start with a strong lower bound for the task of learning to high accuracy the class of balanced monotone Boolean functions (reminiscent of the lower bound obtained by Bshouty and Tamon). Then we combine hardness amplification techniques and results on the noise sensitivity of monotone functions in order to get stronger and more general lower bounds for learning monotone Boolean functions to moderate accuracy. Finally, we use hardness amplification once more to lift this result into a lower bound for learning circuits with few negations to moderate accuracy. An ingredient employed in this last stage is to use a k-alternating combining function which "behaves like" the parity function on (roughly) k^2 variables; this is crucial in order for us to obtain our essentially optimal final lower bound of $2^{\Omega(2^t\sqrt{n}/\varepsilon)}$ for circuits with t negations. These results are discussed in more detail in Section 3.2.

Lastly, we mention an interesting research direction left unanswered by our results. Specifically, we focus in this work on the uniform-distribution learnability to *high accuracy*, i.e. when the error parameter ε is thought of as "small" (or at least bounded away from 1/2). While we provide almost optimal bounds for this regime, the complexity of *weakly* learning circuits with negations – that is obtaining inverse-polynomial advantage over random guessing – remains open. As a concrete question, is there an *efficient* algorithm that learns circuits with a single negation with error at most $1/2 - \Omega(1/n^c)$ for some $c > 0$? (Note that the analogue question for monotone circuits is well-understood [5, 1, 21].)

2 Structural facts about computing and approximating functions with low inversion complexity

2.1 An extension of Markov's theorem

We begin with the proof of our new extension of Markov's theorem. For any $A \subseteq \{0,1\}^n$ let $\mathbf{1}[A]\colon \{0,1\}^n \to \{0,1\}$ be the characteristic function of A. For $f\colon \{0,1\}^n \to \{0,1\}$ and $x \in \{0,1\}^n$, we write $a_f(x)$ to denote

$$a_f(x) \overset{\text{def}}{=} \max\{a(f,X) : X \text{ is a chain that starts at } x\},$$

and note that $a(f) = \max_{x\in\{0,1\}^n}\{a_f(x)\} = a_f(0^n)$. For $0 \leq \ell \leq a(f)$ let us write S_ℓ^f to denote $S_\ell^f \overset{\text{def}}{=} \{x \in \{0,1\}^n : a_f(x) = \ell\}$, and let $T_\ell^f \overset{\text{def}}{=} S_0^f \cup \cdots \cup S_\ell^f$. We note that $S_0^f, \ldots, S_{a(f)}^f$ partition the set of all inputs: $S_i^f \cap S_j^f = \emptyset$ for all $i \neq j$, and $T_{a(f)}^f = S_0^f \cup \cdots \cup S_{a(f)}^f = \{0,1\}^n$.

We will need the following simple observation:

▶ **Observation 2.1.** *Fix any f and any $x \in \{0,1\}^n$. If $x \in S^f_\ell$ and $y \succ x$ then $y \in S^f_{\ell'}$ for some $\ell' \le \ell$. Furthermore, if $f(y) \neq f(x)$ then $\ell' < \ell$.*

▶ **Theorem 1.1.** *(Restated) Fix $f\colon \{0,1\}^n \to \{0,1\}$ and let $k \overset{\text{def}}{=} a(f)$. Then f can be expressed as $f = h\big(\mathbf{1}\big[T^f_0\big], \dots, \mathbf{1}\big[T^f_{k-1}\big]\big)$, where*
(i) *the functions $\mathbf{1}\big[T^f_\ell\big]$ are monotone for all $0 \le \ell \le k$,*
(ii) *$h\colon \{0,1\}^k \to \{0,1\}$ is PAR_k if $f(0^n) = 0$ and $\neg\,\mathsf{PAR}_k$ if $f(0^n) = 1$,*
and $\mathsf{PAR}_k(x) = x_1 \oplus \cdots \oplus x_k$ is the parity function on k variables. Conversely, for any monotone Boolean functions $m_1, \dots, m_k$, any Boolean function of the form $h(m_1, \dots, m_k)$ is k-alternating.

Proof. Claim 1 follows immediately from Observation 2.1 above. The proof of 2 is by induction on k. In the base case $k = 0$, we have that f is a constant function and the claim is immediate.

For the inductive step, suppose that the claim holds for all functions f' that have $a(f') \le k-1$. We define $f'\colon \{0,1\}^n \to \{0,1\}$ as $f' = f \oplus \mathbf{1}\big[S^f_k\big]$. Observation 2.1 implies that $S^{f'}_\ell = S^f_\ell$ for all $0 \le \ell \le k-2$ and $S^{f'}_{k-1} = S^f_{k-1} \cup S^f_k$, and in particular, $a(f) = k-1$. Therefore we may apply the inductive hypothesis to f' and express it as $f' = h'\big(\mathbf{1}\big[T^{f'}_0\big], \dots, \mathbf{1}\big[T^{f'}_{k-2}\big]\big)$. Since $T^{f'}_\ell = T^f_\ell$ for $0 \le \ell \le k-2$, we may use this along with the fact that $\mathbf{1}\big[S^f_k\big] = \neg\,\mathbf{1}\big[T^f_{k-1}\big]$ to get:

$$f = f' \oplus \mathbf{1}\big[S^f_k\big] = h'\big(\mathbf{1}\big[T^{f'}_0\big], \dots, \mathbf{1}\big[T^{f'}_{k-2}\big]\big) \oplus \neg\,\mathbf{1}\big[T^f_{k-1}\big] = h'\big(\mathbf{1}\big[T^f_0\big], \dots, \mathbf{1}\big[T^f_{k-2}\big]\big) \oplus \neg\,\mathbf{1}\big[T^f_{k-1}\big]$$

and the inductive hypothesis holds (note that $0^n \in S^f_k$).

The converse is easily verified by observing that any chain in $\{0,1\}^n$ can induce at most $k+1$ possible vectors of values for $(m_1, \dots, m_k)$ because of their monotonicity. ◀

Theorem 1.1 along with Markov's theorem immediately yields the following corollary:

▶ **Corollary 1.2.** *Every $f \in \mathcal{C}^n_t$ can be expressed as $f = h(m_1, \dots, m_T)$ where h is either PAR_T or its negation, each $m_i\colon \{0,1\}^n \to \{0,1\}$ is monotone, and $T = O(2^t)$.*

2.2 Approximation

As noted earlier, Markov's theorem implies that every n-variable Boolean function can be exactly computed by a circuit with (essentially) $\log n$ negations (since $a(f) \le n$ for all f). If we set a less ambitious goal of *approximating* Boolean functions (say, having a circuit correctly compute f on a $1 - \varepsilon$ fraction of all 2^n inputs), can significantly fewer negations suffice?

We first observe that every Boolean function f is ε-close (with respect to the uniform distribution) to a function f' that has $a(f') \le O(\sqrt{n \log 1/\varepsilon})$. The function f' is obtained from f simply by setting $f'(x) = 0$ for all inputs x that have Hamming weight outside of $[n/2 - O(\sqrt{n \log 1/\varepsilon}), n/2 + O(\sqrt{n \log 1/\varepsilon})]$; a standard Chernoff bound implies that f and f' disagree on at most $\varepsilon 2^n$ inputs. Markov's theorem then implies that the inversion complexity $I(f')$ is at most $\frac{1}{2}(\log n + \log\log \frac{1}{\varepsilon}) + O(1)$. Thus, every Boolean function can be approximated to high accuracy by a circuit with only $\frac{1}{2}\log n + O(1)$ negations.

We now show that this upper bound is essentially optimal: for almost every Boolean function, any 0.01-approximating circuit must contain at least $\frac{1}{2}\log n - O(1)$ negations. To prove this, we recall the notion of the *total influence* of a Boolean function f: this is

$$\mathbf{Inf}[f] = \sum_{i=1}^{n} \mathbf{Inf}_i[f], \quad \text{where} \quad \mathbf{Inf}_i[f] = \mathbf{Pr}_{x\in\{0,1\}^n}[f(x) \neq f(x^{\oplus i})]$$

and $x^{\oplus i}$ denotes x with its i-th coordinate flipped. The total influence of f is easily seen to equal αn, where $\alpha \in [0,1]$ is the fraction of all edges $e = (x, x')$ in the Boolean hypercube that are bichromatic, i.e. have $f(x) \neq f(x')$. In Appendix A.1 we prove the following lemma:

▶ **Lemma 2.2.** *Suppose* $f\colon \{0,1\}^n \to \{0,1\}$ *is such that* $\mathbf{Inf}[f] = \Omega(n)$. *Then* $a(f) = \Omega(\sqrt{n})$.

It is easy to show that a random function has influence $\frac{n}{2}(1 - o(1))$ with probability $1 - 2^{-n}$. Given this, Claim 2.2, together with the elementary fact that whenever f' is ε-close to f then $|\mathbf{Inf}(f') - \mathbf{Inf}(f)| \leq 2\varepsilon n$, directly yields the following:

▶ **Theorem 1.3.** *With probability* $1 - 2^{-n}$, *any* 0.01-*approximator* f' *for a random function* f *must have inversion complexity* $I(f') \geq \frac{1}{2}\log n - O(1)$.

▶ Remark. The results in this section (together with simple information-theoretic arguments showing that random functions are hard to learn) imply that one cannot expect to have a learning algorithm (even to constant accuracy) for the class $\mathcal{C}^n_{\frac{1}{2}\log n + O(1)}$ of circuits with $\frac{1}{2}\log n + O(1)$ negations in time significantly better than 2^n. As we shall see in Section 3.1, for any fixed $\delta > 0$ it *is* possible to learn $\mathcal{C}^n_{(\frac{1}{2}-\delta)\log n}$ to accuracy $1 - \varepsilon$ in time $2^{\tilde{O}(n^{1-\delta})/\varepsilon}$.

3 Learning circuits with few negations

3.1 A learning algorithm for $\mathcal{C}^n_t$

We sketch the learning algorithm and analysis of Bshouty and Tamon [6]; using the results from Section 2 our Theorem 1.4 will follow easily from their approach. Our starting point is the simple observation that functions with good "Fourier concentration" can be learned to high accuracy under the uniform distribution simply by estimating all of the low-degree Fourier coefficients. This fact, established by Linial, Mansour and Nisan, is often referred to as the "Low-Degree Algorithm:"

▶ **Theorem 3.1** (Low-Degree Algorithm ([13])). *Let* $\mathcal{C}$ *be a class of Boolean functions such that for* $\varepsilon > 0$ *and* $\tau = \tau(\varepsilon, n)$,

$$\sum_{|S|>\tau} \widehat{f}(S)^2 \leq \varepsilon$$

for any $f \in \mathcal{C}$. *Then* $\mathcal{C}$ *can be learned from uniform random examples in time* $\mathrm{poly}(n^\tau, 1/\varepsilon)$.

Using the fact that every monotone function $f\colon \{0,1\}^n \to \{0,1\}$ has total influence $\mathbf{Inf}(f) \leq \sqrt{n}$, and the well-known Fourier expression $\mathbf{Inf}(f) = \sum_S \widehat{f}(S) \cdot |S|^2$ for total influence, a simple application of Markov's inequality let Bshouty and Tamon show that every monotone function f has

$$\sum_{|S|>\sqrt{n}/\varepsilon} \widehat{f}(S)^2 \leq \varepsilon.$$

Together with Theorem 3.1, this gives their learning result for monotone functions.

Armed with Corollary 1.2, it is straightforward to extend this to the class $\mathcal{C}_t^n$. Corollary 1.2 and a union bound immediately give that every $f \in \mathcal{C}_t^n$ has $\mathbf{Inf}(f) \leq O(2^t)\sqrt{n}$, so the Fourier expression for influence and Markov's inequality give that

$$\sum_{|S|>O(2^t)\sqrt{n}/\varepsilon} \widehat{f}(S)^2 \leq \varepsilon$$

for $f \in \mathcal{C}_t^n$. Theorem 1.4 follows immediately using the Low-Degree Algorithm.

An immediate question is whether this upper bound on the complexity of learning $\mathcal{C}_t^n$ is optimal; we give an affirmative answer in the next subsection.

3.2 Lower bounds for learning

As noted in the introduction, we prove information-theoretic lower bounds against learning algorithms that make a limited number of membership queries. We start by establishing a new lower bound on the number of membership queries that are required to learn *monotone* functions to high accuracy, and then build on this to provide a lower bound for learning $\mathcal{C}_t^n$. Our query lower bounds are essentially tight, matching the upper bounds (which hold for learning from uniform random examples) up to logarithmic factors in the exponent.

We first state the results; the proofs are deferred to Section 3.2.1. We say that a Boolean function f is *balanced* if $\mathbf{Pr}_x[f(x) = 0] = \mathbf{Pr}_x[f(x) = 1] = 1/2$.

▶ **Theorem 3.2.** *There exists a class $\mathcal{H}_n$ of balanced n-variable monotone Boolean functions such that for any $\varepsilon \in [\frac{1}{n^{1/6}}, 1/2 - c]$, $c > 0$, learning $\mathcal{H}_n$ to accuracy $1 - \varepsilon$ requires $2^{\Omega(\sqrt{n}/\varepsilon)}$ membership queries.*

This immediately implies the following corollary, which essentially closes the gap in our understanding of the hardness of learning monotone functions:

▶ **Corollary 3.3.** *For any $\varepsilon = \Omega(1/n^{1/6})$ bounded away from $1/2$, learning n-variable monotone functions to accuracy $1 - \varepsilon$ requires $2^{\tilde{\Theta}(\sqrt{n})/\varepsilon}$ queries.*

Using this class $\mathcal{H}$ as a building block, we obtain the following hardness of learning result for the class of k-alternating functions:

▶ **Theorem 3.4.** *For any function $k \colon \mathbb{N} \to \mathbb{N}$, there exists a class $\mathcal{H}^{(k)}$ of balanced $k = k(n)$-alternating n-variable Boolean functions such that, for any n sufficiently large and $\varepsilon > 0$ such that (i) $2 \leq k < n^{1/14}$, and (ii) $k^{7/3}/n^{1/6} \leq \varepsilon \leq \frac{1}{2} - c$, learning $\mathcal{H}^{(k)}$ to accuracy $1 - \varepsilon$ requires $2^{\Omega(k\sqrt{n}/\varepsilon)}$ membership queries.*

(We note that the tradeoff between the ranges of k and ε that is captured by condition (ii) above seems to be inherent to our approach and not a mere artifact of the analysis; see Observation 3.16.) This theorem immediately yields the following:

▶ **Corollary 3.5.** *Learning the class of k-alternating functions to accuracy $1 - \varepsilon$ in the uniform-distribution membership-query model requires $2^{\Omega(k\sqrt{n}/\varepsilon)}$ membership queries, for any $k = O(n^{1/28})$ and $\varepsilon \in [1/n^{1/12}, \frac{1}{2} - c]$.*

▶ **Corollary 3.6.** *For $t \leq \frac{1}{28}\log n$, learning $\mathcal{C}_t^n$ to accuracy $1 - \varepsilon$ requires $2^{\Omega(2^t\sqrt{n}/\varepsilon)}$ membership queries, for any $\varepsilon \in [2^{7t/3}/n^{1/6}, \frac{1}{2} - c]$.*

3.2.1 Proofs

We require the following standard notion of *composition* for two functions f and g:

▶ **Definition 3.7** (Composition). For $f\colon \{0,1\}^m \to \{0,1\}$ and $g\colon \{0,1\}^r \to \{0,1\}$, we denote by $g \otimes f$ the Boolean function on $n = mr$ inputs defined by

$$(g \otimes f)(x) \overset{\text{def}}{=} g(\underbrace{f,\dots,f}_{r})(x) = g(f(x_1,\dots,x_m),\dots,f(x_{(r-1)m+1},\dots,x_{rm}))$$

Similarly, for any $g\colon \{0,1\}^r \to \{0,1\}$ and $\mathcal{F}_m$ a class of Boolean functions on m variables, we let

$$g \otimes \mathcal{F}_m = \{\, g \otimes f \;:\; f \in \mathcal{F}_m \,\}$$

and $g \otimes \mathcal{F} = \{g \otimes \mathcal{F}_m\}_{m \ge 1}$.

Overview of the arguments. Our approach is based on hardness amplification. In order to get our lower bound against learning k-alternating functions, we (a) start from a lower bound ruling out very high-accuracy learning of *monotone* functions; (b) use a suitable monotone combining function to get an XOR-like hardness amplification, yielding a lower bound for learning (a subclass of) monotone functions to moderate accuracy; (c) repeat this approach on this subclass with a different (now k-alternating) combining function to obtain our final lower bound, for learning k-alternating functions to moderate accuracy.

$$\underbrace{\begin{bmatrix}\text{high-accuracy}\\ \text{monotone}\end{bmatrix}}_{\text{(a)}} \xrightarrow[\text{monotone}]{\otimes\text{-like}} \underbrace{\begin{bmatrix}\text{moderate accuracy}\\ \text{monotone}\end{bmatrix}}_{\text{(b)}} \xrightarrow[k\text{-alternating}]{\otimes\text{-like}} \underbrace{\begin{bmatrix}\text{moderate accuracy}\\ k\text{-alternating}\end{bmatrix}}_{\text{(c)}} \qquad (1)$$

In more detail, in both steps (b) and (c) the idea is to take as base functions the hard class from the previous step (respectively "monotone hard to learn to high accuracy," and "monotone hard to learn to moderate accuracy"), and compose them with a very noise-sensitive function in order to amplify hardness. Care must be taken to ensure that the combining function satisfies several necessary constraints (being monotone for (b) and k-alternating for (c), and being as sensitive as possible to the correct regime of noise in each case).

Useful tools

We begin by recalling a few notions and results that play a crucial role in our approach.

▶ **Definition 3.8** (Noise stability). For $f\colon \{0,1\}^n \to \{0,1\}$, the *noise stability* of f at $\eta \in [-1,1]$ is

$$\mathrm{Stab}_\eta(f) \overset{\text{def}}{=} 1 - 2\,\mathbf{Pr}[\, f(x) \neq f(y)\,]$$

where x is drawn uniformly at random from $\{0,1\}^n$ and y is obtained from x by independently for each bit having $\mathbf{Pr}[y_i = x_i] = (1+\eta)/2$ (i.e., x and y are *η-correlated*).

▶ **Definition 3.9** (Bias and expected bias). The *bias* of a Boolean function $h\colon \{0,1\}^n \to \{0,1\}$ is the quantity $\mathrm{bias}(h) \overset{\text{def}}{=} \max(\mathbf{Pr}[\,h=1\,], \mathbf{Pr}[\,h=0\,])$, while the *expected bias of h at δ* is defined as $\mathrm{ExpBias}_\delta(h) \overset{\text{def}}{=} \mathbf{E}_\rho[\mathrm{bias}(h_\rho)]$, where ρ is a random restriction on n coordinates where each coordinate is independently left free with probability δ and set to 0 or 1 with same probability $(1-\delta)/2$.

▶ **Fact 3.10** (Proposition 4.0.11 from [19]). *For $\delta \in [0,1/2]$ and $f\colon \{0,1\}^n \to \{0,1\}$, we have*

$$\frac{1}{2} + \frac{1}{2}\mathrm{Stab}_{1-2\delta}(f) \le \mathrm{ExpBias}_{2\delta}(f) \le \frac{1}{2} + \frac{1}{2}\sqrt{\mathrm{Stab}_{1-2\delta}(f)}.$$

Building on Talagrand's probabilistic construction [28] of a class of functions that are sensitive to very small noise, Mossel and O'Donnell [18] gave the following noise stability upper bound. (We state below a slightly generalized version of their Theorem 3, which follows from their proof with some minor changes; see Appendix A.2 for details of these changes.)

▶ **Theorem 3.11** (Theorem 3 of [18]). *There exists an absolute constant K and an infinite family of* balanced *monotone functions $g_r\colon \{0,1\}^r \to \{0,1\}$ such that $\mathrm{Stab}_{1-\tau/\sqrt{r}}(g_r) \le 1 - K\tau$ holds for all sufficiently large r, as long as $\tau \in [16/\sqrt{r}, 1]$.*

Applying Fact 3.10, it follows that for the Mossel–O'Donnell function g_r on r inputs and any τ as above, we have

$$\frac{1}{2} \le \mathrm{ExpBias}_\gamma(g_r) \le \frac{1}{2} + \frac{1}{2}\sqrt{1-K\tau} \le 1 - \frac{K}{4}\tau \tag{2}$$

for $\gamma \overset{\text{def}}{=} \frac{\tau}{\sqrt{r}}$.

We will use the above upper bound on expected bias together with the following key tool from [7], which gives a hardness amplification result for uniform distribution learning. This result builds on the original hardness amplification ideas of O'Donnell [19]. (We note that the original theorem statement from [7] deals with the running time of learning algorithms, but inspection of the proof shows that the theorem also applies to the number of membership queries that the learning algorithms perform.)

▶ **Theorem 3.12** (Theorem 12 of [7]). *Fix $g\colon \{0,1\}^r \to \{0,1\}$, and let $\mathcal{F}$ be a class of m-variable Boolean functions such that for every $f \in \mathcal{F}$, $\mathrm{bias}(f) \le \frac{1}{2} + \frac{\epsilon}{8r}$. Let A be a uniform distribution membership query algorithm that learns $g \otimes \mathcal{F}$ to accuracy $\mathrm{ExpBias}_\gamma(g) + \epsilon$ using $T(m, r, 1/\epsilon, 1/\gamma)$ queries. Then there exists a uniform-distribution membership query algorithm B that learns $\mathcal{F}$ to accuracy $1-\gamma$ using $O(T \cdot \mathrm{poly}(m, r, 1/\epsilon, 1/\gamma))$ membership queries.*

Hardness of learning monotone functions to high accuracy. At the bottom level, corresponding to step (a) in (1), our approach relies on the following simple claim which states that monotone functions are hard to learn to very high accuracy. (We view this claim, as essentially folklore; as noted in the introduction it slightly sharpens a lower bound given in [6]. A proof is given for completeness in Appendix A.3.)

▶ **Claim 3.13** (A slice of hardness). *There exists a class of balanced monotone Boolean functions $\mathcal{G} = \{\mathcal{G}_m\}_{m\in\mathbb{N}}$ and a universal constant C such that, for any constants $0 < \alpha \le 1/10$, learning $\mathcal{G}_m$ to error $0 < \varepsilon \le \alpha/\sqrt{m}$ requires at least 2^{Cm} membership queries.*

We now prove Theorem 3.2, i.e. we establish a stronger lower bound (in terms of the range of accuracy it applies to) against learning the class of *monotone* functions. We do this by amplifying the hardness result of Fact 3.13 by composing the "mildly hard" class of functions $\mathcal{G}$ with a monotone function g – the Mossel–O'Donnell function of Theorem 3.11 – that is very sensitive to small noise (intuitively, the noise rate here is comparable to the error rate from Fact 3.13).

Proof of Theorem 3.2. We will show that there exists an absolute constant $\alpha > 0$ such that for any n sufficiently large and $\tau \in [\frac{1}{n^{1/6}}, 1/2 - c]$, there exist $m = m(n)$, $r = r(n)$ (both of which are $\omega_n(1)$) such that learning the class of (balanced) functions $\mathcal{H}_n = g_r \otimes \mathcal{G}_m$ on $n = mr$ variables to accuracy $1 - \tau$ requires at least $2^{\alpha\sqrt{n}/\tau}$ membership queries.

By contradiction, suppose we have an algorithm A which, for all m, r, τ as above, learns the class $\mathcal{H}_n$ to accuracy $1 - \tau$ using $T = T_A(n, \tau) < 2^{\alpha\sqrt{n}/\tau}$ membership queries. We show that this implies that for infinitely many values of m, one can learn $\mathcal{G}_m$ to error $\varepsilon = .1/\sqrt{m}$ with $2^{o(m)}$ membership queries, in contradiction to Fact 3.13.

Fix any n large enough and $\tau \in [\frac{1}{n^{1/6}}, .1]$, and choose m, r satisfying $mr = n$ and $\frac{5}{K} \cdot \frac{\tau}{\sqrt{r}} = \frac{.1}{\sqrt{m}}$, where K is the constant from Theorem 3.11. Note that this implies $m = \frac{K}{50} \cdot \frac{\sqrt{n}}{\tau} \in [\Theta(n^{1/2}), \Theta(n^{2/3})]$ so indeed both m and r are $\omega_n(1)$. (Intuitively, the value $\frac{.1}{\sqrt{m}}$ is the error we *want* to achieve to get a contradiction, while the value $\frac{5}{K} \cdot \frac{\tau}{\sqrt{r}}$ is the error we *can* get from Theorem 3.12.) Note that we indeed can use the Mossel–O'Donnell function from Theorem 3.11, which requires $\tau > \frac{16}{\sqrt{r}}$ – for our choice of r, this is equivalent to $\tau > \left(\frac{16\sqrt{K}}{\sqrt{50}}\right)^{2/3} \frac{1}{n^{1/6}}$. Finally, set $\varepsilon \overset{\text{def}}{=} .1/\sqrt{m}$.

We apply Theorem 3.12 with $g \overset{\text{def}}{=} g_r$, $\gamma = (5/K)\tau/\sqrt{r}$ and $\epsilon = \tau/4$. (Note that all functions in $\mathcal{G}_m$ are balanced, and thus trivially satisfy the condition that $\text{bias}(f) \leq \frac{\epsilon}{8r}$, and recall that $1 - \gamma$ is the accuracy the theorem guarantees against the original class $\mathcal{G}_m$.) With these parameters we have

$$\text{ExpBias}_\gamma(g) + \epsilon \underset{\text{Eq.(2)}}{\leq} 1 - \frac{K}{4}\frac{5\tau}{K} + \frac{\tau}{4} = 1 - \tau \leq \text{accuracy}(A).$$

Theorem 3.12 gives that there exists a learning algorithm B learning $\mathcal{G}_m$ to accuracy $1 - \gamma \geq 1 - \varepsilon$ with $T_B = O(T \cdot \text{poly}(m, r, 1/\tau, 1/\gamma)) = O(T \cdot \text{poly}(n, 1/\tau))$ membership queries, that is, $T_B = T_A(n, \tau) \cdot \text{poly}(n, 1/\tau) < 2^{\alpha\sqrt{n}/\tau + o(\sqrt{n}/\tau)}$ many queries. However, we have $2^{(\alpha+o(1))\sqrt{n}/\tau} = 2^{(\alpha+o(1))m \cdot \frac{\sqrt{n}}{\tau m}} < 2^{Cm}$, where the inequality comes from observing that $\frac{\sqrt{n}}{\tau m} = \frac{50}{K}$ (so that it suffices to pick α satisfying $50\alpha/K < C$). This contradicts Claim 3.13 and proves the theorem. ◀

▶ Remark (Improving this result). Proposition 1 of [18] gives a lower bound on the best noise stability that can be achieved by any monotone function. If this lower bound were in fact tight – that is, there exists a family of monotone functions $\{f_r\}$ such that for all $\gamma \in [-1, 1]$, $\text{Stab}_{1-\gamma}(f_r) = (1 - \gamma)^{(\sqrt{2/\pi}+o(1))\sqrt{r}}$ – then the above lower bound could be extended to an (almost) optimal range of τ, i.e. $\tau \in [\Phi(n)/\sqrt{n}, \frac{1}{2} - c]$ for Φ any fixed superconstant function.

From hardness of learning monotone Boolean functions to hardness of learning k-alternating functions. We now establish the hardness of learning k-alternating functions. Hereafter we denote by $\mathcal{H} = \{g_r \otimes \mathcal{G}_m\}_{m,r}$ the class of "hard" monotone functions from Theorem 3.2. Since g_r is balanced and every $f \in \mathcal{G}_m$ has bias 1/2, it is easy to see that $\mathcal{H}$ is a class of balanced functions.

We begin by recalling the following useful fact about the noise stability of functions that are close to PAR:

▶ **Fact 3.14** (e.g., from the proof of Theorem 9 in [4]). *Let $r \geq 1$. If f is a Boolean function on r variables which η-approximates* PAR$_r$, *then for all $\delta \in [0, 1]$,*

$$\text{Stab}_{1-2\delta}(f) \leq (1 - 2\eta)^2(1 - 2\delta)^r + 4\eta(1 - \eta). \tag{3}$$

We use the above fact to define a function that is tailored to our needs: that is, a k-alternating function that is very sensitive to noise *and is defined on roughly k^2 inputs.* Without the last condition, one could just use PAR_k, but in our context this would only let us obtain a $\sqrt{k}$ (rather than a k) in the exponent of the lower bound, because of the loss in the reduction. To see why, observe that by using a combining function on k variables instead of k^2, the number of variables of the combined function $g_k \otimes \mathcal{G}_m$ would be only $n = km$. However, to get a contradiction with the hardness of monotone functions we shall need $k\sqrt{n}/\varepsilon \ll \sqrt{m}/\tau$, where $\tau \approx \varepsilon/k$, as the hardness amplification lemma requires the error to scale down with the number of combined functions.

► **Definition 3.15.** For any odd[2] $r \geq k \geq 1$, let $\mathsf{PAR}'_{k,r}$ be the symmetric Boolean function on r inputs defined as follows: for all $x \in \{0,1\}^r$,

$$\mathsf{PAR}'_{k,r}(x) = \begin{cases} 0 & \text{if } |x| \leq \frac{r-k}{2} \\ 1 & \text{if } |x| \geq \frac{r+k}{2} \\ \mathsf{PAR}_r(x) & \text{otherwise.} \end{cases}$$

In particular, $\mathsf{PAR}'_{k,r}$ is k-alternating, and agrees with PAR_r on the $k+1$ middle layers of the hypercube. By an additive Chernoff bound, one can show that $\mathsf{PAR}'_{k,r}$ is η-close to PAR_r, for $\eta = e^{-k^2/2r}$.

Proof of Theorem 3.4. $\mathcal{H}_n^{(k)}$ will be defined as the class $\mathsf{PAR}'_{k,r} \otimes \mathcal{H}_m$ for some r and m such that $n = mr$ (see below). It is easy to check that functions in $\mathcal{H}_n^{(k)}$ are balanced and k-alternating. We show below that for n sufficiently large, $2 \leq k < n^{1/14}$ and $\varepsilon \in [(1/300)(k^{14}/n)^{1/6}, \frac{1}{2} - c]$, learning $\mathcal{H}_n^{(k)}$ to accuracy $1-\varepsilon$ requires $2^{\Omega\left(k\sqrt{n}/\varepsilon\right)}$ membership queries.

By contradiction, suppose we have an algorithm A learning for all n, k, ε as above the class of k-alternating functions to accuracy $1-\varepsilon$ using $T_A(n,k,\varepsilon) < 2^{\beta\frac{k\sqrt{n}}{\varepsilon}}$ membership queries, where $\beta > 0$ is a universal constant to be determined during the analysis. We claim that this implies that for infinitely many values of m, one can learn $\mathcal{H}_m$ to some range of accuracies with a number of membership queries contradicting the lower bound of Theorem 3.2.

Fix any n large enough, k and ε as above (which in particular impose $k = O\left(n^{1/14}\right)$). The constraints we impose on m, r and τ are the following:

$$mr = n; \quad \mathrm{ExpBias}_\tau(\mathsf{PAR}'_{k,r}) + \varepsilon \leq 1 - \varepsilon; \quad m = \omega_n(1); \quad \tau \geq \frac{1}{m^{1/6}}; \tag{4}$$

$$\beta k\frac{\sqrt{n}}{\varepsilon} < \alpha\frac{\sqrt{m}}{\tau}, \tag{5}$$

where the constraints in (4) are for us to apply the previous theorems and lemmas, while (5) is needed to ultimately derive a contradiction.

One can show that by taking $r \stackrel{\text{def}}{=} \left\lfloor \frac{k^2}{2\ln 5} \right\rfloor \geq 1$ and $\tau \stackrel{\text{def}}{=} \frac{100\varepsilon}{r}$, the second constraint of (4) is satisfied, as then $\mathrm{Stab}_{1-\tau}(\mathsf{PAR}'_{k,r}) \leq 1 - 8\varepsilon$ (for the derivation, see Appendix A.4). Then, with the first constraint of (4), we get (omitting for simplicity the floors) $m \stackrel{\text{def}}{=} \frac{n\tau}{100\varepsilon} = (2\ln 5)\frac{n}{k^2}$,

[2] The above definition can be straightforwardly extended to $r \geq k \geq 1$ not necessarily odd, resulting in a similar k-alternating perfectly balanced function $\mathsf{PAR}'_{k,r}$ that agrees with PAR_r on $k + O(1)$ middle layers of the cube and is 0 below and 1 above those layers. For the sake of simplicity we leave out the detailed description of the other cases.

so as long as $k = o(\sqrt{n})$, the third constraint of (4) is met as well. With these settings, the final constraint of (4) can be rewritten as $\varepsilon \geq \frac{1}{100}\left(\frac{r^7}{n}\right)^{1/6} = \frac{1}{100(2\ln 5)^{7/6}}\left(\frac{k^{14}}{n}\right)^{1/6}$. As $(2\ln 5)^{7/6} > 3$, it is sufficient to have $\varepsilon \geq \frac{1}{300}\left(\frac{k^{14}}{n}\right)^{1/6}$, which holds because of the lower bound on ε.

It only remains to check Constraint (5) holds:

$$k\frac{\sqrt{n}}{\varepsilon} = 100k\frac{\sqrt{n}}{\tau r} = 100\frac{k}{\sqrt{r}}\frac{\sqrt{m}}{\tau} \leq \left(100\sqrt{\frac{2\ln 5}{1-2\ln 5/k^2}}\right)\frac{\sqrt{m}}{\tau} \leq 300\sqrt{2\ln 5}\cdot\frac{\sqrt{m}}{\tau},$$

where the first inequality holds because as $\frac{1}{r} \leq \frac{1}{\frac{k^2}{2\ln 5}-1}$ and the second holds because $k \geq 2$. So for the right choice of $\beta = \Omega(1)$, e.g. $\beta = \alpha/600$, $\beta k\frac{\sqrt{n}}{\varepsilon} < \alpha\frac{\sqrt{m}}{\tau}$, and (5) is satisfied.

It now suffices to apply Theorem 3.12 to $\mathsf{PAR}'_{k,r} \otimes \mathcal{H}_m$, with parameters $\gamma = \tau$ and ε, on algorithm A, which has accuracy $\operatorname{acc}(A) \geq 1-\tau \geq \operatorname{ExpBias}_\gamma(\mathsf{PAR}'_{k,r}) + \epsilon$. Since the functions of $\mathcal{H}$ are unbiased, it follows that there exists an algorithm B learning $\mathcal{H}_m$ to accuracy $1-\tau$, with $\tau > 1/2m^{1/6}$, making only

$$T_B(m,\tau) = O(T_A(n,k,\varepsilon)\operatorname{poly}(n,k,1/\varepsilon)) = 2^{\beta k\frac{\sqrt{n}}{\varepsilon}(1+o(1))} < 2^{\alpha\frac{\sqrt{m}}{\tau}}$$

membership queries, which contradicts the lower bound of Theorem 3.2. ◀

▶ **Observation 3.16** (On the relation between ε and k). *The tradeoff in the ranges for k and ε appear to be inherent to this approach. Namely, it comes essentially from Constraint* (4)*, itself deriving from the hypotheses of Theorem 3.2. However, even getting an optimal range in the latter would still require $\tau = \Omega(1/\sqrt{m})$, which along with $r \approx k^2$ and $\tau \approx \varepsilon/r$ impose $k = O(n^{1/6})$ and $\varepsilon = \Omega(k^3/\sqrt{n})$.*

References

1 K. Amano and A. Maruoka. On learning monotone Boolean functions under the uniform distribution. In *International Conference on Algorithmic Learning Theory* (ALT), pages 57–68, 2002.

2 K. Amano and A. Maruoka. A Superpolynomial Lower Bound for a Circuit Computing the Clique Function with At Most (1/6) log log n Negation Gates. *SIAM Journal on Computing*, 35(1):201–216, 2005.

3 D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

4 E. Blais and L-Y. Tan. Approximating Boolean functions with depth-2 circuits. In *Conference on Computational Complexity* (CCC), pages 74–85, 2013.

5 A. Blum, C. Burch, and J. Langford. On learning monotone Boolean functions. In *Symposium on Foundations of Computer Science* (FOCS), pages 408–415, 1998.

6 N. Bshouty and C. Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 1996.

7 V. Feldman, H. K. Lee, and R. A. Servedio. Lower bounds and hardness amplification for learning shallow monotone formulas. *Journal of Machine Learning Research – Proceedings Track*, 19:273–292, 2011.

8 O. Goldreich and R. Izsak. Monotone circuits: One-way functions versus pseudorandom generators. *Theory of Computing*, 8(1):231–238, 2012.

9 S. Guo and I. Komargodski. Negation-limited formulas. Technical Report 22(26), Electronic Colloquium on Computational Complexity (ECCC), 2015.

10 S. Guo, T. Malkin, I. C. Oliveira, and A. Rosen. The power of negations in cryptography. In *Theory of Cryptography Conference* (TCC), pages 36–65, 2015.

11 M. Kearns and L. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.

12 A. D. Korshunov. Monotone Boolean functions. *Russian Mathematical Surveys* (*Uspekhi Matematicheskikh Nauk*), 58(5):929–1001, 2003.

13 N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

14 A. A. Markov. On the inversion complexity of systems of functions. *Doklady Akademii Nauk SSSR*, 116:917–919, 1957. English translation in [15].

15 A. A. Markov. On the inversion complexity of a system of functions. *Journal of the ACM*, 5(4):331–334, October 1958.

16 H. Morizumi. Limiting negations in formulas. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 701–712, 2009.

17 H. Morizumi. Limiting negations in non-deterministic circuits. *Theoretical Computer Science*, 410(38-40):3988–3994, 2009.

18 E. Mossel and R. O'Donnell. On the noise sensitivity of monotone functions. *Random Structures and Algorithms*, 23(3):333–350, 2003.

19 R. O'Donnell. *Computational applications of noise sensitivity*. PhD thesis, MIT, June 2003.

20 R. O'Donnell and R. Servedio. Learning monotone decision trees in polynomial time. *SIAM Journal on Computing*, 37(3):827–844, 2007.

21 R. O'Donnell and K. Wimmer. KKL, Kruskal-Katona, and monotone nets. *SIAM Journal on Computing*, 42(6):2375–2399, 2013.

22 Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *Journal of the ACM*, 39(3):736–744, 1992.

23 A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Doklady Akademii Nauk SSSR*, 281:798–801, 1985. English translation in: Soviet Mathematics Doklady 31:354–357, 1985.

24 B. Rossman. Correlation bounds against monotone NC^1. In *Conference on Computational Complexity* (CCC), 2015.

25 M. Santha and C. Wilson. Limiting negations in constant depth circuits. *SIAM Journal on Computing*, 22(2):294–302, 1993.

26 R. Servedio. On learning monotone DNF under product distributions. *Information and Computation*, 193(1):57–74, 2004.

27 S. Sung and K. Tanaka. Limiting Negations in Bounded-Depth Circuits: an Extension of Markov's Theorem. In *International Symposium on Algorithms and Computation* (ISAAC), pages 108–116, 2003.

28 M. Talagrand. How much are increasing sets positively correlated? *Combinatorica*, 16(2):243–258, 1996.

A Proofs

A.1 Proof of Claim 2.2

Suppose $\mathbf{Inf}[f] \geq \alpha n$ for some $\alpha \in (0,1]$: this means that at least an α fraction of all edges are bichromatic. Define the *weight level* k (denoted $\mathcal{W}_k$) to be the set of all edges going from a vertex of Hamming weight k to a vertex of Hamming weight $k+1$ (in particular, $|\mathcal{W}_k| = (n-k)\binom{n}{k}$), and consider weight levels $n/2 - a\sqrt{n}, \dots, n/2 + a\sqrt{n}-1$ (the "middle levels") for $a \overset{\text{def}}{=} \sqrt{(1/2)\ln(8/\alpha)}$. (We suppose without loss of generality that $n/2 - a\sqrt{n}$ is a whole number.) Now, the fraction of all edges which do *not* lie in these middle levels is at most

$$\frac{1}{n2^{n-1}}\cdot 2\sum_{j=0}^{\frac{n}{2}-a\sqrt{n}-1}|\mathcal{W}_k| \le \frac{2n}{n2^{n-1}}\sum_{j=0}^{\frac{n}{2}-a\sqrt{n}-1}\binom{n}{k} \le \frac{4}{2^n}\sum_{j=1}^{\frac{n}{2}-a\sqrt{n}-1}\binom{n}{k} \le 4e^{-2a^2} = \frac{\alpha}{2}.$$

So no matter how many of these edges are bichromatic, it must still be the case that at least an $\alpha/2$ fraction of all edges in the "middle levels" are bichromatic.

Since the ratio

$$\frac{|\mathcal{W}_{n/2}|}{|\mathcal{W}_{n/2-a\sqrt{n}}|} = \frac{\frac{n}{2}\binom{n}{n/2}}{\left(\frac{n}{2}+a\sqrt{n}\right)\binom{n}{n/2-a\sqrt{n}}}$$

converges monotonically from below (when n goes to infinity) to $C \overset{\text{def}}{=} e^{2a^2}$, any two weight levels amongst the middle ones have roughly the same number of edges, up to a multiplicative factor C. Setting $p = \alpha/6C$ and $q = \alpha/6$, this implies that at least a p fraction of the weight levels in the middle levels have at least a q fraction of their edges being bichromatic. (Indeed, otherwise we would have, letting b_k denote the number of bichromatic edges in weight layer k,

$$\begin{aligned}
\frac{\alpha}{2}\cdot\underbrace{\sum_{k=\frac{n}{2}-a\sqrt{n}}^{\frac{n}{2}+a\sqrt{n}-1}|\mathcal{W}_k|}_{\text{total}} &\le \sum_{k=\frac{n}{2}-a\sqrt{n}}^{\frac{n}{2}+a\sqrt{n}-1} b_k\\
&\le \sum_{\substack{k\in[\frac{n}{2}-a\sqrt{n},\frac{n}{2}+a\sqrt{n}-1]\\ b_k>q|\mathcal{W}_k|}}|\mathcal{W}_k| + \sum_{\substack{k\in[\frac{n}{2}a\sqrt{n},\frac{n}{2}+a\sqrt{n}-1]\\ b_k\le q|\mathcal{W}_k|}} q\cdot|\mathcal{W}_k|\\
&\le p\cdot 2a\sqrt{n}\cdot|\mathcal{W}_{n/2}| + q\cdot\sum_{k=\frac{n}{2}-a\sqrt{n}}^{\frac{n}{2}+a\sqrt{n}-1}|\mathcal{W}_k|\\
&\le p\cdot C\cdot\sum_{k=\frac{n}{2}-a\sqrt{n}}^{\frac{n}{2}+a\sqrt{n}-1}|\mathcal{W}_k| + q\cdot\sum_{k=\frac{n}{2}-a\sqrt{n}}^{\frac{n}{2}+a\sqrt{n}-1}|\mathcal{W}_k|.
\end{aligned}$$

So $\frac{\alpha}{2}\cdot\text{total} \le p\cdot C\cdot\text{total} + q\cdot.\text{total}$, which gives $\frac{\alpha}{2} \le \frac{\alpha}{6C}\cdot C + \frac{\alpha}{6} = \frac{\alpha}{3}$, a contradiction.)

Let S be this collection of at least $2a\sqrt{n}p$ weight levels (from the middle ones) that each have at least a q fraction of edges being bichromatic, and write p_i to denote the fraction of bichromatic edges in $\mathcal{W}_i$, so that for each $i \in S$ it holds that $p_i \ge q$. Consider a random chain from 0^n to 1^n. The marginal distribution according to which an edge is drawn from any given fixed weight level i is uniform on $\mathcal{W}_i$, so by linearity, the expected number of bichromatic edges in a random chain is at least $\sum_{i\in S} p_i \ge 2a\sqrt{n}pq = \Omega(\sqrt{n})$, and hence some chain must have that many bichromatic edges. ◀

A.2 Derivation of Theorem 3.11 using Theorem 3 of [18]

The original theorem is stated for $\tau = 1$, with the upper bound being $1 - \Omega(1)$. However, the proof of [18] goes through for our purposes until the very end, where they set $\epsilon \overset{\text{def}}{=} \frac{1}{\sqrt{r}}$ and need to show that

$$e^{-2}\left(1-(1-\epsilon+2\sqrt{\epsilon/r})^{\sqrt{r}}\right) = \Omega(1).$$

More precisely, the proof goes overall as follows: for some realization of the Talagrand function on r variables g_r, we want (for some absolute constant K) that

$$1 - K\tau \geq \mathrm{Stab}_{1-\frac{\tau}{\sqrt{r}}}(g_r) = 1 - 2\,\mathbf{Pr}\Big[\, g_r \circ N_{1-\frac{\tau}{\sqrt{r}}}(x) \neq g_r(x) \,\Big] .$$

That is, one needs to show $\mathbf{Pr}\Big[\, g_r \circ N_{1-\frac{\tau}{\sqrt{r}}}(x) \neq g_r(x) \,\Big] \geq \frac{K}{2}\tau$; and in turn, it is sufficient to prove that for g a random Talagrand function on r variables,

$$\mathbf{E}_g\Big[\ \mathbf{Pr}\Big[\, g \circ N_{1-\frac{\tau}{\sqrt{r}}}(x) \neq g(x) \,\Big]\ \Big] \geq \frac{K}{2}\tau.$$

This is where we slightly adapt the [18] proof. Where they set a parameter ϵ to be equal to $1/\sqrt{r}$ and analyze $\mathbf{E}_g[\mathbf{Pr}[\, g \circ N_{1-2\epsilon}(x) \neq g(x)\,]]$, we set for our purposes $\epsilon \stackrel{\text{def}}{=} \frac{\tau}{2\sqrt{r}}$. The rest of the argument goes through until the very end, where it only remains to show that

$$ae^{-2}\left(1 - (1 - \epsilon + 2\sqrt{\epsilon/r})^{\sqrt{r}}\right) \geq \frac{K}{2}\tau \tag{6}$$

(a being a small constant resulting from the various conditionings in their proof), or equivalently, that $(1 - \epsilon + 2\sqrt{\epsilon/r})^{\sqrt{r}} \leq 1 - \frac{e^2K}{2a}\tau$. But the left-hand side can be rewritten as

$$\begin{aligned}
(1 - \epsilon + 2\sqrt{\epsilon/r})^{\sqrt{r}} &= e^{\sqrt{r}\ln(1-\epsilon+2\sqrt{\epsilon/r})} = e^{\sqrt{r}\ln(1-\tau/2\sqrt{r}+\sqrt{2\tau}/r^{3/4})} \\
&= e^{\sqrt{r}\ln\left(1-\frac{\tau}{2\sqrt{r}}\left(1-\frac{2\sqrt{2}}{\sqrt{r^{1/2}\tau}}\right)\right)} \\
&\leq e^{-\sqrt{r}\cdot\frac{\tau}{2\sqrt{r}}\left(1-\frac{2\sqrt{2}}{\sqrt{r^{1/2}\tau}}\right)} && \left(\text{as } \frac{\tau}{2\sqrt{r}}\left(1-\frac{2\sqrt{2}}{\sqrt{r^{1/2}\tau}}\right) < 1\right) \\
&= e^{-\frac{\tau}{2}\left(1-\frac{2\sqrt{2}}{\sqrt{r^{1/2}\tau}}\right)} \leq e^{-\frac{\tau}{2}(1-\frac{1}{\sqrt{2}})} && \left(\text{as } \tau > \frac{16}{\sqrt{r}}\right) \\
&\leq e^{-\frac{\tau}{7}} \leq 1 - \frac{\tau}{8} \leq 1 - \frac{e^2K}{2a}\tau.
\end{aligned}$$

(first as $\tau < 1$, then for a suitable choice of K)

◀

A.3 Proof of Fact 3.13

We give the proof for m even; by standard techniques, it extends easily to the odd case. For any $m \in 2\mathbb{N}$, define $\mathcal{C}_m$ as the class of functions f generated as follows: let $R = \{\, x \in \{0,1\}^m \ :\ |x| = m/2 \,\}$, and partition R in $|R|/2$ pairs of elements $(x^\ell, \bar{x}^\ell)$. For all $x \in \{0,1\}^m$,

$$f(x) = \begin{cases} 0 & \text{if } |x| < m/2 \\ r_\ell & \text{if } x \in R \text{ and } x = x^\ell \\ 1 - r_\ell & \text{if } x \in R \text{ and } x = \bar{x}^\ell \\ 1 & \text{if } |x| > m/2 \end{cases}$$

where the $|R|/2$ bits r_ℓ are chosen independently and uniformly at random. Clearly, f is balanced, and we have

$$|R| = \binom{m}{m/2} \underset{m\to\infty}{\sim} \sqrt{\frac{2}{\pi}} \cdot \frac{2^m}{\sqrt{m}} \stackrel{\text{def}}{=} \gamma 2^m.$$

Suppose we have a learning algorithm A for $\mathcal{C}_m$ making $q < 2^{Cm}$ membership queries. Fix $0 < \alpha \leq 1$, and $\varepsilon = \alpha/\sqrt{m}$; to achieve error at most ε overall, A must in particular achieve error at most $\frac{\varepsilon}{\gamma} = \sqrt{\frac{\pi}{2}}\alpha$ on R. But after making q queries, there are still at least $t = \gamma 2^m/2 - 2^{Cm} > 0.99\,|R|$ points in R (for m big enough) A has not queried, and hence with values chosen uniformly at random; on each of these points, A is wrong with probability exactly half, and in particular

$$\begin{aligned}\mathbf{Pr}\left[\text{error } \leq \frac{\varepsilon}{\gamma}\right] < \mathbf{Pr}[\text{error } \leq 2\alpha] = \mathbf{Pr}\left[\sum_{i=1}^{t} X_i \leq 2\alpha\,|R|\right] \\ \leq \mathbf{Pr}\left[\sum_{i=1}^{t} X_i \leq \frac{200}{99}\alpha t\right] \\ \leq e^{-\frac{(1-\frac{400}{99}\alpha)^2 t}{2}} = o(1)\end{aligned}$$

with an additive Chernoff bound. This means that with high probability over the choice of the target concept, A will fail to learn it to accuracy $1-\varepsilon$. ◀

A.4 Derivation of the bound $\mathrm{Stab}_{1-\tau}(\mathsf{PAR}'_{k,r}) \leq 1 - 8\varepsilon$

By setting r as stated we get that $r \leq k^2/\ln(1/\varepsilon)$ and the distance between $\mathsf{PAR}'_{k,r}$ and PAR_r becomes $\eta = e^{-k^2/2r} \leq 1/5$. Since we aim at having $\mathrm{ExpBias}_\tau(\mathsf{PAR}'_{k,r}) \leq 1 - 2\varepsilon$, it is sufficient to have $\sqrt{\mathrm{Stab}_{1-\tau}(\mathsf{PAR}'_{k,r})} \leq 1 - 4\varepsilon$; which would in turn be implied by $\mathrm{Stab}_{1-\tau}(\mathsf{PAR}'_{k,r}) \leq 1 - 8\varepsilon$.

By Fact 3.14, it is sufficient to show that $(1-2\eta)^2(1-\tau)^r + 4\eta(1-\eta) \leq 1 - 8\varepsilon$; note that since $\varepsilon < 1/100$ and by our choice of τ,

$$\begin{aligned}(1-2\eta)^2(1-\tau)^r + 4\eta(1-\eta) &\leq \frac{(1-2\eta)^2}{1+100\varepsilon} + 4\eta(1-\eta) \leq (1-2\eta)^2(1-50\varepsilon) + 4\eta(1-\eta) \\ &\leq (1-4\eta+4\eta^2)(1-50\varepsilon) + 4\eta(1-\eta) \\ &= 1 - 4\eta - 50\varepsilon + 200\eta\varepsilon + 4\eta^2 - 200\varepsilon\eta^2 + 4\eta - 4\eta^2 \\ &= 1 - 50\varepsilon + 200\varepsilon\eta(1-\eta) \leq 1 - 50\varepsilon + 32\varepsilon = 1 - 18\varepsilon \\ &\leq 1 - 8\varepsilon.\end{aligned}$$

◀

Dynamics for the Mean-field Random-cluster Model*

Antonio Blanca and Alistair Sinclair

University of California, Berkeley, USA
`{ablanca,sinclair}@cs.berkeley.edu`

Abstract

The random-cluster model has been widely studied as a unifying framework for random graphs, spin systems and random spanning trees, but its dynamics have so far largely resisted analysis. In this paper we study a natural non-local Markov chain known as the Chayes-Machta dynamics for the mean-field case of the random-cluster model, and identify a critical regime (λ_s, λ_S) of the model parameter λ in which the dynamics undergoes an exponential slowdown. Namely, we prove that the mixing time is $\Theta(\log n)$ if $\lambda \notin [\lambda_s, \lambda_S]$, and $\exp(\Omega(\sqrt{n}))$ when $\lambda \in (\lambda_s, \lambda_S)$. These results hold for all values of the second model parameter $q > 1$. In addition, we prove that the local heat-bath dynamics undergoes a similar exponential slowdown in (λ_s, λ_S).

1998 ACM Subject Classification G.3 Probability and Statistics

Keywords and phrases random-cluster model, random graphs, Markov chains, statistical physics, dynamics

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.528

1 Introduction

Background and previous work. Let $H = (V, E)$ be a finite graph. The *random-cluster model* on H with parameters $p \in (0, 1)$ and $q > 0$ assigns to each subgraph $(V, A \subseteq E)$ a probability

$$\mu_{p,q}(A) \propto p^{|A|}(1-p)^{|E|-|A|}q^{c(A)},$$

where $c(A)$ is the number of connected components in (V, A). A is a configuration of the model.

The random-cluster model was introduced in the late 1960s by Fortuin and Kasteleyn [10] as a unifying framework for studying random graphs, spin systems in physics and random spanning trees; see the book [14] for extensive background. When $q = 1$ this model corresponds to the standard Erdős-Rényi model on subgraphs of H, but when $q > 1$ (resp., $q < 1$) the resulting probability measure favors subgraphs with more (resp., fewer) connected components, and is thus a strict generalization.

For the special case of integer $q \geq 2$ the random-cluster model is, in a precise sense, dual to the classical ferromagnetic *q-state Potts model*, where configurations are assignments of spin values $\{1, \ldots, q\}$ to the vertices of H; the duality is established via a coupling of the models (see, e.g., [9]). Consequently, the random-cluster model illuminates much of the physical theory of the Ising/Potts models. Indeed, recent breakthrough work by Beffara

* Antonio Blanca was supported by a U.C. Berkeley Chancellor's Fellowship, an NSF Graduate Research Fellowship and NSF grants CCF-1016896 and CCF-1420934. Alistair Sinclair was supported by NSF grants CCF-1016896 and CCF-1420934.

and Duminil-Copin [1] uses the geometry of the random-cluster model in $\mathbb{Z}^2$ to establish the critical temperature of the q-state Potts model, settling a long-standing conjecture.

At the other extreme, when $q, p \to 0$ and p approaches zero at a slower rate (i.e., $q/p \to 0$) the random-cluster measure $\mu_{p,q}$ converges to the uniform random spanning tree measure on H. Random spanning trees are fundamental probabilistic objects, whose relevance goes back to Kirchhoff's work on electrical networks [16]. In this paper we investigate the dynamics of the random-cluster model, i.e., Markov chains on random-cluster configurations that are reversible w.r.t. $\mu_{p,q}$ and thus converge to it. The dynamics of physical models are of fundamental interest, both as evolutionary processes in their own right and as Markov chain Monte Carlo (MCMC) algorithms for sampling configurations in equilibrium. In both these contexts the central object of study is the *mixing time*, i.e., the number of steps until the dynamics is close to the equilibrium measure $\mu_{p,q}$ starting from any initial configuration. While dynamics for the Ising and Potts models have been widely studied, very little is known about random-cluster dynamics. The main reason for this appears to be the fact that connectivity is a global property which has led to the failure of existing Markov chains analysis tools.

We focus on the *mean-field* case, where H is the complete graph on n vertices. In this case the random-cluster model may be viewed as the standard random graph model $\mathcal{G}_{n,p}$, enriched by a factor that depends on the component structure. As we shall see, the mean-field case is already quite non-trivial; moreover, it has historically proven to be a useful starting point in understanding the dynamics on more general graphs. The structural properties of the mean-field model are already well understood [3, 19]; in particular, it exhibits a phase transition (analogous to that in $\mathcal{G}_{n,p}$) corresponding to the appearance of a "giant" component of linear size. It is natural here to re-parameterize by setting $p = \lambda/n$; the phase transition then occurs at the critical value $\lambda = \lambda_c(q)$ given by

$$\lambda_c(q) = \begin{cases} q & \text{for } 0 < q \leq 2; \\ 2\left(\frac{q-1}{q-2}\right)\log(q-1) & \text{for } q > 2. \end{cases}$$

For $\lambda < \lambda_c(q)$ all components are of size $O(\log n)$ w.h.p.[1], while for $\lambda > \lambda_c(q)$ there is a unique giant component of size θn (for some constant θ that depends on q and λ). The former regime is called the *disordered phase*, and the latter is the *ordered phase*. Henceforth we assume $q > 1$, since the $q < 1$ regime is structurally quite different; the dynamics are trivial for $q = 1$.

Our main object of study is a non-local dynamics known as the *Chayes-Machta (CM) dynamics* [6]. Given a random-cluster configuration (V, A), one step of this dynamics is defined as follows:

(i) *activate* each connected component of (V, A) independently with probability $1/q$;
(ii) remove all edges connecting active vertices;
(iii) add each edge connecting active vertices independently with probability p, leaving the rest of the configuration unchanged.

It is easy to check that this dynamics is reversible w.r.t. $\mu_{p,q}$ [6]. Until now, the mixing time of the CM dynamics has not been rigorously established for any non-trivial random-cluster measure $\mu_{p,q}$ on any graph. Our goal in this paper is to analyze the CM dynamics in the mean-field case for all values of $q > 1$ and all values of $\lambda > 0$.

[1] We say that an event occurs *with high probability (w.h.p.)* if it occurs with probability approaching 1 as $n \to \infty$.

For integer q, the CM dynamics is a close cousin of the well studied and widely used *Swendsen-Wang (SW) dynamics* [20]. The SW dynamics is primarily a dynamics for the Ising/Potts model, but it may alternatively be viewed as a Markov chain for the random-cluster model using the coupling of these measures mentioned earlier. However, the SW dynamics is only well-defined for integer q, while the random-cluster model makes perfect sense for all $q > 0$. The CM dynamics was introduced precisely in order to allow for this generalization.

The SW dynamics for the mean-field case is fully understood for $q = 2$: recent results of Long, Nachmias, Ning and Peres [18], building on earlier work of Cooper, Dyer, Frieze and Rue [7], show that the mixing time is $\Theta(1)$ for $\lambda < \lambda_c$, $\Theta(\log n)$ for $\lambda > \lambda_c$, and $\Theta(n^{1/4})$ for $\lambda = \lambda_c$. Until recently, the picture for integer $q \geq 3$ was much less complete: Huber [15] gave bounds of $O(\log n)$ and $O(n)$ on the mixing time when λ is far below and far above λ_c respectively, while Gore and Jerrum [13] showed that at the critical value $\lambda = \lambda_c$ the mixing time is $\exp(\Omega(\sqrt{n}))$. All these results were developed for the Ising/Potts model, so their relevance to the random-cluster model is limited to the case of integer q. In work that appeared after the submission of this manuscript [2], Galanis, Štefankovič and Vigoda [11] provide a more comprehensive analysis of the $q \geq 3$ mean-field case. Finally, for the very different case of the d-dimensional torus, Borgs et al. [4, 5] proved exponential lower bounds for the mixing time of the SW dynamics for $\lambda = \lambda_c$ and q sufficiently large.

Our work is the first to provide tight bounds for the mixing time of any random-cluster dynamics for general (non-integer) values of q.

Results. To state our results we identify two further critical points, $\lambda_s(q)$ and $\lambda_S(q)$, with the property that $\lambda_s(q) \leq \lambda_c(q) \leq \lambda_S(q)$. (For $1 < q \leq 2$ these three points coincide; for $q > 2$ they are all distinct.) The definitions of these points are somewhat technical and can be found in Section 2.

Our first result shows that the CM dynamics reaches equilibrium very rapidly for λ outside the "critical" window $[\lambda_s, \lambda_S]$. Moreover, our bounds are tight throughout the fast mixing regime.

▶ **Theorem 1.** *For any $q > 1$, the mixing time of the mean-field CM dynamics is $\Theta(\log n)$ for $\lambda \notin [\lambda_s, \lambda_S]$.*

Our next result shows that, *inside* the critical window (λ_s, λ_S), the mixing time is dramatically larger. (We state this result only for $q > 2$ as otherwise the window is empty.)

▶ **Theorem 2.** *For any $q > 2$, the mixing time of the mean-field CM dynamics is $e^{\Omega(\sqrt{n})}$ for $\lambda \in (\lambda_s, \lambda_S)$.*

We now provide an interpretation of the above results. When $q > 2$ the mean-field random-cluster model exhibits a *first-order* phase transition, which means that at criticality ($\lambda = \lambda_c$) the ordered and disordered phases mentioned earlier *coexist* [19], i.e., each contributes about half of the probability mass. (For $q \leq 2$, there is no phase coexistence.) Phase coexistence suggests exponentially slow mixing for most natural dynamics, because of the difficulty of moving between the phases. Moreover, by continuity we should expect that, within a constant-width interval around λ_c, the effect of the non-dominant phase (ordered below λ_c, disordered above λ_c) will still be felt, as it will form a second mode (local maximum) for the random-cluster measure. This leads to so-called *metastable* states near that local maximum from which it is very hard to escape, so slow mixing should persist throughout this interval. Intuitively, the values λ_s, λ_S mark the points at which the local maxima disappear. A similar phenomenon was captured in the case of the Potts model by Cuff *et al.* [8]. Our results make the above picture for the dynamics rigorous for the random-cluster model for all $q > 2$; notably, in contrast to the Potts model, in the random-cluster model metastability affects the mixing time on *both* sides of λ_c. Note that our results leave open the behavior of the mixing time exactly at λ_s and λ_S.

As a byproduct of our main results above, we deduce new bounds on the mixing time of *local* dynamics for the random-cluster model (i.e., dynamics that modify only a constant-size region of the configuration at each step). For definiteness we consider the canonical *heat-bath (HB) dynamics*, which in each step updates a single edge of the current configuration (V, A) as follows:

(i) pick an edge $e \in E$ u.a.r;
(ii) replace A by $A \cup \{e\}$ with probability $\frac{\mu_{p,q}(A\cup\{e\})}{\mu_{p,q}(A\cup\{e\})+\mu_{p,q}(A\setminus\{e\})}$, else by $A \setminus \{e\}$.

Local dynamics for the random-cluster model are currently very poorly understood (but see [12] for the special case of graphs with bounded tree-width). However, in a recent surprising development, Ullrich [21, 22] showed that the mixing time of the heat-bath dynamics on any graph differs from that of the SW dynamics by at most a poly(n) factor. Thus the previously known bounds for SW translate to bounds for the heat-bath dynamics for integer q. By adapting Ullrich's technology to our CM setting, we are able to obtain a similar translation of our results, thus establishing the first non-trivial bounds on the mixing time of the mean-field heat-bath dynamics for all $q > 1$.

▶ **Theorem 3.** *For any $q > 1$, the mixing time of the heat-bath dynamics for the mean-field random-cluster model is $\tilde{O}(n^4)$ for $\lambda \notin [\lambda_s, \lambda_S]$, and $e^{\Omega(\sqrt{n})}$ for $\lambda \in (\lambda_s, \lambda_S)$.*

The $\tilde{O}$ here hides polylogarithmic factors. We conjecture that the upper bound should be $\tilde{O}(n^2)$ for all $\lambda \notin [\lambda_s, \lambda_S]$; the additional n^2 factor is inherent in Ullrich's spectral approach.

We conclude this introduction with some brief remarks about our techniques. Both our upper and lower bounds on the mixing time of the CM dynamics focus on the evolution of the one-dimensional random process given by the size of the largest component (which approaches θn for $\lambda > \lambda_c$ and $\Theta(\log n)$ for $\lambda < \lambda_c$). A key ingredient in our analysis is a function that describes the expected change, or "drift", of this random process at each step; the critical points λ_s and λ_S discussed above arise naturally from consideration of the zeros of this drift function.

For our upper bounds, we construct a multiple-phase coupling of the evolution of two arbitrary configurations, showing that they converge in $O(\log n)$ steps; this coupling is similar in flavor to that used by Long *et al.* [18] for the SW dynamics for $q = 2$, but there are additional complexities in that our analysis has to identify the "slow mixing" window (λ_s, λ_S) for $q > 2$, and also has to contend with the fact that only a subset of the vertices (rather than the whole graph, as in SW) are active at each step. This latter issue is handled using precise concentration bounds for the number of active vertices, tailored estimates for the component structure of random graphs and a new coupling for pairs of binomial random variables.

For our exponential lower bounds we use the drift function to identify the metastable states mentioned ealier from which the dynamics cannot easily escape. For both upper and lower bounds, we have to handle the sub-critical and super-critical cases, $\lambda < \lambda_c$ and $\lambda > \lambda_c$, separately, even though our final results are insensitive to λ_c, because the structure of typical configurations differs in the two cases.

2 Preliminaries

In this section we gather a number of standard definitions and background results that we will refer to repeatedly in our proofs. For those results that are not available in the literature, we provide proofs in the full version of this paper [2].

Mixing time. Let P be the transition matrix of a finite, ergodic Markov chain M with state space Ω and stationary distribution π. The mixing time of M is defined by

$$\tau_{\text{mix}} = \max_{z\in\Omega} \min_t \left\{ ||P^t(z,\cdot) - \pi(\cdot)||_{\text{TV}} \le 1/4 \right\}$$

where $||\mu - \nu||_{\text{TV}} = \max_{A\subset\Omega} |\mu(A) - \nu(A)|$ is the total variation distance between distributions μ and ν.

A *(one step) coupling* of the Markov chain M specifies for every pair of states $(X_t, Y_t) \in \Omega^2$ a probability distribution over (X_{t+1}, Y_{t+1}) such that the processes $\{X_t\}$ and $\{Y_t\}$, viewed in isolation, are faithful copies of M, and if $X_t = Y_t$ then $X_{t+1} = Y_{t+1}$. The *coupling time* is defined by

$$T_{\text{coup}} = \max_{x,y\in\Omega} \min_t \{X_t = Y_t | X_0 = x, Y_0 = y\}.$$

For any $\delta \in (0,1)$, the following standard inequality (see, e.g., [17]) provides a bound on the mixing time:

$$\tau_{\text{mix}} \le \min_t \{\Pr[T_{\text{coup}} > t] \le 1/4\} \le O\left(\delta^{-1}\right) \cdot \min_t \{\Pr[T_{\text{coup}} > t] \le 1 - \delta\}. \tag{1}$$

Random graphs. Let G_d be distributed as a $G(n, p = d/n)$ random graph where $d > 0$. Let $\mathcal{L}(G_d)$ denote the largest component of G_d and let $L_i(G_d)$ be the *size* of the i-th largest component of G_d. (Thus, $L_1(G_d) = |\mathcal{L}(G_d)|$.) In our proofs we will use several facts about the random variables $L_i(G_d)$, which we gather here for convenience.

▶ **Lemma 4** ([18, Lem. 5.7]). *Let $I(G_d)$ denote the number of isolated vertices in G_d. If $d = O(1)$, then there exists a constant $A > 0$ such that $\Pr[I(G_d) > An] = 1 - O\left(n^{-1}\right)$.*

▶ **Lemma 5.** *If $d = O(1)$, then $L_2(G_d) < 2n^{11/12}$ with probability $1 - O\left(n^{-1/12}\right)$ for sufficiently large n.*

▶ **Lemma 6** ([7, Lem. 7]). *If $d < 1$ is bounded away[2] from 1, then $L_1(G_d) = O(\log n)$ with probability $1 - O\left(n^{-1}\right)$.*

For $d > 1$, let $\beta = \beta(d)$ be the unique positive root of the equation

$$e^{-dx} = 1 - x. \tag{2}$$

▶ **Lemma 7.** *Let $\widetilde{G}_{d_n}$ be distributed as a $G(n+m, d_n/n)$ random graph where $\lim_{n\to\infty} d_n = d$ and $|m| = o(n)$. Assume $1 < d_n = O(1)$ and d_n is bounded away from 1 for all $n \in \mathbb{N}$. Then, $L_2(\widetilde{G}_{d_n}) = O(\log n)$ with probability $1 - O\left(n^{-1}\right)$, and for $A = o(\log n)$ and n large enough, there exists a constant $c > 0$ such that*

$$\Pr[|L_1(\widetilde{G}_{d_n}) - \beta(d)n| > |m| + A\sqrt{n}] \le e^{-cA^2}.$$

▶ **Corollary 8.** *With the same notation as in Lemma 7, $|\operatorname{E}[L_1(\widetilde{G}_{d_n})] - \beta(d)n| < |m| + O(\sqrt{n})$.*

▶ **Lemma 9** ([13, Lem. 6]). *If $d < 1$ is bounded away from 1, then $L_1(G_d) = O(\sqrt{n})$ with probability $1 - e^{-\Omega(\sqrt{n})}$.*

[2] We say that d is bounded away from a if there exists a constant ξ such that $|d - a| \ge \xi$.

The random-cluster model. Recall from the introduction that the mean-field random-cluster model exhibits a phase transition at $\lambda = \lambda_c(q)$ (see [3]): in the sub-critical regime $\lambda < \lambda_c$ the largest component is of size $O(\log n)$, while in the super-critical regime $\lambda > \lambda_c$ there is a unique giant component of size $\sim\theta_r n$, where $\theta_r = \theta_r(\lambda, q)$ is the largest $x > 0$ satisfying the equation

$$e^{-\lambda x} = 1 - \frac{qx}{1+(q-1)x}. \tag{3}$$

(Note that, as expected, this equation is identical to (2) when $q = 1$.)

Drift function. As indicated in the introduction, our analysis relies heavily on understanding the evolution of the size of the largest component under the CM dynamics. To this end, for fixed λ and q let $\phi(\theta)$ be the largest $x > 0$ satisfying the equation

$$e^{-\lambda x} = 1 - \frac{qx}{1+(q-1)\theta}. \tag{4}$$

This equation corresponds to (2) for a $G(\alpha n, \lambda/n)$ random graph where $\alpha = (1+(q-1)\theta)q^{-1}$. Thus, $\phi(\theta) = \beta(\alpha\lambda)$ and consequently ϕ is well-defined when $\alpha\lambda > 1$. In particular, ϕ is well-defined in the interval $(\theta_{\min}, 1]$, where $\theta_{\min} = \max\{(q-\lambda)/\lambda(q-1), 0\}$.

We will see in Section 3 that for a configuration with a unique "large" component of size θn, the expected "drift" in the size of the largest component will be determined by the sign of the function $f(\theta) = \theta - \phi(\theta)$: $f(\theta) > 0$ corresponds to a negative drift and $f(\theta) < 0$ to a positive drift. Thus, let

$$\lambda_s = \max\{\lambda \le \lambda_c : f(\theta) > 0 \ \ \forall\theta \in (\theta_{\min}, 1]\}, \text{ and}$$

$$\lambda_S = \min\{\lambda \ge \lambda_c : f(\theta)(\theta - \theta_r) > 0 \ \ \forall\theta \in (\theta_{\min}, 1]\}.$$

In words, λ_s and λ_S are the maximum and minimum values, respectively, of λ for which the drift in the size of the largest component is always in the right direction (i.e., towards 0 in the sub-critical case and towards $\theta_r n$ in the super-critical case). The following lemma reveals basic information about these quantities.

▶ **Lemma 10.** *For $q \le 2$, $\lambda_s = \lambda_c = \lambda_S = q$; and for $q > 2$, $\lambda_s < \lambda_c < \lambda_S = q$.*

For integer $q \ge 3$, λ_s corresponds to the threshold β_s in the mean-field q-state Potts model at which the local (Glauber) dynamics undergoes an exponential slowdown [8]. In fact, a change of variables reveals that $\lambda_s = 2\beta_s$ for the specific mean-field Potts model normalization in [8].

In Figure 1 we sketch f in its only two qualitatively different regimes: $q \le 2$ and $q > 2$. The following lemma provides bounds for the drift of the size of the largest component under CM steps.

▶ **Lemma 11.** *For all $\theta \in (\theta_{\min}, 1]$,*

(i) *If $\lambda < \lambda_s$, there exists a constant $\delta > 0$ such that $f(\theta) \ge \delta$.*

(ii) *When $\lambda > \lambda_S$, if $\theta > \theta_r$, then $\theta \ge \phi(\theta) \ge \theta_r$ and if $\theta < \theta_r$, then $\theta \le \phi(\theta) \le \theta_r$.*

(iii) *If $\lambda > \lambda_S$, there exists a constant $\delta \in (0,1)$ such that $\delta|\theta - \theta_r| \le |\phi(\theta) - \theta|$.*

Binomial coupling. In our coupling constructions we will use the following fact about the coupling of two binomial random variables.

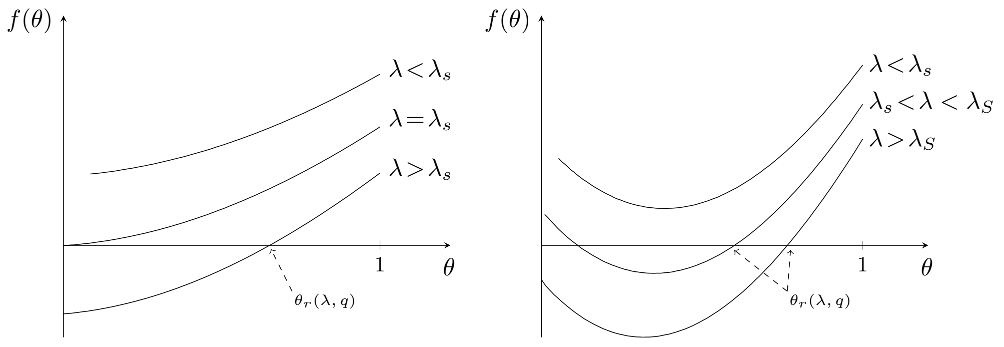


Figure 1 Sketch of the function f. (Left figure corresponds to $q \leq 2$ and right figure to $q > 2$.)

▶ **Lemma 12.** *Let X and Y be binomial random variables with parameters m and r, where $r \in (0,1)$ is a constant. Then, for any $y \in \mathbb{N}$, there exists a coupling (X,Y) such that for a suitable constant $\gamma = \gamma(r) > 0$,*

$$\Pr[X - Y = y] \geq 1 - \frac{\gamma y}{\sqrt{m}}.$$

Moreover if $y = a\sqrt{m}$ for a fixed constant a, then $\gamma a < 1$.

Hitting time for supermartingales. We will require the following easily derived hitting time estimate.

▶ **Lemma 13.** *Consider the stochastic process $\{Z_t\}$ such that $Z_t \in [-n,n]$ for all $t \geq 0$. Assume $Z_0 > a$ for some $a \in [-n,n]$ and let $T = \min\{t > 0 : Z_t \leq a\}$. Suppose $\mathrm{E}[Z_{t+1} - Z_t | \mathcal{F}_t] \leq -A$, where $A > 0$ and $\mathcal{F}_t$ is the history of the first t steps. Then, $\mathrm{E}[T] \leq 4n/A$.*

3 Mixing time upper bounds

In this section we prove the upper bound portion of Theorem 1 from the introduction.

▶ **Theorem 14.** *Consider the CM dynamics for the mean-field random-cluster model with parameters $p = \lambda/n$ and q where $\lambda > 0$ and $q > 1$ are constants independent of n. If $\lambda \notin [\lambda_s, \lambda_S]$, then $\tau_{\text{mix}} = O(\log n)$.*

Proof Sketch. Consider two copies $\{X_t\}$ and $\{Y_t\}$ of the CM dynamics starting from two arbitrary configurations X_0 and Y_0. We design a coupling (X_t, Y_t) of the CM steps and show that $\Pr[X_T = Y_T] = \Omega(1)$ for some $T = O(\log n)$; the result then follows from (1). The coupling consists of four phases. In the first phase $\{X_t\}$ and $\{Y_t\}$ are run independently. In Section 3.1 we establish that after $O(\log n)$ steps $\{X_t\}$ and $\{Y_t\}$ each have at most one large component with probability $\Omega(1)$. We call a component *large* if it contains at least $2n^{11/12}$ vertices; otherwise it is *small*.

In the second phase, $\{X_t\}$ and $\{Y_t\}$ also evolve independently. In Sections 3.2 and 3.3 we show that, conditioned on the success of Phase 1, after $O(\log n)$ steps with probability $\Omega(1)$ the largest components in $\{X_t\}$ and $\{Y_t\}$ have sizes close to their expected value: $O(\log n)$ in the sub-critical case and $\sim\theta_r n$ in the super-critical case. In the third phase, $\{X_t\}$ and $\{Y_t\}$ are coupled to obtain two configurations with the same component structure. This coupling, described in Section 3.4, makes crucial use of the binomial coupling of Section 2, and conditioned on a successful conclusion of Phase 2 succeeds with probability $\Omega(1)$ after $O(\log n)$ steps. In the last phase, a straightforward coupling is used to obtain two identical configurations from configurations with the same component structure. This coupling is described in Section 3.5 and succeeds w.h.p. after $O(\log n)$ steps, conditioned on the success of the previous phases.

Putting all this together, there exists a coupling (X_t, Y_t) such that, after $T = O(\log n)$ steps, $X_T = Y_T$ with probability $\Omega(1)$. The reminder of this section fleshes out the above proof sketch. ◀

We now introduce some notation that will be used throughout the rest of the paper. As before, we will use $\mathcal{L}(X_t)$ for the largest component in X_t and $L_i(X_t)$ for the *size* of the i-th largest component of X_t. (Thus, $L_1(X_t) = |\mathcal{L}(X_t)|$.) For convenience, we will sometimes write $\theta_t n$ for $L_1(X_t)$. Also, we will use $\mathcal{E}_t$ for the event that $\mathcal{L}(X_t)$ is activated, and A_t for the number of activated vertices at time t.

3.1 Convergence to configurations with a unique large component

▶ **Lemma 15.** *For any starting random-cluster configuration X_0, there exists $T = O(\log n)$ such that X_T has at most one large component with probability $\Omega(1)$.*

Proof. Let N_t be the number of new large components created in sub-step (iii) of the CM dynamics at time t. If $A_t < 2n^{11/12}$, then $N_t = 0$. Together with Lemma 5 this implies that $\Pr[N_t > 1 | X_t, A_t = a] \le a^{-1/12}$ for all $a \in [0, n]$. Thus,

$$\begin{aligned}
\mathrm{E}[N_t|X_t] &= \sum_{a=0}^{n} \mathrm{E}[N_t|X_t, A_t = a] \Pr[A_t = a|X_t] \\
&\le \sum_{a=0}^{n} \Big(\Pr[N_t \le 1|X_t, A_t = a] + \frac{a}{2n^{11/12}} \Pr[N_t > 1|X_t, A_t = a]\Big) \Pr[A_t = a|X_t] \\
&\le \sum_{a=0}^{n} \left(1 + \frac{a}{2n^{11/12}} \frac{1}{a^{1/12}}\right) \Pr[A_t = a|X_t] \le 2.
\end{aligned}$$

Let K_t be the number of large components in X_t and let C_t be the number of activated large components in sub-step (i) of the CM dynamics at time t. Then,

$$\mathrm{E}[K_{t+1}|X_t] = K_t - \mathrm{E}[C_t|X_t] + \mathrm{E}[N_t|X_t] \le K_t - \frac{K_t}{q} + 2 \le \left(1 - \frac{1}{2q}\right) K_t$$

provided $K_t \ge 4q$. Assuming that $K_t \ge 4q$ for all $t < T$, we have

$$\mathrm{E}[K_T|X_0] \le \left(1 - \frac{1}{2q}\right)^T K_0.$$

Hence, Markov's inequality implies that $K_T < 4q$ w.h.p. for some $T = O(\log n)$. If at time T the remaining K_T large components become active, then $K_{T+1} \le 1$ w.h.p. by Lemma 5. All K_T components become active simultaneously with probability at least q^{-4q} and thus $K_{T+1} \le 1$ with probability $\Omega(1)$, as desired. ◀

3.2 Convergence to typical configurations: the sub-critical case

▶ **Lemma 16.** *Let $\lambda < \lambda_s$; if X_0 has at most one large component, then there exists $T = O(\log n)$ such that $L_1(X_T) = O(\log n)$ with probability $\Omega(1)$.*

The following fact will be used in the proof. Let $\xi = \sqrt{2n^{23/12}\log n}$.

▶ **Fact 17.** *If X_t has at most one large component, then for sufficiently large n each of the following holds with probability $1 - O(n^{-1})$:*

(i) *If $\mathcal{L}(X_t)$ is inactive, then all new components in X_{t+1} have size $O(\log n)$.*

(ii) *If $\mathcal{L}(X_t)$ is active, then $A_t \in J_t := \left[L_1(X_t) + \frac{n-L_1(X_t)}{q} - \xi, L_1(X_t) + \frac{n-L_1(X_t)}{q} + \xi\right]$.*

(iii) *If there is no large component in X_t, then the largest new component in X_{t+1} have size $O(\log n)$.*

Proof of Lemma 16. If X_0 has at most one large component, then it is easy to check that X_t retains this property for $O(\log n)$ CM steps w.h.p. Thus, we condition on this event throughout this phase. We show first that one step of the CM dynamics contracts the size of the largest component in expectation.

For ease of notation set $\Theta_{\rm s} := \theta_{\min}$, with $\theta_{\min}$ defined as in Section 2. Note that $(\Theta_{\rm s} + (1-\Theta_{\rm s})q^{-1})\lambda = 1$. Hence, if $L_1(X_t) = \Theta_{\rm s}n$ and $\mathcal{L}(X_t)$ is activated, then the percolation step (sub-step (iii) of the CM dynamics) is critical with non-negligible probability. This makes the analysis in the neighborhood of $\Theta_{\rm s}n$ more delicate.

We consider first the case where $\theta_t \geq \Theta_{\rm s} + \varepsilon$ for some small constant $\varepsilon > 0$ to be chosen later. By Fact 17(i), if $\mathcal{L}(X_t)$ is inactive all the new components have size $O(\log n)$ with probability $1 - O(n^{-1})$. Thus,

$$\mathrm{E}[L_1(X_{t+1}) \mid X_t, \neg\mathcal{E}_t] \leq L_1(X_t) + O(1) = \theta_t n + O(1). \tag{5}$$

To bound $\mathrm{E}[L_1(X_{t+1}) \mid X_t, \mathcal{E}_t]$, let $h^+(\theta_t) = \theta_t n + (1-\theta_t)q^{-1}n + \xi$ and let $\ell^+(\theta_t)$ be a random variable distributed as the size of the largest component of a $G(h^+(\theta_t), p)$ random graph. Then, by Fact 17(ii) we have

$$\begin{aligned}\mathrm{E}[L_1(X_{t+1}) \mid X_t, \mathcal{E}_t] &\leq \sum_{a\in J_t} \mathrm{E}[L_1(X_{t+1}) \mid X_t, \mathcal{E}_t, A_t = a]\Pr[A_t = a \mid X_t, \mathcal{E}_t] + O(1) \\ &\leq \mathrm{E}[L_1(X_{t+1}) \mid X_t, \mathcal{E}_t, A_t = h^+(\theta_t)] + O(1) = \mathrm{E}[\ell^+(\theta_t)] + O(1).\end{aligned}$$

When $\theta_t \geq \Theta_{\rm s} + \varepsilon$, $G(h^+(\theta_t), p)$ is a super-critical random graph. Thus, Corollary 8 implies

$$\mathrm{E}[L_1(X_{t+1}) \mid X_t, \mathcal{E}_t] \leq \phi(\theta_t)n + O(\xi), \tag{6}$$

where $\phi(\theta_t)$ is defined as in (4). Since $\lambda < \lambda_s$, by Lemma 11 there exists a constant $\delta > 0$ such that $\theta_t - \phi(\theta_t) \geq \delta$. Therefore, putting (5) and (6) together, we have

$$\mathrm{E}[L_1(X_{t+1}) \mid X_t] \leq (1-q^{-1})\theta_t n + q^{-1}\phi(\theta_t)n + O(\xi) \leq \theta_t n - \delta q^{-1}n + O(\xi). \tag{7}$$

As mentioned before, in a close neighborhood of $\Theta_{\rm s}$ the percolation step is critical with non-negligible probability, so when $\theta_t \in (\Theta_{\rm s} - \varepsilon, \Theta_{\rm s} + \varepsilon)$ we use monotonicity to simplify the analysis. Namely, we assume that $\theta_t = \Theta_{\rm s} + \varepsilon$ and use the previous steps to obtain (7). Thus, there exists a constant $\gamma > 0$ such that for all $\theta_t > \Theta_{\rm s} - \varepsilon$:

$$\mathrm{E}[L_1(X_{t+1}) - L_1(X_t) \mid X_t] \leq -\gamma n.$$

Let $\tau = \min\{t > 0 : L_1(X_t) \leq (\Theta_{\rm s} - \varepsilon)n\}$. Lemma 13 implies $\mathrm{E}[\tau] \leq 4/\gamma$ and thus $\Pr[\tau > 8/\gamma] \leq 1/2$ by Markov's inequality. Hence, $L_1(X_T) \leq (\Theta_{\rm s} - \varepsilon)n$ for some $T = O(1)$ with probability $\Omega(1)$.

To conclude, we show that after $O(\log n)$ additional steps the largest component has size $O(\log n)$ with probability $\Omega(1)$. If $L_1(X_T) \leq (\Theta_s - \varepsilon)n$ and $\mathcal{L}(X_T)$ is activated, then the definition of Θ_s implies that the percolation step of the CM dynamics is sub-critical, and thus X_{T+1} has no large component w.h.p. Hence, X_{T+1} has no large component with probability $\Omega(1)$. Now, by Fact 17(iii) and a union bound, all the new components created during the $O(\log n)$ steps immediately after time $T+1$ have size $O(\log n)$ w.h.p. Another union bound over components shows that during these $O(\log n)$ steps, every component in X_{T+1} is activated w.h.p. Thus, after $O(\log n)$ steps the largest component in the configuration has size $O(\log n)$ with probability $\Omega(1)$, which establishes Lemma 16. ◀

The reader is referred to the full version [2] for the proof of Fact 17, as well as some of the details omitted from the proof of Lemma 16.

3.3 Convergence to typical configurations: the super-critical case

▶ **Lemma 18.** *Let $\lambda > \lambda_S = q$ and $\Delta_t := |L_1(X_t) - \theta_r n|$. If X_0 has at most one large component, then for some $T = O(\log n)$ there exists a constant $c > 0$ such that $\Pr[\,\Delta_T > A\sqrt{cn}\,] < 1/A$ for all $A > 0$.*

The following facts, whose proofs can be found in the full version [2], will be useful. Let $\xi(r) = \sqrt{nr\log n}$, $\Theta_S := 1 - q/\lambda$ and $\mu_t = L_1(X_t) + \frac{n-L_1(X_t)}{q}$.

▶ **Fact 19.** *If X_0 has at most one large component, then there exists $T = O(\log n)$ such that with probability $\Omega(1)$: $L_1(X_T) > (\Theta_S + \varepsilon)n$, $L_2(X_T) = O(\log n)$ and $\sum_{j\geq 2} L_j(X_T)^2 = O(n)$. Moreover, once these properties are obtained they are preserved for a further $T' = O(\log n)$ CM steps w.h.p.*

▶ **Fact 20.** *Assume X_t has exactly one large component and all its other components have size at most $r < 2n^{11/12}$. Then, for a small constant $\varepsilon > 0$ and sufficiently large n, each of the following holds with probability $1 - O\left(n^{-1}\right)$:*

(i) *If $\mathcal{L}(X_t)$ is inactive and $L_1(X_t) > (\Theta_S + \varepsilon)n$, then all new components in X_{t+1} have size $O(\log n)$.*

(ii) *If $\mathcal{L}(X_t)$ is active, then $A_t \in J_{t,r} := [\mu_t - \xi(r), \mu_t + \xi(r)]$ and $G(A_t, p)$ is a super-critical random graph.*

Proof of Lemma 18. We show that one step of the CM dynamics contracts Δ_t in expectation. Observe that by Fact 19 we may assume X_0 is such that $L_1(X_0) > (\Theta_S + \varepsilon)n$, $L_2(X_0) = O(\log n)$ and $\sum_{j\geq 2} L_j(X_0)^2 = O(n)$, and that X_t retains these properties for the $O(\log n)$ steps of this phase w.h.p. Consequently, if $\mathcal{L}(X_t)$ is inactive, then $L_1(X_{t+1}) = L_1(X_t)$ with probability $1 - O(n^{-1})$ by Fact 20(i). Hence,

$$\mathrm{E}[\Delta_{t+1} \mid X_t, \neg\mathcal{E}_t] \leq \mathrm{E}\left[\,|L_1(X_{t+1}) - L_1(X_t)| \mid X_t, \neg\mathcal{E}_t\right] + |L_1(X_t) - \theta_r n| \leq \Delta_t + O(1). \quad (8)$$

To bound $\mathrm{E}[\Delta_{t+1} \,|\, X_t, \mathcal{E}_t]$, let $M_t = A_t - \mu_t$ and let $\ell_t(m)$ denote the size of the largest component of a $G(\mu_t + m, p)$ random graph. Also, let $\Delta'_{t+1} := |L_1(X_{t+1}) - \phi(\theta_t)n|$. Note that, conditioned on $M_t = m$, $L_1(X_{t+1})$ and $\ell_t(m)$ have the same distribution. Moreover, if $A_t \in J_{t,r}$ then $M_t \in J'_{t,r} := [-\xi(r), \xi(r)]$. Hence, Fact 20(ii) with $r = O(\log n)$ implies

$$\begin{aligned} E[\Delta'_{t+1} \mid X_t, \mathcal{E}_t] \quad &\leq \quad \sum_{m \in J'_{t,r}} \mathrm{E}[\Delta'_{t+1} \mid X_t, \mathcal{E}_t, M_t = m]\ \Pr[M_t = m \mid X_t, \mathcal{E}_t] + O(1) \\ &= \quad \sum_{m \in J'_{t,r}} \mathrm{E}[|\ell_t(m) - \phi(\theta_t)n|]\ \Pr[M_t = m \mid X_t, \mathcal{E}_t] + O(1). \end{aligned}$$

Now, by Fact 20(ii), $G(\mu_t + m, p)$ is a super-critical random graph, and thus $\mathrm{E}[|\ell_t(m) - \phi(\theta_t)n|] \leq |m| + O(\sqrt{n})$ by Corollary 8. Hence,

$$E[\Delta'_{t+1} \mid X_t, \mathcal{E}_t] \leq \mathrm{E}[|M_t| \mid X_t, \mathcal{E}_t] + O(\sqrt{n}).$$

Since $\sum_{j\geq 2} L_j(X_t)^2 = O(n)$, it follows from Hoeffding's inequality that $\mathrm{E}[|M_t| \mid X_t, \mathcal{E}_t] = O(\sqrt{n})$ (the explicit calculation is provided in the full version [2]), and thus $E[\Delta'_{t+1} \mid X_t, \mathcal{E}_t] = O(\sqrt{n})$. The triangle inequality then implies

$$\mathrm{E}[\Delta_{t+1} \mid X_t, \mathcal{E}_t] \;\leq\; \mathrm{E}[\Delta'_{t+1} \mid X_t, \mathcal{E}_t] + |\theta_r - \phi(\theta_t)|n \;\leq\; |\theta_r - \phi(\theta_t)|n + O(\sqrt{n}). \tag{9}$$

Putting (8) and (9) together, we have

$$\mathrm{E}[\Delta_{t+1} \mid X_t] \leq (1 - q^{-1})\Delta_t + q^{-1}|\theta_r - \phi(\theta_t)|n + O(\sqrt{n}).$$

By Lemma 11(iii), there exists a constant $\delta \in (0, 1)$ such that $\delta|\theta_t - \theta_r| \leq |\theta_t - \phi(\theta_t)|$. Together Lemma 11(ii), this implies $|\theta_r - \phi(\theta_t)| \leq (1 - \delta)|\theta_t - \theta_r|$. Thus, there exists a constant $\delta' > 0$ such that

$$\mathrm{E}[\Delta_{t+1} \mid X_t] \leq (1 - \delta')\Delta_t + \xi$$

where $\xi = O(\sqrt{n})$. Inducting, $\mathrm{E}[\Delta_t] \leq (1 - \delta')^t\Delta_0 + \xi/\delta'$. Hence, for some $t = O(\log n)$, $\mathrm{E}[\Delta_t] = O(\sqrt{n})$ and so Markov's inequality implies $\Pr[\Delta_t > A\sqrt{cn}] \leq 1/A$ for some constant $c > 0$ and any $A > 0$. ◀

3.4 Coupling to the same component structure

In this section we design a coupling of the CM steps which, starting from two configurations with certain properties (namely, those obtained in Sections 3.2 and 3.3 for the sub-critical and super-critical case respectively), quickly converges to a pair of configurations with the same component structure. (We say that two random-cluster configurations X and Y have the same component structure if $L_j(X) = L_j(Y)$ for all $j \geq 1$.)

The only additional property we will require is that the starting configurations should have a linear number of isolated vertices. Although in Sections 3.2 and 3.3 we do not guarantee this, observe that a single CM step from a configuration with at most one large component activates a linear number of vertices w.h.p., and thus Lemma 4 implies that the new configuration has a linear number of isolated vertices w.h.p. We will focus first on the super-critical case, since a simplified version of the arguments works in the sub-critical case.

▶ **Lemma 21.** *Let $\lambda > q$ and let X_0, Y_0 be random-cluster configurations with $\Omega(n)$ isolated vertices such that: $L_2(X_0) = O(\log n)$, $|L_1(X_0) - \theta_r n| = O(\sqrt{n}\log^2 n)$, $\sum_{j\geq 2} L_j(X_0)^2 = O(n)$ and similarly for Y_0. Then, there exists a coupling of the CM steps such that X_T and Y_T have the same component structure after $T = O(\log n)$ steps with probability $\Omega(1)$.*

Proof. It is straightforward to check that X_t, Y_t retain the above structural properties of X_0, Y_0 for $O(\log n)$ CM steps w.h.p. (The details are provided in the full version [2].)

Our coupling will be a composition of three couplings. Coupling I contracts a certain notion of distance between $\{X_t\}$ and $\{Y_t\}$. This contraction will boost the probability of success of the other two couplings. Coupling II is a one-step coupling which guarantees that the largest components from $\{X_t\}$ and $\{Y_t\}$ have the same size with probability $\Omega(1)$. Coupling III uses the binomial coupling from Lemma 12 to achieve two configurations with the same component structure with probability $\Omega(1)$.

Coupling I: Excluding $\mathcal{L}(X_t)$ and $\mathcal{L}(Y_t)$, consider a maximal matching W_t between the components of X_t and Y_t with the restriction that only components of equal size are matched to each other. Let $M(X_t)$ and $M(Y_t)$ be the components in the matching from X_t and Y_t respectively. Let $D(X_t)$ and $D(Y_t)$ be the complements of $\mathcal{L}(X_t) \cup M(X_t)$ and $\mathcal{L}(Y_t) \cup M(Y_t)$ respectively, and let $d_t := |D(X_t)| + |D(Y_t)|$ where $|\cdot|$ denotes the total number of vertices in the respective components.

The activation of the components in $M(X_t)$ and $M(Y_t)$ is coupled using the matching W_t. That is, $c \in M(X_t)$ and $W_t(c) \in M(Y_t)$ are activated simultaneously with probability $1/q$. The activations of $\mathcal{L}(X_t)$ and $\mathcal{L}(Y_t)$ are also coupled, and the components in $D(X_t)$ and $D(Y_t)$ are activated independently. Let $A(X_t)$ and $A(Y_t)$ denote the set of active *vertices* in X_t and Y_t respectively, and w.l.o.g. assume $|A(X_t)| \geq |A(Y_t)|$. Let R_t be an arbitrary subset of $A(X_t)$ such that $|R_t| = |A(Y_t)|$ and let $Q_t = A(X_t) \setminus R_t$. The percolation step is coupled by establishing an arbitrary vertex bijection $b_t : R_t \to A(Y_t)$ and coupling the re-sampling of each edge $(u, v) \in R_t \times R_t$ with $(b_t(u), b_t(v)) \in A(Y_t) \times A(Y_t)$. Edges within Q_t and in the cut $C_t = R_t \times Q_t$ are re-sampled independently. The following claim establishes the desired contraction in d_t.

▶ **Claim 22.** *Let $\omega(n) = n/\log^4 n$; after $T = O(\log\log n)$ steps, $d_T \leq \omega(n)$ w.h.p.*

Proof. Let $D_a(X_t)$ and $D_a(Y_t)$ be the number of active vertices from $D(X_t)$ and $D(Y_t)$ respectively, and let $\mathcal{F}_t$ be the history of the first t steps. Observe that Coupling I guarantees that R_t and $A(Y_t)$ will have the same component structure internally. However, the vertices in Q_t will contribute to d_{t+1} unless they are part of the new large component, and each edge in C_t could increase d_{t+1} by at most (twice) the size of one component of R_t, which is $O(\log n)$. Thus,

$$\mathrm{E}[d_{t+1} \,|\, A(X_t), A(Y_t), C_t, \mathcal{F}_t] \leq d_t - (|D_a(X_t)| + |D_a(Y_t)|) + |Q_t| + 2|C_t| \times O(\log n). \quad (10)$$

Observe that $\mathrm{E}[|D_a(X_t)| + |D_a(Y_t)| \; |\mathcal{F}_t] = d_t/q$, and $\mathrm{E}[|C_t| \; |A(X_t), A(Y_t), \mathcal{F}_t] = |R_t||Q_t|p \leq \lambda|Q_t|$. Since $|Q_t| = O(\sqrt{n}\log^2 n)$, taking expectations in (10) we get

$$\mathrm{E}[d_{t+1} \,|\, \mathcal{F}_t] \leq d_t - \frac{d_t}{q} + O\left(\sqrt{n}\log^3 n\right) \leq \left(1 - \frac{1}{2q}\right) d_t$$

provided $d_t > \omega(n)$. Thus, Markov's inequality implies $d_T \leq \omega(n)$ for some $T = O(\log\log n)$ w.h.p. Note that for larger values of T, this argument immediately provides stronger bounds for d_T, but neither our analysis nor the order of the coupling time benefits from this. ◀

Coupling II: Assume now that $d_t \leq \omega(n)$ and let $I_{\mathrm{M}}(X_t)$ and $I_{\mathrm{M}}(Y_t)$ denote the isolated vertices in $M(X_t)$ and $M(Y_t)$ respectively. The activation in $X_t \setminus I_{\mathrm{M}}(X_t)$ and $Y_t \setminus I_{\mathrm{M}}(Y_t)$ is coupled as in Coupling I, except we condition on the event that $\mathcal{L}(X_t)$ and $\mathcal{L}(Y_t)$ are activated, which occurs with probability $1/q$. This first part of the activation could activate a different number of vertices from each copy of the chain; let ρ_t be this difference.

First we show that $\rho_t = O(\sqrt{n})$ with probability $\Omega(1)$. By Lemma 18 (with $A = 2$), we have $|L_1(X_t) - L_1(Y_t)| = O(\sqrt{n})$ with probability $\Omega(1)$. If this is the case, then $||D(X_t)| - |D(Y_t)|| = O(\sqrt{n})$. Also, since $\sum_{j\geq 2} L_j(X_t)^2 = O(n)$ and $\sum_{j\geq 2} L_j(Y_t)^2 = O(n)$, by Hoeffding's inequality the numbers of active vertices from $D(X_t)$ and $D(Y_t)$ differ by at most $O(\sqrt{n})$ with probability $\Omega(1)$. Thus, $\rho_t = O(\sqrt{n})$ with probability $\Omega(1)$.

Now we show how to couple the activation in $I_{\mathrm{M}}(X_t)$, $I_{\mathrm{M}}(Y_t)$ in a way such that $|A(X_t)| = |A(Y_t)|$ with probability $\Omega(1)$. The number of active isolated vertices from $I_{\mathrm{M}}(X_t)$ is binomially distributed with parameters $|I_{\mathrm{M}}(X_t)|$ and $1/q$, and similarly for $I_{\mathrm{M}}(Y_t)$. Hence, the activation of the isolated vertices may be coupled using the binomial coupling from Section 2. Since $|I_{\rm M}(X_t)| = |I_{\rm M}(Y_t)| = \Omega(n)$ and $\rho_t = O(\sqrt{n})$, Lemma 12 implies that this coupling corrects the difference ρ_t with probability $\Omega(1)$. If this is the case, then by coupling the edge sampling bijectively as in Coupling I, we ensure that $L_1(X_{t+1}) = L_1(Y_{t+1})$ and $d_{t+1} \leq \omega(n)$ with probability $\Omega(1)$.

Coupling III: Assume $L_1(X_0) = L_1(Y_0)$ and $d_0 \leq \omega(n)$. The component activation is coupled as in Coupling II, but we do not require the two large components to be active; rather, we just couple their activation together.

If $L_1(X_t) = L_1(Y_t)$, then $|D(X_t)| = |D(Y_t)|$ and thus the expected number of active vertices from $D(X_t)$ and $D(Y_t)$ is the same. Consequently, since $d_t \leq \omega(n)$, Hoeffding's inequality implies $\rho_t = O\left(\sqrt{n}\log^{-1} n\right)$ w.h.p. Let $\mathcal{F}_t$ be the event that the coupling of the isolated vertices succeeds in correcting the error ρ_t. Since $|I_{\rm M}(X_t)| = |I_{\rm M}(Y_t)| = \Omega(n)$, $\mathcal{F}_t$ occurs with probability $1 - O(\log^{-1} n)$ by Lemma 12. If this is the case, the updated part of both configurations will have the same component structure; thus, $L_1(X_{t+1}) = L_1(Y_{t+1})$ and $d_{t+1} \leq d_t$. Hence, if $\mathcal{F}_t$ occurs for all $0 \leq t \leq T$, then X_T and Y_T fail to have the same component structure only if at least one of the initial components was never activated. For $T = O(\log n)$ this occurs with at most constant probability. Since $\mathcal{F}_t$ occurs for $T = O(\log n)$ consecutive steps with at least constant probability, then X_T and Y_T have the same component structure with probability $\Omega(1)$. ◀

In the sub-critical case we may assume also that $L_1(X_0)$ and $L_1(Y_0)$ are $O(\log n)$. Therefore, a simplified version of the same coupling works since Coupling II is not necessary.

▶ **Corollary 23.** *If $\lambda < \lambda_s$ and X_0, Y_0 are as in Lemma 21, then there exists a coupling of the CM steps such that X_T and Y_T have the same component structure with probability $\Omega(1)$, for some $T = O(\log n)$.*

3.5 Coupling to the same configuration

▶ **Lemma 24.** *Let X_0 and Y_0 be two random-cluster configurations with the same component structure. Then, there exists a coupling of the CM steps such that after $T = O(\log n)$ steps $X_T = Y_T$ w.h.p.*

Proof. Let B_t a bijection between the vertices of X_t and Y_t. We first describe how to construct B_0. Consider a maximal matching between the components of X_0 and Y_0 with the restriction that only components of equal size are matched to each other. Since the two configurations have the same component structure all components are matched. Using this matching, vertices between matched components are mapped arbitrarily to obtain B_0.

Vertices mapped to themselves we call "fixed". At time t, the component activation is coupled according to B_t. That is, if $B_t(u) = v$ for $u \in X_t$ and $v \in Y_t$, then the components containing u and v are simultaneously activated with probability $1/q$. B_{t+1} is adjusted such that if a vertex w becomes active in both configurations then $B_{t+1}(w) = w$; the rest of the activated vertices are mapped arbitrarily in B_{t+1} and the inactive vertices are mapped like in B_t. The percolation step at time t is then coupled using B_{t+1}. That is, the re-sampling of the active edge $(u, v) \in X_t$ is coupled with the re-sampling of the active edge $(B_{t+1}(u), B_{t+1}(v)) \in Y_t$.

This coupling ensures that the component structures of X_t and Y_t remain the same for all $t \geq 0$. Moreover, once a vertex is fixed it remains fixed forever. The probability that a vertex is fixed in one step is $1/q^2$. Therefore, after $O(\log n)$ steps the probability that a vertex is not fixed is at most $1/n^2$. A union bound over all vertices implies that $X_T = Y_T$ w.h.p. after $T = O(\log n)$ steps. ◀

4 Mixing time lower bounds

In this section we prove the exponential lower bound on the mixing time of the CM dynamics for λ in the critical window (λ_s, λ_S), as stated in Theorem 2 in the introduction. We also prove a $\Omega(\log n)$ lower bound in the "fast mixing" regime, showing that our upper bounds in Section 3 are tight.

Recall from the introduction that when $q = 2$ and $\lambda < \lambda_s = \lambda_c$, the SW dynamics mixes in $\Theta(1)$ steps and thus the CM dynamics requires $\Theta(\log n)$ additional steps to mix. This is due to the fact that the CM dynamics may require as many steps to activate all the components from the initial configuration.

▶ **Theorem 25.** *For any $q > 1$, the mixing time of the CM dynamics is $\exp(\Omega(\sqrt{n}))$ for $\lambda \in (\lambda_s, \lambda_S)$, and $\Omega(\log n)$ for $\lambda \notin [\lambda_s, \lambda_S]$.*

Proof. Note that when $q \le 2$ the interval (λ_s, λ_S) is empty and the exponential lower bound is vacuously true. It is natural to divide the proof into four cases: $\lambda < \lambda_s$, $\lambda \in (\lambda_s, \lambda_c)$, $\lambda \in [\lambda_c, \lambda_S)$ and $\lambda > \lambda_S$. First consider the case when $\lambda < \lambda_s$. Let X_0 be a configuration where all the components have size $\Theta(\log^2 n)$ and let $b = q/(q-1)$. The probability that a particular component is not activated in any of the first $T = \frac{1}{2}\log_b n$ steps is $(1-1/q)^T = n^{-1/2}$. Therefore, the probability that all initial components are activated in the first T steps is $(1-n^{-1/2})^K$ with $K = \Theta(n/\log^2 n)$. Thus, after T steps, $L_1(X_T) = \Theta(\log^2 n)$ w.h.p., and the result follows.

Consider now the case $q > 2$ and $\lambda_c \le \lambda < \lambda_S = q$. Let S be the set of graphs G such that $L_1(G) = \Theta(\sqrt{n})$ and let $X_0 \in S$. Let $\mu := \mathrm{E}[A_0] = n/q$; then by Hoeffding's inequality $\Pr\left[|A_0 - \mu| > \varepsilon n\right] \le 2\exp\left(-2\varepsilon^2\sqrt{n}\right)$. If $A_0 < \mu + \varepsilon n$, the percolation step is sub-critical for sufficiently small ε. Therefore, Lemma 9 implies that $\Pr[X_1 \notin S | X_0 \in S] \le e^{-c\sqrt{n}}$ for some constant $c > 0$. Hence, $\Pr[X_1, ..., X_t \in S | X_0 \in S] \ge 1 - te^{-c\sqrt{n}} \ge 3/4$ for $t = \lfloor e^{c\sqrt{n}}/4 \rfloor$, and the result follows.

The intuition for the other two cases, which are more technically involved, comes directly from Figure 1. When $q > 2$ and $\lambda_s < \lambda < \lambda_c$, the function $f(\theta) = \theta - \phi(\theta)$ has two positive zeros θ^* and θ_r in $(\theta_{\min}, 1]$. Moreover, f is negative in the interval (θ^*, θ_r). Therefore, any configuration with a unique large component of size θn with $\theta \in (\theta^*, \theta_r)$ will drift towards a configuration with a bigger large component. However, a typical random-cluster configuration in this regime does not have a large component. This drift in the incorrect direction is sufficient to prove the exponential lower bound in this regime.

When $\lambda > \lambda_S$, we show that the derivative of f between its unique zero θ_r and 1 is bounded above by a constant. This implies that, starting from the complete graph, it takes at least $\Omega(\log n)$ steps for the size of the largest component to shrink to close to $\theta_r n$. The reader is referred to the full version [2] for the proofs of the last two cases. ◀

5 Local dynamics

In this section we sketch the proof of Theorem 3 from the introduction; the full proof is included in the full version [2]. Consider an arbitrary finite graph $H = (V, E)$ and let $\Omega_{\mathrm{E}} = \{(V, A) : A \subseteq E\}$ be the set of random-cluster configurations on H. Let P be the transition matrix of a finite, ergodic and reversible Markov chain over Ω_{E} with stationary distribution $\mu = \mu_{p,q}$ and eigenvalues $1 = \lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$. The *spectral gap* of P is defined by $\lambda(P) := 1 - \lambda^*$, where $\lambda^* = \max\{|\lambda_2|, |\lambda_n|\}$. Let $\mu_{\min} = \min_{x \in \Omega_{\mathrm{E}}} \mu(x)$; the following bounds on the mixing time are standard (see, e.g., [17]):

$$\lambda^{-1}(P) - 1 \leq \tau_{\mathrm{mix}}(P) \leq \log\left(2e\mu_{\min}^{-1}\right)\lambda^{-1}(P). \tag{11}$$

For $r \in \mathbb{N}$, let $\Omega_{\mathrm{V}} = \{0, 1..., r-1\}^V$ be the set of "r-labelings" of V, and let $\Omega_{\mathrm{J}} = \Omega_{\mathrm{V}} \times \Omega_{\mathrm{E}}$. Assume P can be decomposed as a product of stochastic matrices of the form $P = M(\prod_{i=1}^m T_i)M^*$, where:

(i) M is a $|\Omega_{\mathrm{E}}| \times |\Omega_{\mathrm{J}}|$ matrix indexed by the elements of Ω_{E}, Ω_{J} such that $M(A, (\sigma, B)) \neq 0$ only if $A = B$.

(ii) Each T_i is a $|\Omega_{\mathrm{J}}| \times |\Omega_{\mathrm{J}}|$ matrix indexed by the elements of Ω_{J}, reversible w.r.t. the distribution $\nu = \mu M$ and such that $T_i((\sigma, A), (\tau, B)) \neq 0$ only if $\sigma = \tau$.

(iii) M^* is a $|\Omega_{\mathrm{J}}| \times |\Omega_{\mathrm{E}}|$ matrix indexed by the elements of Ω_{J}, Ω_{E} such that $M^*((\sigma, A), B) = \mathbb{1}(A = B)$.

In words, M assigns a (random) r-labeling to the vertices of H; $(\prod_{i=1}^m T_i)$ performs a sequence of m operations T_i, each of which updates some edges of H; and M^* drops the labels from the vertices.

Consider now the matrix $P_{\mathrm{L}} = M(\frac{1}{m}\sum_{i=1}^m T_i)M^*$. It is straightforward to verify that P_{L} is also reversible w.r.t. μ. The following theorem, which generalizes a recent result of Ullrich [21, 22], relates the spectral gaps of P and P_{L} up to a factor of $O(m \log m)$.

▶ **Theorem 26.** *If M, M^* and T_i are stochastic matrices satisfying (i)–(iii) above, and the T_i's are idempotent commuting matrices, then $\lambda(P_{\mathrm{L}}) \leq \lambda(P) \leq 8m \log m \cdot \lambda(P_{\mathrm{L}})$.*

We pause to note that this fact has a very attractive intuitive basis: P_{L} performs a single update T_i chosen u.a.r., while P performs all m updates T_i, so by coupon collecting one might expect that $O(m \log m)$ P_{L} steps should suffice to simulate a single P step. However, the proof has to take account of the fact that the T_i updates are interleaved with the vertex re-labeling operations M and M^* in P_{L}. The proofs in [21] and [22] are specific to the case where P corresponds to the SW dynamics. Our contribution is the realization that these proofs still go through (without essential modification) under the more general assumptions of Theorem 26, as well as the framework described above that provides a systematic way of deriving P_{L} from P.

The key observation in the proof of Theorem 3 is that we can express P_{CM}, the transition matrix of the CM dynamics, as a product of stochastic matrices as above: specifically, $P_{\mathrm{CM}} = L(\prod_{e \in E} T_e)L^*$ where L is the matrix that assigns a random active-inactive labeling to a random-cluster configuration, T_e samples e with probability p provided both its endpoints are active, and L^* drops the active-inactive labeling from a joint configuration.

Now consider the Markov chain given by the matrix $P_{\mathrm{SU}} = L(\frac{1}{|E|}\sum_{e \in E} T_e)L^*$, which we call the *Single Update (SU) dynamics* (P_{SU} plays the role of the matrix P_{L} above.) The matrices L, L^* and the T_e's satisfy the assumptions in Theorem 26, so we have $\lambda(P_{\mathrm{SU}}) \leq \lambda(P_{\mathrm{CM}}) \leq 8|E| \log |E| \cdot \lambda(P_{\mathrm{SU}})$.

The SU dynamics is very closely related to the heat-bath dynamics defined in the introduction; in fact, their spectral gaps differ by a constant. Hence, Theorem 3 now follows from (11) and Theorems 1 and 2 since in the mean-field case $\log(\mu_{\min}^{-1}) = \tilde{O}(n^2)$.

References

1 V. Beffara and H. Duminil-Copin. The self-dual point of the two-dimensional random-cluster model is critical for $q \geq 1$. *Probability Theory and Related Fields*, 153:511–542, 2012.

2 A. Blanca and A. Sinclair. Dynamics for the mean-field random-cluster model, 2014. Preprint. arXiv:1412.6180v1 [math.PR].

3 B. Bollobás, G. Grimmett, and S. Janson. The random-cluster model on the complete graph. *Probability Theory and Related Fields*, 104(3):283–317, 1996.

4 C. Borgs, J. Chayes, and P. Tetali. Swendsen-Wang algorithm at the Potts transition point. *Probability Theory and Related Fields*, 152:509–557, 2012.

5 C. Borgs, A. Frieze, J.H. Kim, P. Tetali, E. Vigoda, and V. Vu. Torpid mixing of some Monte Carlo Markov chain algorithms in statistical physics. *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 218–229, 1999.

6 L. Chayes and J. Machta. Graphical representations and cluster algorithms II. *Physica A*, 254:477–516, 1998.

7 C. Cooper, M.E. Dyer, A.M. Frieze, and R. Rue. Mixing properties of the Swendsen-Wang process on the complete graph and narrow grids. *Journal of Mathematical Physics*, 41:1499–1527, 2000.

8 P. Cuff, J. Ding, O. Louidor, E. Lubetzky, Y. Peres, and A. Sly. Glauber dynamics for the mean-field Potts model. *Journal of Statistical Physics*, 149(3):432–477, 2012.

9 R.G. Edwards and A.D. Sokal. Generalization of the Fortuin-Kasteleyn-Swendsen-Wang representation and Monte Carlo algorithm. *Physical Review D*, 38(6):2009–2012, 1988.

10 C.M. Fortuin and P.W. Kasteleyn. On the random-cluster model I. Introduction and relation to other models. *Physica*, 57(4):536–564, 1972.

11 A. Galanis, D. Štefankovič, and E. Vigoda. Swendsen-Wang Algorithm on the Mean-Field Potts Model, 2015. Preprint. arXiv:1502.06593v1 [cs.DM].

12 Q. Ge and D. Štefankovič. A graph polynomial for independent sets of bipartite graphs. *Combinatorics, Probability and Computing*, 21(5):695–714, 2012.

13 V.K. Gore and M.R. Jerrum. The Swendsen-Wang process does not always mix rapidly. *Journal of Statistical Physics*, 97(1-2):67–86, 1999.

14 G.R. Grimmett. *The Random-Cluster Model*, volume 333 of *Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 2006.

15 M. Huber. A bounding chain for Swendsen-Wang. *Random Structures & Algorithms*, 22(1):43–59, 2003.

16 G. Kirchhoff. Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Strome gefuhrt wird. *Annalen der Physik und Chemie*, pages 497–508, 1847.

17 D.A. Levin, Y. Peres, and E.L. Wilmer. *Markov Chains and Mixing Times.* American Mathematical Society, 2008.

18 Y. Long, A. Nachmias, W. Ning, and Y. Peres. A power law of order 1/4 for critical mean-field Swendsen-Wang dynamics. *Memoirs of the American Mathematical Society*, 232(1092), 2011.

19 M.J. Luczak and T. Łuczak. The phase transition in the cluster-scaled model of a random graph. *Random Structures & Algorithms*, 28(2):215–246, 2006.

20 R.H. Swendsen and J.S. Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters*, 58:86–88, 1987.

21 M. Ullrich. Rapid mixing of Swendsen-Wang dynamics in two dimensions. *Dissertationes Mathematicae*, 502, 2014.

22 M. Ullrich. Swendsen-Wang is faster than single-bond dynamics. *SIAM Journal on Discrete Mathematics*, 28(1):37–48, 2014.

Correlation in Hard Distributions in Communication Complexity

Ralph Christian Bottesch[1], Dmitry Gavinsky*[2], and Hartmut Klauck†[1,3]

1 **Nanyang Technological University**
50 Nanyang Avenue, Singapore 639798
2 **Institute of Mathematics, Czech Academy of Sciences**
Praha, Czech Republic
3 **Centre for Quantum Technologies, National University of Singapore**
Block S15, 3 Science Drive 2, Singapore 117543

Abstract

We study the effect that the amount of correlation in a bipartite distribution has on the communication complexity of a problem under that distribution. We introduce a new family of complexity measures that interpolates between the two previously studied extreme cases: the (standard) randomised communication complexity and the case of distributional complexity under product distributions.

- We give a tight characterisation of the randomised complexity of Disjointness under distributions with mutual information k, showing that it is $\Theta(\sqrt{n(k+1)})$ for all $0 \leq k \leq n$. This smoothly interpolates between the lower bounds of Babai, Frankl, Simon [4] for the product distribution case ($k = 0$), and the bound of Razborov [22] for the randomised case. The upper bounds improve and generalise what was known for product distributions, and imply that any tight bound for Disjointness needs $\Omega(n)$ bits of mutual information in the corresponding distribution.
- We study the same question in the distributional *quantum* setting, and show a lower bound of $\Omega((n(k+1))^{1/4})$, and an upper bound (via constructing communication protocols), matching up to a logarithmic factor.
- We show that there are total Boolean functions f_d that have distributional communication complexity $O(\log n)$ under all distributions of information up to $o(n)$, while the (interactive) distributional complexity maximised over all distributions is $\Theta(\log d)$ for $n \leq d \leq 2^{n/100}$. This shows, in particular, that the correlation needed to show that a problem is hard can be much larger than the communication complexity of the problem.
- We show that in the setting of one-way communication under product distributions, the dependence of communication cost on the allowed error ϵ is multiplicative in $\log(1/\epsilon)$ – the previous upper bounds had the dependence of more than $1/\epsilon$. This result, for the first time, explains how one-way communication complexity under product distributions is stronger than PAC-learning: both tasks are characterised by the VC-dimension, but have very different error dependence (learning from examples, it costs more to reduce the error).

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases communication complexity, information theory

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.544

* Partially funded by the grant P202/12/G061 of GA ČR and by RVO: 67985840. Most of this work was done while DG was visiting the Centre for Quantum Technologies at the National University of Singapore, and was partially funded by the Singapore Ministry of Education and the NRF.

† This work is funded by the Singapore Ministry of Education (partly through the Academic Research Fund Tier 3 MOE2012-T3-1-009) and by the Singapore National Research Foundation.

1 Introduction

The standard way to attack the problem of showing a lower bound on the randomised communication complexity of a function f is to choose a probability distribution μ on the inputs, and then show that the deterministic distributional complexity is large for f w.r.t. μ – i.e., that any deterministic protocol that computes f with small error under μ must communicate much. This approach eliminates the need to argue about the randomness used by the protocol.[1]

It is well known that this approach can be used without loss of generality, due to von Neumann's minimax theorem (see [20]; the same principle applies to many nonuniform computational models):

$$\max_{\mu} D^{\mu}_{\epsilon}(f) = R_{\epsilon}(f),$$

where $D^{\mu}_{\epsilon}(f)$ denotes the deterministic complexity of protocols that compute f with error ϵ under the distribution μ of input to f, and $R_{\epsilon}(f)$ is the public coin randomised communication complexity of f with worst-case error ϵ.[2]

As a matter of convenience, one first tries to use a simple distribution μ, for instance the uniform distribution, or more generally, product distributions over the inputs to Alice and Bob. This works for some problems, like Inner Product modulo 2 [7]. However, Babai, Frankl, and Simon [4] observed that for the Disjointness problem DISJ one cannot obtain lower bounds larger than $\Omega(\sqrt{n}\log n)$ under *any* product distribution, i.e., they show that an upper bound of $O(\sqrt{n}\log n)$ holds for every product distribution. They also give a lower bound of $\Omega(\sqrt{n})$ under a product distribution. Later, Kalyanasundaram and Schnitger [16] obtained the tight $\Theta(n)$ bound, and Razborov [22] showed that indeed $D^{\mu}_{\epsilon}(DISJ) = \Theta(n)$ for an explicit simple distribution μ, for any sufficiently small constant $\epsilon > 0$ (that such a μ exists is immediate from the result in [16] and the minimax theorem, but their proof does not exhibit such a distribution explicitly). Distributional complexity under product distributions has been also frequently used to show structural properties like direct product theorems (e.g., [15, 12]). Furthermore, distributional communication complexity is the natural average case version of communication complexity, and it makes sense to study this for distributions that are 'easy', in order to get a different model than randomised complexity. It seems natural to measure "easiness" via mutual information.

For many years it was open how large the gap between $R^{I=0}_{\epsilon}(f) = \max_{\mu \text{ product}} D^{\mu}_{\epsilon}(f)$ and $R_{\epsilon}(f)$ (for constant $\epsilon > 0$) can be. Sherstov [25] finally gave a proof that there are total Boolean functions f, where the former is $O(1)$ and the latter is $\Omega(n)$. In his result f is not given explicitly. Very recently Alon et al. [2] give the following optimal explicit separation. Consider the problem where Alice gets a point and Bob a line from a projective plane containing $2^{\Theta(n)}$ points and lines. In this case the VC-dimension of the projective plane is at most 2, which implies that the distributional complexity under any product distribution is at most $O(1)$ (even for one-way protocols), whereas the sign-rank of the communication matrix is $2^{\Omega(n)}$, and hence the randomised (even unbounded error) communication complexity is $\Omega(n)$.

This leaves open a more precise investigation of the *amount* of correlation in μ needed to make $D^{\mu}(f)$ equal to $R(f)$. It is natural to quantify this via the mutual information

[1] We note that the popular information complexity method (see e.g.[5]) also uses distributional complexity, but does not seek to eliminate randomness from protocols.
[2] Throughout the paper we do not consider private coin randomised protocols.

$I(X : Y)$, when the input (X, Y) is drawn from μ. We define the following measure:

$$R_\epsilon^{I \le k}(f) = \max_{\mu : I(X:Y) \le k} D_\epsilon^\mu(f).$$

We note here that the quantity on the right hand side does not change if randomised or deterministic protocols are allowed, because in the distributional setting the randomness can be fixed without increasing the error (under any distribution). The investigation of this measure has been initiated by Jain and Zhang [14] in the setting of one-way communication complexity (we discuss their contribution at the end of Section 1.3). We note that $R_\epsilon^{I \le n}(f) = R_\epsilon(f)$ for all functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$.

This family of complexity measures allows us to investigate how much correlation is needed in the input distribution to get good lower bounds. We have 3 main applications. First, we closely investigate the case of the Disjointness problem. Second, we show that a certain problem exhibits a threshold behaviour, i.e., only with almost maximal correlation can a tight lower bound be proved, and this correlation can also be larger than the actual communication complexity of the problem. Third, we investigate the dependence of one-way communication complexity under product distributions on the allowed error.

1.1 The Disjointness problem

In the *Disjointness problem (DISJ)*, Alice and Bob receive, respectively, subsets $x, y \subseteq \{1, \ldots, n\}$, and their task is to decide whether x and y are disjoint. This is one of the most-studied problem in communication complexity, which arguably has the biggest number of known applications to other models (see [20]). We give a complete characterisation of the information-bounded distributional complexity of Disjointness for all values of $k = I(X : Y)$, both in the randomised and in the quantum case.

▶ **Theorem 1.** *For all $0 \le k \le n$ and constant ϵ we have*
1. $R_\epsilon^{I \le k}(DISJ) = \Theta(\sqrt{n(k+1)})$.
2. $Q_\epsilon^{I \le k}(DISJ) = \tilde{O}((n(k+1))^{1/4})$.
3. $Q_\epsilon^{I \le k}(DISJ) = \Omega((n(k+1))^{1/4})$.

Previously, a lower bound of $\Omega(\sqrt{n})$ was known for a product distribution [4], and the $\Omega(n)$ lower bound by Razborov [22] uses a distribution μ with $I^\mu(X : Y) = \Theta(n)$. Babai et al. [4] also gave an upper bound of $O(\sqrt{n} \log n)$ for the case of product distributions, which we improve by a log-factor. Our results interpolate between the previously-known extreme cases, and also show that one needs input correlation $\Omega(n)$ to prove tight lower bounds. Interestingly, the bounds depend inverse-polynomially on the error probability, except for the extreme cases of zero correlation and of maximal correlation. We also note that a nearly-optimal complexity for randomised protocols can be achieved in a protocol with two rounds of communication (though not in one round).

The tight bound in the randomised case is based on a two-phase protocol, in which the players first remove "uninteresting" elements from their sets, until they are (essentially) small enough to be communicated. For the quantum case this two-phase approach cannot be optimal, because the first phase reveals "too much" information about the input. Therefore we give a completely different protocol for the quantum case, in which the players identify uninteresting elements a priori. This approach is tight up to a log-factor.

1.2 Mutual information in hard distributions

Note that for DISJ the complexity increases with the information parameter, and the randomised communication complexity bound $\Theta(n)$ is reached only once the information in the hard distribution reaches $\Omega(n)$. For other problems like Inner Product mod 2 the tight bound of $\Omega(n)$ is reached already under product distributions [7]. But can the mutual information between the input sides that is required to show a tight lower bound ever be *larger* than the actual communication complexity? I.e., is it ever necessary to use distributions that are (much) more strongly correlated than the communication lower bound we want to show, or is it always possible to prove a tight lower bound for a (total) function f by using a hard distribution with $I(X:Y) \leq poly(R(f))$? A weak example is the quantum complexity of Disjointness, where the tight $\Omega(\sqrt{n})$ bound is only reached when the information reaches $\Omega(n)$, but even here the complexity increases gradually with the information. We resolve this question, although our example is not explicit.

▶ **Theorem 2.** *For every $n \leq d \leq 2^{n/100}$ there is a function $f_d : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that has $R(f_d) = \Theta(\log d)$, but under all bipartite distributions with mutual information less than $n/1000$ the communication bound is $R_{1/10}^{I \leq n/1000}(f_d) \leq O(\log n)$.*

Hence for f_d the complexity stays low until the information is almost maximal, and then shoots up.

1.3 Dependence of $R_\epsilon^{A\to B, I=0}(f)$ on ϵ [3]

Finally, we investigate the error dependence of $R_\epsilon^{I\leq k}(f)$ for arbitrary f. In the unrestricted case, by standard boosting techniques we have $R_\epsilon(f) \leq O(R_{1/3}(f) \cdot \log(1/\epsilon))$. We call a function f and a class C of distributions on the inputs with $\max_{\mu\in C} D_\epsilon^\mu(f) \leq O(\max_{\mu \in C} D_{1/3}^\mu(f) \cdot \log(1/\epsilon))$ *boost-able.* For this definition we require the above to be true for all ϵ. One can easily show that there are distributions μ and functions f, such that e.g. $D_{1/4}^\mu((f) = \Omega(n)$ and $D_{1/3}^\mu(f) = 0$, by placing a hard problem with weight 1/3 in an otherwise constant matrix, so for a fixed distribution μ one cannot in general expect the error dependence to behave nicely.

Boost-ability is a property of a class of distributions. The class of all distributions clearly has the property, but what about the class of distributions with information at most I? In particular, what about $I = 0$?

The issue is particularly interesting for product distributions, because boost-ability can be used to derive upper bounds on $R^{I\leq k}(f)$ from upper bounds on $R^{I=0}(f)$: due to the substate theorem (Fact 4 below), a protocol that solves f under all product distributions with error $\epsilon 2^{-9k/\epsilon}$ can be used to solve f under distributions with $I(X:Y) = k$ with error ϵ, hence boost-ability implies $R_\epsilon^{I\leq k}(f) \leq O((k+1) \cdot R_{1/3}^{I=0}(f)/\epsilon)$.

We will use the super-script "$A \to B$" to denote one-way communication. In this model the class of product distributions is boost-able:

▶ **Theorem 3.** $R_\epsilon^{A\to B,I=0}(f) \leq O(R_{1/3}^{A\to B,I=0}(f) \cdot \log(1/\epsilon))$.

We also show that when the information is between 1 and $n^{1-\Omega(1)}$, then neither randomised nor distributional quantum protocols are, in general, boost-able, see our Corollaries 20 and 27.

[3] The same result has been obtained recently by Molinaro et al. [21] independently. The methods being used in the two works are similar; [21] has been published prior to the current publication, while our results have been presented during a public talk prior to either publication.

It is well known that $R_{1/3}^{A\to B,I=0}(f)=\Theta(VC(f))$ [19], where $VC(f))$ is the VC-dimension of the set of rows of the communication matrix. This even extends to the quantum case [3, 18]. The VC-dimension is also known to characterise the hardness of PAC-learning (see the monograph by Kearns and Vazirani [17]) – in fact, the previous proofs of the upper bound on $R_\epsilon^{A\to B,I=0}(f)$ in terms of VC-dimension have been done by explicitly simulating learning algorithms in the one-way communication model: Random examples are generated using a public coin, and Alice classified the examples in order to teach Bob a row of the communication matrix of f in the PAC sense (examples were generated from the public coin, and Alice labelled those examples spending 1 bit per example).

The main limitation of this approach is that for PAC learning one needs $\Omega(1/\epsilon)$ examples to achieve error $1/\epsilon$. On the other hand, this approach ignores two strengths of the one-way model: First, Alice and Bob know the underlying distribution; second, Alice can do more than simply label examples. One can interpret the one-way communication model under product distributions as a learning model, in which Alice is an (old-fashioned) teacher, who teaches by monologue, but using shared randomness that does not count towards the communication. Does such a teacher offer any advantage over learning from random examples? At first glance no, since both models are characterised by the VC-dimension, and one could conclude that learning from experience is all it takes. Our Theorem 3, however, shows that the final error can be made much smaller when learning from a teacher, comparing to learning "just from experience". Note that in practice $1/\epsilon$ can also easily become the dominating factor in the complexity of a learning algorithm.

The main idea in our protocol is that Alice and Bob can beforehand agree on an ϵ-net among the rows of the communication matrix, and Alice simply sends the name of the nearest row in the net. During a PAC learning algorithm, on the other hand, the ϵ-net is generated from examples, which is more costly.

We can now discuss the previous result of Jain and Zhang [14]. They show that for all total Boolean functions f in the one-way model:

$$R_\epsilon^{A\to B,I\le k}(f)\le O((k+1)\cdot R_{1/3}^{A\to B,I=0}(f)\cdot 1/\epsilon^2\cdot\log(1/\epsilon)).$$

This extends the VC-dimension upper bound to distributions with nonzero information. Their protocol for information-k distributions is constructed by simulating the PAC learning algorithm for the row x, and by generating examples $y', f(x,y')$ using a rejection-sampling protocol. We can improve the error dependence to $1/\epsilon$ by the following idea. Due to the Substate Theorem (Fact 4 below) it is enough to find a protocol that has error $2^{-9k/\epsilon}$ under the product of the marginal distributions of a distribution μ (with information k). But this can be achieved with communication $O((k+1)/\epsilon\cdot R_{1/3}^{A\to B,I=0}(f))$ according to Theorem 3.

2 Preliminaries and Definitions

2.1 Information Theory

We refer to [8] for standard definitions concerning information theory.

The relative entropy of two distributions on a discrete support is denoted by $D(\rho||\sigma)$. The relative max-entropy is $D_\infty(\rho||\sigma)=\max_x\log(\rho(x)/\sigma(x))$. Note that these quantities are infinite, if the support of σ does not contain the support of ρ. We mostly consider bipartite distributions on $\{0,1\}^n\times\{0,1\}^n$. The mutual information is $I(X:Y)=D(\mu||\mu_X\times\mu_Y)$, where μ is the joint distribution of (X,Y) and μ_X, and μ_Y are the two marginal distributions of μ. We also use the quantity $I_\infty(X:Y)=D_\infty(\mu||\mu_X\times\mu_Y)$. If we want to indicate the distribution used we write its name as a superscript, like $I^\mu(X:Y)$.

We first state the following well-known fact, see [13].

▶ **Fact 4** (Substate Theorem).

1. $I(X:Y) \leq I_\infty(X:Y)$.
2. *For a given μ there is a μ' with $||\mu - \mu'|| \leq \epsilon$, and $I_\infty^{\mu'}(X:Y) \leq I^\mu(X:Y) \cdot 4/\epsilon$, where $||\mu - \mu'||$ is the total variation distance between μ and μ'.*

We will use the following lemmas and facts. The first follows from the definition of relative entropy.

▶ **Lemma 5.** *Let μ be a bipartite distribution, $\rho = \mu_A \times \mu_B$, and $\sigma = \sigma_A \times \sigma_B$ any product distribution.*

Then $D(\mu||\sigma) = D(\mu||\rho) + D(\rho||\sigma) = I^\mu(X:Y) + D(\rho||\sigma)$.

The following is a consequence of the log-sum inequality.

▶ **Lemma 6.** *Let μ, σ be distributions (for concreteness on $\{0,1\}^n \times \{0,1\}^n$), and E an event. Then we have that $\sum_{x,y\in E} \mu(x,y)\log(\mu(x,y)/\sigma(x,y)) \geq \max\{-1, \mu(E)\log(\mu(E)/\sigma(E))\}$.*

▶ **Lemma 7.** *Let μ be a distribution on $\{0,1\}^n \times \{0,1\}^n$, E an event, and μ' the distribution μ restricted to E. Furthermore, assume that under μ we have that $Prob(E) = \alpha$. Then $D(\mu'||\sigma) \leq (D(\mu||\sigma) + 1)/\alpha - \log \alpha$.*

Proof. For all $x, y \in E$ we have $\mu'(x,y) = \mu(x,y)/\alpha$, otherwise $\mu'(x,y) = 0$.

$$\begin{aligned}
& D(\mu||\sigma) \\
= & \sum_{x,y} \mu(x,y)\log(\frac{\mu(x,y)}{\sigma(x,y)}) \\
\overset{(*)}{\geq} & \sum_{x,y\in E} \mu(x,y)\log(\frac{\mu(x,y)}{\sigma(x,y)}) - 1 \\
\geq & \sum_{x,y\in E} \mu'(x,y)\cdot\alpha\cdot\log(\frac{\mu'(x,y)\cdot\alpha}{\sigma(x,y)}) - 1 \\
= & D(\mu||\sigma)\cdot\alpha + \alpha\log\alpha - 1,
\end{aligned}$$

where for (*) we use Lemma 6 with the event $\{0,1\}^n \times \{0,1\}^n - E$. ◀

We will use the following *rejection sampling* protocol from [10].

▶ **Fact 8.** *Let μ and ν be distributions on $\{0,1\}^n$ with $D(\mu||\nu) = k$. Assume that Alice and Bob both know ν, and can create samples from ν using a public coin. Then Alice can send a message of expected length $k + 2\log k + O(1)$ to Bob, which allows Bob (and Alice) to obtain a shared sample from the distribution μ. The expectation is over the public coin tosses, and Bob's sample is distributed exactly with μ.*

The next lemma follows from a calculation and shows that a distribution can decrease a joint probability compared to the product of marginal distributions only in the presence of mutual information.

▶ **Lemma 9.** *Let X, Y be Boolean random variables with a joint distribution μ and marginal distributions μ_A, μ_B. If $\mu_A(X=1)\mu_B(Y=1) \geq 2\mu(X=Y=1)$, then $I^\mu(X:Y) \geq \mu_A(X=1)\mu_B(Y=1)/4$.*

Finally, we show that this is true for any product distribution, not just the product of marginals.

▶ **Lemma 10.** *Let X, Y be Boolean random variables with a joint distribution μ (and set $\rho = \mu_A \times \mu_B$), and σ any product distribution. If $\sigma_A(X=1)\sigma_B(Y=1) \geq 4\mu(X=Y=1)$ then $D(\mu||\sigma) \geq \sigma(X=Y=1)/16$.*

Proof. If $\rho(X=Y=1) \geq \sigma(X=Y=1)/2$, then by the above lemma $D(\mu||\sigma) \geq D(\mu||\rho) = I^\mu(X:Y) \geq \rho(X=Y=1)/4 \geq \sigma(X=Y=1)/8$, because σ is a product distribution and the relative entropy of μ and a product distribution is minimal for ρ. If $\rho(X=Y=1) \leq \sigma(X=Y=1)/2$, then we can bound $D(\mu||\sigma) \geq D(\rho||\sigma) = D(\mu_A||\sigma_A) + D(\mu_B||\sigma_B)$. Assume that $\alpha = \mu_A(X=1) \leq \beta/\sqrt{2} = \sigma_A(X=1)/\sqrt{2}$. Then $(1-\alpha)\log((1-\alpha)/(1-\beta)) + \alpha\log(\alpha/\beta) \geq \beta/16$. Hence in this case $D(\rho||\sigma) \geq D(\mu_A||\sigma_A) \geq \beta/16 = \sigma_A(X=1)/16 \geq \sigma_A(X=1)\sigma_B(Y=1)/16$. Other cases follow by symmetry. ◀

2.2 Communication Complexity

We assume familiarity with classical and quantum communication complexity. For the former consult [20], the latter is surveyed in [9]. We concentrate on distributional complexity, which we define here.

▶ **Definition 11.** The distributional complexity $D^\mu_\epsilon(f)$ is the minimal worst case communication cost of any deterministic protocol that computes f with error ϵ under μ. Similarly we define $R^\mu_\epsilon(f)$ for randomised public coin protocols and $Q^\mu_\epsilon(f)$ for quantum protocols (we consider quantum protocols with shared entanglement, but do not use the entanglement in our protocols). When we drop the error ϵ from the notation, we set $\epsilon = 1/3$. When we drop the superscript we mean the ordinary, worst-case communication complexity.

We observe that $R^\mu_\epsilon(f) = D^\mu_\epsilon(f)$ for all f, μ, ϵ, because one can fix the public coin randomness without increasing the error. Hence, we adopt the R-notation, and use randomness in upper bounds and deterministic protocols in lower bounds. Note that $Q^\mu_\epsilon(f)$ can be smaller than $R^\mu_\epsilon(f)$, for instance for Disjointness under the hard distribution exhibited by Razborov [22], where $R^\mu(DISJ) = \Theta(n)$, since the quantum complexity of DISJ is at most $O(\sqrt{n})$ [1].

We consider functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$.

▶ **Definition 12.** Define by $D(k)$ the set of distributions on the inputs that have $I(X:Y) \leq k$.

We define $R^{I\leq k}_\epsilon(f) = \max_{\mu \in D(k)} R^\mu_\epsilon(f)$ and use an analogous definition for the quantum case.

Clearly $R(f) = R^{I\leq n}(f)$ and $R^{I=0}(f)$ is the complexity under the hardest product distribution.

▶ **Definition 13.** One-way protocols allow only a single message from Alice to Bob, who produces the output. We indicate this model by a superscript, like $R^{A\to B, I\leq k}(f)$.

Finally, we note the following fingerprinting technique [20].

▶ **Fact 14.** *There is a public coin protocol that can check equality of strings (of any length) with error $1/2^k$ and communication k.*

3 Randomised Complexity of Disjointness

3.1 Upper Bound

In this section we prove the upper bound for DISJ under bounded information distributions.

First we consider the case of 0 mutual information, for which we show an upper bound of $O(\sqrt{n}\log(1/\epsilon))$. Let μ be a product distribution on the inputs to DISJ. Babai et al. [4] already show a protocol of cost $O(\sqrt{n}\log n\log(1/\epsilon))$ [they do not state the dependence on ϵ, which is however easy to derive from their proof]. Note that one can combine their protocol for product distributions with the Substate Theorem (Fact 4) to get a bound of $O(\sqrt{n}(k+1)\log n/\epsilon)$ on the distributional complexity under distributions with information k: every distribution with information k approximately sits with probability $1/2^{4k/\epsilon}$ inside the product of its marginal distributions, hence it is enough to use a product distribution protocol with very small error. This bound is worse in the dependence on k than what is proved below.

▶ **Theorem 15.** *$R_\epsilon^{I=0}(DISJ) \leq O(\sqrt{n}\cdot\log(1/\epsilon))$.*

The proof is in the appendix. The main issue here is to achieve the small error dependence. The protocol has a 2-phase structure, where in phase 1, assuming that Bob holds a large set and that the probability that $x \cap y' = \emptyset$ is large, random y' are drawn using the public coin and, if disjoint from x, removed from the universe (initially $\{1,\dots,n\}$). After doing this sufficiently many times, the universe becomes small, and in phase 2 we use the small set disjointness protocol due to Hastad and Wigderson [11].

Now we turn to distributions with more information. The protocol has the same structure, but we need to sample from a distribution of y' that is not independent of x, which takes communication. The protocol also does not have the same error dependence, which we show is unavoidable later. Due to this we may just analyse expected communication, and show that the worst case communication cannot be more than $1/\epsilon$ the established bound by appealing to the Markov bound.

▶ **Theorem 16.** *$R_\epsilon^{I\leq k}(DISJ) \leq O(\sqrt{n(k+1)}/\epsilon^2)$.*

The proof is in the appendix. The main idea is to follow the 2-phase approach, and shrink the universe until is has size $S = \sqrt{n(k+1)}$. At this point the Hastad-Wigderson small set Disjointness protocol [11] takes over. To shrink the universe we need to sample inputs y' from the distribution conditioned on x, and on being disjoint from x. This is achieved using the rejection sampling protocol of Fact 8. We need to carefully bound the information increase, but on average we remove S elements from the universe with communication cost $O(k/\epsilon)$, and there are at most n/S iterations in phase 1, hence the expected communication is at most $n/S \cdot k/\epsilon$.

In the next section we will also show a lower bound of $\Omega(\sqrt{n/\epsilon})$, so the error dependence cannot be made logarithmic, in contrast to the the 0 information case.

One more issue we would like to consider is the number of rounds used. The above protocol can easily use a large number of rounds, and it is not immediately clear whether this is necessary. It is well known that the complexity of DISJ under product distributions for *one-way protocols* is $\Theta(n)$ [19]. We have the following modification that saves most of the interaction.

▶ **Theorem 17.**

1. *The complexity of DISJ under distributions with information at most k for protocols with 2 rounds is at most $O(\sqrt{n(k+1)}\log n/\epsilon^2)$.*
2. *The complexity of DISJ under distributions with information at most k for $O(\log^* n)$ rounds is at most $O(\sqrt{n(k+1)}/\epsilon^2)$.*
3. *In the case of 0 mutual information, the error dependence drops to a factor of $\log(1/\epsilon)$.*

Proof. For the first item we observe that in phase 1 Alice can simply act as if Bob's set was large, and continue to let him discover y_i''s that are disjoint with x until U_i is guaranteed to be small. This does not increase the bound on the communication. After this Bob can tell Alice, in which 'round' his set really became small, so that she can recover the proper universe U_j. He also sends her his set using $\sqrt{n(k+1)}\log n$ bits. Note that in this protocol only Alice learns the result.

For the second item we do as above, but when Bob's set is small also repeat the same in reverse until both sets are small. Saglam and Tardos [24] have a protocol that solves the small set disjointness problem in phase 2 in $O(\log^* n)$ rounds with communication $O(\sqrt{n(k+1)}\log(1/\epsilon))$.

Finally, note that for product distributions we can use the same modifications to the protocol described in Theorem 15. ◀

3.2 Lower Bound

In this section we prove that the protocol of the previous section is optimal (except regarding the exact dependence on ϵ).

For the lower bound we employ a distribution, depending on n and k, such that the mutual information of the two inputs according to the marginal distributions is at most k; we then prove an $\Omega(\sqrt{n(k+1)})$ lower-bound for the distributional complexity under this distribution. In what follows we consider $k = k(n)$ as being $\in o(n)$, since for $k = \Omega(n)$ the upper bound on the information is trivial and the lower bound on the communication is known.

Let $c = \frac{1}{\log e}$ and $m = c\sqrt{n(k+1)}$. Note that $m = o(n)$ as well. Now $\mu_{n,k}$ can be defined as the distribution obtained by mixing two distributions, one where a pair of disjoint subsets of $\{1, \dots, n\}$ of size m is chosen uniformly among all such pairs, and one where a pair of subsets of size m with intersection of size 1 is chosen uniformly among all such pairs. This is essentially the distribution used in the proof by Razborov [22], but with smaller sets.

We show in the appendix that the information is bounded by k.

▶ **Theorem 18.** *For any sufficiently small $\epsilon > 0$ we have that $D_\epsilon^{\mu_{n,k}}(DISJ) = \Omega(\sqrt{n(k+1)})$, and hence that $R_\epsilon^{I\le k}(DISJ) = \Omega(\sqrt{n(k+1)})$.*

While the proof is similar to that of the original proof of Razborov [22], two difficulties arise when working with smaller sets: The first is that by mixing the two distributions with equal probability, the weight of any pair of intersecting sets is much larger than that of a pair of disjoint sets. Since the proof relies heavily on the properties of the distribution when conditioned on certain events, and in particular on the proportion of the weight of 1-inputs and the weight of 0-inputs when conditioning, this imbalance complicates several computations.

The second difficulty comes from the fact that Razborov's entropy "counting" argument no longer works in our case, because in that argument a linear number of terms have their entropy upper-bounded as $H\left(\frac{1}{2}\right) = 1$. Since we still have to deal with a linear number of terms while having much less total entropy, we require a finer combinatorial counting argument instead.

Now we give a simple argument that shows that error dependence cannot be logarithmic in $1/\epsilon$.

▶ **Theorem 19.** $R_\epsilon^{I\le 1}(DISJ) = \Omega(\sqrt{n/\epsilon})$ *for* $\epsilon \ge \Omega(1/n)$.

Proof. Above we have described a distribution $\mu_{n,k}$ with information at most k such that $\Omega(\sqrt{n(k+1)})$ communication is needed for some constant error δ. We define $\sigma_{n,k}$ to be $1/(2k) \cdot \mu_{n,k} + (1 - 1/(2k))\rho$, where ρ is some product distribution for DISJ that puts weight 1/2 on 1-inputs. Clearly, for error $\delta/(4k)$ the communication must be at least $\Omega(\sqrt{n(k+1)})$. Set $k = 4\delta/\epsilon$ (note that $k \leq n$).

It remains to show that the information in σ is at most 1. Let E be an indicator variable that indicates that x, y have been chosen according to μ_k. Then $I(X:Y) \leq I(XE:Y) = I(E:Y) + I(X:Y|E) \leq H(E) + (1/2k) \cdot k \leq H(1/(2k)) + 1/2 \leq 1$. ◀

▶ **Corollary 20.** *The class of distributions with information k with $1 \leq k \leq n^{1-\Omega(1)}$ is not boost-able for randomised protocols.*

Proof. Consider $k = 1$. We have that $R^{I\leq 1}_{1/3}(DISJ) \leq O(\sqrt{n})$. If distributions with at most 1 bit information were boost-able, then we would have $R^{I\leq 1}_{\epsilon}(DISJ) \leq O(\sqrt{n}\log(1/\epsilon))$. But the left hand side is at least $\Omega(\sqrt{n/\epsilon})$, which puts a lower bound on ϵ, whereas boost-ability should work for all ϵ.

In the case of larger k we use the same proof, to get that $\sqrt{\epsilon} \cdot \log(1/\epsilon) \geq \Omega(1/\sqrt{k+1})$, which remains a restriction on ϵ until k exceeds $n^{1-\Omega(1)}$, and the assumption of Theorem 19 is violated. ◀

4 Quantum Complexity of Disjointness

4.1 Upper Bound: First Attempt

Consider the two-phase approach from the previous section. The second phase ‘quantises’ readily, if we do not care about log-factors: Simply use distributed quantum search by amplitude amplification to obtain a quadratic speedup in this part [6]. We mention here that the tight protocol for DISJ due to Aaronson and Ambainis [1] does not seem to work well for the small set case and so we do not know if the logarithmic factor is needed or not.

The problem is the first phase of the classical protocol, which seems impossible to quantise. Since phase 2 is now cheaper one can re-balance the costs of the two phases (details are left to the reader) and find a protocol with cost $\tilde{O}((n(k+1))^{1/3}$.

In the next section we will show that this bound is not optimal. We do note here, however, that the error dependence for the case $I(X:Y) = 0$ is a factor of $O(-\log\epsilon)$ for the above, which will not be the case in the protocol we present next.

4.2 Upper Bound: Almost Optimal Protocol

We now describe a different approach that also works in the classical case, but loses a logarithmic factor and has a worse error dependence for product distributions. The approach we use identifies two blocks of “interesting” positions (i.e., the blocks are subsets of $[n]$), such that Alice can conduct a search efficiently on one block, and Bob on the other one, because their sets are expected to be small on their respective blocks, and on the other hand, the situation when the input sets intersect but not on any interesting position is unlikely. This conforms to the rough intuition that if “large” x and y come from a product distribution that puts constant weight on 1-inputs, then there must be many “semi-interesting” and “uninteresting” positions – i.e., such $i \in [n]$ that not both $i \in x$ and $i \in y$ is likely.

Let μ be a distribution on $\{0,1\}^n \times \{0,1\}^n$ with $I^{\mu}(X:Y) \leq k$. Denote by E_i the event that x, y drawn from μ satisfy $\sum_{1\leq j\leq i-1} x_i y_i = 0$, i.e., x and y are disjoint on $\{1,\ldots,i-1\}$.

We set $s_i = Prob_\mu(E_i)$. We assume that $s_i \geq \alpha$ for all i, and some $\alpha \geq \epsilon$. If this is not the case, then the probability that x, y are disjoint is less than ϵ, and the distribution is trivial.

Define $q_i'^x = Prob(Y_i = 1|X = x, X_i = 1, E_i)$ and $p_i'^y = Prob(X_i = 1|Y = y, Y_i = 1, E_i)$. Our protocol follows the simple idea that Alice should search among those positions i, such that $q_i'^x$ is large, similarly for Bob and $p_i'^y$.

The Protocol. A position is *chosen* by Alice, if $q_i'^x \geq \epsilon^3/\sqrt{80000(k+1)n}$ and $i \in x$ and *chosen* by Bob, if $p_i'^y \geq \epsilon^3/\sqrt{(80000(k+1)n}$ and $i \in y$. Denote the former set by C_A and the latter by C_B. Alice is responsible for finding intersecting positions in C_A, Bob for finding intersecting positions in C_B. In the protocol Alice organises a search for an intersecting position based on amplitude amplification on her positions C_A (using a distributed Grover search as in [6]). More precisely, Alice creates a superposition over all positions in C_A, and the two players can mark intersecting positions like in Grover search by communicating $\log n$ qubits back and forth, and conduct amplitude amplification to find an intersection there. In phase 2 the same is done with C_B and the roles of the players reversed. If the players find an intersecting position, they reject, otherwise they accept.

Communication. For all x we have $\sum_{i \in x} q_i'^x s_i \leq Prob(DISJ(x,y) = 0) \leq 1$. Hence $|C_A| \leq O(\sqrt{(k+1)n}/\epsilon^4)$, since all $s_i \geq \alpha \geq \epsilon$. Amplitude amplification needs $O(((k+1)n)^{1/4}/\epsilon^2 \cdot \log(1/\epsilon))$ iterations, each taking $\log n$ communication.

Error Analysis. Let us define some probabilities. By $\vec{x}, \vec{y}$ we denote prefixes of strings x, y of length $i-1$, where i is usually clear from the context. The random variable $\vec{X}$ is the prefix of length $i-1$ of the random variable X (Alice's inputs), and similarly for Y.

Denote $p_i^{\vec{x}} = Prob(X_i = 1|E_i, \vec{X} = \vec{x})$, $q_i^{\vec{x}} = Prob(Y_i = 1|E_i, \vec{X} = \vec{x})$, and similarly for $p_i^{\vec{y}}$ and $q_i^{\vec{y}}$. Denote also $p_i'^{\vec{y}} = Prob(X_i = 1|E_i, Y_i = 1, \vec{Y} = \vec{y})$, and similarly for $p_i'^{\vec{x}}$, $q_i'^{\vec{x}}$ and $q_i'^{\vec{y}}$. Denote also $q_i'^{\vec{x},\vec{y}} = Prob(Y_i = 1|E_i, X_i = 1, \vec{X} = \vec{x}, \vec{Y} = \vec{y})$, and similarly for $p_i'^{\vec{x},\vec{y}}$. Denote by $r_i^{\vec{x}} = p_i^{\vec{x}} q_i'^{\vec{x}} = p_i'^{\vec{x}} q_i^{\vec{x}}$ the probability that $X_i = Y_1 = 1$ under the conditions $\vec{X} = \vec{x}$ and E_i, and similarly for other conditions (i.e., the super-script specifies the condition): say, r_i is the probability that $X_i = Y_i = 1$ conditioned on E_i, and so on.

As a first step we "get rid" of the input positions that are very unlikely to contribute, compared to the average for a position. We say that x with $x_i = 1$ is *A-bad* for i, if $q_i'^{\,x} \leq \epsilon q_i'^{\,\vec{x}}/10$. These are the positions where x depresses the probability of intersection compared to $\vec{x}$. Similarly, y with $y_i = 1$ is *B-bad* for i, if $p_i'^{\,y} \leq \epsilon p_i'^{\,\vec{y}}/10$. Denote by V_i the event that x is A-bad for i, and by W_i the event that y is B-bad for i. Finally, set $\tilde{q}_i'^{\,\vec{x}} = Prob(Y_i = 1 \wedge V_i|X_i = 1, \vec{X} = \vec{x}, E_i)$ and $\tilde{p}_i'^{\,\vec{y}} = Prob(X_i = 1 \wedge W_i|Y_i = 1, \vec{Y} = \vec{y}, E_i)$.

Note that $r_i^{\vec{x}} s_i = p_i^{\vec{x}} q_i'^{\,\vec{x}} s_i$ is the probability that the first intersection between X and Y is on position i when $\vec{X} = \vec{x}$. The probability that x is A-bad for i and the first intersection is on i is $p_i^{\vec{x}} \tilde{q}_i'^{\vec{x}} s_i$. Similarly, the probability that y is B-bad for i and the first intersection is on i is $\tilde{p}_i'^{\vec{y}} q_i^{\vec{y}} s_i$. The following lemma shows that possible intersections at such positions i that either x is A-bad for i or y is B-bad for i can be safely ignored.

▶ **Lemma 21.** *$\tilde{q}_i'^{\vec{x}} \leq \epsilon q_i'^{\vec{x}}/10$ and $\tilde{p}_i'^{\vec{y}} \leq \epsilon p_i'^{\vec{y}}/10$.*

Proof. Denote by $Bad(x, i)$ the property that x is A-bad for i and by $E_i(x, y)$ the property that x, y are disjoint on $\{1, \ldots, i-1\}$.

$$\begin{aligned}
\tilde{q}_i'^{\vec{x}} &= \frac{Prob(V_i \wedge Y_i = 1 \wedge X_i = 1 \wedge \vec{X} = \vec{x} \wedge E_i)}{Prob(X_i = 1 \wedge \vec{X} = \vec{x} \wedge E_i)} \\
&= \sum_{x: x_i = 1, x_1, \ldots, x_{i-1} = \vec{x}, Bad(x,i)} \quad \sum_{y: y_i = 1, E_i(x,y)} \mu(x,y) / Prob(X_i = 1 \wedge \vec{X} = \vec{x} \wedge E_i) \\
&= \sum_{x: x_i = 1, x_1, \ldots, x_{i-1} = \vec{x}, Bad(x,i)} q_i'^{x} \cdot Prob(X = x \wedge E_i) / Prob(X_i = 1 \wedge \vec{X} = \vec{x} \wedge E_i) \\
&\leq (\epsilon/10) \cdot q_i'^{\vec{x}} \cdot \sum_{x: x_i = 1, x_1, \ldots x_{i-1} = \vec{x}} Prob(X = x \wedge E_i) / Prob(X_1 = 1 \wedge \vec{X} = \vec{x} \wedge E_i) \\
&\leq (\epsilon/10) \cdot q_i'^{\vec{x}}.
\end{aligned}$$

◀

Therefore, ignoring possible intersections where x or y are bad for i, one can introduce error at most $\epsilon/5$, because the probability of an A-bad (first) intersections is at most $p_i^{\vec{x}} \tilde{q}_i'^{\vec{x}} s_i \leq (\epsilon/10) p_i^{\vec{x}} q_i'^{\vec{x}} s_i$ for any $\vec{x}$, with a similar bound for B-bad. Hence in the following we assume that all x, y are not bad for i.

We call a position i and inputs x, y *lucky*, if $p_i^{\vec{x}} \leq 400(k+1) p_i'^{\vec{y}} / \epsilon^3$. The remaining positions are unlucky for x, y. There are four possible sources of error in our protocol: There are "bad" intersections (considered above). Among the positions for which the input is not bad, there may be *unchosen lucky* positions and *unchosen unlucky* positions. Finally, some error comes from the amplitude amplification quantum searches.

"Bad" intersections contribute error at most $\epsilon/5$, as shown above. The amplitude amplification error can be pushed below $\epsilon/20$ by increasing communication by a factor of $O(-\log \epsilon)$, which is already absorbed in the stated communication bound above. It remains to deal with the unchosen lucky and unlucky positions (for which the input is not bad).

We first consider the error contributed by lucky positions i that are not chosen by either Alice or Bob – denote these by L. Fix the input prefixes $\vec{x}, \vec{y}$ and assume that the inputs are not bad for i. Positions that are not chosen satisfy $p_i'^{\vec{y}} q_i'^{\vec{x}} \leq (10/\epsilon)^2 \cdot p_i'^{y} q_i'^{x} \leq \epsilon^4 / (800(k+1)n)$. We have that the probability that the first intersection is at position $i \in L$ but i is not chosen, is (conditioned on $\vec{x}$)

$$r_i^{\vec{x}} s_i = p_i^{\vec{x}} q_i'^{\vec{x}} s_i \leq 400(k+1) p_i'^{\vec{y}} q_i'^{\vec{x}} s_i / \epsilon^3 \leq \epsilon/(2n) \cdot s_i \leq \epsilon/(2n),$$

where the first inequality is because of 'lucky', and the second because of 'unchosen'. Summing up, and taking expectations (over $\vec{x}$ under μ_i), this gives $\sum_i Prob(X_i = Y_i = 1 \wedge E_i \wedge i$ lucky, not chosen$) \leq \epsilon/2$, hence error at most $\epsilon/2$.

Now we turn to the error contributed by unlucky positions. For these we have that $p_i^{\vec{x}} > 400(k+1)/\epsilon^3 \cdot p_i'^{\vec{y}}$.

We use the following lemma.

▶ **Lemma 22.** *Assume that for no x or y the conditional probability of non-intersection is less than α, and that for no x and i the probability that $X_i = Y_i = 1$ conditioned on $X = x$ and E_i is larger than $1/2$, and the same for all y, i. Then*

$$\sum_i \mathbf{E}_{\vec{x},\vec{y}}^{\mu_i} \; p_i^{\vec{x}} q_i^{\vec{y}} \leq 16k/\alpha + 68/\alpha^2,$$

where $\vec{x}, \vec{y}$ are prefixes of length $i-1$.

Note that the first assumption of the lemma can be made since otherwise we can just reject x (resp., y) with error α. The second assumption can be made since otherwise i is chosen by Alice (resp., Bob), and no error happens there.

Now we can bound the probability that an unlucky position i has the first intersection (conditioned on $\vec{y}$) by

$$r_i^{\vec{y}} s_i \leq r_i^{\vec{y}} = p_i'^{\vec{y}} q_i^{\vec{y}} \leq \epsilon^3 p_i^{\vec{x}} q_i^{\vec{y}}/(400(k+1)).$$

Summing up over all i (not just the unlucky ones) and taking expectation over μ_i we get by our lemma that

$$\begin{aligned} & \sum_i \mathbf{E}_{\vec{x},\vec{y}}\, \epsilon^3/(400(k+1)) \cdot p_i^{\vec{x}} q_i^{\vec{y}} \\ \leq\ & \epsilon^3/(400(k+1)) \cdot ((16k/\alpha) + 68/\alpha^2) \\ \leq\ & \epsilon/4. \end{aligned}$$

Hence the total error is not more than $\epsilon/20 + \epsilon/4 + \epsilon/5 + \epsilon/2 \leq \epsilon$ and we get the following.

▶ **Theorem 23.** $Q_\epsilon^{I \leq k}(f) \leq O((n(k+1))^{1/4}/\epsilon^2 \cdot \log n \cdot \log(1/\epsilon))$.

It remains to prove the lemma.

Proof of Lemma 22. Denote by μ_i the probability distribution μ, restricted to the event E_i. We know that $k \geq I^\mu(X:Y) = D(\mu||\sigma)$, where σ is the product of marginals of μ. Denote by σ_i the product of marginals of μ_i, and by $\mu_i^{\vec{x},\vec{y},j}$ the distribution μ_i, conditioned on the event $X_1 = x_1, \ldots, X_j = x_j, Y_1 = y_1, \ldots, Y_j = y_j$, which we abbreviate by $F^{\vec{x},\vec{y},j}$. Similarly, $\sigma_i^{\vec{x},\vec{y},j}$ is σ_i conditioned on $F^{\vec{x},\vec{y},j}$. Note that for the latter probability distribution we first take the product of marginals of μ_i, and then condition. This is different from considering conditional mutual information, in which one would first condition and then take the product of marginals. We also stress that here j denotes the length of $\vec{x}, \vec{y}$, unlike before. In the following, when we do not mention j explicitly, it is $i-1$: e.g., $\mu_i^{\vec{x},\vec{y}} = \mu_i^{\vec{x},\vec{y},i-1}$.

By the chain rule for relative entropy we get that

$$\sum_{j=1,\ldots,n} \mathbf{E}^{\mu_i}_{\vec{x},\vec{y},j-1} D(\mu_i^{\vec{x},\vec{y},j-1}(X_j, Y_j)||\sigma_i^{\vec{x},\vec{y},j-1}(X_j, Y_j)) = D(\mu_i||\sigma_i) = I^{\mu_i}(X:Y),$$

where the expectation is over the prefixes $\vec{x}, \vec{y}$ of length $j-1$ under μ_i. We are interested in

$$k_i = \mathbf{E}^{\mu_i}_{\vec{x},\vec{y}} k_i^{\vec{x},\vec{y}} = \mathbf{E}^{\mu_i}_{\vec{x},\vec{y}} D(\mu_i^{\vec{x},\vec{y}}(X_i, Y_i)||\sigma_i^{\vec{x},\vec{y}}(X_i, Y_i)).$$

For this, i determines both the condition on previous positions, and the choice of distribution. The chain rule can be used if we fix μ_i, but here we want to vary μ_i as well. For the moment suppose we can bound $\sum k_i$ by a k' not much larger than k.

Observe that $p_i^{\vec{x}} q_i^{\vec{y}}$ is the probability that $X_i = Y_i = 1$ under the distribution $\sigma_i^{\vec{x},\vec{y}}(X_i, Y_i)$. σ_i is a product distribution, and hence conditioning on $Y's$ does note change the probability of $X_i = 1$ etc., and so we get that $p_i^{\vec{x}} = Prob_{\sigma_i}(X_i = 1|\vec{X} = \vec{x}, \vec{Y} = \vec{y})$.

We can now apply Lemma 10 to learn that either $p_i^{\vec{x}} q_i^{\vec{y}} \leq 4r_i^{\vec{x},\vec{y}}$ or $D(\mu_i^{\vec{x},\vec{y}}(X_i, Y_i)||\sigma_i^{\vec{x},\vec{y}}(X_i, Y_i)) \geq p_i^{\vec{x}} q_i^{\vec{y}}/16$. Hence

$$p_i^{\vec{x}} q_i^{\vec{y}} \leq 4r_i^{\vec{x},\vec{y}} + 16D(\mu_i^{\vec{x},\vec{y}}(X_i, Y_i)||\sigma_i^{\vec{x},\vec{y}}(X_i, Y_i)).$$

Then

$$\sum_i p_i^{\vec{x}} q_i^{\vec{y}} \leq \sum_i 4r_i^{\vec{x},\vec{y}} + 16k_i^{\vec{x}\vec{y}}.$$

Noting that $\sum_i \mathbf{E}^{\mu_i}_{\vec{x},\vec{y}} r_i^{\vec{x},\vec{y}} s_i \leq \sum r_i s_i \leq 1$ and hence $\sum_i \mathbf{E}^{\mu_i}_{\vec{x},\vec{y}} r_i^{\vec{x},\vec{y}} \leq 1/\alpha$ it remains to bound $k' = \sum k_i$ by $k/\alpha + 4/\alpha^2$. For this we need to first compare $\mu_{i+1}(x,y)$ and $\mu_i(x,y)$. If $(x,y) \in E_{i+1}$, then we get $\mu_{i+1}(x,y) = \mu_i(x,y)/(1-r_i)$. Also, we have $\sigma_{i+1}(x,y) = \sum_{y':(x,y')\in E_{i+1}} \mu_i(x,y')/(1-r_i) \cdot \sum_{x':(x',y)\in E_{i+1}} \mu_i(x',y)/(1-r_i)$, and, denoting $r_i^y = Prob_{\mu_i}(X_i = Y_i = 1|Y = y)$ that is equal to $\sum_{y':(x,y')\in E_i} \mu_i(x,y') \cdot (1-r_i^x)/(1-r_i) \cdot \sum_{x':(x',y)\in E_i} \mu_i(x',y) \cdot (1-r_i^y)/(1-r_i)$. Which is $\mu_i(x) \cdot \mu_i(y) \cdot (1-r_i^x)(1-r_i^y)/(1-r_i)^2$.

Let us compute an upper bound on $D(\mu_{i+1}||\sigma_{i+1})$

$$\begin{aligned}
&= \sum_{(x,y)\in E_{i+1}} \mu_{i+1}(x,y) \log \frac{\mu_{i+1}(x,y)}{\sigma_{i+1}(x,y)} \\
&= \sum_{(x,y)\in E_{i+1}} \mu_i(x,y)/(1-r_i) \cdot \log \frac{\mu_i(x,y)\cdot(1-r_i)}{\sigma_i(x,y)(1-r_i^x)(1-r_i^y)} \\
&\overset{(*)}{\leq} \sum_{(x,y)\in E_i} \mu_i(x,y)/(1-r_i) \cdot \log \frac{\mu_i(x,y)\cdot(1-r_i)}{\sigma_i(x,y)(1-r_i^x)(1-r_i^y)} \\
&- \mu_i(E_i - E_{i+1})/(1-r_i) \cdot \log(4\mu_i(E_i - E_{i+1})/\sigma_i(E_i - E_{i+1})) \\
&\leq \sum_{(x,y)\in E_i} \mu_i(x,y) \log \left(\frac{\mu_i(x,y)}{\sigma_i(x,y)}\right) /(1-r_i) \\
&+ \sum_{(x,y)\in E_i} \mu_i(x,y)/(1-r_i) \cdot \log \frac{1-r_i}{(1-r_i^x)(1-r_i^y)} - r_i/(1-r_i)\cdot\log(4r_i) \\
&\overset{(**)}{\leq} D(\mu_i||\sigma_i)/(1-r_i) + 2\sum_{(x,y)\in E_i} \mu_i(x,y)\cdot 2\cdot(r_i^x + r_i^y) - 2r_i\log(4r_i) \\
&\leq D(\mu_i||\sigma_i)/(1-r_i) + 12r_i - 2r_i\log(r_i),
\end{aligned}$$

where in (*) we use Lemma 6, in (**) we use that $-\log(1-\lambda) = -\ln(1-\lambda)/\ln(2) \leq 2\lambda$, for all $0 \leq \lambda \leq 1/2$, and in general use that $r_i^x, r_i^y, r_i \leq 1/2$ by the assumption in the lemma. The conclusion is that the relative entropy increases only slightly.

Now we turn to the terms k_i in the chain rule expansion. Fix $X_1 = x_1, \ldots, X_i = x_i$ and $Y_1 = y_1, \ldots, Y_i = y_i$. We are interested in $D(\mu_{i+1}^{\vec{x},\vec{y},i}||\sigma_{i+1}^{\vec{x},\vec{y},i})$ and its relation to to $D(\mu_i^{\vec{x},\vec{y},i}||\sigma_i^{\vec{x},\vec{y},i})$. Note that the distributions involved are on $X_{i+1},\ldots,X_n,Y_{i+1},\ldots,Y_n$. We assume $x_jy_j \neq 1$ for all $j < i+1$, otherwise the inputs are not in E_{i+1} and have no weight under μ_{i+1}. We have

$$\begin{aligned}
& D(\mu_{i+1}^{\vec{x},\vec{y}}(X_{i+1},\ldots,X_n,Y_{i+1},\ldots,Y_n)||\sigma_{i+1}^{\vec{x},\vec{y}}(X_{i+1},\ldots,X_n,Y_{i+1},\ldots,Y_n)) \\
= & \sum_{x,y\in E_{i+1}:x_1,\ldots,x_i=\vec{x},y_1,\ldots,y_i=\vec{y}} \mu_{i+1}(x,y|\vec{x},\vec{y}) \log\left(\frac{\mu_{i+1}(x,y|\vec{x},\vec{y})}{\sigma_{i+1}(x,y|\vec{x},\vec{y})}\right) \\
\leq & \sum_{x,y\in E_i:x_1,\ldots,x_i=\vec{x},y_1,\ldots,y_i=\vec{y}} \mu_i(x,y|\vec{x},\vec{y}) \log\left(\frac{\mu_i(x,y|\vec{x},\vec{y})}{\sigma_i(x,y|\vec{x},\vec{y})\cdot(1-r_i^x)(1-r_i^y)}\right) \\
\leq & D(\mu_i^{\vec{x},\vec{y},i}(X_{i+1},\ldots,X_n,Y_{i+1},\ldots,Y_n)||\sigma_i^{\vec{x},\vec{y},i}(X_{i+1},\ldots,X_n,Y_{i+1},\ldots,Y_n)) \\
+ & \sum_{x,y:x_1,\ldots,x_i=\vec{x},y_1,\ldots,y_i=\vec{y}} \mu_i(x,y|\vec{x},\vec{y}) \cdot \log\left(\frac{1}{(1-r_i^x)(1-r_i^y)}\right) \\
\leq & D(\mu_i^{\vec{x},\vec{y},i}(X_{i+1},\ldots,X_n,Y_{i+1},\ldots,Y_n)||\sigma_i^{\vec{x},\vec{y},i}(X_{i+1},\ldots,X_n,Y_{i+1},\ldots,Y_n)) \\
+ & \sum_{x,y:x_1,\ldots,x_i=\vec{x},y_1,\ldots,y_i=\vec{y}} \mu_i(x,y|\vec{x},\vec{y})(2r_i^x + 2r_i^y) \\
\leq & D(\mu_i^{\vec{x},\vec{y},i}(X_{i+1}\ldots)||\sigma_i^{\vec{x},\vec{y},i}(X_{i+1},\ldots)) + 2r_i^{\vec{y}} + 2r_i^{\vec{x}}.
\end{aligned}$$

Note here that conditioned on $\vec{x}, \vec{y}$, the condition E_{i+1} is satisfied for all inputs, and no re-scaling happens going from μ_{i+1} to μ_i conditioned on $\vec{x}, \vec{y}$.

We can now bound $\sum_i k_i = \sum_i \mathbf{E}^{\mu_i}_{\vec{x},\vec{y}}\, k_i^{\vec{x},\vec{y}}$. Note that $\mu_1 = \mu$.

$$\begin{array}{rl}
k \geq & D(\mu||\sigma) \\
= & D(\mu_1(X_1,Y_1)||\sigma_1(X_1,Y_1)) \\
+ & \mathbf{E}^{\mu_1}_{x_1,y_1} D(\mu_1^{x_1,y_1}(X_2,\ldots,X_n,Y_2,\ldots,Y_n)||\sigma_1^{x_1,y_1}(X_2,\ldots,X_n,Y_2,\ldots,Y_n)) \\
\geq & D(\mu_1(X_1,Y_1)||\sigma_1(X_1,Y_1)) \\
+ & \mathbf{E}^{\mu_1}_{x_1,y_1} D(\mu_2^{x_1,y_1}(X_2,\ldots,X_n,Y_2,\ldots,Y_n)||\sigma_2^{x_1,y_1}(X_2,\ldots X_n,Y_2,\ldots,Y_n)) - 4r_1 \\
\geq & D(\mu_1(X_1,Y_1)||\sigma_1(X_1,Y_1)) \\
+ & \mathbf{E}^{\mu_2}_{x_1,y_1} D(\mu_2^{x_1,y_1}(X_2,\ldots)||\sigma_2^{x_1,y_1}(X_2,\ldots)) \cdot (1-r_1) - 4r_1 \\
= & D(\mu_1(X_1,Y_1)||\sigma(X_1,Y_1)) \\
+ & \mathbf{E}^{\mu_2}_{x_1,y_1} D(\mu_2^{x_1,y_1}(X_2,Y_2)||\sigma^{x_1,y_1}(X_2,Y_2)) \cdot (1-r_1) \\
+ & \mathbf{E}^{\mu_2}_{x_1,x_2,y_1,y_2} D(\mu_2^{x_1,x_2,y_1,y_2}(X_3,\ldots)||\mu_2^{x_1,x_2,y_1,y_2}(X_3,\ldots)) \cdot (1-r_1) - 4r^1 \\
\geq & D(\mu_1(X_1,Y_1)||\sigma_1(X_1,Y_1)) + \mathbf{E}^{\mu_2}_{x_1,y_1} D(\mu_2^{x_1,y_1}(X_2,Y_2)||\sigma_2^{x_1,y_1}(X_2,Y_2)) \cdot (1-r_1) \\
+ & \mathbf{E}^{\mu_2}_{x_1,x_2,y_1,y_2} D(\mu_3^{x_1,x_2,y_1,y_2}(X_3,\ldots)||\mu_3^{x_1,x_2,y_1,y_2}(X_3,\ldots)) \cdot (1-r_1) \\
- & 4r_1 - 4r_2 \cdot (1-r_1) \\
\geq & D(\mu_1(X_1,Y_1)||\sigma_1(X_1,Y_1)) + \mathbf{E}^{\mu_2}_{x_1,y_1} D(\mu_2^{x_1,y_1}(X_2,Y_2)||\sigma_2^{x_1,y_1}(X_2,Y_2)) \cdot (1-r_1) \\
+ & \mathbf{E}^{\mu_3}_{x_1,x_2,y_1,y_2} D(\mu_3^{x_1,x_2,y_1,y_2}(X_3,\ldots)||\mu_3^{x_1,x_2,y_1,y_2}(X_3,\ldots)) \cdot (1-r_1)(1-r_2) \\
- & 4r_1 - 4r_2 \cdot (1-r_1) \\
\vdots & \\
\geq & \sum_i \mathbf{E}^{\mu_i}_{\vec{x},\vec{y}} D(\mu_i^{\vec{x},\vec{y}}(X_i,Y_i)||\sigma_i^{\vec{x},\vec{y}}(X_i,Y_i)) \cdot \alpha - 4\sum_i r_i \\
= & \sum_i k_i \cdot \alpha - 4/\alpha,
\end{array}$$

where in the last step we use that $\prod_{i=1,\ldots,n}(1-r_i) \geq \alpha$ and $\sum_i r_i \leq 1/\alpha$.

This means that $\sum k_i \leq k/\alpha + 4/\alpha^2$. ◀

4.3 Lower Bound

We use exactly the same hard distribution for the quantum case as for the classical case, see Section 3.2, where also the mutual information of this distribution is shown to be at most k. Conveniently, Razborov [23] has done most of the hard work for us by analysing the quantum complexity of Disjointness for all set sizes. We get the following main result:

▶ **Theorem 24.** *The distributional quantum communication complexity of Disjointness under $\mu_{n,k}$ is at least $\Omega((n(k+1))^{1/4})$.*

Proof. Recall the distributions $\nu_{n,k}, \sigma_{n,k}$ as defined in Section 3.2. These are the distributions of sets of size $s = O(\sqrt{n(k+1)}$ from a size n universe (not intersecting resp. intersecting). We employ the following result by Razborov [23]:

▶ **Fact 25.** *Any quantum protocol that solves DISJ with error ϵ under $\nu_{n,k}$ and error ϵ under $\sigma_{n,k}$ needs communication $\Omega(\sqrt{s}) = \Omega((n(k+1))^{1/4})$.*

This follows from Razborov's proof, in which given a quantum protocol with communication c for DISJ (on inputs of size s from a size n universe), a uni-variate polynomial of degree $O(c)$ on $\{0, 1, \dots, s\}$ is constructed such that $p(i)$ is close to 0 for all $\{0, 1, \dots, s-1\}$ and $p(s) = 1$. Such a polynomial must have degree $\Omega(\sqrt{s})$. The construction is done by averaging of the acceptance probabilities on all inputs x, y where x, y have size s, and hence it is enough if the given protocol for DISJ is correct on average inputs under $\nu_{n,k}$ and under $\sigma_{n,k}$. But any protocol with small error under $\mu_{n,k}$ must also have small error under both of these distributions, and we get the same lower bound under this distribution as in the worst case, as stated by Razborov. ◀

We also note that again, the error dependence cannot be polylogarithmic. The proof is the same as in the classical case.

▶ **Theorem 26.** $Q_\epsilon^{I\leq 1}(DISJ) \geq \Omega((n/\epsilon)^{1/4})$.

We again obtain this following.

▶ **Corollary 27.** *The class of distributions with information k with $1 \leq k \leq n^{1-\Omega(1)}$ is not boost-able for quantum protocols.*

5 Large Correlation is Needed for Tight Bounds

In this section we show that there is a function, for which the distributional communication complexity is far from the randomised communication complexity if the information in the distribution is less than $\Omega(n)$. The main idea is that random sparse problems make it hard for low information distributions to 'focus' on the 1-inputs.

Define $f_{n,d}$ as a random variable that takes as its values functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. The functions are generated randomly as follows. Each input x, y is chosen to be a 1-input independently with probability $d/2^n$.

Note that the communication matrix of $f_{n,d}$ has expected d 1-inputs for each row and column. In the following d should be thought of as some value like $2^{\sqrt{n}}$. We need $2^{n/100} \geq d \geq 6n$.

We first show that the complexity of $f_{n,d}$ is $\Theta(\log d)$ with high probability. Then, we show that with high probability $f_{n,d}$ has a property that allows an $O(\log n)$ protocol under all low information distributions.

First we note that by the Chernoff bound the probability that a row or column has more than $2d$ or less than $d/2$ 1-inputs is at most $2e^{-d/3} \leq 2^{-2n}$. By the union bound it is true for all rows and columns (with high probability) that they contains between $d/2$ and $2d$ 1-inputs. Throughout this section we assume that $f_{n,d}$ has this property.

▶ **Lemma 28.** *$R(f_{n,d}) \leq O(\log d)$ with high probability.*

Proof. With high probability there are at most $2d$ 1-inputs $(x_1, y), \dots, (x_{2d}, y)$ in Bob's column. If Alice sends a fingerprint of x as in Fact 14, using $2 \log d$ bits, then Bob can check whether $x = x_j$ for some $1 \leq j \leq 2d$ with error $2d \cdot 2^{-2\log d} \leq 2/d$. If so, then he accepts, otherwise he rejects. ◀

▶ **Lemma 29.** *$R(f_{n,d}) \geq \Omega(\log d)$ with high probability.*

Proof. The proof is by the probabilistic method. We use the minimax theorem and the following hard distribution: Put 1/2 weight on 1-inputs and 1/2 weight on 0-inputs to $f_{n,d}$. Note that the mutual information of this distribution is $\Omega(n)$: for 1-inputs, given x there are at most d inputs y out of 2^n such that x, y is a 1-input. Hence the information is at least $(n - \log d)/2$.

We employ the 1-sided discrepancy method. The 1-sided discrepancy under a distribution μ is $disc'(f, \mu) = \max_R \mu(f^{-1}(1) \cap R) - \mu(f^{-1}(0) \cap R)$, where the maximum is over all rectangles. Then $R^\mu(f) \geq -\log disc'(f, \mu)$ for all μ that put weight 1/2 on the 1-inputs. Our goal is to show that the 1-sided discrepancy is small with high probability over the choice of $f_{n,d}$.

Fix a rectangle R and consider a random $f_{n,d}$. We would like to compute the probability that $disc'(R) = \mu(f_{n,d}^{-1}(1) \cap R) - \mu(f_{n,d}^{-1}(0) \cap R)$ is large. Note that this is a random variable and that μ depends on $f_{n,d}$

If $\mu(R \cap f^{-1}(1)) \leq 4/d^{1/4}$, then $disc'(R) \leq 4/d^{1/4}$ and we are done. Hence we assume the opposite. For R to contain at least a $4/d^{1/4}$ fraction of all 1-inputs it must be the case that R contains at least $(4/d^{1/4}) \cdot 2^n d/2$ 1-inputs, and no row or column contains more than $2d$ of them, which implies that R must have at least $2^n/d^{1/4}$ rows and columns.

Write $R = A \times B$, where $|A|, |B| \geq 2^n/d^{1/4}$. The expected number of 1-inputs in R is at most $|A| \cdot |B| \cdot d/2^n$. The 1-inputs are chosen independently, and the Chernoff bound yields that $Prob(R$ contains more than $(1 + d^{-1/2})|A||B|d/2^n$ 1-inputs$) \leq e^{-|A||B|d/(3 \cdot 2^n d^{1/4})} \leq e^{-2^n d^{1/4}/3}$. Similarly, we can bound $Prob(R$ contains less than $(1 - d^{-1/2})|A||B|d/2^n$ 1-inputs).

Furthermore, since there are at most $2^{2^{n+1}}$ rectangles, and by the union bound with high probability these estimates are correct for *all* rectangles with enough rows and columns (in particular the rectangle consisting of all inputs).

Note that R contains at least $|A| \cdot |B| - |A|2d$ 0-inputs, each of which have weight at least $1/(2^{2n+1})$, for a total 0-weight of at least $|A||B|/2^{2n+1} - d/2^n$. The weight of a single 1-input is at most $1/(1 - d^{-1/2}) \cdot 1/(d2^{n+1})$ and the total 1-weight of R is at most $(1 + d^{-1/2})/(1 - d^{-1/2}) \cdot |A||B|/2^{2n+1}$ by the above. Hence the one-sided discrepancy is at most $O(d^{-1/2}|A||B|/2^{2n+1}) \leq O(d^{-1/2})$. ◀

We will now show that most functions $f_{n,d}$ are easy under all low information distributions, but hard for information n distributions, by showing that $f_{n,d}$ has a certain property with high probability. We assume in the following that $d \leq 2^{\epsilon^2 n}$ and set $\epsilon = 1/10$.

▶ **Definition 30.** We say a Boolean $2^n \times 2^n$ matrix is *good*, if it is true that every rectangle $A \times B$ with $\min\{|A|, |B|\} \leq 2^{2n/3}$ has no more than $100 \max\{|A|, |B|\}$ 1-entries. We also call any rectangle $A \times B$ with $\min\{|A|, |B|\} \leq 2^{2n/3}$ in a good matrix *good.*

▶ **Lemma 31.** *With high probability the communication matrix of $f_{n,d}$ is good.*

Proof. Fix A, B. Assume that $|B| \geq |A|$ and that $|A| \leq 2^{2n/3}$. The probability that a fixed x, y is a 1-input is $d/2^n$. The probability that there are at least $100|B|$ 1-inputs in R is at most $\binom{|A||B|}{100|B|} \cdot (d/2^n)^{100|B|} \leq (\frac{|A|d}{2^n})^{100|B|} \leq \frac{d}{2^{n/3}}^{100|B|}$.

There are $\binom{2^n}{|A|}\binom{2^n}{|B|} \leq (e2^n/|B|)^{2|B|}$ rectangles of this size. By the union bound the probability that there is a rectangle that is not good is small. ◀

Now assume that f (or rather its matrix) is good. Consider any ν such that $I(X : Y) \leq \epsilon^3 n$. We have to give a protocol for f under ν. By Fact 4 there is another distribution μ, that is $\epsilon/2$-close to ν and has $I_\infty(X : Y) \leq 8\epsilon^2 n$. We describe a protocol for f under μ with error $\epsilon/2$. The same protocol has error at most ϵ under ν. We assume $d \leq 2^{\epsilon^2 n}$.

Alice and Bob consider the marginal distributions μ_A and μ_B. Alice sends 0, if $\mu_A(x) \leq 2^{-n/2 - \epsilon n}$, and 1, otherwise, and Bob does the same for $\mu_B(y)$. We first consider the rectangle

R_{00} the messages were 00. Then $\mu_A(x) \cdot \mu_B(y) \leq 2^{-n-2\epsilon n}$ for all x, y in R_{00}. Hence on this rectangle $\sum_{x,y \in R: f(x,y)=1} \mu_A(x)\mu_B(y) \leq 2d2^{-2\epsilon n}$. That means that under $\mu_A \times \mu_B$ the probability of 1-inputs in R_{00} is at most $2d2^{-2\epsilon n}$. But since $I_\infty(X : Y) \leq 8\epsilon^2 n$, the probability of 1-inputs there under μ is at most $2^{-2\epsilon n + O(\epsilon^2 n)}$. We can reject on R_{00} without introducing much error.

Now consider one of the remaining rectangles, say $R_{10} = A \times B$. Clearly, this rectangle has $|A| \leq 2^{n/2+\epsilon n}$. Assume $|A| \leq |B|$. By the above lemma this means that $A \times B$ is good, i.e., contains relatively few 1-inputs, on average only 100 per column.

On R_{10} Alice and Bob send public coin fingerprints of x, y each, with error guarantee $\epsilon/1000$. This takes communication $O(-\log \epsilon)$. If a row or column contains few 1-inputs Alice resp. Bob can test with the fingerprint whether x, y is one of these. But R_{10} only contains few 1-inputs on average, and it is quite possible that both the row and the column of x, y have many 1-inputs.

Let $A = A_0$ and $B = B_0$. Assume that $|A| \leq |B|$. Define A_i as the set of $x \in A_{i-1}$ such that there are at least 1000 1-inputs x, y' with $y' \in B_{i-1}$ and B_i the set of $y \in B_{i-1}$ such that there are at least 1000 1-inputs x', y with $x' \in A_{i-1}$.

Clearly, all $A_i \times B_i$ are good. Assume that $|A_i| \leq |B_i|$. $A_i \times B_i$ has at most $100|B_i|$ 1-inputs. $A_i \times B_{i+1}$ has at least $1000|B_{i+1}|$ 1-inputs, hence $|B_{i+1}| \leq |A_i|/10$, because $A_i \times B_{i+1}$ is good: $1000|B_{i+1}| \leq 100 \max\{|A_i|, |B_{i+1}|\}$. That means that for odd i we have $|B_i| \leq |A_{i-1}|/10$ and for even i we have $|A_i| \leq |B_{i-1}|/10$.

All sets A_i, B_i are known to Alice and Bob without communication. Also, due to the shrinking sizes, all $i \leq O(n)$.

The protocol works as follows: Alice determines the first i such that on $A_i \times B_{i-1}$ her row contains at most 1000 1-inputs and sends this information. Bob also sends the index j, such that on $A_{j-1} \times B_j$ his column contains at most 1000 1-inputs. If $i < j$, then Bob also sends a fingerprint of y with error guarantee $1/10000$ (see Fact 14). If there is a $y' \in B_{i-1}$ with the same fingerprint and $f(x, y') = 1$ then Alice accepts, otherwise she rejects. If $i > j$, then Alice sends the fingerprint, and Bob accepts if and only if there is an $x' \in A_{j-1}$ with $f(x', y) = 1$. Clearly the communication is $2 \log n + O(1)$, and is done in 2 rounds.

Correctness: Assume $i < j$. The players can be sure that $x, y \in A_i \times B_{i-1}$. There are at most 1000 1-inputs in row x in B_{i-1}. If $f(x, y) = 1$, then certainly the fingerprints will coincide, and Alice accepts. Otherwise the probability that the fingerprints equal is at most $100/10000 = 1/10$.

▶ **Lemma 32.** *Under ν with information at most $\epsilon^3 n$ and for $6n \leq d \leq 2^{\epsilon^2 n}$ we have that $R^\nu_\epsilon(f) \leq O(\log n)$, if f is good.*

▶ **Theorem 33.** *For every $6n \leq d \leq 2^{n/100}$ there is a function f_d such that*

- $R(f_d) = \Theta(\log d)$,
- $R^{I \leq n/1000}_{1/10}(f_d) \leq O(\log n)$.

6 One-Round Error Dependence

We now consider the general question of error dependence under distributions with limited information. In the case, where the information is bounded only by n, we get the standard randomised (resp. quantum) communication complexity, for which the usual boosting techniques (i.e., the Chernoff bound) show that the error dependence is at most factor of $O(\log(1/\epsilon))$. Furthermore, Corollary 20 shows that for all information parameters $1 < k < n^{1-\Omega(1)}$ the error dependence is polynomial. This leaves the case of product distributions, where in the randomised two-way communication case DISJ has logarithmic error dependence. In this section we show that for *all* total functions, in the case of one-way communication complexity the error dependence is small under product distributions. The corresponding statement about two-way protocols remains open.

In [19] Kremer et al. show that the complexity of one-way protocols for total functions under product distributions is determined by the VC-dimension (see also [17]).

▶ **Definition 34.** The VC-dimension of a Boolean matrix M is the largest k such that there is a $2^k \times k$ rectangle R in M such that R contains all Boolean strings of length k as rows.

The VC-dimension in turn characterises the number of examples needed to PAC-learn the concept class given by the rows of the communication matrix of f, under any distribution on the columns. Usually in learning theory a concept class is a set of Boolean functions ('concepts'), and here we view rows of the communication matrix of f as functions $f_x(y) = f(x,y)$. The task of PAC learning is for the learner to be able to compute $f_x(y)$ for most y under a distribution μ, after having seen labelled examples from the same distribution. It is well known, that $O(VC(f) \cdot 1/\epsilon \cdot \log(1/\epsilon))$ examples suffice [17].

Kremer et al. [19] proved the following upper bound on one-way communication complexity: $R_\epsilon^{A\to B,I=0}(f) \leq O(VC(f)\cdot 1/\epsilon \cdot \log(1/\epsilon))$. The idea is that Alice and Bob can choose examples y' from the public coin, which Alice can label by sending $f(x,y')$. Bob simulates the PAC learning algorithm for the rows of the communication matrix, and hence he can successfully predict $f(x,y)$ for most y, including (likely) his own input. Note that there is also a lower bound of $Q^{A\to B,I=0}(f) \geq (1-H(\epsilon))VC(f)$ (which is even true in the entanglement assisted case with an additional factor of 1/2)[19, 3, 18].

While it is known, that the number of examples needed to PAC-learn is at least $\Omega(VC(f)/\epsilon)$ [17], we get an exponentially better dependence on the error here for the one-way communication model under product distributions.

Our result has an appealing interpretation. Both the one-way model under product distributions and the PAC model can be viewed as learning models (for this it is crucial that the distributional one-way model is considered under product distributions). In the PAC model Alice (or nature) labels random examples drawn from a distribution, and Bob has to end up being able to label new examples mostly correct (under the same, unknown distribution). In the one-way model, there is a known distribution on examples (columns), and a known distribution on concepts (rows). The one-way model under product distributions can clearly simulate any PAC algorithm. But Alice can send any information she deems useful, not just label examples. Nevertheless, in both models the complexity is determined by the VC-dimension. Is a teacher like Alice not more useful than random labelled examples? We show that the one-way model (i.e., a teacher) is better in the sense that making the error small is exponentially cheaper there, compared to the PAC model.

▶ **Theorem 35.** *For all total f: $R_\epsilon^{A\to B,I=0}(f) \leq O(Q_{1/3}^{A\to B,I=0}(f)\cdot \log(1/\epsilon))$*

Proof. First, $Q_{1/3}^{A\to B,I=0}(f) = \Theta(VC(f))$. Hence we need to show only that $R_\epsilon^{A\to B,I=0}(f) \leq O(VC(f)\cdot \log(1/\epsilon))$.

For a given distribution μ on the columns, an ϵ-net among the rows of the communication matrix is a subset N of the set of rows, such that for every row x there is a row $x' \in N$ which coincides with x with probability $1-\epsilon$ under μ. We have the following simple observation, due to the fact that Alice can simply send the name of the closest $x' \in N$ to Bob.

▶ **Lemma 36.** *$R_\epsilon^{\mu_A\times\mu_B}(f)$ is upper bounded by the logarithm of the size of the smallest ϵ-net for f and μ_B.*

Hence instead of the simulation Alice and Bob can agree on an ϵ-net beforehand, and the size of the ϵ-net determines the complexity of the protocol. Note that PAC-learners also try to find an ϵ-net, but they are restricted to finding one from random examples. The size of the constructed ϵ-net is much smaller than the number of examples (this is not surprising, since otherwise the concept is not learned yet). Indeed, Sauer's lemma tells us enough about the size of the ϵ-net, when the specified number of examples have been chosen.

▶ **Fact 37** (Sauer). *Let M be a Boolean matrix with r rows and c columns and VC-dimension d. Then $r \leq \Phi(c,d)$, where $\Phi(c,d) = \sum_{i=0,\ldots,d} \binom{c}{i} \leq d \cdot \binom{c}{d}$.*

We now state the fundamental result from PAC learning (see Theorem 3.3 in [17]).

▶ **Fact 38.** *Consider any function $f : \{0,1\} \times \{0,1\}^n \to \{0,1\}$. Assume we fix any x, and there is a distribution μ on y's that does not depend on x. We are given $c = O(VC(f) \cdot 1/\epsilon \cdot \log(1/\epsilon))$ random examples $y_1, \ldots, y_c$ from the distribution and labels $\ell_1 = f(x, y_1), \ldots, \ell_c = f(x, y_c)$. If we use any x' that is consistent with these values, i.e., $f(x', y_i) = \ell_i$ for all $i = 1, \ldots, c$, then the probability that $f(x', y) \neq f(x, y)$ is at most ϵ under μ, i.e., if we choose a string x' consistent with any vector $\ell_1, \ldots, \ell_c$, then we get an ϵ-net for f, μ.*

The size of this ϵ-net is clearly at most 2^c. Sauer's lemma can be used to show that the constructed ϵ-net is actually much smaller. The size of the ϵ-net constructed in Fact 38 is at most the size of the set of *distinct* rows in the matrix for f, when we restrict the matrix to the c chosen columns (we may choose one x' for every distinct value of the c labels appearing and add it into the ϵ-net).

The size of the number of distinct rows is bounded now by Sauer's lemma as follows: $VC(f) \cdot \binom{c}{VC(f)} = VC(f) \cdot \binom{const \cdot VC(f) \cdot 1/\epsilon \cdot \log(1/\epsilon)}{VC(f)} \leq (1/\epsilon)^{O(VC(f))}$. Hence the communication is at most the logarithm of this size, which yields the theorem. ◀

7 Open Problems

- Can the error dependence of a tight upper bound on $Q_\epsilon^{I=0}(DISJ)$ be improved to $\log(1/\epsilon)$?
- Can the error dependence of $R_\epsilon^{I=0}(f)$ be improved to $\log(1/\epsilon)$ for *every* total function f?
- What is the trade-off between the number of rounds and the randomised complexity of DISJ under product distributions?
- What is the quantum communication complexity of DISJ where the inputs are sets of size $\sqrt{n}$ from a size n universe? The best known lower bound is $\Omega(n^{1/4})$, the best known upper bound is $O((n^{1/4}) \log n)$.
- What is the largest gap between $Q^{I=0}(f)$ and $R^{I=0}(f)$? In the one-way model there is at most a constant gap for any total function. We have shown a quadratic gap for DISJ.

References

1 S. Aaronson and A. Ambainis. Quantum search of spatial regions. *Theory of Computing*, 1(1):47–79, 2005. Earlier version in FOCS'03. quant-ph/0303041.

2 Noga Alon, Shay Moran, and Amir Yehudayoff. Sign rank, VC dimension and spectral gaps. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:135, 2014.

3 A. Ambainis, A. Nayak, A. Ta-Shma, and U. V. Vazirani. Dense quantum coding and quantum finite automata. *Journal of the ACM*, 49(4):496–511, 2002. Earlier version in STOC'99.

4 L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory. In *Proceedings of 27th IEEE FOCS*, pages 337–347, 1986.

5 Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *Proceedings of 43rd IEEE FOCS*, pages 209–218, 2002.

6 H. Buhrman, R. Cleve, and A. Wigderson. Quantum vs. classical communication and computation. In *Proceedings of 30th ACM STOC*, pages 63–68, 1998. quant-ph/9802040.

7 B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988. Earlier version in FOCS'85.

8 T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley, 1991.

9 Ronald de Wolf. Quantum communication and complexity. *Theoretical Computer Science*, 287(1):337–353, 2002.

10 Prahladh Harsha, Rahul Jain, David A. McAllester, and Jaikumar Radhakrishnan. The communication complexity of correlation. *IEEE Transactions on Information Theory*, 56(1):438–449, 2010.

11 Johan Håstad and Avi Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007.

12 R. Jain, H. Klauck, and A. Nayak. Direct product theorems for classical communication complexity via subdistribution bounds. In *Proc. of 40th ACM STOC*, pages 599–608, 2008.

13 R. Jain, J. Radhakrishnan, and P. Sen. Privacy and interaction in quantum communication complexity and a theorem about the relative entropy of quantum states. In *Proceedings of 43rd IEEE FOCS*, pages 429–438, 2002.

14 R. Jain and S. Zhang. New bounds on classical and quantum one-way communication complexity. *Theoretical Computer Science*, 410(26):2463–2477, 2009.

15 Rahul Jain, Jaikumar Radhakrishnan, and Pranab Sen. A direct sum theorem in communication complexity via message compression. In *ICALP*, page 187, 2003.

16 B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992. Earlier version in Structures'87.

17 Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, 1994.

18 H. Klauck. On quantum and probabilistic communication: Las Vegas and one-way protocols. In *Proceedings of 32nd ACM STOC*, pages 644–651, 2000.

19 I. Kremer, N. Nisan, and D. Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999. Earlier version in STOC'95. Correction at `http://www.eng.tau.ac.il/~danar/Public/KNR-fix.ps`.

20 E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge Univ. Press, 1997.

21 Marco Molinaro, David P. Woodruff, and Grigory Yaroslavtsev. Amplification of one-way information complexity via codes and noise sensitivity. In *ICALP*, pages 960–972, 2015.

22 A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.

23 A. Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya of the Russian Academy of Sciences, mathematics*, 67(1):159–176, 2003. quant-ph/0204025.

24 Mert Saglam and Gábor Tardos. On the communication complexity of sparse set disjointness and exists-equal problems. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 678–687, 2013.

25 Alexander A. Sherstov. Communication complexity under product and nonproduct distributions. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity, CCC*, pages 64–70, 2008.

A Randomised Protocol for DISJ under Product Distributions

Proof of Theorem 15. Fix any product distribution μ on $\{0,1\}^n \times \{0,1\}^n$. The main idea is (just like in [4]) to have a first phase in which large sets are reduced in size until both sets have size $O(\sqrt{n})$. In phase 2 we employ the randomised protocol for DISJ on small sets given by Hastad and Wigderson [11] (instead of communicating the sets). To simplify our presentation we describe a randomised protocol.

Set $S = \sqrt{n}$. In phase 1 Alice and Bob try to shrink the universe U (without removing positions in $x \cap y$) until the size of U is at most S. At that point also $|x \cap U|$ and $|y \cap U|$ are at most of size S and the players move to phase 2. The protocol starts with the universe $U_0 = \{1, \ldots, n\}$. The players maintain a current universe U_i until U_i is small at some point.

The protocol proceeds in rounds during phase 1 (we later explain how to get rid of all but two rounds). In each round Alice and Bob exchange a bit each, indicating whether $|x|, |y| \geq S$ or not. If both are smaller, they move to phase 2. The players also maintain a current rectangle of inputs $R_i = A_i \times B_i$ (this would be immediate in a deterministic protocol, but needs to be maintained in the randomised case).

After this exchange, Alice and Bob each compute $Prob(x, y$ are disjoint) on the current distribution restricted to R_i and their row/column. If this probability is less than ϵ for someone, they reject and quit the protocol. Otherwise, one player who has a large set still, say Alice, uses the public coin to generate samples $y' \in B_i$. These are disjoint from x with probability at least ϵ. Hence, Alice can name a disjoint y' with expected communication $O(\log(1/\epsilon))$. Since $x \cap y$ is disjoint with y' they set $U_{i+1} = U_i - y'$. The size of the universe decreases by at least $\sqrt{n}$ in each round in phase 1, the communication is expected $O(\log(1/\epsilon))$ per round, and there are at most $\sqrt{n}$ rounds.

Phase 2, as mentioned, is the protocol from [11], which solves DISJ with communication $O(\sqrt{n}\log(1/\epsilon))$ and worst case error ϵ on sets of size at most $\sqrt{n}$.

Hence the total expected communication is at most $O(\sqrt{n}\log(1/\epsilon))$. We need a protocol with a *worst case* communication bound, though, but note that during each round in phase 1, using the public coin to pick a new y' corresponds to a Bernoulli trial with success probability at least ϵ. The communication cost is the logarithm of the number of the first successful trial. The probability that this is larger than $t\log(1/\epsilon)$ is at most $e^{-1/\epsilon^{t-1}}$. Assume there are T rounds in phase 1. The probability that the message length in any round is more than $(T+1)\log(1/\epsilon)$ is at most $T \cdot e^{-1/\epsilon^T} \leq \epsilon$. Hence we can assume that the message length is at most $(T+1)\log(1/\epsilon)$ in all rounds (the probability that this is not the case is bounded by ϵ).

We now bound the probability that the total message length is more than $10T\log(1/\epsilon)$, by appealing to the Hoeffding bound. Note that the message lengths of all rounds are (still) independent, and that we just established an upper bound on the message length. The Hoeffding bound now implies that the probability of the total message length being larger than the stated bound is at most ϵ. Furthermore, we have that $T \leq \sqrt{n}$ with certainty. This shows that the communication of phase 1 is at most $O(\sqrt{n}\log(1/\epsilon))$. Note that the protocol needs to be modified such that it aborts if the communication in phase 1 exceeds this bound. This introduces error at most ϵ. ◀

B Randomised Protocol and Distributions with Bounded Mutual Information

Proof of Theorem 16. Fix any distribution μ' that has information at most k. The protocol we describe again has 2 phases. Informally, the first phase shrinks the sets of Alice and

Bob (which could be arbitrarily large) until their sizes are both small enough. The second phase is small set disjointness, as considered before by Hastad and Wigderson [11], and more recently by Saglam and Tardos [24]. We will establish an upper bound of $O(\sqrt{n(k+1)}/\epsilon)$ on the *expected* communication complexity with error ϵ. Then the theorem (which claims a worst-case bound) follows via the Markov inequality: if the stated communication bound is violated, stop the protocol and output a random bit.

Set $S = \sqrt{(k+1)n}$. The goal of the first phase is to make both sets smaller than S. Suppose Alice holds x and Bob y. They communicate to determine one of them has a set larger than S. This needs communication $O(1)$. If both sets are small we move to phase 2 described below.

In phase 1 Alice and Bob try to shrink the universe U until the size of U is at most S. At that point also $|x \cap U|$ and $|y \cap U|$ are at most S and the players move to phase 2. The protocol starts with the universe $U_0 = \{1, \dots, n\}$. The players maintain a current universe U_i until U_i is small at some point.

Note that while the information under $\mu_0 = \mu$ is at most k, in some branches of the protocol the information on the current sub-rectangle can grow, and we need that on average it is bounded by k. We keep a transcript $T_i = U_i, V_i, R_i$, which contains the messages exchanged in phase 1 up to round i (in every round either Alice or Bob sends a message, which goes into U_i resp. V_i), as well as the random variable R_i containing the public coins used so far. Note that conditioned on a fixed value r of R_i the message transcript $U_i(r) \times V_i(r)$ is a rectangle in the communication matrix.

Then $I(X:Y|T_i) = H(XU_iV_i|R_i) + H(YU_iV_i|R_i) - H(U_iV_i|R_i) - H(XYU_iV_i|R_i) \leq H(XU_i|R_i) + H(YV_i|R_i) - H(XYU_iV_i|R_i) + I(U_i:V_i|R_i) = I(XU_i:YV_i|R_i) = I(X:Y|R_i) = I(X:Y)$, hence the information does not increase on average.

Denote by μ_{t_i} the distribution on inputs conditional on the transcript being $T_i = t_i$. $\mu^x_{t_i}$ is μ_{t_i} restricted to the row $X = x$. $\tilde{\mu}, \tilde{\mu}_{t_i}, \tilde{\mu}^x_{t_i}$ denote the distributions restricted to 1-inputs of DISJ. $\mu_{t_i,Y}$ is the marginal of μ_{t_i} on Bob's inputs. $\tilde{\mu}^x_{t_i,Y}$ is the distribution on y's under $T_i = t_i$, for fixed x and conditioned on $x \cap y = \emptyset$. $\mu^x_{t_i,Y}$ is the distribution on y's under $T_i = t_i$, for fixed x.

Here is the protocol for phase 1. Explanations follow.

1. Alice and Bob check whether $|x| \leq S$ and $|y| \leq S$ on U_i. If both are, they move to phase 2. W.l.o.g. assume that $|y| \geq S$, otherwise the following steps are done by Bob in an analogous fashion.
2. Alice computes the probability that $DISJ(x, y') = 1$ if y' is chosen from $\mu^x_{t_i}$. If this probability is less than $\epsilon/2$, she ends the protocol with output 0.
3. Alice computes $\tilde{\mu}^x_{t_i,Y}$. Another distribution, this one known to both players, is $\mu_{t_i,Y}$.
4. Alice and Bob use rejection sampling as in Fact 8 (using the distributions $\tilde{\mu}^x_{t_i,Y}$ and $\mu_{t_i,Y}$) to discover a y'_i distributed according to $\tilde{\mu}^x_{t_i,Y}$.
5. Alice and Bob set $U_{i+1} = U_i - y'_i$.
6. t_{i+1} is t_i together with the message and randomness from 1. $\mu_{t_{i+1}}$ is μ conditioned on $T_{i+1} = t_{i+1}$.
7. Move to step 1.

We note the following on the different steps.

1. Communication is $O(1)$.
2. Clearly the total error introduced by these steps under μ can never be more than $\epsilon/2$. If the protocol moves ahead the probability of $DISJ(x, y) = 1$ is at least $\epsilon/2$ under $\mu^x_{t_i}$.
3. Since $I(X:Y|T_i) \leq k$ we have that $E_{t_i,x} D(\mu^x_{t_i,Y} || \mu_{t_i,Y}) \leq k$.

4. $D(\tilde{\mu}^x_{t_i,Y}||\mu_{t_i,Y}) \le 2(D(\mu^x_{t_i,Y}||\mu_{t_i,Y})+1)/\epsilon - \log(\epsilon/2)$ due to Lemma 7 and hence the rejection sampling protocol from Fact 8 uses expected communication $O((k+1)/\epsilon)$. Drawn y'_i are always disjoint from x.
5. $|y'_i| \ge S$. Hence $|U_i - U_{i+1}| \ge S$. This step can be performed at most $n/\sqrt{n/(k+1)}$ times.

The protocol ends phase 1 with sets $x \cap U_j$ held by Alice and $y \cap U_j$ held by Bob, and $|x \cap U_j|, |y \cap U_j| \le S$, and $DISJ(x,y) = 1 \Leftrightarrow DISJ(x \cap U_j, y \cap U_j) = 1$. The probability that the protocol ends during phase 1 and makes an error is at most $\epsilon/2$. The expected communication is at most $O(\sqrt{n(k+1)}/\epsilon$.

Phase 2 is simply the Hastad Wigderson protocol for small set disjointness [11], that finishes the protocol in communication $O(\sqrt{n(k+1)}\log(1/\epsilon))$ and with worst case error $\epsilon/2$. Hence we get a protocol with error ϵ, and expected communication $O(\sqrt{n(k+1)}/\epsilon)$. ◀

C Randomised Lower Bound for DISJ

We first bound the information. Letting X and Y follow the marginal distributions of $\mu_{n,k}$, respectively, we have:

$$\begin{aligned}
I(X:Y) = H(X) - H(X|Y) &= \log\binom{n}{m} - \mathbf{E}_{y\in Y}(Pr(y)H(X|Y=y)) \\
&= \log\binom{n}{m} - H(X|Y=y_0) \text{ (where } y_0 \text{ is any set with } P(y_0)>0) \\
&= \log\binom{n}{m} - \left(2\log\binom{n-m}{m} + 2\log\binom{n-m}{m-1} + 2\right) \\
&\le \log\binom{n}{m} - \log\binom{n-m}{m} = \log\frac{n(n-1)\cdot\ldots\cdot(n-m+1)}{(n-m)\cdot\ldots\cdot(n-2m+1)} \\
&\le \log\left(1+\frac{m}{n-2m+1}\right)^m \le (\log e)\frac{m^2}{n-2m+1} \\
&= c^2(\log e)(1+o(1))(k+1) \le k
\end{aligned}$$

for any sufficiently large n.

Proof of Theorem 18. We may assume that $k = o(n)$, since otherwise (if $k = \Omega(n)$), the original proof by Razborov [22] applies directly. Let $l \in \mathbb{N}$ be given and assume that $n = 4l-1$. Let $\gamma = \log_l(c\sqrt{n(k+1)})$, where $c = (\log e)^{-1}$. Thus $\gamma \in \left(\frac{1}{2},1\right)$ (for n sufficiently large) and our distribution will pick sets of size $l^\gamma = c\sqrt{n(k+1)}$. Throughout the proof we will treat numbers like l^γ as natural numbers, and avoid using the floor function for the sake of readability. We will also identify $\mathcal{P}(\{1,\ldots,n\})$ with $\{0,1\}^n$.

We now give an alternative definition for the distribution $\mu = \mu_{n,k}$, as the distribution induced by the following process: First, a triple $T = (T_1, T_2, i)$ is chosen uniformly among all such triples, where $|T_1| = |T_2| = 2l-1$ and $\{T_1, T_2, \{i\}\}$ form a partition of the set $\{1,\ldots,n\}$. Then, with probability $\frac{1}{2}$ the set x is chosen uniformly among all subsets of $T_1 \cup \{i\}$ with l^γ elements and such that they contain i, and with probability $\frac{1}{2}$ the set x is chosen as a subset of T_1 with l^γ elements, again uniformly among all such subsets of T_1. Similarly, and independently of the choice of x, y is chosen with probability $\frac{1}{2}$ uniformly as a subset of $T_2 \cup \{i\}$ with l^γ elements and such that it contains i, and with probability $\frac{1}{2}$ uniformly among the subsets of T_2 with l^γ elements (not containing i). Thus non-zero probabilities are assigned only on the set $\{(x,y) \mid x,y \subseteq \{1,\ldots,n\}, |x| = |y| = l^\gamma, |x \cap y| \in \{0,1\}\}$.

Now the statement that $D_{\epsilon}^{\mu_{n,k}}(DISJ) = \Omega(\sqrt{n(k+1)})$ for any sufficiently small constant $\epsilon > 0$, follows directly from Lemma 39 below. ◀

▶ **Lemma 39.** *Let γ and μ be defined as in the proof of Theorem 18. Let $A = \{(x, y) \mid \mu(x, y) > 0$ and $x \cap y = \emptyset\}$ and $B = \{(x, y) \mid \mu(x, y) > 0$ and $x \cap y \neq \emptyset\}$. For any sufficiently small $\epsilon > 0$ we have for any rectangle $R = C \times D \subseteq \mathcal{P}(\{1, \dots, n\})^2$ that*

$$\mu(B \cap R) \geq \Omega(\mu(A \cap R)) - 2^{-\Omega(n^{\gamma})}.$$

Proof. We consider $\epsilon > 0$ to be fixed (but will specify its value later). We begin by defining for any triple $T = (T_1, T_2, \{i\})$ as above, the numbers $Row(T) = Pr[x \in C \mid x \subseteq T_1 \cup \{i\}]$, $Row_0(T) = Pr[x \in C \mid x \subseteq T_1 \cup \{i\}, i \notin x]$ and $Row_1(T) = Pr[x \in C \mid x \subseteq T_1 \cup \{i\}, i \notin x]$, and similarly $Col(T) = Pr[y \in D \mid y \subseteq T_2 \cup \{i\}]$, $Col_0(T) = Pr[y \in D \mid y \subseteq T_2 \cup \{i\}, i \notin y]$ and $Col_1(T) = Pr[y \in D \mid y \subseteq T_2 \cup \{i\}, i \notin y]$. It is important to note that $Row(T) = \frac{1}{2}(Row_0(T) + Row_1(T))$ and $Col(T) = \frac{1}{2}(Col_0(T) + Col_1(T))$, just as in the case of Razborov's original distribution, and for the same reasons.

Next, for a triple $T = (T_1, T_2, \{i\})$ (and under the above distribution μ) we say that T is *x-bad* if $Row_1(T) < \frac{1}{6}Row_0(T) - 2^{-\epsilon n^{\gamma}}$, and that T is *y-bad* if $Col_0(T) < \frac{1}{6}Col_0(T) - 2^{-\epsilon n^{\gamma}}$. If T is x-bad or y-bad, we say that T is *bad*. Let $Bad_x(T)$, $Bad_y(T)$ and $Bad(T)$ be the respective event indicators.

▶ **Claim 40.** *For all $t_2 \subseteq \{1, \dots, n\}$, with $|t_2| = 2l - 1$, we have that $Pr[Bad_x(T) = 1 \mid T_2 = t_2] \leq \frac{1}{5}$ and $Pr[Bad_y(T) = 1 \mid T_2 = t_2] \leq \frac{1}{5}$.*

Proof of the Claim. We prove the first statement, the second one having an almost identical proof.

Let $t_2 \subseteq \{1, \dots, n\}$, with $|t_2| = 2l - 1$, be fixed. Under our distribution, $Row(T)$ can take different values even when T is restricted to partitions for which $T_2 = t_2$. Thus we first treat the case when $\max\{Row(T) \mid T_2 = t_2\} \leq 2^{-\epsilon n^{\gamma}}$. If this inequality holds, then for all T with $T_2 = t_2$ we have: $Row(T) \leq 2^{-\epsilon n^{\gamma}}$, and hence $Row_0(T) \leq 2Row(T) \leq 2 \cdot 2^{-\epsilon n^{\gamma}}$ so that $\frac{Row_0(T)}{6} - 2^{-\epsilon n^{\gamma}} < 0 \leq Row_1(T)$ holds trivially (and hence $Pr[Bad_x(T) = 1 \mid T_2] = 0$).

Next we treat the case where $\max\{Row(T) \mid T_2 = t_2\} > 2^{-\epsilon n^{\gamma}}$. Define $S = \{x \in C \mid |x| = l^{\gamma}, x \subset \{1, \dots, n\} \setminus t_2\}$. Note that for any T with $T_2 = t_2$, $Row(T)$ measures the conditional probability (conditioned on T) of the same set S, with each $x \in S$ having a different (conditional) probability depending on whether $i \in x$. Specifically, if $i \in x$ then the probability of x being chosen, conditioned on T, is $\frac{1}{2}\binom{2l-1}{l^{\gamma}-1}^{-1}$, otherwise the probability is $\frac{1}{2}\binom{2l-1}{l^{\gamma}}^{-1} = \frac{1}{2}\binom{2l-1}{l^{\gamma}-1}^{-1}\frac{l^{\gamma}}{2l-l^{\gamma}} = \frac{1}{2}\binom{2l-1}{l^{\gamma}-1}^{-1}\frac{1}{2l^{1-\gamma}-1}$. Thus, when T is fixed, the probability of each set x containing i is $2l^{1-\gamma} - 1$ times that of a set which does not contain i.

The proof of this case will proceed as follows: First, we show that under the assumption that a sufficiently large part of the partitions T with $T_2 = t_2$ are x-bad, three quarters of the elements of S (which are subsets of $\{1, \dots, n\} \setminus T_2$) must have at least $\frac{21}{25}$ of their elements in a subset of $\{1, \dots, n\} \setminus T_2$ of size $\frac{8l}{5}$. We will then upper-bound the number of subsets of $\{1, \dots, n\} \setminus T_2$ of size l^{γ} that have this property (regardless of whether they are in C or not). Next, we will lower-bound $\frac{3}{4}|S|$ in terms of ϵ, and show that for a suitable choice of ϵ, the lower bound for $\frac{3}{4}|S|$ is in fact larger than the upper bound we computed before, which is a contradiction showing that it is not possible for that T with $T_2 = t_2$ to be x-bad for that many choices of i.

Note first that whenever T_2 is fixed (in our case to t_2), the choice of $i \in \{1, \dots, n\} \setminus T_2$ also fixes T_1 and hence all of T, and that the choice of i determines the proportion of $x \in S$ whose weights are counted in $Row_1(T)$. If for a particular choice of i the resulting T is

x-bad, then by definition we have that $Row_1(T) < \frac{1}{6}Row_0(T) - 2^{-\epsilon n^\gamma}$, and in particular that $Row_1(T) < \frac{1}{6}Row_0(T)$. If we let S' be the set of $x \in S$ with $i \in x$, then we may rewrite this inequality as:

$$\frac{|S'|}{\binom{2l-1}{l^\gamma-1}} < \frac{|S|-|S'|}{6\binom{2l-1}{l^\gamma}} \iff |S'| < \frac{|S|-|S'|}{6(2l^{1-\gamma}-1)}$$
$$\iff |S'|\left(1+\frac{1}{6(2l^{1-\gamma}-1)}\right) < \frac{|S|}{6(2l^{1-\gamma}-1)},$$

and we may conclude that for l sufficiently large, $|S'| < \frac{|S|}{10l^{1-\gamma}}$ (under the assumption that T is x-bad). For the last inequality we have used the fact that $\lim_{n\to\infty} l^{1-\gamma} = \infty$, which holds because: $\lim_{n\to\infty} \log l^{1-\gamma} = \lim_{n\to\infty}(1-\gamma)\log l = \lim_{n\to\infty}(1-\log_l(c\sqrt{n(k+1)}))\log l \geq \lim_{n\to\infty}(\log l - \log\sqrt{n(k+1)}) \geq \lim_{n\to\infty}\log\sqrt{\frac{n+1}{16(k+1)}} = \infty$ (since $k = o(n)$).

Let $B = \{i \in \{1,\ldots,n\} \mid \text{the partition } (\{1,\ldots,n\}\setminus(t_2\cup\{i\}), t_2, \{i\}) \text{ is } x\text{-bad}\}$, and assume that $|B| \geq \frac{2l}{5}$, that is, assume that for at least one fifth of the possible choices for i the corresponding partition is x-bad. By excluding some elements of B, we may assume that $|B| = \frac{2l}{5}$. Now, if we consider the number of pairs (x,i) with $x \in S$ and $i \in x$, we have by the inequality in the last paragraph that each of the $i \in B$ can be the second element of at most $\frac{|S|}{10l^{1-\gamma}}$ such pairs, and hence B can contribute the second element of at most $\frac{2l}{5}\frac{|S|}{10l^{1-\gamma}} = \frac{1}{25}l^\gamma|S|$ of the total of $l^\gamma|S|$ pairs. Applying the Colouring Lemma below with $X = S$, $Y = \{1,\ldots,l^\gamma\}$, $c(x,i) = 0$ if and only if the i-th smallest element of x is in B (so that $p \geq \frac{24}{25}$) and $r = \frac{21}{25}$, we have that at least three quarters of all $x \in S$ have the property that more than $\frac{21}{25}$ of their elements lie in $G = \{1,\ldots,n\}\setminus(t_2\cup B)$. Let Q be the set of subsets $x \subseteq B\cup G = \{1,\ldots,n\}\setminus t_2$, with $|x| = l^\gamma$ and the property that $|x\cap G| \geq \frac{21}{25}l^\gamma$. Then we must have that $|Q| \geq \frac{3}{4}|S|$. We will now upper-bound the size of the set Q.

Since every $x \in Q$ can have a proportion of at most $4/25$ of its elements in B, we have that

$$\begin{aligned}
\log|Q| &\leq \log\left[\sum_{i=0}^{\frac{4}{25}l^\gamma}\binom{\frac{2l}{5}}{i}\binom{\frac{8l}{5}}{l^\gamma-i}\right] \leq \log\left[\sum_{i=0}^{\frac{4}{25}l^\gamma}\left(\frac{2le}{5i}\right)^i\left(\frac{8le}{5(l^\gamma-i)}\right)^{l^\gamma-i}\right] \\
&\leq \log\left[\frac{4}{25}l^\gamma\left(\frac{2le}{5}\frac{25}{4l^\gamma}\right)^{\frac{4}{25}l^\gamma}\left(\frac{8le}{5}\frac{25}{21l^\gamma}\right)^{\frac{21}{25}l^\gamma}\right] \\
&= \log\left[\frac{4}{25}l^\gamma\left(\frac{5e}{2}l^{1-\gamma}\right)^{\frac{4}{25}l^\gamma}\left(\frac{40e}{21}l^{1-\gamma}\right)^{\frac{21}{25}l^\gamma}\right] \\
&\leq \gamma\log l + \frac{4}{25}l^\gamma\log\left(\frac{5e}{2}l^{1-\gamma}\right) + \frac{21}{25}l^\gamma\log\left(\frac{40e}{21}l^{1-\gamma}\right) + O(1) \\
&= (1-\gamma)l^\gamma\log l + \left(\frac{4}{25}\log\frac{5e}{2} + \frac{21}{25}\log\frac{40e}{21}\right)l^\gamma + O(\log l) \\
&\leq (1-\gamma)l^\gamma\log l + 2.43508\cdot l^\gamma + O(\log l),
\end{aligned}$$

where in the first line we used the inequality $\binom{m}{k} \leq \left(\frac{em}{k}\right)^k$ for each term of the sum. The inequality sign between the first and second line can be justified as follows: For $x \in (0,\frac{1}{2})$, consider the expression

$$\log\left[\left(\frac{\frac{2}{5}el}{x\cdot l^\gamma}\right)^{x\cdot l^\gamma}\left(\frac{\frac{8}{5}el}{(1-x)l^\gamma}\right)^{(1-x)l^\gamma}\right] = (1-\gamma)l^\gamma\log l + l^\gamma\left(x\log\frac{2e}{5x} + (1-x)\log\frac{8e}{5(1-x)}\right)$$

and set $f(x) = x \log \frac{2e}{5x} + (1-x) \log \frac{8e}{5(1-x)} = x \log \frac{1}{x} + (1-x) \log \frac{1}{1-x} + (1 + \log e)x + (3 + \log e)(1-x) - \log 5 = 3 + \log e - \log 5 + H(x) - 2x$. Then $f'(x) = H'(x) - 2 = \log \frac{1-x}{x} - 2$. Note that the function $\frac{1-x}{x}$ is decreasing but positive on $(0, 1)$, and we have that the smallest value $x_0 \in (0, \frac{1}{2})$ for which we can have $f'(x_0) = 0$ is $x_0 = \frac{1}{5}$, which implies that $f(x)$, and hence also the argument of the logarithm in the expression above, is strictly increasing on $(0, \frac{1}{5})$. Thus the terms of the sum

$$\sum_{i=0}^{\frac{4}{25}l^\gamma} \left(\frac{2le}{5i}\right)^i \left(\frac{8le}{5(l^\gamma - i)}\right)^{l^\gamma - i}$$

are increasing, so that each term is upper-bounded by the final term, which justifies the inequality between the the first and second line above.

Next we compute a lower bound for $\frac{3}{4}|S|$. Let T^* be a partition with $T_2^* = t_2$ and $Row(T^*) = \max\{Row(T) \mid T_2 = t_2\}$. Then we have that $\frac{3}{4}|S| = \frac{3}{4}[Row_0(T^*)\binom{2l-1}{l^\gamma} + Row_1(T^*)\binom{2l-1}{l^\gamma-1}] \geq \frac{3}{4}(Row_0(T^*) + Row_1(T^*))\binom{2l-1}{l^\gamma-1} = \frac{6}{4} Row(T^*)\binom{2l-1}{l^\gamma-1} > 2^{-\epsilon n^\gamma}\binom{2l-1}{l^\gamma-1}$. Finally we have:

$$\begin{aligned}
\log \frac{3}{4}|S| > \log \left[2^{-\epsilon n^\gamma}\binom{2l-1}{l^\gamma - 1}\right] &\geq l^\gamma \log \left((e - o(1))\frac{2l-1}{l^\gamma - 1}\right) - \epsilon n^\gamma - \Theta(\log l) \\
&\geq (1-\gamma) l^\gamma \log l + l^\gamma \log(2(e - o(1))) - \epsilon \cdot (4l-1)^\gamma - \Theta(\log l) \\
(\text{for large } l) &\geq (1-\gamma) l^\gamma \log l + 2.4426 \cdot l^\gamma - \epsilon \cdot (4l)^\gamma - \Theta(\log l).
\end{aligned}$$

For $\epsilon \leq \frac{1}{1000 \cdot 4^\gamma}$ we get the desired contradiction, that $\frac{3}{4}|S| > |Q|$.

The lower-bound for $\binom{2l-1}{l^\gamma-1}$ above can be obtained using the Stirling bounds for the factorial, $\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \leq n! \leq e\sqrt{n}\left(\frac{n}{e}\right)^n$, as follows:

$$\begin{aligned}
\binom{2l-1}{l^\gamma - 1} &\geq \frac{\sqrt{2\pi(2l-1)} \cdot (2l-1)^{2l-1}}{e^2\sqrt{(l^\gamma-1)(2l-l^\gamma)} \cdot (l^\gamma-1)^{l^\gamma-1} \cdot (2l-l^\gamma)^{2l-l^\gamma}} \\
&= \frac{\sqrt{2\pi(2l-1)}}{e^2\sqrt{(l^\gamma-1)(2l-l^\gamma)}} \left(\frac{2l-1}{l^\gamma-1}\right)^{l^\gamma-1} \left(\frac{2l-1}{(2l-1)-(l^\gamma-1)}\right)^{2l-l^\gamma} \\
&= \frac{\sqrt{2\pi(2l-1)}}{e^2\sqrt{(l^\gamma-1)(2l-l^\gamma)}} \left(\frac{2l-1}{l^\gamma-1}\right)^{l^\gamma-1} \left[\left(1 + \frac{l^\gamma-1}{2l-l^\gamma}\right)^{\frac{2l-l^\gamma}{l^\gamma-1}}\right]^{l^\gamma-1} \\
&\geq \frac{\sqrt{2\pi}}{e^2} \frac{1}{\sqrt{(l^\gamma-1)}} \left(\frac{2l-1}{l^\gamma-1}\right)^{l^\gamma-1} (e - o(1))^{l^\gamma-1}.
\end{aligned}$$

◀

▶ **Claim 41.** $\mathbf{E}[Row_0(T)Col_0(T)(1 - Bad(T))] > \frac{1}{5}\mathbf{E}[Row_0(T)Col_0(T)]$.

Proof of the Claim. Since $Bad(T) \leq Bad_x(T) + Bad_y(T)$, it is enough to prove that $\mathbf{E}[Row_0(T)Col_0(T)Bad_x(T)] \leq \frac{2}{5}\mathbf{E}[Row_0(T)Col_0(T)]$, with a similar statement for $Bad_y(T)$ being proved in the same fashion. For each $t_2 \subseteq \{1, \ldots, n\}$, with $|t_2| = 2l - 1$, we will show that the desired inequality holds when conditioned on $T_2 = t_2$, which implies that the unconditioned inequality holds. All triples T with $T_2 = t_2$ have the same value for $Col_0(T)$, so let this value be called c'. Also let $r = \mathbf{E}[Row(T) \mid T_2 = t_2]$. Now we have:

$$\begin{aligned}
\mathbf{E}[Row_0(T)Col_0(T)Bad_x(T) \mid T_2 = t_2] &\leq c'\mathbf{E}[Row_0(T)Bad_x(T) \mid T_2 = t_2] \\
&\leq c'\mathbf{E}\left[2 \cdot \mathbf{E}[Row(T) \mid T_2 = t_2] \cdot Bad_x(T) \mid T_2 = t_2\right] \\
&\leq 2c'r\mathbf{E}[Bad_x(T) \mid T_2 = t_2]
\end{aligned}$$

$$\begin{aligned}
&\leq \frac{2}{5}c'r \text{ (by Claim 1)}\\
&= \frac{2}{5}c'\mathbf{E}[Row(T) \mid T_2 = t_2]\\
&= \frac{2}{5}c'\mathbf{E}[Row_0(T) \mid T_2 = t_2]\\
&= \frac{2}{5}\mathbf{E}[Row_0(T)Col_0(T) \mid T_2 = t_2]
\end{aligned}$$

The inequality between the second and the third line can be justified as follows: Recall that, as observed in the proof of Claim 1, even when considering only triples $T_2 = t_2$, the value of $Row(T)$ can differ by a factor of at most $2l^{1-\gamma} - 1$. This is due to the fact that $Row(T)$ measures the probability (conditioned on T) of the same set $S = \{x \in C \mid |x| = l^\gamma, x \subseteq \{1, \ldots, n\} \setminus t_2\}$, but depending on whether $i \in x$ (for a particular choice of i and hence of T), an $x \in S$ will have (conditional) probability either $\frac{1}{2}\binom{2l-1}{l^\gamma-1}^{-1}$ or $\frac{1}{2}\binom{2l-1}{l^\gamma}^{-1}$. Thus if T^* is a triple with $T_2^* = t_2$ for which $Row_0(T^*) = \max\{Row_0(T) \mid T_2 = t_2\}$, then $Row(T^*)$ must be the minimum among all values of $Row(T)$ when $T_2 = t_2$, because when $T = T^*$ the largest portion of elements of S have probability $\frac{1}{2}\binom{2l-1}{l^\gamma}^{-1}$ instead of $\frac{1}{2}\binom{2l-1}{l^\gamma-1}^{-1}$. It follows that for all T with $T_2 = t_2$ we have $Row(T) \geq Row(T^*) \geq \frac{1}{2}Row_0(T^*)$, and hence that $\mathbf{E}[2Row(T) \mid T_2 = t_2] \geq Row_0(T^*)$. On the other hand we have that for all T with $T_2 = t_2$, $Row_0(T) \leq Row_0(T^*)$, so finally we get that $Row_0(T) \leq 2\mathbf{E}[Row(T) \mid T_2 = t_2]$ for all T with $T_2 = t_2$. ◀

▶ **Claim 42.** *For any rectangle R: $\mu(B \cap R) = \frac{1}{4}\mathbf{E}[Row_1(T)Col_1(T)]$ and $\mu(A \cap R) = \frac{3}{4}\mathbf{E}[Row_0(T)Col_0(T)]$ (with the expectation taken over all partitions T).*

The proof of this claim is identical to the case where μ is the distribution in Razborov's proof (see [20]), since the relevant observations also apply to our modified distribution: 1. $\mu(B) = \frac{1}{4}$ (and hence $\mu(A) = \frac{3}{4}$), because for every fixed partition T, $i \in x$ with probability $\frac{1}{2}$ and $i \in y$ with probability $\frac{1}{2}$, independently. 2. $i \in x$ and $i \in y$ are independent events (for the same reason). 3. For every (x, y) with $x \cap y = \emptyset$ we have that $Pr[(x, y) \mid (i \notin x) \wedge (i \notin y)] = Pr[(x, y) \mid ((i \notin x) \wedge (i \notin y)) \vee ((i \in x) \wedge (i \notin y)) \vee ((i \notin x) \wedge (i \in y))]$, because conditioning on either one of the two events induces the uniform distribution on the set $\{(x, y) \mid x, y \subset \{1, \ldots, n\}, x \cap y = \emptyset, |x| = |y| = l^\gamma\}$.

We now use claims 2 and 3 to prove the statement of the lemma:

$$\begin{aligned}
&\mu(B \cap R) = \frac{1}{4}\mathbf{E}[Row_1(T)Col_1(T)]\\
&\geq \frac{1}{4}\mathbf{E}[Row_1(T)Col_1(T)(1 - Bad(T))]\\
&\geq \frac{1}{4}\mathbf{E}\left[\left(\frac{Row_0(T)}{6} - 2^{-\epsilon n^\gamma}\right)\left(\frac{Col_0(T)}{6} - 2^{-\epsilon n^\gamma}\right)(1 - Bad(T))\right] \text{ (by def. of } Bad)\\
&= \frac{1}{4}\mathbf{E}\left[\left(\frac{Row_0(T)Col_0(T)}{36} - \frac{2^{-\epsilon n^\gamma}}{6}(Row_0(T) + Col_0(T)) + 2^{-2\epsilon n^\gamma}\right)(1 - Bad(T))\right]\\
&\geq \Omega\left(\mathbf{E}[Row_0(T)Col_0(T)(1 - Bad(T))]\right) - 2^{-\epsilon n^\gamma} \text{ (since } Row_0(T) + Col_0(T) \leq 2)\\
&\geq \Omega\left(\mathbf{E}[Row_0(T)Col_0(T)]\right) - 2^{-\epsilon n^\gamma} \text{ (by Claim 2)}\\
&\geq \Omega(\mu(A \cap R)) - 2^{-\epsilon n^\gamma} \text{ (by Claim 3)}
\end{aligned}$$

Choosing ϵ to be smaller than both the constant in front of $\mu(A \cap R)$ and $\frac{1}{1000 \cdot 4^\gamma}$ completes the proof. ◀

▶ **Lemma 43** (Colouring Lemma.). *Let X and Y be non-empty finite sets, and let $c : X \times Y \mapsto \{0,1\}$ be a colouring of $X \times Y$ such that a proportion $p \in (0,1)$ of the elements of $X \times Y$ are mapped to 1, that is, such that $|c^{-1}(1)|/|X \times Y| = p$. Then for any $r \in (0,p)$ such that $r|Y| \in \mathbb{N}$, we have that for at least $\frac{p-r}{1-r}|X|$ elements $x \in X$, $|(\{x\} \times Y) \cap c^{-1}(1)| > r|Y|$.*

Proof. We call sets of the form $\{x\} \times Y$ *rows*, and let the number $w(x) = \sum_{y \in Y} c(x,y) = |(\{x\} \times Y) \cap c^{-1}(1)|$ be the *weight* of the row $\{x\} \times Y$, for each $x \in X$. Let c be a colouring of $X \times Y$ as above, but such that the smallest possible proportion of rows have weight $> r|Y|$, and denote this proportion by q. Thus q is such that for any colouring c' satisfying the conditions of the lemma, at least $q|X|$ elements $x \in X$ satisfy $|(\{x\} \times Y) \cap c^{-1}(1)| > r|Y|$.

We may assume that all rows with weight $\leq r|Y|$ have weight exactly $r|Y|$: If this is not the case, we may repeatedly perform the operation of changing a 0 into 1 on a row with weight $< r|Y|$, and a 1 into 0 on a row with weight $> r|Y|$, until the above statement is true. (It is easy to see that the colouring c must have rows with weight $> r|Y|$, since otherwise the overall proportion of elements mapped to 1 would be $\leq r < p$.) This operation leaves the proportion of elements that are mapped to 1 unchanged, and the minimality of the chosen colouring c guarantees that the number of rows with weight $> r|Y|$ does not decrease (and therefore remains unchanged).

Next, we may assume that all but at most one of the rows with weight $> r|Y|$ have weight exactly $|Y|$: If this is not the case, we may fix one such row, replace all zeroes with ones on all other rows of weight $> r|Y|$ (thus making their weight exactly $|Y|$), and on the fixed row change the same number of ones into zeroes so as to match the changes made on all other rows. Again the overall proportion of elements being mapped to 1 does not change, and the minimality of the colouring c guarantees that the weight of the fixed row stays $> r|Y|$.

Based on the above we now have: $p|X||Y| = q|X||Y| - \alpha|Y| + (1-q)|X|r|Y|$, where $\alpha \in [0, 1-r)$ is the proportion of zeroes on the one row that has weight $> r|Y|$ but not necessarily $= |Y|$. Thus we have:

$$p \leq q + (1-q)r \iff p \leq (1-r)q + r \iff \frac{p-r}{1-r} \leq q. \qquad ◀$$

Zero-One Laws for Sliding Windows and Universal Sketches

Vladimir Braverman[*1], **Rafail Ostrovsky**[†2], **and Alan Roytman**[‡3]

1 **Department of Computer Science, Johns Hopkins University, USA**
`vova@cs.jhu.edu`

2 **Department of Computer Science and Department of Mathematics University of California, Los Angeles, USA**
`rafail@cs.ucla.edu`

3 **School of Computer Science, Tel-Aviv University, Israel**
`alan.roytman@cs.tau.ac.il`

Abstract

Given a stream of data, a typical approach in streaming algorithms is to design a sophisticated algorithm with small memory that computes a specific statistic over the streaming data. Usually, if one wants to compute a different statistic after the stream is gone, it is impossible. But what if we want to compute a different statistic after the fact? In this paper, we consider the following fascinating possibility: can we collect some small amount of specific data during the stream that is "universal," i.e., where we do not know anything about the statistics we will want to later compute, other than the guarantee that had we known the statistic ahead of time, it would have been possible to do so with small memory? This is indeed what we introduce (and show) in this paper with matching upper and lower bounds: we show that it is possible to collect universal statistics of polylogarithmic size, and prove that these universal statistics allow us after the fact to compute all other statistics that are computable with similar amounts of memory. We show that this is indeed possible, both for the standard unbounded streaming model and the sliding window streaming model.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Streaming Algorithms, Universality, Sliding Windows

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.573

[*] This material is based upon work supported in part by the National Science Foundation under Grant No. 1447639, by the Google Faculty Award and by DARPA grant N660001-1-2-4014. Its contents are solely the responsibility of the authors and do not represent the official view of DARPA or the Department of Defense.

[†] Research supported in part by NSF grants DMS-9206267; CNS-0430254; CNS-0716835; CNS-0716389; CNS-0830803; CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; US-Israel BSF grants BSF-2002354; BSF-2008411; OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

[‡] The work of Alan Roytman was partially supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement number 337122, and in part by the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11).

1 Introduction

With the vast amount of data being generated today, algorithms for data streams continue to play an important role for many practical applications. As the data being generated continues to grow at a staggering rate, streaming algorithms are increasingly becoming more important as a practical tool to analyze and make sense of all the information. Data streams have recently received a lot of attention with good reason, as evidenced by their wide array of applications. In particular, applications for streaming algorithms which operate over input that arrives on the fly and use a small amount of memory are numerous, ranging from monitoring packets flowing across a network to analyzing patterns in DNA sequences. In practice, such applications generate vast amounts of data in a very short period of time, so it is infeasible to store all this information. This presents a pressing question: when is it possible to avoid storing all the information while still providing approximate solutions with good theoretical guarantees?

Typically, algorithms are developed for data streams in the unbounded model, where some statistic is maintained over the entire history of the stream. For certain applications, it is useful to only compute such statistics over recent data. For instance, we may wish to analyze stock market transactions in a particular timeframe or monitor packets transmitted over a network in the last hour to identify suspicious activity. This framework is known as the sliding window model, where we maintain statistics over the current window of size at most N, which slides as time progresses. In the sequence-based model, exactly one element arrives and expires from the window per time step. In the timestamp-based model, any number of elements may arrive or expire. Clearly, the timestamp-based model is more general.

In a landmark paper that influenced the streaming field, the work of Alon, Matias and Szegedy [3] studied the following fundamental framework. For a universe $U = \{1, \ldots, n\}$ and an input stream (i.e., a sequence of integers drawn from U), let $M = (m_1, \ldots, m_n)$ be the vector where m_i denotes the frequency that element $i \in U$ appears in the stream. At any point in time, the paper of [3] showed how to approximate various frequency moments in sublinear space. Informally, for the k^{th} frequency moment $F_k = \sum_{i \in U} m_i^k$, it was shown that F_0, F_1, and F_2 can be approximated in polylogarithmic space, while for $k > 2$, an upper bound of $O^*(n^{1-1/k})$ was shown (the notation $O^*(f(n))$ hides polylogarithmic factors). In addition, a lower bound of $\Omega(n^{1-5/k})$ was shown for every $k \geq 6$. As discussed in [3], such frequency functions are very important in practice and have many applications in databases, as they indicate the degree to which the data is skewed. The fundamental work of Indyk and Woodruff [24] showed how to compute F_k for $k > 2$ in space $O^*(n^{1-2/k})$, which was the first optimal result for such frequency moments. They reduced the problem of computing F_k to computing heavy hitters, and indeed our construction builds on their methods. Recently, Li, Nguyễn, and Woodruff [28] showed that any one-pass streaming algorithm that approximates an arbitrary function in the turnstile model can be implemented via linear sketches. Our work is related, as our algorithms are based on linear sketches of [3].

Such works have opened a line of research that is still extremely relevant today. In particular, what other types of frequency-based functions admit efficient solutions in the streaming setting, and which functions are inherently difficult to approximate? In our paper, we strive to answer this question for frequency-based, monotonically increasing functions in the sliding window model. We make progress on two significant, open problems outlined in [2] by Nelson and [1] by Sohler. Specifically, we are the first to formalize the notion of universality for streaming over sliding windows (since the sliding window model is more general than the standard unbounded model, our universality result is also the first such contribution in the unbounded model). Our main result is the construction of a universal algorithm in the timestamp-based sliding window model for a broad class of functions. That is, we define a class of functions and design a single streaming algorithm that produces a data structure with the following guarantee. When querying the data structure with a function G taken from the class, our algorithm approximates $\sum_{i=1}^{n} G(m_i)$ without knowing G in advance (here, m_i denotes the frequency that element i appears in the window). This is precisely the notion of universality that we develop in our paper, and it is an important step forward towards resolving the problem in [2].

Along the way, we design a zero-one law for a broader class of monotonically increasing functions G which are zero at the origin that specifies when $\sum_{i=1}^{n} G(m_i)$ can be approximated with high probability in one pass, using polylogarithmic memory. If G satisfies the conditions specified by the test, then given the function G we construct an explicit, general algorithm that is able to approximate the summation to within a $(1 \pm \epsilon)$-factor using polylogarithmic memory. If the function G does not pass the test, then we provide a lower bound which proves it is impossible to do so. This result generalizes the work of [9] to the sliding window setting, and makes important progress towards understanding the question posed in [1].

1.1 Contributions and Techniques

Our contributions in this paper make progress on two important problems:

1. We are the first to formally define the notion of universality in the streaming setting. We define a large class of functions $\mathcal{U}$ such that, for the entire class, we design a single, universal algorithm for data streams in the sliding window model which maintains a data structure with the following guarantee. When the data structure is queried with any function $G \in \mathcal{U}$, it outputs a $(1 \pm \epsilon)$-approximation of $\sum_{i=1}^{n} G(m_i)$ without knowing G in advance (note that the choice of G can change). Our algorithm uses polylogarithmic memory, makes one pass over the stream, and succeeds with high probability.
2. We give a complete, algebraic characterization for the class of tractable functions over sliding windows. We define a broader set of functions $\mathcal{T}$ such that, for any non-decreasing function G with $G(0) = 0$, if $G \in \mathcal{T}$, then we have an algorithm that gives a $(1 \pm \epsilon)$-approximation to $\sum_{i=1}^{n} G(m_i)$, uses polylogarithmic space, makes one pass over the stream, and succeeds with high probability. Moreover, if $G \notin \mathcal{T}$, we give a lower bound which shows that super-polylogarithmic memory is necessary to approximate $\sum_{i=1}^{n} G(m_i)$ with high probability. This extends the work of [9] to the sliding window model.

Our algorithms work in the timestamp-based sliding window model and maintain the sum approximately for every window. The value ϵ can depend on n and N, so that the approximation improves as either parameter increases. Our construction is very general, applying to many functions using the same techniques. In stark contrast, streaming algorithms typically depend specifically on the function to be approximated (e.g., F_2 [3, 22] and F_0 [17, 13, 3]). The problems we study have been open for several years, and our construction and proofs are non-trivial. Surprisingly, despite us using existing techniques, their solutions have remained elusive.

For our main result, item 1, it is useful to understand our techniques for solving item 2. When designing the correct zero-one law for tractable functions, a natural place to begin is to understand whether the predicate from [9] suffices for designing an algorithm in the sliding window model. As it turns out, there are some functions which are tractable in the unbounded model but not the sliding window model, and hence the predicate is insufficient. Part of the novelty of our techniques is the identification of an extra smoothing assumption for the class of tractable functions over sliding windows.

If a function does not satisfy our smoothing assumption, we show a super-polylogarithmic lower bound, inspired by the proof of [15]. For our positive result, we observe that the sliding window model adds extra error terms relative to the unbounded model, which our smoothness condition can bound. We also draw on the methods of [9, 10, 24] by finding heavy elements according to the function G, and then reducing the sum problem to the heavy elements problem. Our work sheds light on the question posed in [1], by exhibiting a strict separation result between the unbounded and sliding window models. A function which serves as a witness to this separation (i.e., tractable in the unbounded model as defined in [9] but not in the sliding window model) is a monotonically increasing, piecewise linear function that alternates between being constant and linearly increasing. The function can be seen as a linear approximation to $\log(x)$.

To obtain our main result, we observe that one can remove the assumption from our initial constructions that G is given up front (so that all applications of G happen at the end of the window). However, some technical issues arise, as our construction relies on some parameters of G that stem from our zero-one law. To address these issues, we parameterize our class of functions $\mathcal{U}$ by a constant, allowing us to build a single algorithm to handle the entire parameterized class.

1.2 Related Work

The paper of Braverman and Ostrovsky [9] is the most closely related to our paper. We extend their result from the unbounded model to the timestamp-based sliding window model (by formalizing a new characterization of tractable functions) and by designing a universal algorithm for a large class of functions. Our results build on [9, 10, 24].

Approximating frequency moments and L_p norms has many applications, and there are indeed a vast number of papers on the subject. Compared to such works, we make minimal assumptions and our results are extremely broad, as we design general algorithms that can not only handle frequency moments, but other functions as well. Flajolet and Martin [17] gave an algorithm to approximate F_0 (i.e., counting distinct elements), and Alon, Matias, and Szegedy [3] showed how to approximate F_k for $0 \leq k \leq 2$ using polylogarithmic memory, while for $k > 2$ they showed how to approximate F_k using $O^*(n^{1-1/k})$ memory. They also showed an $\Omega(n^{1-5/k})$ lower bound for $k \geq 6$. Indyk [22] used stable distributions to approximate L_p norms for $p \in (0, 2]$. Indyk and Woodruff [24] gave the first optimal algorithm for F_k ($k > 2$), where an $O^*(n^{1-2/k})$ upper bound was developed. In a followup work, Bhuvanagiri, Ganguly, Kesh, and Saha [6] improved the space by polylogarithmic factors. Bar-Yossef, Jayram, Kumar, and Sivakumar [4] gave an $\Omega(n^{1-(2+\epsilon)/k})$ lower bound, which was improved to $\Omega(n^{1-2/k})$ by Chakrabarti, Khot, and Sun [11] for any one-pass streaming algorithm. The literature is vast, and other results for such functions include [23, 31, 5, 12, 13, 16, 18, 26, 27].

There is also a vast literature in streaming for sliding windows. In their foundational paper, Datar, Gionis, Indyk, and Motwani [15] gave a general technique called exponential histograms that allows many fundamental statistics to be computed in optimal space, including count, sum of positive integers, average, and the L_p norm for $p \in [1, 2]$. Gibbons and Tirthapura [19] made improvements for the sum and count problem with algorithms that are optimal in space and time. Braverman and Ostrovsky [8] gave a general framework for a large class of smooth functions, which include the L_p norm for $p > 0$. Our work complements their results, as the functions they studied need not be frequency based. Many works have studied frequency estimation and frequent item identification, including [20, 25, 14, 32, 21, 30, 7]. Many of our constructions rely on computing frequent elements, but we must do so under a broad class of functions.

1.3 Roadmap

In Section 2, we describe notation used throughout this paper, give some definitions, and formalize the main problems we study. In Section 3, we give a lower bound for functions that are not tractable (i.e., we show the "zero" part of our zero-one law) and we give an algorithm for any tractable function (i.e., we show the "one" part of our zero-one law). Finally, in Section 4, we show the main result of this paper by giving a universal streaming algorithm.

2 Notation and Problem Definition

We have a universe of n elements $[n]$, where $[n] = \{1, \dots, n\}$, and an integer N. A stream $D(n, N)$ is a (possibly infinite) sequence of integers $a_1, a_2, \dots$, each from the universe $[n]$, where N is an upper bound on the size of the sliding window. Specifically, at each time step, there is a current window W that contains *active* elements, where $|W| \leq N$. The window W contains the most recent elements of the stream, and elements which no longer belong in the window are *expired.* We use the timestamp-based model for sliding windows (i.e., any number of elements from the stream may enter or leave the window at each time step). We denote the frequency vector by $M(W)$, where $M(W) = (m_1, \dots, m_n)$ and each m_i is the frequency of element $i \in [n]$ in window W (i.e., $m_i = |\{j \mid a_j = i \wedge j \text{ is active}\}|$). For the window W, the k^{th} frequency moment $F_k(M(W)) = \sum_{i=1}^n m_i^k$. For a vector $V = (v_1, \dots, v_n)$, we let $|V|$ be the L_1 norm of V, namely $|V| = \sum_i |v_i|$. For a vector $V = (v_1, \dots, v_n)$ and a function f, we define the f-Vector as $f(V) = (f(v_1), \dots, f(v_n))$. We say that x is a $(1 \pm \epsilon)$-approximation of y if $(1 - \epsilon)y \leq x \leq (1 + \epsilon)y$. We define $O^*(f(n, N)) = O(\log^{O(1)}(nN) f(n, N))$. We say a probability p is negligible if $p = O^*\left(\frac{1}{nN}\right)$. Consider the following problem:

▶ **Problem 1** (G-Sum). *Let $G : \mathbb{R} \to \mathbb{R}$ be an arbitrary non-decreasing function such that $G(0) = 0$. For any stream $D(n, N)$, any k, and any $\epsilon = \Omega(1/\log^k(nN))$, output a $(1 \pm \epsilon)$-approximation of $\sum_{i=1}^n G(m_i)$ for the current window W.*

We first give some definitions which will be useful throughout the paper and help us define our notion of tractability, beginning with the local jump:

▶ **Definition 2** (Local Jump). $\forall \epsilon > 0, x \in \mathbb{N}$, we define the local jump $\pi_\epsilon(x)$ as

$$\min\{x, \min\{z \in \mathbb{N} \mid G(x + z) > (1 + \epsilon)G(x) \vee G(x - z) < (1 - \epsilon)G(x)\}\}.$$

That is, $\pi_\epsilon(x)$ is essentially the minimum amount needed to cause G to jump by a $(1 \pm \epsilon)$-factor by shifting either to the left or right of x.

▶ **Definition 3** (Heavy Element). For a vector $V = (v_1, \dots, v_n)$, parameter $d > 0$, and function f, we say an element i is (f, d)-*heavy* with respect to the vector V if $f(v_i) > d \sum_{j \neq i} f(v_j)$.

▶ **Definition 4** (Residual Second Moment). If there is an $(F_2, 1)$-heavy element v_i with respect to $V = (v_1, \dots, v_n)$, we define the *residual second moment* as $F_2^{res}(V) = F_2(V) - v_i^2 = \sum_{j \neq i} v_j^2$.

▶ **Definition 5** (Sampled Substream). For a stream $D(n, N)$ and function $H : [n] \to \{0, 1\}$, we denote by D_H the *sampled substream* of D consisting of all elements that are mapped to 1 by the function H. More formally, $D_H = D \cap H^{-1}(1)$.

We analogously define W_H to be the corresponding window for the sampled substream D_H. We are now ready to define our zero-one law.

▶ **Definition 6** (Tractability). We say a function G is tractable if $G(1) > 0$ and:

$$\forall k \ \exists N_0, t \ \forall x, y \in \mathbb{N}^+ \ \forall R \in \mathbb{R}^+ \ \forall \epsilon : \\ \left(R > N_0, \frac{G(x)}{G(y)} = R, \epsilon > \frac{1}{\log^k(Rx)}\right) \Rightarrow \left(\left(\frac{\pi_\epsilon(x)}{y}\right)^2 \geq \frac{R}{\log^t(Rx)}\right) \quad \textit{and} \qquad (1)$$

$$\forall k \ \exists N_1, r \ \forall x \geq N_1 \ \forall \epsilon : \epsilon > \frac{1}{\log^k(x)} \Rightarrow \pi_\epsilon(x) \geq \frac{x}{\log^r(x)}. \qquad (2)$$

We let Tractable be the set of functions that satisfy the above predicate. We now turn to our universal setting and develop an analogous notion of tractability in the context of universality. It is similar to Definition 6, except we need to upper bound some parameters by a constant.

▶ **Definition 7** (Universal Tractability). Fix a constant C. Let $\mathcal{U}(C)$ denote the set of non-decreasing functions G where $G(0) = 0$, $G(1) > 0$, and:

$$\forall k \leq C \ \exists N_0, t \leq 10C \ \forall x, y \in \mathbb{N}^+ \ \forall R \in \mathbb{R}^+ \ \forall \epsilon : \\ \left(R > N_0, \frac{G(x)}{G(y)} = R, \epsilon > \frac{1}{\log^k(Rx)}\right) \Rightarrow \left(\left(\frac{\pi_\epsilon(x)}{y}\right)^2 \geq \frac{R}{\log^t(Rx)}\right) \quad \textit{and} \qquad (3)$$

$$\forall k \leq C \ \exists N_1, r \leq 10C \ \forall x \geq N_1 \ \forall \epsilon : \epsilon > \frac{1}{\log^k(x)} \Rightarrow \pi_\epsilon(x) \geq \frac{x}{\log^r(x)}. \qquad (4)$$

Some examples of functions that are tractable in the universal sense include the moments x^p for $p \leq 2$, for which $\pi_\epsilon(x) = \Omega(\epsilon x)$, along with other functions such as $(x+1)\log(x+1)$.

▶ **Definition 8** (Universal Core Structure). A data structure S is a universal core structure for a fixed vector $V = (v_1, \ldots, v_n)$ with parameters $\epsilon, \delta, \alpha > 0$, and a class of functions $\mathcal{G}$, where $G \in \mathcal{G}$ satisfies $G : \mathbb{R} \to \mathbb{R}$, if given any $G \in \mathcal{G}$, S outputs a set $T = \{(x_1, j_1), \ldots, (x_\ell, j_\ell)\}$ such that with probability at least $1 - \delta$ we have: 1) For each $1 \leq i \leq \ell$, $(1-\epsilon)G(v_{j_i}) \leq x_i \leq (1+\epsilon)G(v_{j_i})$, and 2) If there exists i such that v_i is (G, α)-heavy with respect to V, then $i \in \{j_1, \ldots, j_\ell\}$.

▶ **Definition 9** (Universal Core Algorithm). An algorithm $\mathcal{A}$ is a universal core algorithm with parameters $\epsilon, \delta, \alpha > 0$, and a class of functions $\mathcal{G}$, where $G \in \mathcal{G}$ satisfies $G : \mathbb{R} \to \mathbb{R}$, if, given any stream $D(n, N)$, $\mathcal{A}$ outputs a universal core structure for the vector $M(W)$ with parameters ϵ, δ, α, and $\mathcal{G}$.

▶ **Definition 10** (Universal Sum Structure). A data structure S is a universal sum structure for a fixed vector $V = (v_1, \ldots, v_n)$ with parameters $\epsilon, \delta > 0$, and a class of functions $\mathcal{G}$, where $G \in \mathcal{G}$ satisfies $G : \mathbb{R} \to \mathbb{R}$, if given any $G \in \mathcal{G}$, S outputs a value x such that with probability at least $1 - \delta$ we have: $(1-\epsilon)\sum_{i=1}^n G(v_i) \leq x \leq (1+\epsilon)\sum_{i=1}^n G(v_i)$.

▶ **Definition 11** (Universal Sum Algorithm). An algorithm $\mathcal{A}$ is a universal sum algorithm with parameters $\epsilon, \delta > 0$, and a class of functions $\mathcal{G}$, where $G \in \mathcal{G}$ satisfies $G : \mathbb{R} \to \mathbb{R}$, if, given any stream $D(n, N)$, $\mathcal{A}$ outputs a universal sum structure for the vector $M(W)$ with parameters ϵ, δ, and $\mathcal{G}$.

In this paper, our main result is the proof of the following theorem:

▶ **Theorem 12.** *Fix a constant C and let $\mathcal{U}(C)$ be the universally tractable set from Definition 7. There is a universal sum algorithm that has parameters $\epsilon = \Omega(1/\log^k(nN))$ (for $0 \leq k \leq C$), $\delta = 0.3$, and $\mathcal{G} = \mathcal{U}(C)$, uses polylogarithmic space in n and N, and makes a single pass over the input stream $D(n, N)$.*

We can reduce the constant failure probability to inverse polynomial via standard methods. To formalize our other main result, we define the following class:

▶ **Definition 13** (STREAM-POLYLOG). We say function $G \in$ STREAM-POLYLOG if $\forall k = O(1)$, $\exists t = O(1)$ and an algorithm $\mathcal{A}$ such that for any universe size n, window size N, $\epsilon \geq 1/\log^k(nN)$, and stream $D(n, N)$: 1) $\mathcal{A}$ makes one pass over D, 2) $\mathcal{A}$ uses $O(\log^t(nN))$ space, and 3) For any window W, $\mathcal{A}$ maintains a $(1 \pm \epsilon)$-approximation of $|G(M(W))|$ except with probability at most 0.3.

Note that the constant error probability can be made to be as small as an inverse polynomial by standard techniques. Our other main result is the following:

▶ **Theorem 14.** *Let G be a non-decreasing function such that $G(0) = 0$. Then we have $G \in$ STREAM-POLYLOG $\iff G \in$ Tractable.*

3 A Characterization for Tractable Functions

In this section, we prove Theorem 14 by first giving a lower bound for non-tractable functions. We first show a deterministic lower bound for any algorithm that approximates G-Sum. Our technique is inspired by the lower bound proof in [15] for estimating the number of 1's for sliding windows.

▶ **Theorem 15.** *Let G be a function such that $G \notin$ Tractable. Then, any deterministic algorithm that solves the G-Sum problem with relative error $\epsilon' = 1/\log^b(nN)$ (for some constant b) must use space at least $\Omega(\log^a(nN))$, where a is arbitrarily large.*

Proof. We construct a set of input streams such that, for any pair of data streams in the set, the algorithm must distinguish between these two inputs at some point as the window slides. Therefore, the space of the algorithm must be at least logarithmic in the size of this set.

Since $G \notin$ Tractable, in Definition 6, either Predicate (1) or Predicate (2) does not hold. If Predicate (1) is not true, then the lower bound from [9] applies and the theorem follows. Hence, we assume that Predicate (2) does not hold, which implies the following: $\exists k, \forall r, N_1, \exists x \geq N_1, \epsilon : \epsilon > \frac{1}{\log^k(x)} \wedge \pi_\epsilon(x) < \frac{x}{\log^r(x)}$. Let k be given, and let r be arbitrarily large. This negation implies that there are infinitely many increasing points $x_1, x_2, x_3, \ldots$ and corresponding values $\epsilon_1, \epsilon_2, \epsilon_3, \ldots$, where $\epsilon_i > \frac{1}{\log^k(x_i)}$ and $\pi_{\epsilon_i}(x_i) < \frac{x_i}{\log^r(x_i)}$.

Surprisingly, we construct our lower bound with a universe of size $n = 1$, namely $U = \{1\}$. For each x_i, we construct a set of streams with a fixed, upper bounded window size of $N = x_i$, and argue that the algorithm must use memory at least $\log^r(x_i)$ (note that, as the x_i are monotonically increasing, our lower bound applies for asymptotically large N). We assume without loss of generality that $G(x_i - \pi_{\epsilon_i}(x_i)) < (1 - \epsilon_i)G(x_i)$. Our constructed streams are defined as follows. For each $N = x_i$, note that our window consists of elements which have arrived in the past x_i time steps. For the first x_i time steps, we construct many streams by choosing $\lfloor \frac{x_i}{\pi_{\epsilon_i}(x_i)} \rfloor$ of these time steps (each choice defining a different stream). For each chosen time step, we insert $\pi_{\epsilon_i}(x_i)$ 1's into the stream, and for each time step that is not chosen, we insert zero elements. For technical reasons, we pad the last time step x_i in the first window with $x_i - \pi_{\epsilon_i}(x_i) \lfloor \frac{x_i}{\pi_{\epsilon_i}(x_i)} \rfloor$ 1's. Note that the number of elements in the first window at time x_i is $\pi_{\epsilon_i}(x_i)\lfloor\frac{x_i}{\pi_{\epsilon_i}(x_i)}\rfloor + (x_i - \pi_{\epsilon_i}(x_i)\lfloor\frac{x_i}{\pi_{\epsilon_i}(x_i)}\rfloor) = x_i$. We insert nothing at time step $x_i + 1$. For the remaining time steps $x_i + 2, \ldots, 2x_i - 1$, we simply repeat the first $x_i - 2$ time steps of the stream (i.e., if time step t was chosen in the first x_i time steps, $1 \le t \le x_i - 2$, then we insert $\pi_{\epsilon_i}(x_i)$ 1's at time step $x_i + t + 1$).

Now, we argue that for any such pair of constructed streams A, B which are different, any algorithm with relative error smaller than $\epsilon' = 1/\log^k(nN)$ must distinguish between these two inputs. To see this, consider the earliest time d when the two streams differ (note that $1 \le d \le x_i - 1$). Let W_A be the window for stream A (similarly, we define W_B as the window for stream B). Let c be the number of chosen time steps in the first d time steps of stream A. Without loss of generality, we assume that time step d was chosen in stream A but not in stream B. Hence, the number of chosen time steps in stream B up to time d is $c - 1$. Consider the windows at time step $x_i + d$. The number of elements in W_A at this time is given by $\pi_{\epsilon_i}(x_i)[\lfloor\frac{x_i}{\pi_{\epsilon_i}(x_i)}\rfloor - c + (c-1)] + x_i - \pi_{\epsilon_i}(x_i)\lfloor\frac{x_i}{\pi_{\epsilon_i}(x_i)}\rfloor = x_i - \pi_{\epsilon_i}(x_i)$. Moreover, the number of elements in W_B is given by $\pi_{\epsilon_i}(x_i)[\lfloor\frac{x_i}{\pi_{\epsilon_i}(x_i)}\rfloor - (c-1) + (c-1)] + x_i - \pi_{\epsilon_i}(x_i)\lfloor\frac{x_i}{\pi_{\epsilon_i}(x_i)}\rfloor = x_i$. Hence, the G-Sum value at time $x_i + d$ for W_A is $G(x - \pi_{\epsilon_i}(x_i)) < (1 - \epsilon_i)G(x_i)$. As long as the algorithm has relative error $\epsilon' = 1/\log^k(nN) < \epsilon_i$, streams A and B must be distinguished at some point in time as the window slides.

Thus, the algorithm's memory is lower bounded by the logarithm of the number of constructed streams, of which there are $\binom{x_i}{\lfloor\frac{x_i}{\pi_{\epsilon_i}(x_i)}\rfloor}$ for each x_i. We have $\log\left(\binom{x_i}{\lfloor\frac{x_i}{\pi_{\epsilon_i}(x_i)}\rfloor}\right) \ge \lfloor\frac{x_i}{\pi_{\epsilon_i}(x_i)}\rfloor \log(\pi_{\epsilon_i}(x_i)) \ge \frac{\log^r(x_i)}{2}\log(\pi_{\epsilon_i}(x_i))$. If $\pi_{\epsilon_i}(x_i) = 1$, we repeat the proof inserting two 1's at each time step and the proof goes through. Observing that r can be made arbitrarily large gives the proof. ◀

We now have a randomized lower bound by appealing to Yao's minimax principle [29] and building on top of our deterministic lower bound, similarly to [15] (applying the principle with the uniform distribution suffices).

▶ **Theorem 16.** *Let G be a function where $G \notin$ Tractable. Then, any randomized algorithm that solves G-Sum with relative error smaller than $\epsilon' = 1/\log^b(nN)$ for some constant b and succeeds with at least constant probability $1 - \delta$ must use memory $\Omega(\log^a(nN))$, where a is arbitrarily large.*

We now complete the proof of Theorem 14 by first approximating heavy elements (note that we reduce the G-Sum problem to the following problem):

▶ **Problem 17** (**G-Core**)**.** *We have a stream $D(n, N)$ and parameters $\epsilon, \delta > 0$. For each window W, with probability at least $1-\delta$, maintain a set $S = \{g'_1, \ldots, g'_\ell\}$ such that $\ell = O^*(1)$ and there exists a set of indices $\{j_1, \ldots, j_\ell\}$ where $(1-\epsilon)G(m_{j_i}) \le g'_i \le (1+\epsilon)G(m_{j_i})$ for each $1 \le i \le \ell$. If there is a $(G, 1)$-heavy element m_i with respect to $M(W)$, then $i \in \{j_1, \ldots, j_\ell\}$.*

We begin solving the above problem via the following lemma (taken from [9]).

▶ **Lemma 18.** *Let $V = (v_1, \ldots, v_n)$ be a vector with non-negative entries of dimension n and H be a pairwise independent random vector of dimension n with entries $h_i \in \{0, 1\}$ such that $P(h_i = 1) = P(h_i = 0) = \frac{1}{2}$. Denote by H' the vector with entries $1 - h_i$. Let $K > 10^4$ be a constant, and let $X = \langle V, H\rangle$ and $Y = \langle V, H'\rangle$. If there is an (F_1, K)-heavy element v_i with respect to V, then: $P((X > KY) \vee (Y > KX)) = 1$. If there is no $(F_1, \frac{K}{10^4})$-heavy element with respect to V, then: $P((X > KY) \vee (Y > KX)) \le \frac{1}{2}$.*

We now give some lemmas related to how approximating values can affect the function G.

▶ **Lemma 19.** *Let $0 < \epsilon \le \frac{1}{2}$, and let $x, u, v, y \ge 0$ satisfy $|x - u| \le 0.1\pi_\epsilon(x)$ and $v, y < 0.1\pi_\epsilon(x)$, where $\pi_\epsilon(x) > 1$. Then $(1 - 4\epsilon)G(u + v + y) \le G(u) \le (1 + 4\epsilon)G(u - v - y)$.*

Proof. First, we note that $u + v + y \le x + 0.1\pi_\epsilon(x) + v + y \le x + 0.3\pi_\epsilon(x) \le x + \pi_\epsilon(x) - 1$ (recalling $\pi_\epsilon(x) > 1$). We can similarly get that $u - v - y \ge x - (\pi_\epsilon(x) - 1)$. Hence, we get that $(1 - \epsilon)G(x) \le G(x - (\pi_\epsilon(x) - 1)) \le G(u - v - y) \le G(u) \le G(u + v + y) \le G(x + (\pi_\epsilon(x) - 1)) \le (1 + \epsilon)G(x)$.

We conclude by noting: $(1+4\epsilon)G(u-v-y) \ge (1+4\epsilon)(1-\epsilon)G(x) \ge \frac{(1+4\epsilon)(1-\epsilon)}{1+\epsilon}G(u) \ge G(u)$. Similarly, we get $(1 - 4\epsilon)G(u + v + y) \le (1 - 4\epsilon)(1 + \epsilon)G(x) \le \frac{(1-4\epsilon)(1+\epsilon)}{1-\epsilon}G(u) \le G(u)$. ◀

▶ **Lemma 20.** *Let $x, u, v, y \ge 0$ be such that $|x - u| \le v + y$, and let $0 < \epsilon < 1$. If $(1 - \epsilon)G(u + v + y) \le G(u) \le (1 + \epsilon)G(u - v - y)$, then $(1 - \epsilon)G(x) \le G(u) \le (1 + \epsilon)G(x)$.*

Proof. We have $(1 - \epsilon)G(x) \le (1 - \epsilon)G(u + v + y) \le G(u) \le (1 + \epsilon)G(u - v - y) \le (1 + \epsilon)G(x)$. ◀

We now give a useful subroutine over sliding windows which we use in our main algorithm for approximating heavy elements and prove its correctness (there is a similar algorithm and proof in [9], though it must be adapted to the sliding window setting).

```
1 for $i = 1$ to $O(\log(nN))$ do
2     for $j = 1$ to $C = O(1)$ do
3         Generate a random hash function $H : [n] \to \{0, 1\}$ with pairwise independent
          entries.
4         Let $H' = 1 - H$ (i.e., $h'_k = 1 - h_k$, where $h_k$ is the $k^{th}$ entry of $H$).
5         Let $f_H$ be a $(1 \pm .1)$-approximation of $F_2$ on $D_H$ (with negligible error
          probability), via the smooth histogram method for sliding windows [8].
6         Let $f_{H'}$ be a $(1 \pm .1)$-approximation of $F_2$ on $D_{H'}$ (with negligible error
          probability), via the smooth histogram method for sliding windows [8].
7         Let $X_{ij} = 10 \min\{f_H, f_{H'}\}$.
8     Let $Y_i = \frac{X_{i1}+\cdots+X_{iC}}{C}$ (i.e., $Y_i$ is the average of $C$ independent $X_{ij}$'s).
9 Output $r = \sqrt{median_i\{Y_i\}}$ for the current window $W$.
```

Algorithm 1: Residual-Approximation(D)

▶ **Lemma 21.** *Let $D(n, N)$ be any input stream. Algorithm Residual-Approximation makes a single pass over D and uses polylogarithmic space in n and N. Moreover, if the current window W contains an $(F_2, 2)$-heavy element m_i with respect to $M(W)$, then the algorithm maintains and outputs a value r such that $2\sqrt{F_2^{res}(M(W))} < r < 3\sqrt{F_2^{res}(M(W))}$ (except with negligible probability).*

Proof. Assume the current window W has an $(F_2, 2)$-heavy element m_k with respect to the vector $M(W)$. Due to the properties of smooth histograms from [8], we know that $.9F_2(M(W_H)) \le f_H \le 1.1F_2(M(W_H))$, where $M(W_H)$ is the multiplicity vector of the current window in substream D_H (similarly for $f_{H'}$). Hence, the random variable $X_{ij} = 10 \min\{f_H, f_{H'}\}$ is a $(1 \pm .1)$-approximation of the random variable $Z = 10 \sum_\ell \mathbf{1}_{H(\ell) \ne H(k)} m_\ell^2$ (here, $\mathbf{1}_{H(\ell) \ne H(k)}$ is the indicator random variable which is 1 if $H(\ell) \ne H(k)$ and 0 otherwise). To see why, suppose that element k is mapped to 1 by H, so that k belongs to the sampled substream D_H. Then observe that

$$f_H \ge .9F_2(M(W_H)) \ge .9m_k^2 > 1.8 \sum_{\ell \ne k} m_\ell^2 \ge 1.1 \sum_\ell \mathbf{1}_{H(\ell) \ne H(k)} m_\ell^2 \ge f_{H'}.$$

Thus, the minimum of f_H and $f_{H'}$ is indeed a $(1 \pm .1)$-approximation to $\sum_\ell \mathbf{1}_{H(\ell)\neq H(k)} m_\ell^2$, since this is the second moment of the vector $M(W_{H'})$ (the case is symmetric if element k is mapped to 0 by H).

Now, since H is pairwise independent, we have that $E(Z) = 5F_2^{res}(M(W))$. In particular, since we always have $0 \leq Z \leq 10F_2^{res}(M(W))$, we can bound the variance by $Var(Z) \leq E(Z^2) \leq 100(F_2^{res}(M(W)))^2$. If we denote by A the random variable which is the average of C independent Z's, then we have $Var(A) = \frac{1}{C}Var(Z) \leq \frac{100}{C}(F_2^{res}(M(W)))^2$. Hence, if we choose C to be sufficiently large, then by Chebyshev's inequality we have:

$$P(|A - 5F_2^{res}(M(W))| \geq 0.1F_2^{res}(M(W))) \leq \frac{100Var(A)}{(F_2^{res}(M(W)))^2} \leq \frac{10^4}{C} \leq 0.1$$

(for instance, $C = 10^5$ is sufficient).

Now, if we take the median T of $O(\log(nN))$ independent A's, then by Chernoff bound this would make the probability negligible. That is, we have $4.9F_2^{res}(M(W)) \leq T \leq 5.1F_2^{res}(M(W))$ except with negligible probability. We can repeat these arguments and consider the median of $O(\log(nN))$ averages (i.e., the Y_i's) of $O(1)$ independent X_{ij}'s. Since there are only $O(\log(nN))$ X_{ij}'s total (with each one being a $(1 \pm .1)$-approximation to its corresponding random variable Z, except with negligible probability), then by the union bound all the X_{ij}'s are $(1 \pm .1)$-approximations except with negligible probability (since the sum of polylogarithmically many negligible probabilities is still negligible). Therefore, the median of averages would give a $(1 \pm .1)$-approximation to T. Taking the square root guarantees that $2\sqrt{F_2^{res}(M(W))} < r < 3\sqrt{F_2^{res}(M(W))}$ (except with negligible probability).

Note that the subroutine for computing an approximation to F_2 on sliding windows using smooth histograms can be done in one pass and in polylogarithmic space (even if we demand a $(1 \pm .1)$-approximation and a negligible probability of failure). ◀

Now, we claim that Algorithm Compute-Hybrid-Major solves the following:

▶ **Problem 22** (Hybrid-Major(D, ϵ)). *Given a stream D and $\epsilon > 0$, maintain a value $r \geq 0$ for each window W such that: 1) If $r \neq 0$, then r is a $(1 \pm 4\epsilon)$-approximation of $G(m_j)$ for some m_j, and 2) If the current window W has an element m_i such that $\pi_\epsilon(m_i) \geq 20^5\sqrt{F_2^{res}(M(W))}$, then r is a $(1 \pm 4\epsilon)$-approximation of $G(m_i)$.*

```
1 Let $a$ be a $(1 \pm \epsilon')$-approximation of $L_2$ for window $W$ using the smooth histogram
  method [8] (with negligible probability of error), for $\epsilon' = \frac{1}{\log^{\Omega(1)}(N)}$.
2 Repeat $O(\log(nN))$ times, independently and in parallel:
3     Generate a uniform pairwise independent vector $H \in \{0,1\}^n$.
4     Maintain and denote by $X'$ a $(1 \pm .2)$-approximation of the second moment for the
      window $W_H$ using a smooth histogram [8] (with negligible probability of error).
5     Similarly define $Y'$ for the window $W_{\mathbf{1}-H}$.
6     If $X' < (20)^4 Y'$ and $Y' < (20)^4 X'$, output 0 and terminate the algorithm.
7 In parallel, apply Residual-Approximation($D$) to maintain the residual second moment
  approximation, let $b$ denote the output of the algorithm.
8 If $(1 - 4\epsilon)G(a + b + 2\epsilon' a) > G(a)$ or $G(a) > (1 + 4\epsilon)G(a - b - 2\epsilon' a)$, output 0.
9 Otherwise, output $G(a)$.
```
Algorithm 2: Compute-Hybrid-Major(D, ϵ)

Before delving into the proof, we show the following lemma.

▶ **Lemma 23.** *Suppose the current window W has an $(F_2, 1)$-heavy element m_i. Moreover, let a be a $(1 \pm \epsilon')$-approximation of the L_2 norm of the current window W, where $\epsilon' < 1$. Then $-\epsilon' m_i \le a - m_i \le (1+\epsilon')\sqrt{F_2^{res}(M(W))} + \epsilon' m_i \le 2\sqrt{F_2^{res}(M(W))} + \epsilon' m_i$.*

Proof. Since a is a $(1 \pm \epsilon')$-approximation of the L_2 norm of the vector $M(W)$, we know $(1-\epsilon')\sqrt{\sum_{k=1}^n m_k^2} \le a \le (1+\epsilon')\sqrt{\sum_{k=1}^n m_k^2}$. Hence, we have that

$$a - m_i \le (1+\epsilon')\sqrt{\sum_{k=1}^n m_k^2} - m_i \le (1+\epsilon')m_i + (1+\epsilon')\sqrt{\sum_{j \ne i} m_j^2} - m_i$$
$$\le \epsilon' m_i + (1+\epsilon')\sqrt{F_2^{res}(M(W))}.$$

We conclude by noting that $m_i - a \le m_i - (1-\epsilon')\sqrt{\sum_{k=1}^n m_k^2} \le m_i - (1-\epsilon')m_i$, which gives the lemma. ◀

▶ **Lemma 24.** *For any function $G \in$ Tractable, Algorithm Compute-Hybrid-Major solves the Hybrid-Major problem with negligible probability of error.*

Proof. First, we show that if there is no $(F_2, 2)$-heavy entry in the current window W, then the output is 0 except with negligible probability. Consider a single iteration of the main loop of the algorithm. Let M' be the vector with entries m_i^2 and denote $X = \langle M', H \rangle, Y = |M'| - \langle M', H \rangle$. Since we have an F_2 approximation over sliding windows, except with negligible probability, X' and Y' are $(1 \pm .2)$-approximations of X and Y, respectively. Hence, $\frac{4}{5}X \le X' \le \frac{5}{4}X$ and $\frac{4}{5}Y \le Y' \le \frac{5}{4}Y$. By Lemma 18, except with probability at most $0.5 + o(1)$: $X' \le \frac{5}{4}X \le \frac{5}{2}(10)^4 Y < (20)^4 Y'$ and $Y' < (20)^4 X'$. Thus, the algorithm outputs 0 except with negligible probability.

Assume that there is an $(F_2, 2)$-heavy entry m_i. Then, applying Lemma 23 with some $0 < \epsilon' < 1$ to be set later, we know $|m_i - a| \le 2\sqrt{F_2(M(W))} + \epsilon' m_i$ and $a \ge (1-\epsilon')m_i$ (except with negligible probability). By Lemma 21, it follows that $2\sqrt{F_2^{res}(M(W))} < b < 3\sqrt{F_2^{res}(M(W))}$ except with negligible probability. Hence, we have $|m_i - a| \le b + \epsilon' m_i \le b + 2\epsilon' a$, since $2\epsilon' a \ge 2\epsilon'(1-\epsilon')m_i \ge \epsilon' m_i$ (assuming $\epsilon' \le \frac{1}{2}$). Now, observe that if the algorithm outputs $G(a)$, then it must be that $(1-4\epsilon)G(a + b + 2\epsilon' a) \le G(a) \le (1+4\epsilon)G(a - b - 2\epsilon' a)$. Thus, by applying Lemma 20 with parameters $x = m_i, u = a, v = b$, and $y = 2\epsilon' a$, it follows that if the algorithm outputs $G(a)$, then $G(a)$ is a $(1 \pm 4\epsilon)$-approximation of $G(m_i)$. Thus, the first condition of Hybrid-Major follows.

Finally, assume $\pi_\epsilon(m_i) \ge (20)^5\sqrt{F_2^{res}(M(W))}$. By definition, $m_i \ge \pi_\epsilon(m_i)$ and so m_i is $(F_2, 20^{10})$-heavy with respect to $M(W)$. By Lemma 18, we have (except with negligible probability): $X' > 20^4 Y'$ or $Y' > 20^4 X'$. Hence, except with negligible probability, the algorithm does not terminate before the last line. Let N_1 be the constant given by the definition of tractability in Definition 6 (N_1 may depend on the parameter k from Definition 6, but we apply the definition for $k = O(1)$ determined by ϵ). We assume $m_i \ge N_1$ (otherwise the number of elements in the window is constant). Also, let r be given by Definition 6. By applying Lemma 23 with $\epsilon' = \frac{1}{\log^{r+1}(N)}$, we have $|m_i - a| \le 2\sqrt{F_2^{res}(M(W))} + \epsilon' m_i \le 0.01\pi_\epsilon(m_i) + \frac{1}{\log N}\frac{m_i}{\log^r(m_i)} \le .01\pi_\epsilon(m_i) + \frac{\pi_\epsilon(m_i)}{\log N} \le .02\pi_\epsilon(m_i)$ for sufficiently large N (since G is tractable) and $b \le 3F_2^{res}(M(W)) < 0.01\pi_\epsilon(m_i)$. Since $b \le .1\pi_\epsilon(m_i)$ and $2\epsilon' a \le 2\epsilon'(m_i + b + \epsilon' m_i) \le 2 \cdot (.03\pi_\epsilon(m_i)) \le .1\pi_\epsilon(m_i)$, then by Lemmas 19 and 20 (which we apply with the same parameters, $x = m_i, u = a, v = b$, and $y = 2\epsilon' a$), the algorithm outputs $G(a)$ which is a $(1 \pm 4\epsilon)$-approximation of $G(m_i)$. Thus, the second condition of Hybrid-Major follows, which gives the lemma. ◀

Consider the following lemma, which is from [9] (the proof of Lemma 25 uses Predicate (1) from Definition 6).

▶ **Lemma 25.** *Let G be a non-decreasing tractable function. Then for any $k = O(1)$, there exists $t = O(1)$ such that for any n, N and for any $\epsilon > \log^{-k}(nN)$ the following holds. Let $D(n, N)$ be a stream and W be the current window. If there is a $(G, 1)$-heavy element m_i with respect to $M(W)$, then there is a set $S \subseteq [n]$ such that $|S| = O(\log(N))$ and: $\pi^2_\epsilon(m_i) = \Omega\left(\log^{-t}(nN) \sum_{j \notin S \cup \{i\}} m_j^2\right)$.*

We now give the algorithm Compute-G-Core, which solves G-Core (i.e., Problem 17), and prove its correctness. A similar algorithm appears in [9], we repeat it here for completeness, and to help design and understand our main result on universality.

```
1 Generate a pairwise independent hash function $H : [n] \mapsto \tau$, where $\tau = O^*\left(\frac{1}{p}\right)$.
2 $\forall k \in [\tau]$, compute in parallel $c_k =$ Compute-Hybrid-Major$(D_{H_k}, \frac{\epsilon}{4})$, where
  $H_k(i) = \mathbf{1}_{H(i)=k}$.
3 Output $S = \{c_k : c_k > 0\}$.
```

Algorithm 3: Compute-G-Core(D, ϵ, p)

▶ **Theorem 26.** *Algorithm Compute-G-Core solves the G-Core problem, except with probability asymptotically equal to p. The algorithm uses $O^*(1)$ memory bits if $p = \Omega(1/\log^u(nN))$ and $\epsilon = \Omega(1/\log^k(nN))$ for some $u, k \geq 0$.*

Proof. Let W denote the current window. First, except with negligible probability, every positive c_i is a $(1 \pm 4 \cdot \frac{\epsilon}{4})$-approximation of some distinct entry $G(m_j)$, which implies that c_i is a $(1 \pm \epsilon)$-approximation of $G(m_j)$. Second, assume that there exists a $(G, 1)$-heavy entry m_i with respect to $M(W)$. Denote $X = \sum_{j \neq i} m_j^2 \mathbf{1}_{H(j)=H(i)}$. By pairwise independence of H, we have $E(X) = \frac{1}{\tau}(F_2(M) - m_i^2)$. By Lemma 25, there exists a set S and $t \geq 0$ such that $|S| = O(\log N)$ and:

$$\pi^2_\epsilon(m_i) = \Omega\left(\frac{\sum_{j \notin S \cup \{i\}} m_j^2}{\log^t(nN)}\right). \tag{5}$$

Let $\mathcal{L}$ be the event that $\pi^2_\epsilon(m_i) > 20^{10} X$, and let $\mathcal{B}$ be the event that $\forall j \in S : H(j) \neq H(i)$. By Markov's inequality, by pairwise independence of H, and by Equation (5), there exists $\tau = O^*\left(\frac{1}{p}\right)$ such that:

$$\begin{aligned} P(\neg\mathcal{L}) &= P(\neg\mathcal{L} \mid \mathcal{B}) \cdot P(\mathcal{B}) + P(\neg\mathcal{L} \mid \neg\mathcal{B}) \cdot P(\neg\mathcal{B}) \\ &\leq \frac{E(X \mid \mathcal{B}) 20^{10}}{\pi^2_\epsilon(m_i)} \cdot 1 + 1 \cdot \frac{O(\log N)}{\tau} \leq O^*\left(\frac{1}{\tau}\right) = p. \end{aligned}$$

If $\mathcal{L}$ occurs, which happens with probability at least $1-p$, then $c_{H(i)}$ is a $(1 \pm \epsilon)$-approximation of $G(m_i)$ except with negligible probability (by Lemma 24). Thus, the final probability of error is approximately equal to p.

It is not too hard to see that Algorithm Compute-G-Core uses polylogarithmic memory. The subroutine depth is constant, and there are only polylogarithmically many subroutine calls at each level. At the lowest level, we only do direct computations on the stream that require polylogarithmic space or a smooth histogram computation for F_2 or L_2, which also requires polylogarithmic space. We get that for any constant k, there exists a constant t such that we can solve G-Core (except with probability p) using $O(\log^t(nN))$ space, where $\epsilon \geq \log^{-k}(nN)$. ◀

In Appendix A, we show how to reduce the G-Sum problem to the G-Core problem. In particular, we prove the following theorem. The algorithm and proof of correctness follow from [10]. We restate the algorithm and results using our notation for completeness.

▶ **Theorem 27.** *If there is an algorithm that solves G-Core using memory $O^*(1)$ and makes one pass over D except with probability $O(\log^{-u}(nN))$ for some $u > 0$, then there is an algorithm that solves G-Sum using memory $O^*(1)$ and makes one pass over D except with probability at most* 0.3.

We can reduce the failure probability to inverse polynomial using standard methods. Combining this with Theorem 26 and Theorem 16, we have Theorem 14.

4 Universality

In this section, we show the main result of this paper, Theorem 12, by designing a universal sum algorithm. We first construct a universal core algorithm, which we call UCA. That is, given a data stream, the algorithm produces a universal core structure with respect to the frequency vector $(m_1, \ldots, m_n)$ defined by the current window W without knowing the function G to be approximated in advance. Let C be a constant and let $\mathcal{U}(C)$ be the set according to Definition 7. The structure guarantees that, when queried with any function G from $\mathcal{U}(C)$ (after processing the stream), it outputs the set T according to Definition 8.

Universal Core Algorithm (UCA): The algorithm constructs a universal core structure S and our techniques build on the results from Section 3. Algorithm Residual-Approximation from Section 3 does not depend on the function G, and hence it clearly carries over to our universal setting.

Algorithm Compute-Hybrid-Major depends on G, so we modify it accordingly. We do not rewrite the whole algorithm, as there are only a few modifications. In Step 1, we set $\epsilon' = \frac{1}{\log^{10C+1}(N)}$. We get rid of Steps 8 and 9, and instead create a new Step 8 where we find the index j of an $(F_2, 2)$-heavy element m_j, if it exists (finding such an index can be done using standard methods, the details of which we omit for brevity). We also create a new Step 9 where we output the triple (a, b, j) (assuming none of the parallel copies from Step 2 outputs 0).

We also modify Algorithm Compute-G-Core. In particular, the value of τ in Step 1 should depend on C, and we set it to be $\frac{\log^{10C+2}(nN)}{p}$. Moreover, we remove Step 3 from the algorithm and store c_k for each $k \in [\tau]$ as part of our data structure S (recall that c_k is either 0 or a triple (a_k, b_k, j_k), where a_k, b_k are the values computed in the k^{th} parallel instance of the subroutine Compute-Hybrid-Major and j_k is the index of the corresponding $(F_2, 2)$-heavy element).

Querying the Structure: Given a function $G \in \mathcal{U}(C)$ as a query to our universal core structure, we explain how to produce the set T according to Definition 8. For each stored c_k in the data structure S ($k \in [\tau]$), if $c_k = 0$, then we do not include it in our output set T. Otherwise, if c_k is a triple (a_k, b_k, j_k), then we include the pair $(G(a_k), j_k)$ in our set T as long as $(1 - 4\epsilon)G(a_k + b_k + 2\epsilon' a_k) \leq G(a_k) \leq (1 + 4\epsilon)G(a_k - b_k - 2\epsilon' a_k)$ (recall $\epsilon' = \frac{1}{\log^{10C+1}(N)}$).

▶ **Theorem 28.** *Fix a constant C and let $\mathcal{U}(C)$ be the set of tractable functions corresponding to the definition of universal tractability. Then UCA is a universal core algorithm with parameters $\epsilon = \Omega(1/\log^k(nN))$ (for $0 \le k \le C$), $\delta = \Omega(1/\log^u(nN))$ (for $u \ge 0$), $\alpha = 1$, and $\mathcal{G} = \mathcal{U}(C)$.*

Proof. The correctness of UCA essentially follows from the proofs of the results in Section 3. In particular, Lemma 21 still holds since Algorithm Residual-Approximation is unchanged.

Lemma 24 still mostly holds without much modification. Using the same notation as in the original proof, if there is no $(F_2, 2)$-heavy element, then the proof of Lemma 24 can still be applied and the modified version of Compute-Hybrid-Major outputs 0 (except with negligible probability). In such a case, the universal core structure stores the value 0. If there is an $(F_2, 2)$-heavy element m_{i_k} and the structure stores (a_k, b_k, i_k), then again the proof applies. The reason is that, when querying the universal core structure with a function G, we check if $(1-4\epsilon)G(a_k + b_k + 2\epsilon' a_k) \le G(a_k) \le (1+4\epsilon)G(a_k - b_k - 2\epsilon' a_k)$, in which case the proof argues that $G(a_k)$ is a $(1 \pm 4\epsilon)$-approximation of $G(m_{i_k})$. In the case that $\pi_\epsilon(m_{i_k}) \ge (20)^5 \sqrt{F_2^{res}(M(W))}$, the proof still goes through since we apply Lemma 23 with $\epsilon' = \frac{1}{\log^{10C+1}(N)}$, and we have $|m_{i_k} - a_k| \le 2\sqrt{F_2^{res}(M(W))} + \epsilon' m_{i_k} \le 0.01\pi_\epsilon(m_{i_k}) + \frac{1}{\log N}\frac{m_{i_k}}{\log^{10C}(m_{i_k})} \le 0.01\pi_\epsilon(m_{i_k}) + \frac{1}{\log N}\frac{m_{i_k}}{\log^{r}(m_{i_k})} \le .01\pi_\epsilon(m_{i_k}) + \frac{\pi_\epsilon(m_{i_k})}{\log N} \le .02\pi_\epsilon(m_{i_k})$ (here, similarly to Lemma 24, r is the constant given by the definition of universal tractability for $\mathcal{U}(C)$, and hence $r \le 10C$).

Finally, we must argue the correctness of Theorem 26. Using some notation taken from the proof, consider an output $c_k = (a_k, b_k, i_k)$ (if $c_k = 0$, the data structure does not output it to the set T) and observe that $G(a_k)$ for any a_k satisfying $(1-4\epsilon)G(a_k + b_k + 2\epsilon' a_k) \le G(a_k) \le (1+4\epsilon)G(a_k - b_k - 2\epsilon' a_k)$ is a $(1 \pm 4 \cdot \frac{\epsilon}{4})$-approximation of a distinct entry $G(m_{i_k})$. Moreover, if there is a $(G,1)$-heavy element m_{i_k}, then we again have $\pi_\epsilon^2(m_{i_k}) = \Omega\left(\log^{-(t+1)}(nN)\sum_{j \notin S \cup \{i_k\}} m_j^2\right)$. In fact, delving into the proof of Lemma 25 (found in [9]), we see that the specific value of t depends on G, and is given by the definition of universal tractability for $\mathcal{U}(C)$. Since $t \le 10C$ and we choose $\tau = \frac{\log^{10C+2}(nN)}{p}$, we get the probability of the bad event $\neg\mathcal{L}$ (using the same notation from Theorem 26) is bounded by:

$$\frac{E(X \mid \mathcal{B})20^{10}}{\pi_\epsilon^2(m_{i_k})} + \frac{O(\log N)}{\tau} = \frac{20^{10}\log^{t+1}(nN)\sum_{j\notin S\cup\{i_k\}} m_j^2}{\tau \sum_{j \notin S \cup \{i_k\}} m_j^2} + \frac{O(\log N)}{\tau} \le p.$$

The rest of the proof goes through in the same way, and hence this gives the theorem. ◀

We now argue how to use our universal core algorithm UCA as a subroutine to give the main result of the paper. The proof of the theorem below can be found in Appendix B, the argument of which follows a similar one found in [10].

▶ **Theorem 29.** *Fix a constant C and let $\mathcal{U}(C)$ be the set of tractable functions from the definition of universal tractability. Suppose there is a universal core algorithm that has parameters $\epsilon = \Omega(1/\log^k(nN))$ (for $0 \le k \le C$), $\delta = \Omega(1/\log^u(nN))$ (for $u \ge 0$), $\alpha = 1$, and $\mathcal{G} = \mathcal{U}(C)$, uses polylogarithmic memory in n and N, and makes one pass over D. Then there is a universal sum algorithm that has parameters $\epsilon = \Omega(1/\log^k(nN))$ (for $0 \le k \le C$), $\delta = 0.3$, and $\mathcal{G} = \mathcal{U}(C)$, uses polylogarithmic space in n and N, and makes one pass over D.*

We can reduce the failure probability to inverse polynomial using standard techniques. Our main result, Theorem 12, follows from Theorem 28 and Theorem 29.

References

1 List of open problems in sublinear algorithms: Problem 20. `http://sublinear.info/20`.

2 List of open problems in sublinear algorithms: Problem 30. `http://sublinear.info/30`.

3 N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC*, 1996.

4 Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *FOCS*, 2002.

5 Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *RANDOM*. Springer, 2002.

6 L. Bhuvanagiri, S. Ganguly, D. Kesh, and C. Saha. Simpler algorithm for estimating frequency moments of data streams. In *SODA*, 2006.

7 V. Braverman, R. Gelles, and R. Ostrovsky. How to catch l_2-heavy-hitters on sliding windows. In *COCOON*. Springer, 2013.

8 V. Braverman and R. Ostrovsky. Smooth histograms for sliding windows. In *FOCS*, 2007.

9 V. Braverman and R. Ostrovsky. Zero-one frequency laws. In *STOC*, 2010.

10 V. Braverman and R. Ostrovsky. Generalizing the layering method of Indyk and Woodruff: Recursive sketches for frequency-based vectors on streams. In *APPROX-RANDOM*. Springer, 2013.

11 A. Chakrabarti, S. Khot, and X. Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *CCC*, 2003.

12 D. Coppersmith and R. Kumar. An improved data stream algorithm for frequency moments. In *SODA*, 2004.

13 G. Cormode, M. Datar, P. Indyk, and S. Muthukrishnan. Comparing data streams using hamming norms (how to zero in). *IEEE TKDE*, 15(3):529–540, 2003.

14 G. Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. *ACM TODS*, 30(1):249–278, 2005.

15 M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *SODA*, 2002.

16 J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate l_1-difference algorithm for massive data streams. In *FOCS*, 1999.

17 P. Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *JCSS*, 31(2):182–209, 1985.

18 S. Ganguly and G. Cormode. On estimating frequency moments of data streams. In *APPROX-RANDOM*. Springer, 2007.

19 P. B. Gibbons and S. Tirthapura. Distributed streams algorithms for sliding windows. In *SPAA*, 2002.

20 L. Golab, D. DeHaan, E. D. Demaine, A. Lopez-Ortiz, and J. I. Munro. Identifying frequent items in sliding windows over on-line packet streams. In *IMC*, 2003.

21 R. Hung, L. K. Lee, and H. F. Ting. Finding frequent items over sliding windows with constant update time. *IPL*, 110(7):257–260, 2010.

22 P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *JACM*, 53(3):307–323, 2006.

23 P. Indyk and D. Woodruff. Tight lower bounds for the distinct elements problem. In *FOCS*, 2003.

24 P. Indyk and D. Woodruff. Optimal approximations of the frequency moments of data streams. In *STOC*, 2005.

25 C. Jin, W. Qian, C. Sha, J. X. Yu, and A. Zhou. Dynamically maintaining frequent items over a data stream. In *CIKM*, 2003.

26 D. M. Kane, J. Nelson, and D. Woodruff. On the exact space complexity of sketching and streaming small norms. In *SODA*, 2010.

27 D. M. Kane, J. Nelson, and D. Woodruff. An optimal algorithm for the distinct elements problem. In *PODS*, 2010.
28 Y. Li, H. Nguyễn, and D. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *STOC*, 2014.
29 R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.
30 G. Nie and Z. Lu. Approximate frequency counts in sliding window over data stream. In *CCECE*, 2005.
31 D. Woodruff. Optimal space lower bounds for all frequency moments. In *SODA*, 2004.
32 L. Zhang and Y. Guan. Frequency estimation over sliding windows. In *ICDE*, 2008.

A G-Sum from G-Core

We now show how to reduce the G-Sum problem to the G-Core problem. In particular, we prove the following theorem. The algorithm and proof of correctness follow from [10]. We restate the algorithm and results using our notation for completeness.

▶ **Theorem 30.** *If there is an algorithm that solves G-Core using memory $O^*(1)$ and makes one pass over D except with probability $O(\log^{-u}(nN))$ for some $u > 0$, then there is an algorithm that solves G-Sum using memory $O^*(1)$ and makes one pass over D except with probability at most* 0.3.

Note that we can reduce the failure probability from constant to inverse polynomial using standard techniques. Combining this with Theorem 26 and Theorem 16, we have Theorem 14.

Let G be a tractable function according to Definition 6, and let $D(n, N)$ be a stream given as input. We show how to construct an algorithm that solves the G-Sum problem by using our algorithm for G-Core as a subroutine. In particular, consider the Compute-G-Core algorithm that solves the G-Core problem. Note that for the output set $S = \{g'_1, \dots, g'_\ell\}$ maintained by Compute-G-Core, using standard techniques one can easily obtain the explicit set of indices $\{j_1, \dots, j_\ell\}$ such that $(1-\epsilon)G(m_{j_i}) \le g'_i \le (1+\epsilon)G(m_{j_i})$ for each $1 \le i \le \ell$. Hence, we assume that Compute-G-Core outputs a set of pairs of the form $\{(g'_1, j_1), \dots, (g'_\ell, j_\ell)\}$.

In the language of [10], Compute-G-Core produces a $(1, \epsilon)$-cover with respect to the vector $G(M(W)) = (G(m_1), \dots, G(m_n))$ with probability at least $1 - \delta$, where $\epsilon = \Omega(1/\log^k(nN))$ (for any $k \ge 0$) and $\delta = \Omega(1/\log^u(nN))$ (for any $u \ge 0$). Given the tractable function G, our algorithm for G-Sum is as follows:

```
1 Generate $\phi = O(\log(n))$ pairwise independent, uniform zero-one vectors
  $H_1, \dots, H_\phi : [n] \to \{0, 1\}$, and let $h^k_i = H_k(i)$. Let $D_k$ be the substream defined by
  $D_{H_1 H_2 \dots H_k}$, and let $G(M(W_k))$ denote $(G(m_1), \dots, G(m_n))$ for the substream $D_k$ and
  window $W$ (where $k \in [\phi]$).
2 Maintain, in parallel, the cores $Q_k$ = Compute-$G$-Core$(D_k, \frac{\epsilon^2}{\phi^3}, \epsilon, \frac{1}{\phi})$ for each $k \in [\phi]$.
3 If $F_0(G(M(W_\phi))) > 10^{10}$, then output 0.
4 Otherwise, precisely compute $Y_\phi = |G(M(W_\phi))|$.
5 For each $k = \phi - 1, \dots, 0$, compute $Y_k = 2Y_{k+1} - \sum_{(g'_i, j_i) \in Q_k} (1 - 2h^k_{j_i}) g'_i$.
6 Output $Y_0$.
```

Algorithm 4: Compute-G-Sum(D, ϵ)

Note that, in our paper, Compute-G-Core(D, ϵ, δ) only takes three parameters (the stream D, error bound ϵ, and failure probability δ), while the algorithm from [10] assumes four parameters of the form Compute-G-Core$(D, \alpha, \epsilon, \delta)$. Here, D, ϵ, and δ have the same meaning as in our paper. The parameter α controls how heavy an element needs to be (according to the function G) in order to necessarily be in the output set of Compute-G-Core. That is, in the set $T = \{(g'_1, j_1), \dots, (g'_\ell, j_\ell)\}$ output by Compute-G-Core, if there is an i such that m_i is (G, α)-heavy with respect to $M(W)$, then $i \in \{j_1, \dots, j_\ell\}$. We solve the G-Core problem for $\alpha = 1$, but Algorithm Compute-G-Sum needs the problem solved for $\alpha = \frac{\epsilon^2}{\phi^3}$. However, using standard techniques, we can reduce the problem of solving G-Core for $\alpha = \frac{\epsilon^2}{\phi^3}$ to the same problem for $\alpha = 1$.

▶ **Theorem 31.** *For any tractable function G, Algorithm Compute-G-Sum outputs a $(1 \pm \epsilon)$-approximation of $|G(M(W))|$ except with probability at most 0.3, where $\epsilon = \Omega(1/\log^k(nN))$ for any $k \geq 0$. The algorithm uses memory that is polylogarithmic in n and N.*

Proof. The proof of this theorem follows directly from Theorem 1 in [10]. ◀

B Universal Sum from Universal Core

We now prove Theorem 29. The algorithm and proof are similar to that of the reduction from the G-Sum problem to the G-Core problem found in Appendix A, except that we need to carry out the argument within our universal framework. As mentioned, the algorithm and correctness follow from [10]. We do not rewrite the whole algorithm, but instead describe the necessary modifications that need to be made from Appendix A.

Let $D(n, N)$ be a stream given as input to our universal sum algorithm. Let UCA be our universal core algorithm from Theorem 28, Section 4, the parameters of which are specified in our universal sum algorithm description.

Universal Sum Algorithm: We describe the modifications that need to be made to Algorithm Compute-G-Sum from Appendix A.

In Step 2, instead we need to maintain and store the output $Q_k = UCA$ with parameters $\alpha = \frac{\epsilon^2}{\phi^3}$, ϵ (i.e., the one given as input to our universal sum algorithm), $\delta = \frac{1}{\phi}$, and $\mathcal{G} = \mathcal{U}(C)$ for each $k \in [\phi]$ (in the k^{th} parallel iteration, UCA is given the stream D_k as input). As in Appendix A, we construct a universal core structure for $\alpha = 1$, but we can reduce the problem of $\alpha = \frac{\epsilon^2}{\phi^3}$ to $\alpha = 1$. Note that Q_k is of the form $\{(a_1, b_1, j_1), \dots, (a_\ell, b_\ell, j_\ell)\}$ (Q_k may have 0's as well, which we simply ignore). For each such triple (a_i, b_i, j_i), we also store the value of $h^k_{j_i} = H_k(j_i)$.

In Step 3, instead we check if $F_0(M(W_\phi)) \leq 10^{10}$, and if so we store $M(W_\phi)$ (recall $M(W_\phi)$ denotes the frequency vector $(m_1, \dots, m_n)$ for the substream D_ϕ induced by W). We remove Steps 4, 5, and 6.

Querying the Structure: Now, given a function $G \in \mathcal{U}(C)$, we explain how to query the universal sum structure output by our universal sum algorithm to approximate $|G(M(W))|$. In particular, for each k we first query the universal core structure output by UCA to get a set $Q'_k = \{(x_1, j_1), \dots, (x_{\ell'}, j_{\ell'})\}$. Then, we compute $Y_\phi = |G(M(W_\phi))|$ and, for each $k = \phi - 1, \dots, 0$, we recursively compute Y_k according to:

$$Y_k = 2Y_{k+1} - \sum_{(x_i, j_i) \in Q_k} (1 - 2h^k_{j_i}) x_i.$$

Once each Y_k has been computed for $0 \leq k \leq \phi$, we output Y_0.

▶ **Theorem 32.** *Fix a constant C and let $\mathcal{U}(C)$ be the set of tractable functions corresponding to the definition of universal tractability. There is a universal sum algorithm with parameters $\epsilon = \Omega(1/\log^k(nN))$ (for $0 \leq k \leq C$), $\delta = 0.3$, and $\mathcal{G} = \mathcal{U}(C)$. The algorithm uses polylogarithmic space in n and N and makes a single pass over D. When querying the universal sum structure (output by the universal sum algorithm) with a function $G \in \mathcal{U}(C)$, it outputs a $(1 \pm \epsilon)$-approximation of $|G(M(W))|$ except with probability at most* 0.3.

Proof. The proof of this theorem follows directly from Theorem 1 in [10]. ◀

Universal Sketches for the Frequency Negative Moments and Other Decreasing Streaming Sums

Vladimir Braverman[*1] **and Stephen R. Chestnut**[†2]

1 **Department of Computer Science, Johns Hopkins University**
3400 N. Charles St, Baltimore MD, USA
`vova@cs.jhu.edu`

2 **Department of Mathematics, ETH Zürich**
Rämistrasse 101, Zürich, Switzerland
`stephen.chestnut@ifor.math.ethz.ch`

Abstract

Given a stream with frequencies f_d, for $d \in [n]$, we characterize the space necessary for approximating the frequency negative moments $F_p = \sum |f_d|^p$, where $p < 0$ and the sum is taken over all items $d \in [n]$ with nonzero frequency, in terms of n, ϵ, and $m = \sum |f_d|$. To accomplish this, we actually prove a much more general result. Given any nonnegative and nonincreasing function g, we characterize the space necessary for any streaming algorithm that outputs a $(1 \pm \epsilon)$-approximation to $\sum g(|f_d|)$, where again the sum is over items with nonzero frequency. The storage required is expressed in the form of the solution to a relatively simple nonlinear optimization problem, and the algorithm is universal for $(1 \pm \epsilon)$-approximations to any such sum where the applied function is nonnegative, nonincreasing, and has the same or smaller space complexity as g. This partially answers an open question of Nelson (IITK Workshop Kanpur, 2009).

1998 ACM Subject Classification F.1.2 Models of Computation, F.2.1 Numerical Algorithms and Problems

Keywords and phrases data streams, frequency moments, negative moments

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.591

1 Introduction

A *stream* is a sequence $S = ((d_1, \delta_1), (d_2, \delta_2), \ldots, (d_N, \delta_N))$, where $d_i \in [n]$ are called the items or elements in the stream and $\delta_i \in \mathbb{Z}$ is an update to the d_ith coordinate of an implicitly defined n-dimensional vector. Specifically, the *frequency* of $d \in [n]$ after $k \le N$ updates is

$$f_d^{(k)} = \sum \{\delta_j | j \le k, d_j = d\},$$

and the implicitly defined vector $f := f^{(N)}$ is commonly referred to as the *frequency vector* of the stream S. Let $M := \max\{n, |f_d^{(k)}| : d \in [n], 0 \le k \le N\}$ and $m = \sum_d |f_d|$; thus, it requires $O(\log M)$ bits to exactly determine the frequency of a single item. This model is commonly known as the *turnstile* streaming model, as opposed to the *insertion-only*

[*] This material is based upon work supported in part by the National Science Foundation under Grant No. 1447639, by the Google Faculty Award and by DARPA grant N660001-1-2-4014. Its contents are solely the responsibility of the authors and do not represent the official view of DARPA or the Department of Defense.

[†] This material is based upon work supported in part by the National Science Foundation under Grant No. 1447639 and completed while the author was at Johns Hopkins University.

18th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'15) / 19th Int'l Workshop on Randomization and Computation (RANDOM'15).
Editors: Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim; pp. 591–605
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

model which has $\delta_i = 1$, for all i, but is the same otherwise. In an insertion-only stream $N = m \geq M$.

Streams model computing scenarios where the processor has very limited access to the input. The processor reads the updates one-at-a-time, without control of their order, and is tasked to compute a function on the frequency vector. The processor can perform its computation exactly if it stores the entire vector f, but this may be undesirable or even impossible when the dimension of f is large. Thus, the goal is to complete the computation using as little storage as possible. Typically, exact computation requires storage linear in n, so we seek approximations.

Given a stream with frequencies f_d, for $d \in [n]$, we consider the problem of approximating the frequency negative moments, specifically $F_p = \sum |f_d|^p$ where $p < 0$ and the sum is taken over all items $d \in [n]$ with nonzero frequency. We characterize, up to factors of $O(\epsilon^{-1} \log^2 n \log M)$ in the turnstile model and $O(\epsilon^{-1} \log M)$ in the insertion-only model, the space necessary to produce a $(1 \pm \epsilon)$-approximation to F_p, for $p < 0$, in terms of the accuracy ϵ, the dimension n, and the L_1 length m of f.

Negative moments, also known as "inverse moments", of a probability distribution have found several applications in statistics. Early on, they were studied in application to sampling and estimation problems where the sample size is random [35, 18] as well as in life-testing problems [31]. More recently, they appear in the design of multi-center clinical trials [24] and in the running time analysis of a quantum adiabatic algorithm for 3-SAT [37, 38]. F_0/F_{-1} is the harmonic mean of the (nonzero) frequencies in the insertion-only model, and more generally, the value $(F_p/F_0)^{1/p}$ is known as the pth power mean [10]. The harmonic mean is the truest average for some types of data, for example speeds, parallel resistances, and P/E ratios [34].

To our knowledge this is the first paper to consider streaming computation of the frequency negative moments and the first to determine the precise dependence of the space complexity of streaming computations on m. In fact, in the process of characterizing the storage necessary to approximate the frequency negative moments, we actually characterize the space complexity of a much larger class of streaming sum problems. Specifically, given any nonnegative, nonincreasing function $g : \mathbb{N} \to \mathbb{R}$ we determine to within a factor of $O(\epsilon^{-1} \log^2 n \log M)$ the space necessary to approximate

$$g(f) := \sum_{d \in \text{supp}(f)} g(|f_d|),$$

where $\text{supp}(f) := \{d \in [n] : f_d \neq 0\}$ is the support of f. Furthermore, the sketch providing a $(1 \pm \epsilon)$-approximation for $g(f)$ is universal for a $(1 \pm \epsilon)$-approximation for any nonnegative nonincreasing function with the same or smaller space complexity as g. This partially answers a question of Nelson [33] – which families of functions admit universal sketches?

The attention on m is warranted; in fact, the complexity in question depends delicately on this parameter. If we forget about m for a moment, then a standard reduction from the communication problem INDEX implies that computing a $(1 \pm \frac{1}{2})$-approximation to F_p, for $p < 0$, requires $\Omega(n)$ bits of storage – nearly enough to store the entire vector f. However, the reduction requires $m = \Omega(n^{1-1/p})$, recall that $p < 0$. If $m = o(n^{1-1/p})$ then, as we show, one can often get away with $o(n)$ bits of memory.

The next two sections outline our approach to the decreasing streaming sum problem and state our main results. Section 1.3 reviews previous work on streaming sum problems. In Section 2 we show how our results solve the frequency negative moments problem. Sections 3 and 5 prove the main results. Section 4 and Section 6 describe the implementation details for the streaming setting.

1.1 Preliminaries

Let $\mathcal{F} = \{f \in \mathbb{N}^n : \sum f_d \leq m\}$ and let $\mathcal{T}$ and $\mathcal{I}$ denote the sets of turnstile streams and insertion-only streams, respectively, that have their frequency vector f satisfying $|f| \in \mathcal{F}$. The set $\mathcal{F}$ is the set of all nonnegative frequency vectors with L_1 norm at most m. Clearly, $\mathcal{F}$ is the image under coordinate-wise absolute value of the set of all frequency vectors with L_1 norm at most m. We assume $n \leq m$.

In order to address the frequency negative moments problem we will address the following more general problem. Given a nonnegative, nonincreasing function $g : \mathbb{N} \to \mathbb{R}$, how much storage is needed by a streaming algorithm that $(1 \pm \epsilon)$-approximates $g(f)$, for the frequency vector f of any stream $S \in \mathcal{T}$ or $S \in \mathcal{I}$? Equivalently, we can assume that $g(0) = 0$, g is nonnegative and nonincreasing on the interval $[1, \infty)$, and extend the domain of g to $\mathbb{Z}$ by requiring it to be symmetric, i.e., $g(-x) = g(x)$. Therefore, $g(f) = \sum_{i=1}^{n} g(f_d)$. For simplicity, we call such functions "decreasing functions".

A randomized algorithm $\mathcal{A}$ is a *turnstile streaming* $(1 \pm \epsilon)$-*approximation algorithm* for $g(f)$ if

$$P\{(1-\epsilon)g(f) \leq \mathcal{A}(S) \leq (1+\epsilon)g(f)\} \geq \frac{2}{3}$$

holds for every stream $S \in \mathcal{T}$, and insertion only algorithms are defined analogously. For brevity, we just call such algorithms "approximation algorithms" when g, ϵ, and the streaming model are clear from the context. We consider the maximum number of bits of storage used by the algorithm $\mathcal{A}$ with worst case randomness on any valid stream.

A sketch is a, typically randomized, data structure that functions as a compressed version of the stream. Let $\mathcal{G} \subseteq \mathbb{R}^{\mathbb{N}} \times (0, 1/2]$. We say that a sketch is *universal* for a class $\mathcal{G}$ if for every $(g, \epsilon) \in \mathcal{G}$ there is an algorithm that, with probability at least $2/3$, extracts from the sketch a $(1 \pm \epsilon)$-approximation to $g(f)$. The probability here is taken over the sketch as well as the extraction algorithm.

Our algorithms assume a priori knowledge of the parameters m and n, where $m = \|f\|_1$ and n is the dimension of f. In practice, one chooses n to be an upper bound on the number of distinct items in the stream. Our algorithm remains correct if one instead only knows $m \geq \|f\|_1$, however if $m \gg \|f\|_1$ the storage used by the algorithm may not be optimal. We assume that our algorithm has access to an oracle that computes g on any valid input. In particular, the final step of our algorithms is to submit a list of frequencies, i.e., a sketch, as inputs for g. We do not count the storage required to evaluate g or to store its value.

1.2 Our results

Our lower bound is proved by a reduction from the communication complexity of disjointness wherein we parameterize the reduction with the coordinates of $|f|$, the absolute value of a frequency vector. The parameterization has the effect of giving a whole collection of lower bounds, one for each frequency vector among a set of many. Specifically, if $f \in \mathcal{F}$ and $g(f) \leq \epsilon^{-1}g(1)$ then we find an $\Omega(|\operatorname{supp}(f)|)$ lower bound on on the number of bits used by any approximation algorithm. This naturally leads us to the following nonlinear optimization problem

$$\sigma(\epsilon, g, m, n) := \max\left\{|\operatorname{supp}(f)| : f \in \mathcal{F}, g(f) \leq \epsilon^{-1}g(1)\right\}, \tag{1}$$

which gives us the "best" lower bound. We will use $\sigma = \sigma(\epsilon, g, m, n)$ when ϵ, g, m, and n are clear from the context. Our main lower bound result is the following.

▶ **Theorem 1.** *Let g be a decreasing function, then any k-pass insertion-only streaming $(1 \pm \epsilon)$-approximation algorithm requires $\Omega(\sigma/k)$ bits of space.*

Before we consider approximation algorithms, let us consider a special case. Suppose there is an item d^* in the stream that satisfies $g(f_{d^*}) \geq \epsilon g(f)$. An item such as d^* is called an ϵ-heavy element. If there is an ϵ-heavy element in the stream, then $g(1) \geq g(f_{d^*}) \geq \epsilon g(f)$ which implies $|\operatorname{supp}(f)| \leq \sigma$, by the definition of σ. Of course, in this case it is possible compute $g(f)$ with $O(\sigma \log M)$ bits in one pass in the insertion-only model and with not much additional space in the turnstile model simply by storing a counter for each element of $\operatorname{supp}(f)$. Considering the $\Omega(\sigma)$ lower bound, this is nearly optimal. However, it only works when f contains an ϵ-heavy element.

Our approximation algorithm is presented next. It gives a uniform approach for handling all frequency vectors, not just those with ϵ-heavy elements.

Algorithm 1 $(1 \pm \epsilon)$-approximation algorithm for $g(f)$.

```
1: Compute $\sigma = \sigma(\epsilon, g, m, n)$ and let
       $$q \geq \min\left\{1, \frac{9\sigma}{\epsilon|\operatorname{supp}(f)|}\right\}. \qquad (2)$$
2: Sample pairwise independent random variables $X_d \sim \text{Bernoulli}(q)$, for $d \in [n]$, and let
   $W = \{d \in \operatorname{supp}(f) : X_d = 1\}$.
3: Compute $f_d$, for each $d \in W$.
4: Output $q^{-1} \sum_{d \in W} g(f_d)$.
```

Algorithm 1 outlines the important components of our streaming algorithm and suppresses the details needed to implement it on a stream. In particular, $|\operatorname{supp}(f)|$ is not known ahead of time, and in fact, any streaming algorithm that computes it exactly requires $\Omega(n)$ bits of storage. This and the remaining details can be handled with existing streaming technology as described in Section 6.

Algorithm 1 simply samples each element of $\operatorname{supp}(f)$ pairwise independently with probability q. The expected sample size is $q|\operatorname{supp}(f)|$, so in order to achieve optimal space we should take equality in Equation 2. The choice yields, in expectation, $q|\operatorname{supp}(f)| = O(\sigma/\epsilon)$ samples. Section 4 explains how to compute σ quickly and with small storage, and the correctness of Algorithm 1 is established by the following theorem. It is proved in Section 3.

▶ **Theorem 2.** *There is a turnstile streaming algorithm that, with probability at least $2/3$, outputs a $(1 \pm \epsilon)$-approximation to $g(f)$ and uses $O(\epsilon^{-1}\sigma \log^2(n) \log(M))$ bits of space. The algorithm can be implemented in the insertion-only model with $O(\epsilon^{-1}\sigma \log(M) + \log^2 n)$ bits of space.*

It is worth mentioning that the suppressed constants in the asymptotic bounds of Theorems 1 and 2 are independent of g, ϵ, m, and n.

The optimization problem (1) reappears in the proof of Theorem 2. The key step is the observation mentioned above. Namely, for the particular frequency vector f that is our input, if there is an item d satisfying $g(|f_d|) \geq \epsilon g(f)$ then $|\operatorname{supp}(f)| \leq \sigma$.

Let us now emphasize a particular feature of this algorithm. Previously, we commented that choosing equality in (2) is optimal in terms of the space required. However, Algorithm 1 is still correct when the inequality is strict. Notice that the sketch is just a (pairwise independent) random sample of $\operatorname{supp}(f)$ and its only dependence on g and ϵ is through the parameter σ/ϵ. Let g' and ϵ' be another decreasing function and error parameter satisfying $\frac{\sigma(\epsilon',g',m,n)}{\epsilon'} \leq \frac{\sigma(\epsilon,g,m,n)}{\epsilon}$, then

$$q' = \min\left\{1, \frac{9\sigma'}{\epsilon'|\operatorname{supp}(f)|}\right\} \leq q = \min\left\{1, \frac{9\sigma}{\epsilon|\operatorname{supp}(f)|}\right\}.$$

In particular, this means that the sketch that produces an $(1 \pm \epsilon)$-approximation to $g(f)$ also suffices for an $(1 \pm \epsilon')$-approximation to g'. For example, if one takes $g' \geq g$, pointwise with $g'(1) = g(1)$, then $\sigma(\epsilon, g', m, n) \leq \sigma(\epsilon, g, m, n)$ so one can extract from the sketch $(1 \pm \epsilon)$-approximations to $g(f)$ and $g'(f)$, each being separately correct with probability $2/3$. Thus, the sketch is universal for any decreasing function g' and accuracy ϵ' where $\frac{\sigma(\epsilon',g',m,n)}{\epsilon'} \leq \frac{\sigma(\epsilon,g,m,n)}{\epsilon}$. In the context of the frequency negative moments, this implies that the sketch yielding a $(1 \pm \epsilon)$-approximation to F_p, for $p < 0$, is universal for $(1 \pm \epsilon)$-approximations of $F_{p'}$, for all $p \leq p' < 0$.

Computing the sketch requires a priori knowledge of σ. If one over-estimates σ the algorithm remains correct, but the storage used increases. To know σ requires knowledge of m, or at least an good upper bound on m. This is a limitation, but there are several ways to mitigate it. If one does not know m but is willing to accept a second pass through the stream, then using the algorithm of [26] one can find a $(1 \pm \frac{1}{2})$-approximation to m with $O(\log M)$ bits of storage in the first pass and approximate $g(f)$ on the second pass. A $(1 \pm \frac{1}{2})$-approximation to m is good enough to determine σ to within a constant, which is sufficient for the sketch. Alternatively, one can decide first on the space used by the algorithm and, in parallel within one pass, run the algorithm and approximate m. After reading the stream one can determine for which decreasing functions g and with what accuracy ϵ does the approximation guarantee hold.

1.3 Background

Much of the effort dedicated to understanding streaming computation, so far, has been directed at the frequency moments $F_p = \sum |f_i|^p$, for $0 < p < \infty$, as well as F_0 and F_∞, the number of distinct elements and the maximum frequency respectively. In the turnstile model, F_0 is distinguished from $L_0 = |\operatorname{supp}(f)|$, the number of elements with a nonzero frequency.

The interest in the frequency moments began with the seminal paper of Alon, Matias, and Szegedy [1], who present upper and lower bounds of $O(\epsilon^{-2} n^{1-1/p})$ and $\Omega(n^{1-5/p})$, respectively, on the space needed to find a $(1 \pm \epsilon)$-approximation to F_p, and a separate $O(\epsilon^{-2} \log m)$ space algorithm for F_2. Since then, many researchers have worked to push the upper and lower bounds closer together. We discuss only a few of the papers in this line of research, see [36] an the references therein for a more extensive history of the frequency moments problem.

To approximate F_p, Alon, Matias, and Szegedy inject randomness into the stream and then craft an estimator for F_p on the randomized stream. A similar approach, known as stable random projections, is described by Indyk [22] for F_p, when $0 < p \leq 2$ (also referred to as ℓ_p approximation). Kane, Nelson, and Woodruff [26] show that Indyk's approach, with a more careful derandomization, is optimal. Using the method of stable random projections, Li [29] defined the so-called *harmonic mean estimator* for F_p, when $0 < p < 2$, which improves upon the sample complexity of previous methods. We stress that this is not an estimator for the harmonic mean of the frequencies in a data stream, rather it is an estimator for F_p that takes the form of the harmonic mean of a collection of values.

For $p > 2$, the AMS approach was improved upon [15, 16] until a major shift in the design of streaming algorithms began with the algorithm of Indyk and Woodruff [23] that solves the frequency moments problem with, nearly optimal, $n^{1-2/p}(\frac{1}{\epsilon}\log n)^{O(1)}$ bits. Their algorithm introduced a recursive subsampling technique that was subsequently used to further reduce space complexity [5, 7], which now stands at $O(\epsilon^{-2}n^{1-2/p}\log n)$ in the turnstile model [17] with small ϵ and $O(n^{1-2/p})$ in the insertion-only model with $\epsilon = \Omega(1)$ [6].

Recently, there has been a return to interest in AMS-type algorithms motivated by the difficulty of analyzing algorithms that use recursive subsampling. "Precision Sampling" of Andoni, Krauthgamer, and Onak [2] is one such algorithm that accomplishes nearly optimal space complexity without recursive subsampling. Along these lines, it turns out that one can approximate $g(f)$ by sampling elements $d \in [n]$ with probability roughly $q_d \approx g(f_d)/\epsilon^2 g(f)$, or larger, and then averaging and scaling appropriately, see Proposition 4. Algorithm 1 takes this approach, and also fits in the category of AMS-type algorithms. However, it is far from clear how to accomplish this sampling optimally in the streaming model for a completely arbitrary function g.

A similar sampling problem has been considered before. Monemizadeh and Woodruff [32] formalized the problem of sampling with probability $q_d = g(f_d)/g(f)$ and then go on to focus on L_p sampling, specifically $g(x) = |x|^p$, for $0 \le p \le 2$. In follow-up work, Jowhari, Săglam, and Tardos offer L_p sampling algorithms with better space complexity [25].

As far as the frequency moments lower bounds go, there is a long line of research following AMS [4, 13, 19, 3] that has led to a lower bound matching the best known turnstile algorithm of Ganguly [17] to within a constant [30], at least for some settings of m and ϵ. The insertion-only algorithm of Braverman et al. [6] matches the earlier lower bound of Chakrabarti, Khot, and Sun [13].

For a general function g not much is known about the space-complexity of approximating $g(f)$. Most research has focused on specific functions. Chakrabarti, Do Ba, and Muthukrishnan [12] and Chakrabarti, Cormode, and Muthukrishnan [11] sketch the Shannon Entropy. Harvey, Nelson, and Onak [21] approximate Renyi $\log(\|f\|_\alpha^\alpha)/(1-\alpha)$, Tsallis $(1-\|x\|_\alpha^\alpha)/(\alpha-1)$, and Shannon entropies. Braverman, Ostrovsky, and Roytman [8, 9] characterized nonnegative, nondecreasing functions that have polylogarithmic-space approximation algorithms and they present a universal sketching algorithm for this class of functions. Their algorithm is based on the subsampling technique. Guha, Indyk, McGregor [20] study the problem of sketching common information divergences between the streams, i.e., statistical distances between the probability distributions with p.m.f.s $e/\|e\|_1$ and $f/\|f\|_1$.

2 The frequency negative moments

Before proving Theorems 1 and 2, let us deploy them to determine the streaming space complexity of the frequency negative moments. It will nicely illustrate the trade-off between the length of the stream and the space complexity of the approximation.

The first step is to calculate $\sigma(\epsilon, g, m, n)$, where $g(x) = |x|^p$, for $x \neq 0$ and $p < 0$, and $g(0) = 0$. There is a maximizer of (1) with L_1 length m because g is decreasing. The convexity of g on $[0, \infty)$ implies that $\sigma \le \max\{s \in \mathbb{R} : s(m/s)^p \le \epsilon^{-1}\}$, and σ is at least the minimum of n and $\max\{s \in \mathbb{N} : s(m/s)^p \le \epsilon^{-1}\}$ by definition. Thus, we can take $\sigma = \min\left\{n, \theta\left(\epsilon^{\frac{-1}{1-p}} m^{\frac{-p}{1-p}}\right)\right\}$. This gives us the following corollary to Theorems 1 and 2.

▶ **Corollary 3.** *For any $p < 0$ and $\epsilon > 0$, any algorithm that outputs a $(1 \pm \epsilon)$-approximation to F_p requires $\Omega(\min\{n, \epsilon^{\frac{-1}{1-p}} m^{\frac{-p}{1-p}}\})$ bits of space. Such an approximation can be found with $O(\epsilon^{-\frac{2-p}{1-p}} m^{\frac{-p}{1-p}} \log^2 n \log M)$ bits in a turnstile stream and $O(\epsilon^{-\frac{2-p}{1-p}} m^{\frac{-p}{1-p}} \log M)$ bits in an insertion-only stream.*

For example, taking $p = -1$, which is what we need to estimate the harmonic mean, we find that the complexity is approximately $\frac{\sigma}{\epsilon} = \min\{n, \theta(\epsilon^{-3/2}m^{1/2})\}$. This is also the space complexity of approximating the harmonic mean of the nonzero frequencies. It is apparent from the formula that the relationship between m and n is important for the complexity.

3 Correctness of the algorithm

This section presents the proof that our approximation algorithm is correct. Algorithm 1 describes the basic procedure, and Section 6 describes how it can be implemented in the streaming setting. The correctness relies on our ability to perform the sampling and the following simple proposition.

▶ **Proposition 4.** *Let g be a nonnegative function and let $X_d \sim$ Bernoulli(p_d) be pairwise independent random variables with $p_d \geq \min\left\{1, \frac{9g(f_d)}{\epsilon^2 g(f)}\right\}$, for all $d \in [n]$. Let $\hat{G} = \sum_{d=1}^n p_d^{-1} X_d g(f_d)$, then $P(|\hat{G} - g(f)| \leq \epsilon g(f)) \geq \frac{8}{9}$.*

Proof. We have $E\hat{G} = g(f)$ and $Var(\hat{G}) \leq \sum_d p_d^{-1} g(f_d)^2 = \frac{1}{9}(\epsilon g(f))^2$, by pairwise independence. The proposition now follows from Chebyshev's inequality. ◀

The algorithm samples each element of supp(f) with probability about $\sigma/\epsilon\, \text{supp}(f)$. In order to show that this sampling probability is large enough for Proposition 4 we will need one lemma. It gives us some control on $\sigma(\epsilon, g, m, n)$ as ϵ varies.

▶ **Lemma 5.** *If $\alpha < \epsilon$, then $\epsilon(1 + \sigma(\epsilon, g, m, n)) \geq \alpha\sigma(\alpha, g, m, n)$.*

Proof. Let $\sigma_\epsilon = \sigma(\epsilon, g, m, n)$ and define σ_α similarly. Let $f \in \mathcal{F}$ such that $\sigma_\alpha = |\,\text{supp}(f)|$ and $g(f) \leq \alpha^{-1}g(1)$, without loss of generality the coordinates are ordered such that $f_1 \geq f_2 \geq \cdots \geq f_{\sigma_\alpha} > 0$. Let $s' = \frac{\alpha}{\epsilon}\sigma_\alpha$, and let f' be the vector that takes the first $\lfloor s' \rfloor$ coordinates from f and is 0 thereafter. The choice is made so that $f' \in \mathcal{F}$ and

$$g(f') \leq \frac{\alpha}{\epsilon} g(f) \leq \epsilon^{-1} g(1).$$

Then, by definition of σ_ϵ, we have

$$\sigma_\epsilon \geq |\,\text{supp}(f')| = \left\lfloor \frac{\alpha}{\epsilon}\sigma_\alpha \right\rfloor \geq \frac{\alpha}{\epsilon}\sigma_\alpha - 1. \quad ◀$$

For brevity, we only state here the correctness of the streaming model sampling algorithm, which uses standard techniques. The details of the algorithm are given in the Section 6.

▶ **Lemma 6.** *Given $s \leq n$, there is an algorithm using $O(s \log^2 n \log M)$ bits of space in the turnstile model and $O(s \log M + \log^2 n)$ bits in the insertion-only model that samples each item of supp(f) pairwise-independently with probability at least $\min\{1, s/|\,\text{supp}(f)|\}$ and, with probability at least $7/9$, correctly reports the frequency of every sampled item and the sampling probability.*

Finally, we prove the correctness of our approximation algorithm. Here is where we will again use the optimality of σ in its definition (1). In regards to the lower bound of Theorem 1, this upper bound leaves gaps of $O(\epsilon^{-1}\log^2 n \log M)$ and $O(\epsilon^{-1}\log M)$ in the turnstile and insertion-only models, respectively.

Proof of Theorem 2. We use the algorithm of Lemma 6 to sample with probability at least $q = \min\{1, 9(\sigma+1)/\epsilon|\operatorname{supp}(f)|\}$. Let us first assume that $q \geq \min\{1, 9g(f_d)/\epsilon^2 g(f)\}$, for all d, so that the hypothesis for Proposition 4 is satisfied. The algorithm creates samples W_i, for $i = 0, 1, \ldots, O(\log n)$, where each item is sampled in W_i with probability $q_i = 2^{-i}$. For each i such that $q_i \geq q$, Proposition 4 guarantees that $\hat{G}_i = q_i^{-1} \sum_{d \in W_i} g(f_d)$ is a $(1 \pm \epsilon)$-approximation with probability at least 8/9. With probability at least 7/9, the algorithm returns one of these samples correctly, and then the approximation guarantee holds. Thus, the approximation guarantee holds with probability at least 2/3.

It remains to show that $q \geq \min\{1, 9g(f_d)/\epsilon g(f)\}$, for all $d \in [n]$. Let $\alpha = g(1)/g(f)$ then define $\sigma_\epsilon = \sigma(\epsilon, g, m, n)$ and $\sigma_\alpha = \sigma(\alpha, g, m, n)$. By definition $|\operatorname{supp}(f)| \leq \sigma_\alpha$, thus if $\alpha \geq \epsilon$ then $|\operatorname{supp}(f)| \leq \sigma_\alpha \leq \sigma_\epsilon$, so the sampling probability is 1 and the claim holds.

Suppose that $\alpha < \epsilon$. For all $d \in [n]$, we have

$$\frac{g(f_d)}{g(f)} \leq \frac{g(1)}{g(f)} = \alpha \leq \frac{\epsilon(1+\sigma_\epsilon)}{\sigma_\alpha} \leq \frac{\epsilon(1+\sigma_\epsilon)}{|\operatorname{supp}(f)|},$$

where the second inequality comes from Lemma 5 and the third from the definition of σ_α as a maximum. In particular, this implies that

$$\frac{9\epsilon^{-1}(\sigma+1)}{|\operatorname{supp}(f)|} \geq \frac{9g(f_d)}{\epsilon^2 g(f)},$$

which completes the proof. ◀

4 Computing σ

The value σ is a parameter that is needed for Algorithm 1. That means we need a way to compute it for any decreasing function. As we mentioned before, the only penalty for overestimating σ is inflation of the storage used by the algorithm so to over-estimate σ by a constant factor is acceptable. This section shows that one can find σ' such that $\sigma \leq \sigma' \leq 4\sigma$ quickly, with $O(\log n)$ bits of storage, and by evaluating g at just $O(\log m)$ points.

Because g is decreasing, the maximum of (1) will be achieved by a vector f of length m. This is regardless of whether $m \leq n$ or $m > n$. Lemma 7 says that we might as well take all of the other frequencies to be equal, so we can find a near maximizer by enumerating the single value of those frequencies. Specifically,

$$s(y) = \min\left\{\frac{m}{y}, \frac{g(1)}{\epsilon g(y)}\right\}$$

is the maximum bound we can achieve using y as the single frequency. The value of σ is at most twice $\max\{s(y) : (m/n) \leq y \leq m\}$, by Lemma 7.

But we do not need to check every $y = 1, 2, \ldots, m$ to get a pretty good maximizer. It suffices to check only values where y is a power of two. Indeed, suppose that y^* maximizes $s(y)$ and let $y^* \leq y' \leq 2y^*$. We will show that $s(y') \geq s(y^*)/2$, and since there is a power of two between y^* and $2y^*$ this implies that its s value is at least $s(y^*)/2 \geq \sigma/4$.

Since y^* is a maximizer we have $s(y') \leq s(y^*)$, and because $y' \geq y^*$ and g is decreasing we have $g(y') \leq g(y^*)$. This gives us

$$\frac{g(1)}{\epsilon g(y')} \geq \frac{g(1)}{\epsilon g(y^*)} \geq s(y^*).$$

We also have

$$\frac{m}{y'} \geq \frac{m}{2y^*} \geq \frac{1}{2}s(y^*).$$

Combining these two we have $s(y') \geq s(y^*)/2$.

Thus, one can get by with enumerating at most $\lg m$ values to approximate the value of the parameter σ. Take the largest of the $\lg m$ values tried and quadruple it to get the approximation to σ.

5 Lower bounds for decreasing streaming sums

It bears repeating that if $g(x)$ decreases to 0 as $x \to \infty$ then one can always prove an $\Omega(n)$ lower bound on the space complexity of approximating $g(f)$. However, the stream needed for the reduction may be very long (as a function of n). Given only the streams in $\mathcal{T}$ or $\mathcal{I}$, those with L_1-length m or less, a weaker lower bound may be the best available. The present section proves this "best" lower bound, establishing Theorem 1.

The proof uses a reduction from the communication complexity of disjointness, see the book of Kushilevitz and Nisan [28] for background on communication complexity. The proof strategy is to parameterize the lower bound reduction in terms of the frequencies f. Optimizing the parameterized bound over $f \in \mathcal{F}$ gives the best possible bound from this reduction.

The proof of Theorem 1 is broken up with a two lemmas. The first lemma is used in the reduction from DISJ(s), the s-element disjointness communication problem. It will show up again later when we discuss a fast scheme for computing σ for general functions.

▶ **Lemma 7.** *Let $y_i \in \mathbb{R}_{\geq 0}$, for $i \in [s]$, and let $v : \mathbb{R} \to \mathbb{R}_{\geq 0}$. If $\sum y_i \leq Y$ and $\sum v(y_i) \leq V$, then there exists i such that $\frac{s}{2} y_i \leq Y$ and $\frac{s}{2} v(y_i) \leq V$.*

Proof. Without loss of generality $y_1 \leq y_2 \leq \cdots \leq y_s$. Let i_j, $j \in [\sigma]$, order the sequence such that $v(y_{i_1}) \leq v(y_{i_2}) \leq \cdots \leq v(y_{i_s})$ and let $I = \{i_j | j \leq \lfloor s/2 \rfloor + 1\}$. By the Pigeon Hole Principle, there exists $i \in I$ such that $i \leq \lfloor s/2 \rfloor + 1$. Thus $\frac{s}{2} y_i \leq \sum_{j=\lfloor s/2 \rfloor+1}^{s} y_{i_j} \leq Y$ and $\frac{s}{2} v(y_i) \leq \sum_{j=\lfloor s/2 \rfloor+1}^{s} v(y_j) \leq V$. ◀

▶ **Lemma 8.** *Let g be decreasing and $\epsilon > 0$. If $f = (y, y, \ldots, y, 0, \ldots, 0) \in \mathcal{F}$ and $g(f) \leq \epsilon^{-1} g(1)$, then any k-pass $(1 \pm \epsilon)$-approximation algorithm requires $\Omega(|\operatorname{supp}(f)|/k)$ bits of storage.*

Proof. Let $s = \lfloor |\operatorname{supp}(f)|/2 \rfloor$ and let $\mathcal{A}$ be an $(1 \pm \epsilon)$-approximation algorithm. The reduction is from DISJ$(s, 2)$ where Alice receives $A \subseteq [s]$ and Bob receives $B \subseteq [s]$. Their goal is to jointly determine whether $A \cap B = \emptyset$ or not. Our protocol will answer the equivalent question: is $B \subseteq A^c$ or not? Alice and Bob will answer the question by jointly creating a notional stream, running $\mathcal{A}$ on it, and thresholding the outcome.

For each $d \in A^c$, Alice puts $(d, 1)$ in the stream y times. She then runs $\mathcal{A}$ on her portion of the stream and sends the contents its memory to Bob. For each $d \in B$, Bob adds $(d, 1)$ to the stream. Bob runs $\mathcal{A}$ on his portion of the stream and sends the memory back to Alice. She recreates her portion of the stream, advances $\mathcal{A}$, sends the memory to Bob, etc., until each player has acted k times. In addition to the algorithm's memory, on each pass Alice sends at most $\lceil k^{-1} \lg |A| \rceil$ binary digits of $|A|$ so that Bob knows $|A|$ at the end of the protocol.

The stream is a member of $\mathcal{I}$ by construction; let f' be its frequency vector. At the end, Bob finishes computing $\mathcal{A}(f')$. All of the frequencies are y, $y + 1$, or 1. If

$$\mathcal{A}(f') \leq (1 + \epsilon)[|B| g(y + 1) + (s - |A| - |B|) g(y)],$$

then Bob declares $B \subseteq A^c$ and otherwise $B \not\subseteq A^c$.

The exact value of $g(f')$ is

$$|A \cap B|g(1) + |B \setminus A|g(y+1) + (s - |A| - |B| + |A \cap B|)g(y).$$

If $B \subseteq A^c$ this value is

$$V_0 := |B|g(y+1) + (s - |A| - |B|)g(y),$$

and otherwise, because g is decreasing, it is at least

$$V_1 := g(1) + (|B| - 1)g(y+1) + (s - |B| - |A| + 1)g(y).$$

We find

$$V_1 - V_0 \geq g(1) \geq \epsilon g(f) \geq 2\epsilon s g(y) \geq 2\epsilon V_0$$

Hence, if $\mathcal{A}(f')$ is a $(1 \pm \epsilon)$-approximation to $g(f')$, then Bob's decision is correct. The protocol with solves Disj(s) which requires, in the worst case, $\Omega(s)$ bits of communication including $O(k^{-1} \lg s)$ bits to send $|A|$ and $\Omega(s) = \Omega(|\operatorname{supp}(f)|)$ bits for $(2k-1)$ transmissions of the memory of $\mathcal{A}$. Thus, in the worst case, at least one transmission has size $\Omega(|\operatorname{supp}(f)|/k)$. ◀

Proof of Theorem 1. Let $f \in \mathcal{F}$ be a maximizer of (1) and apply Lemma 7 to the positive elements of f. From this we find that there exists y such that $ys' \leq \|f\|_1$ and $g(1) \geq \epsilon s' g(y)$, for $s' = \sigma/2$. Therefore, $f' = (y, y, \ldots, y, 0, \ldots, 0) \in \mathcal{F}$ with $\lfloor s' \rfloor$ coordinates equal to y. Applying Lemma 8 to f' implies the desired bound. ◀

With Lemma 7 in mind, one may ask: why not restrict the maximization problem in (1), the definition of σ, to streams that have all frequencies equal and still get the same order lower bound? This is valid alternative definition. In fact, doing so does appreciably affect the effort needed to compute σ, it is one of the main steps used by our algorithm to approximate σ in Section 4. However, it makes reasoning about σ a bit messier. For example, in Section 1.2 we comment that if the frequency vector f contains an ϵ-heavy element then $|\operatorname{supp}(f)| \leq \sigma$. This comes directly from the fact that $\{f' \in \mathcal{F} : g(f') \leq \epsilon^{-1} g(1)\}$ is the feasible set for (1). If we restrict the feasible set, then we cannot so directly draw the conclusion. Rather, we must compare $g(f)$ to points in the restricted feasible set by again invoking Lemma 7.

6 Details of the algorithm

The streaming implementation in the turnstile model will make use of the Count Sketch algorithm of Charikar, Chen, and Farach-Colton [14]. It is easy to adapt their algorithm for the purpose of finding $\operatorname{supp}(f)$. This gives us the following theorem.

▶ **Theorem 9** (Charikar, Chen, Farach-Colton [14]). *Suppose that S is a stream with at most s items of nonzero frequency. There is a turnstile streaming algorithm* Count Sketch(S, s, δ) *using $O(s \log \frac{n}{\delta} \log M)$ bits that, with probability at least $1 - \delta$, returns all of the nonzero frequencies in S.*

The sampling algorithm follows. Since we do not know $|\operatorname{supp}(f)|$ at the start of the stream, we guess $O(\log n)$ possible values for it and try each one. After parsing the entire stream, we can use an estimate of $L_0 = |\operatorname{supp}(f)|$ in order to determine which guess is correct. We use $\widehat{L_0}(S^{(i)}, \epsilon, \delta)$ to denote the output of an algorithm that produces a $(1 \pm \frac{1}{8})$-approximation to L_0 with probability at least $1 - \delta$, for example the algorithm of Kane, Nelson, and Woodruff [27]. After the formal presentation of the algorithm we prove its correctness and the claimed storage bounds.

Algorithm 2 Pairwise independent sampling with probability $q \geq s/|\operatorname{supp}(f)|$.

```
 1: procedure SKETCH(Stream $S$, $s > 0$)
 2:     $\ell \leftarrow \lceil \lg(n/s) \rceil$
 3:     for $0 \leq i \leq \ell$ do
 4:         Sample pairwise independent r.v.s $X_{i,d} \sim$ Bernoulli$(2^{-i})$, for $d \in [n]$
 5:         Let $S^{(i)}$ be the substream of $S$ with items $\{d : X_{i,d} = 1\}$
 6:         $U^{(i)} \leftarrow$ COUNT SKETCH$(S^{(i)}, 96s, 1/48)$
 7:     end for
 8:     $L \leftarrow \widehat{L_0}(S^{(i)}, 1/8, 1/18)$
 9:     $i^* \leftarrow \max\left\{0, \left\lfloor \lg \frac{L}{18s} \right\rfloor\right\}$
10:     return $U^{(i^*)}, q = 2^{-i^*}$
11: end procedure
```

▶ **Theorem 10.** *With probability at least $7/9$, Algorithm 2 samples each item in $\operatorname{supp}(f)$ with probability $q \geq s/|\operatorname{supp}(f)|$ and the resulting sample of size $O(s)$. The algorithm can be implemented with $O(s \log(M) \log^2(n))$ bits of space.*

Proof. Let

$$k = \left\lfloor \lg \frac{|\operatorname{supp}(f)|}{16s} \right\rfloor .$$

If $i^* \in \{k-1, k\}$, the streams $S^{(k-1)}$ and $S^{(k)}$ both have small enough support, and the two outputs $U^{(k-1)}$ and $U^{(k)}$ of COUNT SKETCH are correct, then the output is correct. We show that the intersection of these events occurs with probability at least $7/9$.

First, with probability at least $17/18$ L is $(1 \pm 1/8)$-approximation to $|\operatorname{supp}(S)|$. A direct calculations then shows that $i^* \in \{k-1, k\}$.

The following two inequalities arise from the definition of k

$$\frac{64s}{|\operatorname{supp}(f)|} \geq 2^{-(k-1)} \geq 2^{-k} \geq \frac{16s}{|\operatorname{supp}(f)|}. \tag{3}$$

The first inequality implies that the expected support sizes of $S^{(k-1)}$ and $S^{(k)}$ and their variances are all at most $64s$. Chebyshev's inequality implies that each of these values exceeds $96s$ with probability no larger than $64/32^2 = 1/16$. So long as they don't, both streams are valid inputs to COUNT SKETCH. The last inequality of (3), with Theorem 9, implies that the sampling probability is correct.

Putting it together, the total probability of failure is no larger than

$$\frac{1}{18} + \frac{2}{16} + \frac{2}{48} = \frac{2}{9}, \tag{4}$$

where the terms come from the $|\operatorname{supp}(f)|$ estimation, the support sizes of substreams $k-1$ and k, and COUNT SKETCH.

The space bound for turnstile streams follows from Theorem 9. Approximating the support size of the stream with $\widehat{L_0}$ can accomplished with $O(\log n \log \log nM)$ bits using the algorithm of Kane, Nelson, and Woodruff [27]. ◀

Because of deletions in the turnstile model, we need to wait until the end of the stream to rule out any of the guesses of $|\operatorname{supp}(f)|$. This is not the case in the insertion only model. As soon as the number of nonzero counters grows too large we can infer that the sampling probability is too large and discard the sample. It turns out that doing so is enough to cut a $\log n$ factor from the space complexity of Algorithm 2. A further $\log n$ factor can be saved because Count Sketch is not needed in the insertion-only model.

▶ **Corollary 11.** *Algorithm 2 can be implemented with $O(s \log M + \log^2 n)$ bits of storage for insertion-only streams.*

Proof. Define ℓ independent collections of pairwise independent random variables $Y_{i,d} \sim$ Bernoulli$(1/2)$, for $d \in [n]$, and choose the random variables in the algorithm to be

$$X_{i,d} = \prod_{j=1}^{i} Y_{i,d}.$$

One easily checks that each collection $\{X_{i,d}\}_{d\in[n]}$ is pairwise independent and that $P(X_{i,d} = 1) = 2^{-i}$, for all i and d. Storing the seeds for the collection $Y_{i,d}$ requires $O(\log^2 n)$ bits.

We can first save a $\log n$ factor by bypassing Count Sketch and instead simply storing counters for each element that appears in each of the ℓ substreams. The counters should be stored in a hash table or other data structure with no space overhead and a small look-up time. Let us label the maximum number of counters to be stored for each substream as t. We choose $t = \max\{96s, \ell\}$. If the set of counters for each substream is discarded as soon as the number of nonzero counters exceeds the limit of $O(t)$, then the total storage cannot grow to large.

According to Lemma 12, the algorithm uses more than $12t$ counters with probability at most $1/6\ell$, at any given instant.

For each $0 \le i \le \ell$ let $T^{(i)}$ be the longest prefix of stream $S^{(i)}$ such that $|\operatorname{supp}(T^{(i)})| \le s$ and let $k^{(i)}$ denote the number of updates in $T^{(i)}$. Now, notice that the number of counters stored locally maximum at each $k^{(i)}$ and increasing for updates between $k^{(i)}$ and $k^{(i+1)}$. Thus, it is sufficient to bound the storage used by the algorithm at these points.

By a union bound, the probability that the number of counters used by the algorithm at any point $k^{(1)}, k^{(2)}, \ldots, k^{(\ell)}$ is more than $12t$ is at most $\ell \cdot 1/6\ell = 1/6$. Finally, adapting the final union bound of (4) in the previous proof we have that the probability of error is at most $(1/18) + (1/6) = 2/9$. ◀

▶ **Lemma 12.** *Let $v \in \{0,1\}^n$, define ℓ independent collections of pairwise independent random variables $Y_{i,d} \sim$ Bernoulli$(1/2)$, for $s \in [n]$ and $i \in [\ell]$, and set*

$$X_{i,d} = \prod_{j=1}^{i} Y_{i,d}.$$

For a given $s \in \mathbb{N}$, set $k = 0$ if $\sum_d v_d \le s$ or $k = \max\{i : v^T X_i > s\}$ otherwise, where $X_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,n}) \in \{0,1\}^n$. Then

$$P\left(\sum_{i=k+1}^{\ell} v^T X_i > 4s\right) \le \frac{1}{2s}.$$

Proof. The sum is clearly monotonically increasing, so without loss of generality assume $\ell = \infty$. Notice that if $k > 0$, the sum is unchanged (i.e., it remains the same random variable) upon replacing v with the coordinate-wise product of v and X_k. Thus we may also assume that $k = 0$, i.e., $|\operatorname{supp}(v)| \le s$.

For each $d \in \operatorname{supp}(v)$, let $Z_d = \sup\{i : X_{i,d} = 1\}$. Notice that $\{Z_d\}_{d\in\operatorname{supp}(v)}$ is a pairwise independent collection of Geometric$(1/2)$ random variables and let $Z = \sum_{d\in\operatorname{supp}(v)} Z_d$. We have that

$$Z = \sum_{i=0}^{\infty} v^T X_i,$$

because $X_{i,d} = 0$ implies $X_{j,d} = 0$ for all $j > i$.

Pairwise independence implies $EZ = Var(Z) = 2|\operatorname{supp}(v)| \leq 2s$, and by Chebyshev's inequality

$$P(|Z - 2s| > 2s) \leq \frac{Var(Z)}{4s^2} \leq \frac{1}{2s}. \qquad ◀$$

7 Conclusion

It may be possible to apply our methods in order to parameterize according to lengths other than L_1, for example L_∞, or in terms of more general constraints on the set of feasible streams. One challenge with such an adaptation is to ensure that the reduction "preserves" these constraints. For example, if one replaces the L_1 length constraint defining $\mathcal{F}$ with the constraint that the L_∞ length, i.e., maximum frequency, is at most M, then she must take care to ensure that the reduction from Disj never produces a stream with maximum frequency larger than M. It is immediate from the structure of the reduction that it preserves upper bounds on the L_1 size of the frequencies, but our reduction may not preserve an upper bound on the maximum frequency.

Recall that we require an upper bound on the L_1 length of the frequency vector at the start of the algorithm (in order to compute σ). Here the L_1 length has an additional practical advantage over other lengths because it can be computed exactly for insertion-only and strict-turnstile streams by a one pass algorithm with at most $O(\log M)$ bits of memory, and a $(1 \pm \epsilon)$-approximation requires only $O(\epsilon^{-2} \log M)$ bits in the turnstile model [26]. Thus, one can determine after the fact whether the supposed upper bound actually holds.

Acknowledgements. The authors would like to thank several anonymous reviewers for their careful reading of an earlier draft of this paper and their helpful comments.

References

1 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Symposium on the Theory of Computing*, pages 20–29, 1996.

2 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *IEEE Foundations of Computer Science*, pages 363–372, 2011.

3 Alexandr Andoni, Huy L. Nguyễn, Yury Polyanskiy, and Yihong Wu. Tight lower bound for linear sketches of moments. In *Automata, languages, and programming*, volume 7965 of *Lec. Notes in Comput. Sci.*, pages 25–32. Springer, Heidelberg, 2013.

4 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. System Sci.*, 68(4):702–732, 2004.

5 Lakshminath Bhuvanagiri, Sumit Ganguly, Deepanjan Kesh, and Chandan Saha. Simpler algorithm for estimating frequency moments of data streams. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 708–713, 2006.

6 Vladimir Braverman, Jonathan Katzman, Charles Seidell, and Gregory Vorsanger. Approximating large frequency moments with $O(n^{1-2/k})$ bits. *arXiv preprint arXiv:1401.1763*, 2014.

7 Vladimir Braverman and Rafail Ostrovsky. Recursive sketching for frequency moments. *arXiv preprint arXiv:1011.2571*, 2010.

8 Vladimir Braverman and Rafail Ostrovsky. Zero-one frequency laws. In *ACM Symposium on the Theory of Computing*, pages 281–290, 2010.

9 Vladimir Braverman, Rafail Ostrovsky, and Alan Roytman. Universal streaming. *arXiv preprint arXiv:1408.2604*, 2014.

10 Peter S. Bullen. *Handbook of means and their inequalities.* Springer Science & Business Media, 2003.

11 Amit Chakrabarti, Graham Cormode, and Andrew McGregor. A near-optimal algorithm for computing the entropy of a stream. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 328–335. Society for Industrial and Applied Mathematics, 2007.

12 Amit Chakrabarti, Khanh Do Ba, and S. Muthukrishnan. Estimating entropy and entropy norm on data streams. *Internet Math.*, 3(1):63–78, 2006.

13 Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *IEEE Conference on Computational Complexity*, pages 107–117, 2003.

14 Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Automata, languages and programming*, volume 2380 of *Lec. Notes in Comput. Sci.*, pages 693–703. Springer, Berlin, 2002.

15 Don Coppersmith and Ravi Kumar. An improved data stream algorithm for frequency moments. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 151–156, 2004.

16 Sumit Ganguly. Estimating frequency moments of data streams using random linear combinations. In *Approximation, Randomization, and Combinatorial Optimization*, pages 369–380. Springer, 2004.

17 Sumit Ganguly. Polynomial estimators for high frequency moments. *arXiv preprint arXiv:1104.4552*, 2011.

18 Edwin L Grab and I Richard Savage. Tables of the expected value of 1/X for positive bernoulli and poisson variables. *J. Am. Stat. Assoc.*, 49(265):169–177, 1954.

19 André Gronemeier. Asymptotically optimal lower bounds on the NIH-multi-party information complexity of the AND-function and disjointness. In *Symposium on Theoretical Aspects of Computer Science*, 2009.

20 Sudipto Guha, Piotr Indyk, and Andrew McGregor. Sketching information divergences. In *Learning theory*, volume 4539 of *Lec. Notes in Comput. Sci.*, pages 424–438. Springer, Berlin, 2007.

21 Nicholas JA Harvey, Jelani Nelson, and Krzysztof Onak. Sketching and streaming entropy via approximation theory. In *IEEE Symposium on Foundations of Computer Science*, pages 489–498, 2008.

22 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. of the ACM*, 53(3):307–323, 2006.

23 Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. In *ACM Symposium on the Theory of Computing*, pages 202–208, 2005.

24 C.M̃atthew Jones and Anatoly A. Zhigljavsky. Approximating the negative moments of the Poisson distribution. *Statistics & Probability letters*, 66(2):171–181, 2004.

25 Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *ACM Symposium on Principles of Database Systems*, pages 49–58, 2011.

26 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1161–1178, 2010.

27 Daniel M Kane, Jelani Nelson, and David P Woodruff. An optimal algorithm for the distinct elements problem. In *ACM Symposium on Principles of Database Systems*, pages 41–52, 2010.

28 Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, Cambridge, 1997.

29 Ping Li. Estimators and tail bounds for dimension reduction in l_α $(0 < \alpha \leq 2)$ using stable random projections. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 10–19, 2008.

30 Yi Li and David P Woodruff. A tight lower bound for high frequency moment estimation with small error. In *Approximation, Randomization, and Combinatorial Optimization*, pages 623–638. Springer, 2013.

31 W. Mendenhall and H. Lehman, E.˙ An approximation to the negative moments of the positive binomial useful in life testing. *Technometrics*, 2(2):227–242, 1960.

32 Morteza Monemizadeh and David P Woodruff. 1-pass relative-error l_p-sampling with applications. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1143–1160, 2010.

33 Jelani Nelson. List of open problems in sublinear algorithms: Problem 30. `http://sublinear.info/30`.

34 Robert F Reilly and Robert P Schweihs. *The handbook of business valuation and intellectual property analysis.* McGraw Hill, 2004.

35 Frederick F Stephan. The expected value and variance of the reciprocal and other negative powers of a positive Bernoullian variate. *Ann. Math. Stat.*, 16(1):50–61, 1945.

36 David P Woodruff. Data streams and applications in computer science. *Bulletin of EATCS*, 3(114), 2014.

37 Marko Znidaric. Asymptotic expansion for inverse moments of binomial and Poisson distributions. *arXiv preprint math/0511226*, 2005.

38 Marko Žnidarič and Martin Horvat. Exponential complexity of an adiabatic algorithm for an NP-complete problem. *Phys. Rev. A*, 73(2), 2006.

Dependent Random Graphs and Multi-Party Pointer Jumping*

Joshua Brody and Mario Sanchez

Swarthmore College
Swarthmore, PA USA
`joshua.e.brody@gmail.com, msanche1@cs.swarthmore.edu`

Abstract

We initiate a study of a relaxed version of the standard Erdős-Rényi random graph model, where each edge may depend on a few other edges. We call such graphs *dependent random graphs*. Our main result in this direction is a thorough understanding of the clique number of dependent random graphs. We also obtain bounds for the chromatic number. Surprisingly, many of the standard properties of random graphs also hold in this relaxed setting. We show that with high probability, a dependent random graph will contain a clique of size $\frac{(1-o(1))\log(n)}{\log(1/p)}$, and the chromatic number will be at most $\frac{n\log(1/(1-p))}{\log n}$. We expect these results to be of independent interest. As an application and second main result, we give a new communication protocol for the k-player Multi-Party Pointer Jumping ($\textsc{mpj}_k$) problem in the number-on-the-forehead (NOF) model. Multi-Party Pointer Jumping is one of the canonical NOF communication problems, yet even for three players, its communication complexity is not well understood. Our protocol for $\textsc{mpj}_3$ costs $O(n(\log\log n)/\log n)$ communication, improving on a bound from [9]. We extend our protocol to the non-Boolean pointer jumping problem $\widehat{\textsc{mpj}}_k$, achieving an upper bound which is $o(n)$ for any $k \geqslant 4$ players. This is the first $o(n)$ protocol for $\widehat{\textsc{mpj}}_k$ and improves on a bound of Damm, Jukna, and Sgall [12], which has stood for almost twenty years.

1998 ACM Subject Classification F.1.3 Modes of Computation

Keywords and phrases random graphs, communication complexity, number-on-the-forehead model, pointer jumping

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.606

1 Introduction

Random Graphs

The study of random graphs revolves around understanding the following distribution on graphs: Given n and p, define a distribution $G(n,p)$ on n vertex graphs $G = (V, E)$ by placing each edge $(i,j) \in E$ **independently** with probability p. The first paper on this topic, authored by Erdős and Rényi [14], focused on connectivity of graphs. Later, Bollobás and Erdős [7] found the interesting result that almost every graph has a clique number of either r or $r+1$, for some $r \approx \frac{2\log n}{\log 1/p}$. This remarkable concentration of measure result led to further investigations of these graphs. Then, Bollobás [5] solved the question of the chromatic number and showed that almost every graph has chromatic number $(1+o(1))\frac{-n\log(1-p)}{2\log n}$. For more details, consult Bollobás [6] and Alon and Spencer [2].

* This research was supported in part by a grant to Swarthmore College by the Howard Hughes Medical Institute through the Precollege and Undergraduate Science Education Program. The authors gratefully acknowledge the support.

We extend this model by allowing each edge to depend on up to d other edges. We make no a priori assumptions on *how* the edges depend on each other except that edges must be independent of all but at most d other edges. This defines a family of graph distributions $G_d(n,p)$. We initiate a study of dependent random graphs by considering the clique number and the chromatic number. As far as we know, this is the first work to systematically study such distributions. However, other relaxations of the standard random graph model have been studied. The most relevant for us is that of Alon and Nussboim [1], who study random graphs where edges are k-wise independent. Alon and Nussboim give tight bounds for several graph properties, including the clique number, the chromatic number, connectivity, and thresholds for the appearance of subgraphs. The bounds for k-wise independent graph properties are not as tight as the standard random graphs, but this is to be expected since k-wise independent random graphs are a family of distributions rather than a single distribution. Our dependent random graphs similarly represent a family of graph distributions. However, dependent random graphs are generally not even almost k-wise independent, even for small values of d.

NOF Communication Complexity

As an application of our dependent random graphs, we study multi-party communication problems in the Number-On-The-Forehead (NOF) communication model defined by Chandra et al. [11]. In this model, there are k players $\text{PLR}_1, \cdots, \text{PLR}_k$ who wish to compute some function $f(x_1,\ldots,x_k)$ of their inputs using the minimal communication possible. Initially, players share a great deal of information: each PLR_i sees every input *except* x_i.[1] Note that a great deal of information is shared before communication begins; namely, all players except PLR_i see x_i. As a result, for many functions little communication is needed. Precisely how this shared information affects how much communication is needed is not currently well understood, even when limiting how players may communicate. We consider two well-studied models of communication. In the *one-way* communication model, players each send exactly one message in order (i.e., first PLR_1 sends his message, then PLR_2, etc.) In the *simultaneous-message* (or SM) model, each player simultaneously sends a single message to a referee, who processes the messages and outputs an answer. We use $\mathrm{D}(f)$ and $\mathrm{D}^{\|}(f)$ to denote the communication complexity of f in the one-way and simultaneous-message models respectively.

To date, no explicit function is known which requires a polynomial amount of communication for $k = \Theta(\text{polylog}\, n)$ players in the SM model. Identifying such a function represents one of the biggest problems in communication complexity. Furthermore, a chain of results [22, 16, 4] showed that such a lower bound would place f outside of the complexity class ACC^0. ACC^0 lies at the frontier of our current understanding of circuit complexity, and until the recent work of Williams [21] it wasn't even known that $\mathsf{NEXP} \nsubseteq \mathsf{ACC}^0$. The Multi-Party Pointer Jumping problem is widely conjectured to require enough communication to place it outside of ACC^0. This motivates our study.

The Pointer Jumping Problem

There are many variants of the pointer jumping problem. Here, we study two: a Boolean version MPJ_k^n, and a non-Boolean version $\widehat{\text{MPJ}}_k^n$. (From now on, we suppress the n to ease notation). We shall formally define these problems in Section 2, but for now, each may

[1] Imagine x_i being written on PLR_i's forehead. Then, PLR_i sees inputs on other players' foreheads, but not his own.

be described as problems on a directed graph that has $k+1$ layers of vertices $L_0, \ldots, L_k$. The first layer L_0 contains a single vertex s_0, and layers $L_1, \ldots, L_{k-1}$ contain n vertices each. In the Boolean version, L_k contains two vertices, while in the non-Boolean version L_k contains n vertices. For inputs, each vertex in each layer except L_k has a single directed edge pointing to some vertex in the next layer. The output is the the unique vertex in L_k reachable from s_0; i.e., the vertex reached by starting at s_0 and "following the pointers" to the kth layer. Note that the output is a single bit for MPJ_k and a $\log n$-bit string for $\widehat{\text{MPJ}}_k$. To make this into a communication game, we place on PLR_i's forehead all edges from vertices in L_{i-1} to vertices in L_i. If players speak in any order except $\text{PLR}_1, \cdots, \text{PLR}_k$, there is an easy $O(\log n)$-bit protocol for MPJ_k.

This problem was first studied by Wigderson,[2] who gave an $\Omega(\sqrt{n})$ lower bound for MPJ_3. This was later extended by Viola and Wigderson [20], who showed that MPJ_k requires $\tilde{\Omega}(n^{1/(k-1)})$ communication, even under randomized communication. On the upper-bounds side, Pudlak et al. [19] showed a protocol for MPJ_3 that uses only $O(n(\log\log n)/\log n)$ communication, but only works when the input on PLR_2's forehead is a permutation. Damm et al. [12] show that $D(\widehat{\text{MPJ}}_3) = O(n \log\log n)$ and $D(\widehat{\text{MPJ}}_k) = O(n \log^{(k-1)} n)$, where $\log^{(r)} n$ is the rth iterated log of n. Building on [19], Brody and Chakrabarti [9] showed $D(\text{MPJ}_3) = O\left(n\sqrt{(\log\log n)/\log n}\right)$; they give marginal improvements for MPJ_k for $k > 3$. Several works [19, 12, 15, 10, 9, 8] have shown strong lower bounds for certain restricted classes of protocols motivated by the above upper bounds. Despite the attention devoted to this problem, the upper and lower bounds for general protocols remain far apart, even for $k = 3$ players, where $D(\text{MPJ}_3) = \Omega(\sqrt{n})$ and $D(\text{MPJ}_3) = O(n\sqrt{(\log\log n)/\log n})$. For this reason, in this work we focus on MPJ_k and $\widehat{\text{MPJ}}_k$ for small values of k. We strongly believe that fully understanding the communication complexity of MPJ_3 will shed light on the general problem as well.

1.1 Our Results

We give two collections of results: one for dependent random graphs, and the other for the communication complexity of MPJ_k and $\widehat{\text{MPJ}}_k$. For our work on dependent random graphs, we focus on the clique number and on the chromatic number. The clique number of a graph G, denoted $\text{clique}(G)$, is the size of the largest clique; the chromatic number $\chi(G)$ is the number of colors needed to color the vertices such that the endpoints of each edge have different colors. We use $\text{clique}(G_d(n,p))$ and $\chi(G_d(n,p))$ to refer to $\text{clique}(G)$ and $\chi(G)$ for some $G \sim G_d(n,p)$. We achieve upper and lower bounds for each graph property. Say that a graph property P holds **almost surely (a.s.)** if it holds with probability approaching 1 as n approaches ∞ i.e. if P holds with probability $1 - o(1)$.

Our strongest results[3] give a lower bound for $\text{clique}(G_d(n,p))$ and an upper bound for $\chi(G_d(n,p))$.

▶ **Theorem 1.** *If $0 < p < 1/4$ and $d/p << \sqrt{n}$, then $G_d(n,p)$ almost surely has a clique of size $\Omega\left(\frac{\log n}{\log 1/p}\right)$.*

▶ **Theorem 2.** *If $3/4 < p < 1$ and $d = n^{o(1)}$ then almost surely $\chi(G_d(n,p)) \leqslant (1+\varepsilon)\frac{-n\log(1-p)}{\log n}$.*

[2] This was unpublished, but an exposition appears in [3].
[3] Our choice of p is motivated by what was needed to obtain the communication complexity bounds for MPJ_k. We suspect that tweaking our technical lemmas will give bounds for any constant p.

These bounds nearly match similar results for Erdős-Rényi random graphs. Our bounds on the other side are not as tight.

▶ **Theorem 3.** *If* $0 < p < 1$ *and* $d \leqslant n/\log^2 n$*, then almost surely* $\text{clique}(G_d(n,p)) \leqslant d\log n$.

▶ **Theorem 4.** *If* $0 < p < 1$ *and* $d \leqslant n/\log^2 n$*, then almost surely* $\chi(G_d(n,p)) \geqslant n/(d\log n)$.

For large values of d, there are wide gaps in the upper and lower bounds of clique number and chromatic number. Are these gaps necessary? The existing bounds for random graphs show that Theorems 1 and 2 are close to optimal. Our next result witnesses the tightness for $\text{clique}(G_d(n,p))$.

▶ **Lemma 5.** *For any* $d = o(n)$ *and any* $0 < p < 1$
1. *there are* d*-dependent random graphs that almost surely contain cliques of size* $\Omega(d)$.
2. *there are* d*-dependent random graphs that almost surely contain cliques of size* $\Omega(\sqrt{d}\log n)$.

This result shows that Theorem 3 is also close to optimal. It also demonstrates that tight concentration of measure does not generally hold for dependent random graphs, even for small values of d. Nevertheless, we expect that for many specific dependent random graphs, tight concentration of measure results will hold. Finally, we give two simple constructions which show that with too much dependence, very little can be said about $\text{clique}(G_d(n,p))$.

▶ **Lemma 6.** *For any* $d \geqslant 2n$*, the following statements hold.*
1. *For any* $0 < p < 1$*, there exists a* d*-dependent random graph* $G_d(n,p)$ *that is bipartite with certainty.*
2. *For any* $1/2 \leqslant p < 1$*, there exists a* d*-dependent random graph* $G_d(n,p)$ *that contains a clique of size* $n/2$ *with certainty.*

Results for Multi-Party Pointer Jumping

Our main NOF communication complexity result is a new protocol for MPJ_3.

▶ **Theorem 7.** $\text{D}(\text{MPJ}_3) = O(n(\log\log n)/\log n)$.

This is the first improvement in the communication complexity of MPJ since the work of Brody and Chakrabarti [9]. Next, we use this protocol to get new bounds for the non-Boolean version.

▶ **Theorem 8.** $\text{D}(\widehat{\text{MPJ}}_4) = O\left(n\frac{(\log\log n)^2}{\log n}\right)$.

Our protocol for $\widehat{\text{MPJ}}_4$ is the first sublinear-cost protocol for $\widehat{\text{MPJ}}_k$ for any value of k and improves on the protocol of Damm et al. [12] which has stood for nearly twenty years.[4] Our last pointer jumping results give upper bounds in the SM setting. First we show how to convert our protocol from Theorem 7 to a simultaneous messages protocol.

▶ **Lemma 9.** $\text{D}^{\|}(\text{MPJ}_3) = O\left(n\frac{\log\log n}{\log n}\right)$.

[4] The $\widehat{\text{MPJ}}_k$ protocol of Damm et al uses $O(n\log\log\cdots\log n)$ communication, with $k-1$ logs and is quite simple. It works by having players send an increasing number of bits of information about each possible pointer; their main result is a strong lower bound for *conservative* protocols, in which each player i has access only to $f_1 \circ \cdots \circ f_{i-1}$ instead of $f_1, \ldots, f_{i-1}$. It is worth noting that their protocol is conservative. Our protocols are not conservative.

Note that to solve $\widehat{\text{MPJ}}_3$, players can compute each bit of $f_3(f_2(i))$ using an MPJ_3 protocol. By running $\log n$ instances in parallel, players compute all of $\widehat{\text{MPJ}}_3(i, f_2, f_3)$. Thus, we get the following bound for $\widehat{\text{MPJ}}_3$.

▶ **Corollary 10.** $\mathrm{D}^{\|}(\widehat{\text{MPJ}}_3) = O(n \log\log n)$.

This matches the bound from [12] but holds in the more restrictive SM setting.

1.2 Obtaining Bounds for Dependent Random Graph Properties

In this subsection, we describe the technical hook we obtained to prove our bounds for Theorems 1 and 2. A key piece of intuition is that when looking at only small subgraphs of $G \sim G_d(n,p)$, the subgraph usually *looks* like $G(n,p)$. This intuition is formalized in the following definition and lemma.

▶ **Definition 11.** Given a dependent random graph $G_d(n,p)$, call a subset of vertices $S \subseteq V$ UNCORRELATED if any two edges in the subgraph induced by S are independent.

▶ **Lemma 12.** *Suppose d and k are integers such that $dk^3 \leqslant n$. Fix any d-dependent graph $G_d(n,p)$, and let S be a set of k vertices uniformly chosen from V. Then, we have*

$$\Pr[S \textit{ is } \text{UNCORRELATED}] \geqslant 1 - \frac{3dk^3}{2n} .$$

At first glance, it might appear like we are now able to appeal to the existing arguments for obtaining bounds for clique$(G(n,p))$ and then $\chi(G(n,p))$. Unfortunately, this is not the case – while most potential k-cliques are UNCORRELATED, allowing correlation between edges drives up the *variance*. In effect, we might expect to have roughly the same number of k-cliques, but these cliques bunch together. Nevertheless, we are able to show that when d is small enough, these cliques don't bunch up too much. Appropriately bounding the variance is the most technically involved hurdle in this work, and is necessary to obtain both the upper bound on the chromatic number, and the efficient pointer jumping protocol. We leave details to Section 5.

1.3 Road Map

The rest of the paper is organized as follows. In Section 2 we specify some notation, give formal definitions for the problems and models we consider, and provide some technical lemmas on probability which we'll need in later sections. We develop our results for dependent random graphs in Section 3, deferring some technical lemmas to Section 5. We present our main result for MPJ_3 in Section 4, deferring the secondary MPJ_k results to Section 7. In Section 6 we prove Lemmas 5 and 6.

2 Preliminaries and Notation

We use $[n]$ to denote the set $\{1, \ldots, n\}$, N to denote $\binom{n}{2}$, and $\exp(z)$ to denote e^z. For a string $x \in \{0,1\}^n$, let $x[j]$ denote the jth bit of x. For a sequence of random variables $X_0, X_1, \ldots$, we use $\mathbf{X}_i$ to denote the subsequence $X_0, \ldots, X_i$. For a graph $G = (V, E)$, $\bar{G}$ denotes the complement of G. Given sets $A \subset B \subset V$, we use $B \backslash A$ to denote the set of edges $\{(u,v) : u, v \in B \text{ and } \{u,v\} \nsubseteq A\}$.

For a communication problem, we refer to players as $\text{PLR}_1, \ldots, \text{PLR}_k$. When $k = 3$, we anthropomorphize players as Alice, Bob, and Carol. Our communication complexity measures were defined in Section 1; for an in-depth development of communication complexity, consult the excellent standard textbook of Kushilevitz and Nisan [18].

2.1 Probability Theory and Random Graphs

Next, we formalize our notion of dependent random graphs and describe the tools we use to bound clique($G_d(n,p)$).

▶ **Definition 13** ([13], Definition 5.3). A sequence of random variables $Y_0, Y_1, \ldots, Y_n$ is a *martingale* with respect to another sequence $X_0, X_1, \ldots, X_n$ if for all $i \geqslant 0$ we have

$$Y_i = g_i(\mathbf{X}_i)$$

for some functions $\{g_i\}$ and, for all $i \geqslant 1$ we have

$$E[Y_i|\mathbf{X}_{i-1}] = Y_{i-1} \ .$$

▶ **Theorem 14** (Azuma's Inequality). *Let $Y_0, \ldots, Y_n$ be a martingale with respect to $X_0, \ldots, X_n$ such that $a_i \leqslant Y_i - Y_{i-1} \leqslant b_i$ for all $i \geqslant 1$. Then*

$$\Pr[Y_n > Y_0 + t], \Pr[Y_n < Y_0 - t] \leqslant \exp\left(-\frac{2t^2}{\sum_i (b_i - a_i)^2}\right) \ .$$

Of particular relevance for our work is the *edge-exposure martingale.* Let G be a random graph. Arbitrarily order possible edges of the graph $e_1, \ldots, e_N$, and let X_i be the indicator variable for the event that $e_i \in G$. Let $f : \binom{n}{2} \to \mathbb{R}$ be any function on the edge set, and let $Y_i := E[f(X_1, \ldots, X_N)|\mathbf{X}_i]$. It is easy to verify that for any f, $E[Y_i|\mathbf{X}_{<i}] = Y_{i-1}$, and therefore $\{Y_i\}$ are a martingale with respect to $\{X_i\}$. We say that $\{Y_i\}$ is the edge-exposure martingale for G.

It is worth noting that martingales make no assumptions about the independence of $\{X_i\}$. We'll use martingales on graph distributions where each edges may depend on a small number of other edges. This notion of *local dependency* is formalized below.

A *dependency graph* for a set of random variables $X = \{X_1, \ldots, X_N\}$ is a graph H on $[N]$ such that for all i, X_i is independent of $\{X_j : (i,j) \notin H\}$. We say that a set of variables X is d**-locally dependent** if there exists a dependency graph for X where each vertex has degree at most d.

The following lemma of Janson [17] (rephrased in our notation) bounds the probability that the sum of a series of random bits deviates far from its expected value, when the random bits have limited dependence.

▶ **Lemma 15** ([17]). *Let $X = \{X_i\}_{i\in[N]}$ be a d-locally dependent set of identically distributed binary variables, and let $Y = \sum_{i\in[N]} X_i$. Then, for any t we have*

$$\Pr[|Y - \mathbb{E}[Y]| \geqslant t] \leqslant e^{\frac{-2t^2}{(d+1)N}} \ .$$

For more details and results on probability and concentration of measure, consult the textbook of Dubhashi and Panconesi [13].

▶ **Definition 16.** A distribution $G_d(n,p)$ is *d-dependent* if each edge is placed in the graph with probability p, and furthermore that the set of edges are d-locally dependent.

Note that taking $d = 0$ gives the standard Erdős-Rényi graph model. As with k-wise independent random graphs, d-dependent random graphs are actually a family of graph distributions. We make no assumptions on the underlying distribution beyond the fact that each edge depends on at most d other edges. We abuse notation somewhat and let $G_d(n,p)$ denote both the family of d-dependent random graph distributions as well as an arbitrary

d-dependent random graph distribution. Unless specified otherwise (by stating e.g. "there exists a d-dependent random graph distribution…") all of our results apply to an arbitrary d-dependent random graph distribution. Following convention in the random graph literature, we use "a depedent random graph" to denote a random variable distributed according to a d-dependent random graph distribution.

A clique in a graph $G = (V, E)$ is a set of vertices S such that the subgraph induced on S is complete. Similarly, an independent set T is a set of vertices whose induced subgraph is empty. A *clique cover* of G is a partition of V into cliques. We let clique(G) denote the size of the largest clique in G. Let $\chi(G)$ denote the chromatic number of G; i.e., the minimum number of colors needed to color the vertex set such that no two adjacent vertices are colored the same. Note that $\chi(G)$ is the size of the smallest clique cover of $\bar{G}$.

2.2 Multi-Party Pointer Jumping

Finally, we formally define the Boolean Multi-Party Pointer Jumping function. Let $i \in [n]$, and let $f_2, \ldots, f_k : [n]^n$, be functions from $[n]$ to $[n]$. Let $x \in \{0,1\}^n$. We define the k-player pointer jumping function $\textsc{mpj}_k^n : [n] \times ([n]^n)^{k-2} \times \{0,1\}^n$ recursively as follows:

$$\textsc{mpj}_3^n(i, f_2, x) := x[f_2(i)] \ ,$$
$$\textsc{mpj}_k^n(i, f_2, \ldots, f_{k-1}, x) := \textsc{mpj}_{k-1}^n(f_2(i), f_3, \ldots, f_{k-1}, x) \ .$$

The non-Boolean version $\widehat{\textsc{mpj}}_k^n : [n] \times ([n]^n)^{k-1}$ is defined similarly recursively:

$$\widehat{\textsc{mpj}}_3^n(i, f_2, f_3) := f_3(f_2(i)) \ ,$$
$$\widehat{\textsc{mpj}}_k^n(i, f_2, \ldots, f_k) := \widehat{\textsc{mpj}}_{k-1}^n(f_2(i), f_3, \ldots, f_k) \ .$$

Henceforth, we drop the superscript n to ease notation. Each problem is turned into a communication game in the natural way. $\textsc{plr}_1$ is given i; for each $2 \leqslant j < k$, $\textsc{plr}_j$ receives f_j, and $\textsc{plr}_k$ receives x. Players must communicate to output $\textsc{mpj}_k(i, f_2, \ldots, f_{k-1}, x)$.

3 Dependent Random Graphs

In this section, we prove our main results regarding dependent random graphs, namely that with high probability they contain a large clique, and with high probability the chromatic number is not too large. The two theorems are formally stated below.

▶ **Theorem 17** (Formal Restatement of Theorem 1). *For all $0 < \varepsilon < 1/4$ there exists n_0 such that*

$$\Pr[\text{clique}(G_d(n,p)) > k] > 1 - \exp(-n^{1+\varepsilon}) \ ,$$

for all $n \geqslant n_0$, for all $n^{-\varepsilon/4} < p < \frac{1}{4}$ and for all d, k such that $k \leqslant \frac{\log(n/(2d \log^3 n))}{\log(1/p)}$ and $d/p \leqslant n^{1/2-\varepsilon}$.

This theorem shows clique$(G_d(n,p)) = \Omega\left(\frac{\log n}{\log 1/p}\right)$ with high probability, as long as d/p is bounded away from $\sqrt{n}$. Furthermore, when $d = n^{o(1)}$, clique$(G_d(n,p)) \geqslant (1-\varepsilon)\frac{\log n}{\log(1/p)}$ with high probability.

Proof. This proof follows the classic technique of Bollobás [5], modified to handle dependent random graphs. We need to show that $G_d(n,p)$ contains clique of size k. To that end, let Y be the largest number of *edge-disjoint* UNCORRELATED k-cliques. First, we give a lower bound on $E[Y]$; we defer its proof to Section 5.

▶ **Lemma 18.** $\mathbb{E}[Y] \geqslant \frac{n^2 p}{19k^5}$.

Now, we use the edge-exposure martingale on $G_d(n,p)$ to show that with high probability, Y does not stray far from its expectation. Let $Y_0, Y_1, \cdots Y_N$, be the edge exposure martingale on $G_d(n,p)$. Recall that $Y_0 = E[Y], Y_N = Y$, and $Y_i = E[Y|\mathbf{X}_i]$. In a standard random graph model where all edges are independently placed in G, it is easy to see that conditioning on whether or not an edge is in the graph changes the expected number of *edge-disjoint* UNCORRELATED k-cliques by at most one. This no longer holds when edges are dependent. However, if the graph distribution is d-dependent, then conditioning on X_i changes the expected number of *edge-disjoint* UNCORRELATED k-cliques by at most d. Therefore, $|Y_{i+1} - Y_i| \leqslant d$. Then, by Azuma's inequality, Lemma 18, and our assumption that $d/p \leqslant n^{1/2-\varepsilon}$, we have

$$\begin{aligned}\Pr[Y=0] &\leqslant \Pr[Y - \mathbb{E}[Y] \leqslant -\mathbb{E}[Y]] \\ &\leqslant \exp\left(\frac{-\mathbb{E}[Y]^2}{2Nd^2}\right) \\ &= \exp\left(-\frac{n^2p^2}{19^2 d^2 k^{10}}(1+o(1))\right) \\ &\leqslant \exp(-n^{1+\varepsilon}) \ .\end{aligned}$$

Thus, it follows that $G_d(n,p)$ contains an UNCORRELATED k-clique with probability at least $1 - \exp(-n^{1+\varepsilon})$. Since every UNCORRELATED clique is still a clique, it is clear that

$$\Pr[\text{clique}(G_d(n,p)) \geqslant k] \geqslant 1 - \exp(-n^{1+\varepsilon}) \ .$$

◀

Next, we use the lower bound on clique$(G_d(n,p))$ to obtain an upper bound on $\chi(G_d(n,p))$.

▶ **Theorem 19.** *For all $0 < \varepsilon < 1/8$ there exists n_0 such that*

$$\Pr\left[\chi(G_d(n,q)) < (1+4\varepsilon)\frac{-n\log(1-q)}{\log n}\right] > 1 - \exp(n^{1+\varepsilon}) \ ,$$

for all $3/4 < q < 1 - n^{-\varepsilon/4}$, all $d \leqslant n^{o(1)}$, and all $n \geqslant n_0$.

Proof. This follows a greedy coloring approach similar to [5, 19], but adapted to dependent random graphs. Set $m = \frac{n}{\log^2 n}$, $\varepsilon' = 2\varepsilon$, and $p = 1-q$. Let $\mathcal{E}$ be the event that every induced subgraph H of $G_d(n,q)$ with m vertices has an independent set of size at least $k := (1-\varepsilon')\frac{\log m}{-\log(1-q)}$. Independent sets in $G_d(n,q)$ correspond to cliques in the complement graph $\overline{G_d(n,q)}$, which is distributed identically to $G_d(n,p)$. Thus, we're able to leverage Theorem 17 to bound $\Pr[\mathcal{E}]$. In particular, since $d \leqslant n^{o(1)} \leqslant m^{o(1)}$,[5] by Theorem 17 and a union bound we have

$$\Pr[\mathcal{E}] > 1 - \binom{n}{m}\exp(-n^{1+\varepsilon'}) > 1 - \exp\left(\frac{n}{\log n} - n^{1+\varepsilon'}\right) > 1 - \exp(-n^{1+\varepsilon}) \ .$$

Now, assume $\mathcal{E}$ holds. We iteratively construct a coloring for $G_d(n,q)$. Start with each vertex uncolored. Repeat the following process as long as more than m uncolored vertices remain:

[5] note that $n^\delta = m^{\delta'}$, where $\delta' = \delta\frac{\log n}{\log n - 2\log\log n}$. If $\delta = o(1)$ then $\delta' = o(1)$ as well.

Select m uncolored vertices. From their induced subgraph, identify an independent set I of size at least k. Then, color each vertex in I using a new color. When at most m uncolored vertices remain, color each remaining vertex using a different color. Since two vertices share the same color only if they are in an independent set, it's clear this is a valid coloring. Moreover, for each color in the first phase, we color at least $k > (1-\varepsilon')\frac{\log m}{-\log p} > (1-(3/2)\varepsilon)\frac{\log n}{-\log p}$ vertices. Hence, the overall number of colors used is at most

$$\frac{n-m}{(1-(3/2)\varepsilon')(\log n)/(-\log(1-q))} + m \leqslant (1+4\varepsilon)\frac{-n\log(1-q)}{\log n} .$$

Therefore, $\chi(G_d(n,q)) \leqslant (1+4\varepsilon)\frac{-n\log(1-q)}{\log n}$ as long as $\mathcal{E}$ holds. This completes the proof. ◀

Finally, we give an upper bound on clique$(G_d(n,p))$ and a lower bound on $\chi(G_d(n,p))$, which follow directly from Lemma 15.

▶ **Theorem 20.** *For all* $0 < p < 1$ *and* $d \leqslant n/\log^2 n$*, almost surely* clique$(G_d(n,p)) = O(d\log n)$.

Proof. Let $G \sim G_d(n,p)$, and fix some constant c to be determined later. For a set of vertices $S \subseteq V$ of size $|S| = cd\log n$, let BAD_S denote the event that S is a clique, and let $BAD := \bigvee_S BAD_S$. Note that there are $\binom{n}{cd\log n} \leqslant exp(cd\log^2 n)$ such events. Since G is d-dependent and $S \subset V$, then the subgraph induced by S is also d-dependent. Now, define $z := \binom{cd\log n}{2}$ and let $X_1, \ldots, X_z$ be indicator variables for the edges in the subgraph induced by S. Finally, let $Y := \sum_i X_i$. Then, $E[Y] = pz$, and BAD_S amounts to having $Y = z$. By Lemma 15,

$$\begin{aligned}
\Pr[BAD_S] &= \Pr[Y = z] \\
&= \Pr[Y - E[Y] \geqslant z(1-p)] \\
&\leqslant \exp\left(-\frac{2z^2(1-p)^2}{(d+1)z}\right) \\
&= \exp\left(-\frac{2z(1-p)^2}{d+1}\right) .
\end{aligned}$$

Choosing $c = 1/(1-p)^2$ and using a union bound yields

$$\begin{aligned}
\Pr[BAD] &\leqslant \binom{n}{z}\Pr[BAD_S] \leqslant \exp\left(cd\log^2 n - \frac{2(1-p)^2}{d+1}(cd\log n)^2\right) \\
&= \exp\left(cd\log^2 n(1-2c(1-p)^2)\right) \\
&< \exp(-\Omega(d\log^2 n)) .
\end{aligned}$$

Thus, almost surely $G_d(n,p)$ has no clique of size $\geqslant cd\log n$. ◀

Our lower bound on $\chi(G_d(n,p))$ follows as a direct corollary, since any independent set in $G_d(n,p)$ is a clique in the complement graph $\overline{G_d(n,p)}$, which is also d-dependent.

▶ **Corollary 21.** *If* $0 < p < 1$ *and* $d \leqslant n/\log^2 n$*, then almost surely* $\chi(G_d(n,p)) \geqslant n/(d\log n)$.

4 A New Protocol for MPJ$_3$

Below, we describe a family of MPJ$_3$ protocols $\{\mathcal{P}_H\}$ parameterized by a bipartite graph $H = (A \cup B, E)$ with $|A| = |B| = n$. In each protocol $\mathcal{P}_H$, Alice and Bob each independently send a single message to Carol, who must take the messages and the input she sees and output $\textsc{mpj}_3(i, f, x)$. Bob's communication in each protocol is simple: given i, he sends x_j for each j such that $(i, j) \in H$. Alice's message is more involved. Given H and f, she partitions $[n]$ into *clusters*. For each cluster in the partition, she sends the XOR of the bits for x. (e.g. if one cluster is $\{1, 3, 5\}$, then Alice would send $x[1] \oplus x[3] \oplus x[5]$) This partition of $[n]$ into clusters is carefully chosen and depends on H and f. Crucially, it is possible to make this partition so that for any inputs i, f, Bob sends $x[j]$ for each j in the cluster containing $f(i)$, except for possibly $x[f(i)]$ itself. We formalize this clustering below. Thus, Carol can compute $x[f(i)]$ by taking the relevant cluster from Alice's message and "XOR-ing out" the irrelevant bits using portions of Bob's message.

Each protocol $\mathcal{P}_H$ will correctly compute $\textsc{mpj}_3(i, f, x)$; we then use the probabilistic method to show that there exists a graph H such that $\mathcal{P}_H$ is *efficient*. At the heart of this probabilistic analysis is a bound on the chromatic number of a dependent random graph. For functions with large preimages, this dependency becomes too great to handle.

▶ **Definition 22.** A function $f : [n] \to [n]$ is *d-limited* if $|f^{-1}(j)| \leqslant d$ for all $j \in [n]$.

We end up with a protocol $\mathcal{P}_H$ that is efficient for all inputs (i, f, x) as long as f is d-limited ($d \approx \log n$ suffices); later, we generalize $\mathcal{P}_H$ to work for all inputs.

▶ Remark. This construction is inspired by the construction of Pudlák et al. [19], who gave a protocol for $\textsc{mpj}_3$ that works in the special case that the middle layer is a *permutation* π instead of a general function f. They also use the probabilistic method to show that one $\mathcal{P}_H$ must be efficient. The probabilistic method argument in our case depends on the chromatic number of a dependent random graph; the analysis of the permutation-based protocol in [19] relied on the chromatic number of the standard random graph $G(n, p)$.

Description of $\mathcal{P}_H$

Let $H = (A \cup B, E)$ be a bipartite graph with $|A| = |B| = n$. Given H and f, define a graph $G_{f,H}$ by placing $(i, j) \in G_{f,H}$ if and only if both $(i, f(j))$ and $(j, f(i))$ are in H. Let $C_1, \ldots, C_k$ be a clique cover of $G_{f,H}$, and for each $1 \leqslant \ell \leqslant k$, let $S_\ell := \{f(j) : j \in C_\ell\}$.

The protocol $\mathcal{P}_H$ proceeds as follows. Given f and x, Alice constructs $G_{f,H}$. For each clique C_ℓ, Alice sends $b_\ell := \bigoplus_{j \in S_\ell} x[j]$. Bob, given i and x, sends $x[j]$ for all $(i, j) \in H$. We claim these messages enable Carol to recover $\textsc{mpj}_3(i, f, x)$. Indeed, given i and f, Carol computes $G_{f,H}$. Let C be the clique in the clique cover of $G_{f,H}$ containing i, and let $S := \{f(j) : j \in C\}$ and $b := \bigoplus_{j \in S} x[j]$. Note that Alice sends b. Also note that for any $j \neq i \in C$, there is an edge $(i, j) \in G_{f,H}$. By construction, this means that $(i, f(j)) \in H$, so Bob sends $x[f(j)]$. Thus, Carol computes $x[f(i)]$ by taking b (which Alice sends) and "XOR-ing out" $x[f(j)]$ for any $j \neq i \in C$. In this way, $\mathcal{P}_H$ computes $\textsc{mpj}_3$.

While $\mathcal{P}_H$ computes $\textsc{mpj}_3$, it might not do so in a communication-efficient manner. The following lemma shows that there is an efficient protocol whenever f has small preimages.

▶ **Lemma 23.** *For any $d \leqslant n^{o(1)}$, there exists a bipartite graph H such that for all $i \in [n], x \in \{0,1\}^n$, and all d-limited functions f, we have*

$$\operatorname{cost}(\mathcal{P}_H) = O\left(n \frac{\log\log n}{\log n}\right) .$$

Before proving Lemma 23, let us see how this gives the general upper bound.

▶ **Theorem 24** (Restatement of Theorem 7)**.** $\mathrm{D}(\textsc{mpj}_3) = O(n(\log\log n)/\log n)$.

Proof. Fix $d = \log n$ and let $\mathcal{P}_H$ be the protocol guaranteed by Lemma 23. We construct a general protocol $\mathcal{P}$ for $\textsc{mpj}_3$ as follows. Given f, Alice and Carol select a d-limited function g such that $g(j) = f(j)$ for all j such that $|f^{-1}(f(j))| \leqslant d$. Note that Alice and Carol can do this without communication, by selecting (say) the lexicographically least such g. On input (i, f, x), Alice sends the message she would have sent in $\mathcal{P}_H$ on input (i, g, x), along with $x[j]$ for all j with large preimages. Bob merely sends the message he would have sent in $\mathcal{P}_H$. If the preimage of $f(i)$ is large, then Carol recovers $x[f(i)]$ directly from the second part of Alice's message. Otherwise, Carol computes $\textsc{mpj}_3(i, g, x)$ using $\mathcal{P}_H$. Since $f(i)$ has a small preimage, we know that $x[g(i)] = x[f(i)] = \textsc{mpj}_3(i, f, x)$, so in either case Carol recovers $\textsc{mpj}_3(i, f, x)$.

The communication cost of $\mathcal{P}$ is the cost of $\mathcal{P}_H$, plus one bit for each j with preimage $|f^{-1}(j)| > d$. There are at most n/d such j. With $d = \log n$ and using Lemma 23, the cost of $\mathcal{P}$ is

$$\mathrm{cost}(\mathcal{P}) \leqslant \mathrm{cost}(\mathcal{P}_H) + n/d = O(n(\log\log n)/\log n) + O(n/\log n) = O(n(\log\log n)/\log n) \,. \blacktriangleleft$$

Proof of Lemma 23. We use the Probabilistic Method. Place each edge in H independently with probability $p = \Theta\left(\frac{\log\log n}{\log n}\right)$. Now, for any d-limited function f, consider the graph $G_{f,H}$. Each edge (i, j) is in $G_{f,H}$ with probability p^2, but the edges are not independent. However, we claim that if f is d-limited, then $G_{f,H}$ is $(2d-2)$-dependent. To see this, note that (i, j) is in $G_{f,H}$ if both $(i, f(j))$ and $(j, f(i))$ are in H. Therefore, (i, j) is dependent on (i) any edge (i, j') such that $f(j') = f(j)$, and (ii) any edge (i', j) such that $f(i) = f(i')$. Since f is d-limited, there are at most $d-1$ choices each for i' and j'. Thus, each edge depends on at most $2d-2$ other edges, and $G_{f,H}$ is $(2d-2)$-dependent.

In $\mathcal{P}_H$, Alice sends one bit per clique in the clique cover of $G_{f,H}$. Bob sends one bit for each neighbor of i in H. Thus, we'd like a graph H such that every $i \in [n]$ has a few neighbors and every d-limited function f has a small clique cover.

Let BAD_i denote the event that i has more than $2pn$ neighbors in H. By a standard Chernoff bound argument, $\Pr[BAD_i] \leqslant \exp(-np^2/2)$. Next, let BAD_f be the event that at least $(1+\varepsilon)\frac{-n\log(p^2)}{\log n}$ cliques are needed to cover the vertices in $G_{f,H}$. Note that any clique in $G_{f,H}$ is an independent set in the complement graph $\overline{G_{f,H}}$, so the clique cover number of $G_{f,H}$ equals the chromatic number of $\overline{G_{f,H}}$. Also note that $\overline{G_{f,H}}$ is itself a d-dependent random graph, with edge probability $q = 1-p^2$. Therefore, by Theorem 19, $\Pr[BAD_f] < \exp(-n^{1+\varepsilon})$. Finally, let $BAD := (\bigvee_i BAD_i) \bigvee \left(\bigvee_{\text{d-limited } f} BAD_f\right)$. There are n indices i and at most $n^n \leqslant \exp(n\log n)$ d-limited functions f. Therefore, by a union bound we have

$$\Pr[BAD] < n\Pr[BAD_i] + n^n \Pr[BAD_f] < ne^{-\frac{np^2}{2}} + n^n e^{-n^{1+\varepsilon}} < 1.$$

Therefore, there exists a good H. Also note that in $\mathcal{P}_H$ for a good H, Alice and Bob each communicate $O(n\frac{\log\log n}{\log n})$ bits. This completes the proof. $\blacktriangleleft$

Simultaneous Messages

We conclude this section by showing how to convert $\mathcal{P}_H$ into an SM protocol. Observe that Carol selects a bit from Alice's message (namely, the clique containing $f(i)$) and a few bits from Bob's message (the neighbors of i in H) and $\textsc{xor}$s them together. To convert $\mathcal{P}_H$ to an SM protocol, Alice and Bob send the same messages as in $\mathcal{P}_H$. Carol, given i and f, sends a bit mask describing which bit from Alice's message and which bits from Bob's message are relevant. The Referee then $\textsc{xor}$s these bits together, again producing $\textsc{mpj}_3(i, f, x)$. Carol sends one bit for each bit of communication sent by Alice and Bob. Thus, this SM protocol costs twice as much as the cost of $\mathcal{P}_H$. We get the following result.

▶ **Lemma 25** (Restatement of Lemma 9). $\mathrm{D}^{\|}(\textsc{mpj}_3) = O(n\frac{\log\log n}{\log n})$.

5 Proofs of Main Technical Lemmas

In this section, we state and prove three technical lemmas which form key insights to our contribution. The first lemma states that most sets of k vertices "look independent". The second bounds the expected number of *intersecting* k-cliques. The final lemma gives a lower bound on the expected number of *disjoint* UNCORRELATED k-cliques.

We remind the reader that all three lemmas apply to arbitrary d-dependent random graph distributions.

▶ **Lemma 26** (Restatement of Lemma 12). *Suppose d and k are integers such that $dk^3 \leqslant n$. Fix any d-dependent graph $G_d(n,p)$, and let S be a set of k vertices uniformly chosen from V. Then, we have*

$$\Pr[S \textit{ is } \textsc{uncorrelated}] \geqslant 1 - \frac{3dk^3}{2n} \ .$$

Proof. We divide the possible conflicts into two classes, bound the probability of each, and use a union bound. Say that correlated edges are *local* if they share a vertex. Otherwise, call them *remote*. Let $\mathcal{L}$ and $\mathcal{R}$ be the events that S contains a local and remote dependency respectively.

First, we bound $\Pr[\mathcal{R}]$. Imagine building S by picking vertices $v_1, \ldots, v_k$ one at a time uniformly. Let $S_i := \{v_1, \ldots, v_i\}$, and let B_i be the the set of vertices that would create a remote dependency if added to S_i. Note that $B_1 = \varnothing$ since there are no edges in S_1 (it contains only one vertex). More importantly, for $i > 1$, there are at most $\binom{i}{2} \cdot (2d) < di^2$ vertices in B_i, because S_i contains $\binom{i}{2}$ edges; each edge depends on at most d other edges, and each of these edges contributes at most two vertices to B_i. It follows that $\mathcal{R}$ is avoided if $v_{i+1} \notin B_i$ for each $i = 2 \ldots k-1$. There are $(n-i)$ choices for v_{i+1}, so

$$\Pr[\neg\mathcal{R}] \geqslant \prod_{i=2}^{k-1}\left(1 - \frac{di^2}{n-i}\right) \geqslant \left(1 - \frac{dk^2}{n-k}\right)^{k-2} \geqslant 1 - \frac{dk^3}{n} \ ,$$

Hence $\Pr[\mathcal{R}] \leqslant dk^3/n$. At first glance, it might appear like we've handled local dependencies as well. However, it is possible that when adding v_i, we add local dependent edges, if these edges are both adjacent to v_i. Thus, we handle this case separately.

Let $\mathcal{L}_{ij}$ denote the event that $i, j \in S$ and there are no local dependencies in S involving (i,j). Call a vertex ℓ bad for (i,j) if either (i,ℓ) or (j,ℓ) depend on (i,j). There are at most d bad vertices for (i,j). Note that $\Pr[i,j \in S] = \binom{n-2}{k-2}/\binom{n}{k} = k(k-1)/n(n-1)$ and that

$$\begin{aligned}
\Pr[\neg\mathcal{L}_{ij} | i,j \in S] &\geqslant \binom{n-2-d}{k-2}\Big/\binom{n-2}{k-2} \\
&\geqslant \prod_{z=0}^{d-1}\left(1 - \frac{k-2}{n-2-z}\right) \\
&\geqslant \left(1 - \frac{k-2}{n-2-d}\right)^d \\
&\geqslant 1 - \frac{d(k-2)}{n-2-d} \\
&\geqslant 1 - \frac{dk}{n} \ .
\end{aligned}$$

It follows that $\Pr[\mathcal{L}_{ij}] = \Pr[i,j \in S]\Pr[\mathcal{L}_{ij}|i,j \in S] \leqslant \frac{k(k-1)}{n(n-1)} \cdot \frac{dk}{n}$. There are $\binom{n}{2}$ possible pairs i,j, so by a union bound, we have $\Pr[\mathcal{L}] \leqslant \frac{n(n-1)}{2}\frac{k(k-1)}{n(n-1)}\frac{dk}{n} \leqslant \frac{dk^3}{2n}$. Another union bound on $\mathcal{R}$ and $\mathcal{L}$ completes the lemma. ◀

▶ **Lemma 27.** *Let d,p,k be such that $k < \frac{\log(n/(2d\log^3 n))}{\log 1/p}$. Fix a d-dependent random graph distribution $G_d(n,p)$. Let $G \sim G_d(n,p)$, and let W be the set of ordered pairs (S,T) such that S,T are intersecting* UNCORRELATED *k-cliques. Then,*

$$E[|W|] \leqslant 2k\binom{n}{k}p^{2\binom{k}{2}-1}\binom{k}{2}\binom{n}{2} .$$

Note: To understand the relationship between d,k,p,n, it is helpful to consider the case $d = n^{o(1)}$. In this setting, the lemma holds as long as $k \leqslant (1-o(1))\frac{\log n}{\log 1/p}$.

Proof. Let S,T be arbitrary sets of k vertices, and let $X = S \cap T$. We calculate $E[|W|]$ by iterating over all possible values of S,X and for each pair, counting the expected number of T such that $S \cap T = X$ and S,T are both k-cliques. For S,X, let $F(S,X)$ be the expected number of UNCORRELATED k-cliques T such that $S \cap T = X$, conditioned on S being a k-clique. Also let $F(\ell)$ be the maximum of all $F(S,X)$, taken over all S and all $X \subset S$ with $|X| = \ell$. We have

$$E[|W|] = \sum_S \Pr[S \text{ is k-clique}] \sum_{X\subset S}\sum_{T:S\cap T=X} \Pr[T \text{ is k-clique}|S \text{ is k-clique}] \tag{1}$$

$$= \sum_S p^{\binom{k}{2}} \sum_{X\subset S} F(S,X) \tag{2}$$

$$\leqslant \sum_S p^{\binom{k}{2}} \sum_{\ell=2}^{k-1} \sum_{\substack{X\subset S\\|X|=\ell}} F(\ell) \tag{3}$$

$$\leqslant \binom{n}{k}p^{\binom{k}{2}}\sum_{\ell}\binom{k}{\ell}F(\ell) . \tag{4}$$

Next, we obtain an upper bound on $F(\ell)$. Since we need only an upper bound, we take a very pessimistic approach. Let $M \subset [n]\setminus S$ be the set of vertices adjacent to an edge e that depends on some edge from $S\backslash X$. Each edge in $S\backslash X$ depends on at most d other edges, and there are $\binom{k}{2}-\binom{\ell}{2}$ edges in $S\backslash X$. Therefore, $|M| \leqslant d(\binom{k}{2}-\binom{\ell}{2})$. Now, let $E(M)$ be the set of edges with one endpoint in M and the other endpoint in $M \cup X$. Each of these edges may be correlated with edges in $S\backslash X$, so for any $e \in E(M)$ we assume only $\Pr[e|S \text{ is k-clique}] \leqslant 1$. On the other hand, by construction any edge e *not* in $E(M)$ is independent of S, and therefore $\Pr[e \in G|S \text{ is k-clique}] = p$. Next, we sum over all possible T, grouping by how much T intersects M. Suppose $|T \cap M| = \ell'$ for some $0 \leqslant \ell' \leqslant k-\ell$. Then, T contains $\binom{k}{2}$ edges, $\binom{\ell}{2}$ of these edges have both endpoints in X, and are fixed after conditioning on S being a k-clique. Of the remaining edges, $\ell \cdot \ell' + \binom{\ell'}{2}$ are in $E(M)$; the rest are independent of S. Thus, when $|T \cap M| = \ell'$, then $\Pr[T \text{ is k-clique}|S \text{ is k-clique}] \leqslant p^{\binom{k}{2}-\binom{\ell}{2}-\ell\ell'-\binom{\ell'}{2}}$.

$$\begin{aligned}
F(\ell) &= \sum_{T:S\cap T=X} \Pr[T \text{ is k-clique}|S \text{ is k-clique}] && (5)\\
&= \sum_{\ell'=0}^{k-\ell} \sum_{\substack{T:S\cap T=X\\ |T\cap M|=\ell'}} \Pr[T \text{ is k-clique}|S \text{ is k-clique}] && (6)\\
&\leqslant \sum_{\ell'=0}^{k-\ell} \binom{M}{\ell'}\binom{n-k-M}{k-\ell-\ell'} p^{\binom{k}{2}-\binom{\ell}{2}-\ell\ell'-\binom{\ell'}{2}} && (7)\\
&= p^{\binom{k}{2}-\binom{\ell}{2}} \sum_{\ell'=0}^{k-\ell} F^*(\ell')\ , && (8)
\end{aligned}$$

where $F^*(\ell') := \binom{M}{\ell'}\binom{n-k-M}{k-\ell-\ell'}p^{-\ell\ell'-\binom{\ell'}{2}}$. Next, we show that the summation in Equation (8) telescopes.

▶ **Claim 28.** *If* $k \leqslant \frac{\log\left(\frac{n}{2d\log^3 n}\right)}{\log 1/p}$ *then* $\sum_{\ell'=0}^{k-1} F^*(\ell') \leqslant 2F^*(0)$.

Proof. Fix any $0 \leqslant i < k-\ell$, and consider $F^*(i+1)/F^*(i)$. Using $\binom{a}{b+1}/\binom{a}{b} = \frac{a-b}{b+1}$ and $\binom{a}{b-1}/\binom{a}{b} = \frac{b}{a-b-1}$ and recalling that $M < d\binom{k}{2}$, we have:

$$\begin{aligned}
\frac{F^*(i+1)}{F^*(i)} &= \frac{\binom{M}{i+1}\binom{n-k-M}{k-(i+1)}p^{-\ell(i+1)-(i+1)i/2}}{\binom{M}{i}\binom{n-k-M}{k-i}p^{-\ell i-i(i-1)/2}}\\
&= \frac{M-1}{i+1}\frac{k-i}{n-k-M-k+i}p^{-\ell-i}\\
&\leqslant \frac{dk^2}{2}\frac{k}{n-o(n)}\left(\frac{1}{p}\right)^k\\
&< \frac{dk^3}{n}\left(\frac{1}{p}\right)^k\\
&< \frac{k^3}{2\log^3 n}\\
&< 1/2\ ,
\end{aligned}$$

where the penultimate inequality holds because of our assumption on k, and the final inequality holds because $k < \log n$. We've shown that for all i, $F^*(i+1)/F^*(i) < 1/2$. Hence $F^*(i) < F^*(0)2^{-i}$, and so $\sum_{\ell'} F^*(\ell') \leqslant \sum_{\ell'} F^*(0)2^{-\ell'} \leqslant 2F^*(0)$. ◀

From claim 28, we see that

$$F(\ell) \leqslant p^{\binom{k}{2}-\binom{\ell}{2}} \sum_{\ell'=0}^{k-\ell} F^*(\ell') \leqslant 2p^{\binom{k}{2}-\binom{\ell}{2}}F^*(0) = 2p^{\binom{k}{2}-\binom{\ell}{2}}\binom{n-k-M}{k-\ell}\ .$$

Now, plugging this inequality back into Equation 4, we get

$$E[|W|] \leqslant \binom{n}{k}p^{\binom{k}{2}}\sum_{\ell}\binom{k}{\ell}F(\ell) \leqslant 2\binom{n}{k}p^{\binom{k}{2}}\sum_{\ell}\binom{k}{\ell}p^{\binom{k}{2}-\binom{\ell}{2}}\binom{n-k-M}{k-\ell}\ .$$

Let $G(\ell) := p^{\binom{k}{2}-\binom{\ell}{2}}\binom{k}{\ell}\binom{n-k-M}{k-\ell}$, and for $2 \leqslant \ell < k-1$, let $G^*(\ell) := G(\ell)/G(\ell+1)$. Note that

$$G^*(\ell) = p^{\ell}\frac{\ell+1}{k-\ell}\frac{n-2k-M+\ell+1}{k-\ell}\ .$$

We claim that $G^*(\ell)$ decreases as long as $p < 8/27 - \Omega(1)$. To see this, note that

$$\frac{G^*(\ell)}{G^*(\ell+1)} = p\frac{\ell+1}{\ell} \cdot \left(\frac{k-\ell+1}{k-\ell}\right)^2 \frac{n-2k-M+\ell+1}{n-2k-M+\ell} < p(3/2)^3(1+o(1)) ,$$

where the inequality holds because $(a+1)/a = 1+1/a$ and because $\ell, k-\ell \geqslant 2$ for the range of ℓ we need when calculating $G^*(\ell)$. In a way, saying that $G^*(\ell)$ is decreasing amounts to saying that $G(\ell)$ is convex – once $G(i) \leqslant G(i+1)$, then $G(j) \leqslant G(j+1)$ for all $j > i$. Next, a straightforward calculation using our choice of k shows that $G(k-1) \leqslant G(2)$. Thus, it must be the case that $G(i) \leqslant G(2)$ for all i, and therefore

$$\begin{aligned} E[|W|] &\leqslant 2\binom{n}{k}p^{\binom{k}{2}}kG(2) \\ &= 2k\binom{n}{k}p^{2\binom{k}{2}-1}\binom{k}{2}\binom{n-k-M}{k-2} \\ &< 2k\binom{n}{k}p^{2\binom{k}{2}-1}\binom{k}{2}\binom{n}{k-2} . \end{aligned}$$

This completes the proof of Lemma 27. ◀

Finally, we prove the lemma that in any d-dependent graph distribution, the expected number of *disjoint* UNCORRELATED k-cliques is large. Recall that Y is the maximal number of disjoint UNCORRELATED k-cliques.

▶ **Lemma 29** (Restatement of Lemma 18). $\mathbb{E}[Y] \geqslant \frac{n^2p}{19k^5}$.

Proof. We construct Y probabilistically, by selecting each potential UNCORRELATED k-clique with small probability and removing any pairs of k-cliques that intersect. Let K denote the family of UNCORRELATED k-cliques. By Lemma 12 and our choice of d, a randomly chosen set S of k vertices is UNCORRELATED with probability at least 2/3. By this and our choice of k, we have

$$\mathbb{E}[|K|] \geqslant \frac{2}{3}\binom{n}{k}p^{\binom{k}{2}} .$$

Recall that W is the set of ordered pairs $\{S, T\}$ of UNCORRELATED k-cliques such that $2 \leqslant |S \cap T| < k$. For our argument, we require an upper bound on $\mathbb{E}[|W|]$. In the standard random graph model, if $|S \cap T| = \ell$, then $\Pr[S, T \text{ both k-cliques}] = p^{\binom{k}{2}-\binom{\ell}{2}}$. However, this no longer holds for d-dependent distributions, even if S and T are both UNCORRELATED. This is because while edges in S and T are independent, edges in S but not T may be correlated with edges in T but not S. As an extreme case, suppose all edges in S are independent, but each edge in $S \setminus T$ is completely correlated with an edge in $T \setminus S$. Then, $\Pr[S, T \text{ k-cliques}] = \Pr[S \text{ is k-clique}] = \Pr[T \text{ is k-clique}] = p^{\binom{k}{2}}$. Essentially, allowing edges to be correlated has the potential to drive up the variance on the number of k-cliques, even when these k-cliques are UNCORRELATED. This is perhaps to be expected. Nevertheless, in Lemma 27, we were able to show that when d is small, this increase is not much more than in the standard graph model.

With this claim, we are now able to construct a large set of disjoint UNCORRELATED k-cliques with high probability. Create $K' \subseteq K$ by selecting each uncorrelated $S \in K$ independently with probability

$$\Pr[S \in K'] = \gamma = \frac{1}{12kp^{\binom{k}{2}-1}\binom{k}{2}\binom{n}{k-2}} .$$

Finally, create L from K' by removing each pair $S, T \in K'$ such that $S, T \in W$. By construction, L is a set of edge-disjoint UNCORRELATED k-cliques; furthermore, we have

$$\begin{aligned}
E[|L|] &= \gamma E[|K|] - 2\gamma^2 E[|W|] \\
&\geqslant \frac{2\gamma}{3}\binom{n}{k}p^{\binom{k}{2}} - \frac{2\gamma \cdot 2k\binom{n}{k}p^{2\binom{k}{2}-1}\binom{k}{2}\binom{n}{k-2}}{12kp^{\binom{k}{2}-1}\binom{k}{2}\binom{n}{k-2}} \\
&= \frac{2\gamma}{3}\binom{n}{k}p^{\binom{k}{2}} - \frac{\gamma}{3}\binom{n}{k}p^{\binom{k}{2}} \\
&= \frac{\gamma}{3}\binom{n}{k}p^{\binom{k}{2}} \\
&= \frac{\binom{n}{k}p^{\binom{k}{2}}}{3 \cdot 12kp^{\binom{k}{2}-1}\binom{k}{2}\binom{n}{2}} \\
&\geqslant \frac{\binom{n}{k}}{\binom{n}{k-2}} \frac{p}{36k} \frac{1}{\binom{k}{2}} \\
&\geqslant \frac{p}{18k^3} \frac{\binom{n}{k}}{\binom{n}{k-2}} \\
&= \frac{p}{18k^3} \frac{(n-k-2)(n-k-1)}{k(k-1)} \\
&\geqslant \frac{p}{18k^3} \frac{18n^2}{19k^2} \\
&= \frac{n^2 p}{19k^5} ,
\end{aligned}$$

where the final inequality holds for large enough n. ◀

6 Dependent Graphs with Large Cliques or Large Dependency

In this section, we provide results that witness the tightness of our current bounds. The next lemma shows that there exist dependent random graphs that almost surely contain cliques of size $\Omega(d)$, and others that almost surely have cliques of size $\Omega(\sqrt{d}\log(n))$.

▶ **Lemma 30** (Restatement of Lemma 5). *For all constant $0 < p < 1$ and $d = o(n)$,*

1. *there exists a d-dependent random graph $G_d(n, p)$ such that*

$$\Pr\left[\text{clique}(G_d(n,p)) > \frac{d\sqrt{p}}{2} - d^{\frac{1}{2}}p^{\frac{1}{4}}\right] > 1 - e^{-2n/d} .$$

2. *there exists a d-dependent random graph $G_d(n, p)$ such that almost surely*

$$\text{clique}(G_d(n,p)) = \Omega(\sqrt{d}\log(n)) .$$

Proof. We give two constructions.

For the first result, fix $d' := \frac{d\sqrt{p}}{2} - \sqrt{d\sqrt{p}}$ and $M_1 := 2n/d$. Partition the vertices into M_1 sets $V_1, \ldots, V_{M_1}$ each of size $d/2$. Let $c(i)$ denote the part containing i (we think of i has having color c). Now, let $\{X_{i,c} : i \in V, 1 \leqslant c \leqslant M_1\}$ be a series of i.i.d. random bits with $\Pr[X_{i,c} = 1] = \sqrt{p}$, and place $(i, j) \in G_d(n, p)$ if $X_{i,c(j)} \bigwedge X_{j,c(i)} = 1$. Thus, (i, j) is an edge with probability p. Also note that edges (i, j) and (i', j') are dependent if either $c(i) = c(i')$ or $c(j) = c(j')$. Since there are $d/2$ vertices in each V_ℓ, (i, j) is dependent on at most d other edges and $G_d(n, p)$ is d-dependent.

Now, fix a color c, and let $S_c := \{i : c(i) = c \wedge X_{i,c} = 1\}$. For any $i, j \in S_c$ we have $X_{i,c} = X_{j,c} = 1$ and that $c(i) = c(j) = c$. Therefore, $(i, j) \in G_d(n, p)$ for any $i, j \in S_c$, hence S_c is a clique.

Next, consider $|S_c|$. There are $d/2$ vertices with color c, so $E[|S_c|] = \frac{d\sqrt{p}}{2}$. By the Chernoff bound, $\Pr[|S_c| < d'] < \frac{1}{e}$, so the probability that there is some color c with $|S_c| \geqslant d'$ is at least $1 - e^{-2n/d}$. Therefore, $G_d(n, p)$ almost surely contains a clique of size at least d'.

For the second graph, partition the vertices $[n]$ into $M_2 := n/\sqrt{d}$ subsets $V_1, \ldots, V_{M_2}$, each of size $\sqrt{d}$. Let $c(i)$ be the subset containing i. Let $\{X_{c_1,c_2} : 1 \leqslant c_1, c_2 \leqslant M_2\}$ be a set of independent, identically distributed binary variables with $\Pr[X_{c,c'} = 1] = p$. Now, place edge (i, j) in the graph if $X_{c(i),c(j)} = 1$. In this way, for any V_s, V_t, either all edges between V_s and V_t exist, or none do, and similarly for any V_s, either all edges between vertices in V_s will be in the graph, or none will.

Next, let S be the set of all i such that edges between vertices in V_i are in the graph. Each $i \in S$ with probability p. By standard Chernoff bounds, $|S| \geqslant pM_2/2$ with high probability. Let $M' := pM_2/2$. The construction above induces a new random graph G' on M' vertices where all edges are i.i.d. in G' with probability p. i.e., G' is an Erdős-Rényi random graph on M' vertices. By [7], $\text{clique}(G') \geqslant 2\log(M')/\log(1/p) = \Omega(\log(n)/\log(1/p))$ with high probability. Finally, a clique of size k in G' gives a clique of size $k\sqrt{d}$ in G, hence G contains a clique of size $\Omega(\sqrt{d}\log(n)/\log(1/p))$ with high probability. ◀

Our second result in this section shows that when the dependency factor becomes $\Omega(n)$, essentially nothing can be said about the clique number of dependent random graphs.

▶ **Lemma 31** (Restatement of Lemma 6). *Fix $d := 2n - 2$. Then, the following statements hold.*

1. *For any $0 < p < 1$, there exists a d-dependent random graph $G_d(n, p)$ that is bipartite with certainty.*
2. *For any $1/2 \leqslant p < 1$, there exists a d-dependent random graph $G_d(n, p)$ such that* $\text{clique}(G_d(n, p)) \geqslant n/2$ *with certainty.*

Proof. We again provide two constructions. For the first construction, set $q_1 := 1 - \sqrt{1-p}$, and let $X_1, \ldots, X_n$ be i.i.d. random bits such that $X_i = 1$ with probability q_1. Think of each X_i as being assigned to vertex v_i. Now, place edge $(i, j) \in G_d(n, p)$ iff $X_i \oplus X_j = 1$. Note that $(i, j) \in G_d(n, p)$ with probability $2q(1 - q) = p$. It is easy to see that (i, j) depends on (i', j') only if either $i = i'$ or $j = j'$. There are at most $2(n-1)$ such edges, hence the random graph is d-dependent. Finally, we claim that the graph is bipartite. To see this, suppose for the sake of contradiction that $G_d(n, p)$ contains an odd cycle $(1, 2, \ldots, 2k + 1, 1)$. Without loss of generality, assume that $X_1 = 1$ (the proof is similar if $X_1 = 0$.) Since each edge $(i, i + 1) \in G_d(n, p)$, we must have that $X_2, X_4, \ldots, X_{2k}$ all equal 0, and $X_1, X_3, \ldots, X_{2k+1}$ all equal 1. But then $X_1 = X_{2k+1} = 1$, hence $(1, 2k + 1) \notin G_d(n, p)$. This contradicts the assumption that $(1, 2, \ldots, 2k + 1, 1)$ is a cycle.

We proceed with the second construction in a similar manner. Let $q_2 := \frac{1}{2}(1 - \sqrt{2p - 1})$, and let $X_1, \ldots, X_n$ be i.i.d. random bits with $\Pr[X_i = 1] = q_2$. This time, place $(i, j) \in G_d(n, p)$ iff $X_i = X_j$. Note that (i, j) is an edge with probability $q^2 + (1 - q)^2 = p$. Now, let $S_0 := \{i : X_i = 0\}$ and similarly $S_1 := \{i : X_i = 1\}$. It is easy to see that S_0 and S_1 both cliques in $G_d(n, p)$. One of them must contain at least half the vertices. ◀

7 Results for Non-Boolean Pointer Jumping

In this section, we leverage the protocol for $\textsc{mpj}_3$ to achieve new results for the non-Boolean Pointer Jumping problem $\widehat{\textsc{mpj}}$. Let $\mathcal{Q}$ be the protocol for $\textsc{mpj}_3$ given in Lemma 9. First, we give a protocol for $\widehat{\textsc{mpj}}_3$. The cost matches the upper bound from [12] but has the advantage of working in the Simultaneous Messages model.

▶ **Lemma 32** (Restatement of Lemma 10). *There is an $O(n \log\log n)$-bit SM protocol for $\widehat{\textsc{mpj}}_3$.*

Proof. Run $\mathcal{Q}$ $\log n$ times in parallel, on inputs $(i, f_2, z_1), (i, f_2, z_2), \ldots, (i, f_2, z_{\log n})$, where z_j denotes the jth most significant bit of f_3. This allows the Referee to recover each bit of $f_3(f_2(i)) = \widehat{\textsc{mpj}}(i, f_2, f_3)$. ◀

Next we give a new upper bound for $\widehat{\textsc{mpj}}_4$. As far as we know, this is the first protocol for $\widehat{\textsc{mpj}}_k$ for any k that uses a sublinear amount of communication.

▶ **Theorem 33** (Restatement of Theorem 8). *There is a one-way protocol for $\widehat{\textsc{mpj}}_4$ with cost $O(n\frac{(\log\log n)^2}{\log n})$.*

Proof. Let i, f_2, f_3, f_4 be the inputs to $\widehat{\textsc{mpj}}_4$, and for $1 \leqslant j \leqslant \log n$, let $z_j \in \{0,1\}^n$ be the string obtained by taking the jth most significant bit of each $f_3(w)$ (i.e., $z_j[w]$ is the jth most significant bit of $f_3(w)$.) Fix a parameter k to be determined shortly. $\textsc{plr}_1, \textsc{plr}_2$, and $\textsc{plr}_3$ run $\mathcal{Q}$ on $\{(i, f_2, z_j) : 1 \leqslant j \leqslant k\}$. From this, $\textsc{plr}_3$ learns the first k bits of $f_3(f_2(i))$. She then sends $f_4(z)$ for every $z \in \{0,1\}^{\log n}$ whose k most significant bits match those of $f_3(f_2(i))$. $\textsc{plr}_4$ sees i, f_2, and f_3, computes $z^* := f_3(f_2(i))$, and recovers $f_4(z^*)$ from $\textsc{plr}_3$'s message. Note that there are $n/2^k$ strings that agree on the first k bits, and for each of these strings, $\textsc{plr}_3$ sends $\log n$ bits. Therefore, the cost of this protocol is $k\,\mathrm{cost}(\mathcal{Q}) + n\log(n)/2^k = O\left(kn\frac{\log\log n}{\log n} + n\log(n)2^{-k}\right)$. Setting $k := 2\log\frac{\ln 2 \log n}{\log\log n} = \Theta(\log\log n)$ minimizes the communication cost, giving a protocol with cost $O\left(n\frac{(\log\log n)^2}{\log n}\right)$. ◀

Acknowledgements. This research was supported in part by a grant to Swarthmore College by the Howard Hughes Medical Institute through the Precollege and Undergraduate Science Education Program. The authors gratefully acknowledge the support.

References

1 Noga Alon and Asaf Nussboim. k-wise independent random graphs. In *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 813–822, 2008.
2 Noga Alon and Joel H. Spencer. *The Probabilistic Method.* Wiley-Interscience, New York, NY, 2000.
3 László Babai, Thomas P. Hayes, and Peter G. Kimmel. The cost of the missing bit: Communication complexity with help. *Combinatorica*, 21(4):455–488, 2001.
4 Richard Beigel and Jun Tarui. On ACC. *Comput. Complexity*, 4:350–366, 1994.
5 Béla Bollobás. The chromatic number of random graphs. *Combinatorica*, 8(1):49–55, 1988.
6 Béla Bollobás. *Random graphs.* Springer, 1998.
7 Béla Bollobás and Paul Erdős. Cliques in random graphs. *Mathematical Proceedings of the Cambridge Philosophical Society*, 80:419–427, 11 1976.
8 Joshua Brody. The maximum communication complexity of multi-party pointer jumping. In *Proc. 24th Annual IEEE Conference on Computational Complexity*, pages 379–386, 2009.

9 Joshua Brody and Amit Chakrabarti. Sublinear communication protocols for multi-party pointer jumping and a related lower bound. In *Proc. 25th International Symposium on Theoretical Aspects of Computer Science*, pages 145–156, 2008.

10 Amit Chakrabarti. Lower bounds for multi-player pointer jumping. In *Proc. 22nd Annual IEEE Conference on Computational Complexity*, pages 33–45, 2007.

11 Ashok K. Chandra, Merrick L. Furst, and Richard J. Lipton. Multi-party protocols. In *Proc. 15th Annual ACM Symposium on the Theory of Computing*, pages 94–99, 1983.

12 Carsten Damm, Stasys Jukna, and Jiří Sgall. Some bounds on multiparty communication complexity of pointer jumping. *Comput. Complexity*, 7(2):109–127, 1998. Preliminary version in *Proc. 13th International Symposium on Theoretical Aspects of Computer Science*, pages 643-654, 1996.

13 Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms.* Cambridge University Press, 2009.

14 Paul Erdős and Alfréd Rényi. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

15 Andre Gronemeier. Nof-multiparty information complexity bounds for pointer jumping. In *Proc. 31st International Symposium on Mathematical Foundations of Computer Science*, pages 459–470. Springer, 2006.

16 Johan Håstad and Mikael Goldmann. On the power of small-depth threshold circuits. *Comput. Complexity*, 1:113–129, 1991.

17 Svante Janson. Large deviations for sums of partly dependent random variables. *Random Structures & Algorithms*, 24(3):234–248, 2004.

18 Eyal Kushilevitz and Noam Nisan. *Communication Complexity.* Cambridge University Press, Cambridge, 1997.

19 Pavel Pudlák, Vojtěch Rödl, and Jiří Sgall. Boolean circuits, tensor ranks and communication complexity. *SIAM J. Comput.*, 26(3):605–633, 1997.

20 Emanuele Viola and Avi Wigderson. One-way multi-party communication lower bound for pointer jumping with applications. In *Proc. 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 427–437, 2007.

21 Ryan Williams. Nonuniform acc circuit lower bounds. *J. ACM*, 61(1):32, 2014.

22 Andrew C. Yao. On ACC and threshold circuits. In *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 619–627, 1990.

Weighted Polynomial Approximations: Limits for Learning and Pseudorandomness

Mark Bun* and Thomas Steinke†

Harvard University, School of Engineering and Applied Sciences, USA
`{mbun,tsteinke,}@seas.harvard.edu`

Abstract

Low-degree polynomial approximations to the sign function underlie pseudorandom generators for halfspaces, as well as algorithms for agnostically learning halfspaces. We study the limits of these constructions by proving inapproximability results for the sign function. First, we investigate the derandomization of Chernoff-type concentration inequalities. Schmidt et al. (SIAM J. Discrete Math. 1995) showed that a tail bound of δ can be established for sums of Bernoulli random variables with only $O(\log(1/\delta))$-wise independence. We show that their results are tight up to constant factors. Secondly, the "polynomial regression" algorithm of Kalai et al. (SIAM J. Comput. 2008) shows that halfspaces can be efficiently learned with respect to log-concave distributions on $\mathbb{R}^n$ in the challenging agnostic learning model. The power of this algorithm relies on the fact that under log-concave distributions, halfspaces can be approximated arbitrarily well by low-degree polynomials. In contrast, we exhibit a large class of non-log-concave distributions under which polynomials of any degree cannot approximate the sign function to within arbitrarily low error.

1998 ACM Subject Classification F.2.0 Analysis of Algorithms and Problem Complexity – General

Keywords and phrases Polynomial Approximations, Pseudorandomness, Concentration, Learning Theory, Halfspaces

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.625

1 Introduction

Approximation theory is a classical area of mathematics that studies how well functions can be approximated by simpler ones. It has found many applications in computer science. Most of these applications of approximation theory focus on the approximation of functions by polynomials in the uniform norm (or infinity norm). For instance, *approximate degree*, which captures how well a boolean function can be approximated by low-degree polynomials in the uniform norm, underlies important lower bounds in circuit complexity [6, 7, 66], quantum query complexity [5, 1], and communication complexity [65]. It also underlies state-of-the art algorithms in learning theory [35, 41], streaming [31], and in spectral methods [63].

While it is compelling to study polynomial approximations under the uniform norm, there are scenarios where it is more natural to study *weighted polynomial approximations*, where error is measured in terms of an L_p norm under some distribution. For instance, in agnostic learning, the polynomial regression algorithm of Kalai et al. [35] has guarantees based on how well functions in a concept class of interest can be approximated by low-degree polynomials in L_1 distance.

* Supported by an NDSEG Fellowship and NSF grant CNS-1237235.
† Supported by NSF grant CCF-1116616 and the Lord Rutherford Memorial Research Fellowship.

In this work, we show how ideas from weighted approximation theory can yield tight lower bounds for several problems in theoretical computer science. As our first application, in the area of derandomization, we give a tight characterization of the amount of k-wise independence necessary to establish Chernoff-like concentration inequalities. Second, we establish a strong limitation on the distributions under which halfspaces can be learned using the polynomial regression algorithm of Kalai et al.

1.1 Tail Bounds for Limited Independence

The famous Hoeffding bound [32] implies that if $X \in \{\pm 1\}^n$ is a uniform random variable and $r \in \mathbb{R}^n$ is fixed, then, for all $T \geq 0$,

$$\underset{X}{\mathbb{P}}\left[|X \cdot r| \geq T\right] \leq 2e^{-\frac{T^2}{2||r||_2^2}}.$$

We ask the following question:

For what *pseudorandom* X is the Hoeffding bound true?

More precisely, given T and δ, can we construct a pseudorandom $X \in \{\pm 1\}^n$ such that $\underset{X}{\mathbb{P}}\left[|X \cdot r| \geq T\right] \leq \delta$ for all $r \in \{\pm 1\}^n$?[1] Of particular interest is the parameter regime $\delta = 1/\operatorname{poly}(n)$ and $T = \Theta(||r||_2 \sqrt{\log(1/\delta)})$, which is natural in the context of derandomizing efficient randomized algorithms.[2] The probabilistic method gives a non-constructive proof that there exists such an X which can be sampled with seed length $O(\log(n/\delta))$. The challenge is to give an explicit construction of such an X which can be *efficiently* sampled with a short seed.

This is a very natural pseudorandomness question: Concentration of measure is a fundamental property of independent random variables and one of the key objectives of pseudorandomness research is to replicate such properties for random variables with low entropy. Finding a pseudorandom X exhibiting good concentration is also a relaxation of a more general and well-studied pseudorandomness question, namely that of constructing pseudorandom generators that fool linear threshold functions [21, 48, 28, 22]. This problem can also be viewed as a special case of constructing pseudorandom generators for space-bounded computation [57, 33, 60, 14, 15, 43, 61].

For $\delta = 1/\operatorname{poly}(n)$ and $T = \Theta(\sqrt{n \log(1/\delta)})$, we can construct explicit X that can be sampled with seed length $O(\log^2 n)$ using a variety of methods (including [57, 48]). In particular, it suffices for X to be $O(\log(1/\delta))$-wise independent:

▶ **Theorem 1** (Tail Bound for Limited Independence). *Let $n \geq 1$, $\eta > 0$, and $\delta \in (0,1)$ be given. Let $X \in \{\pm 1\}^n$ be k-wise independent for $k = 2\lceil \eta \log_e(1/\delta)\rceil$. Let $r \in \mathbb{R}^n$ and set $T = e^{(\eta+1)/2\eta}\sqrt{k}\,||r||_2$. Then*

$$\mathbb{P}\left[|X \cdot r| \geq T\right] \leq \delta.$$

Limited independence is a very general and intuitively appealing technique in pseudorandomness. As a tool for derandomization, it has been studied in the contexts of hashing [46, 50], dimensionality reduction [37], random graphs [3], and circuits [4, 13]. A k-wise

[1] For simplicity we restrict our attention to $r \in \{\pm 1\}^n$, instead of arbitrary real-valued r.
[2] Smaller values of δ are also interesting and our results apply in these settings. However, our results are most stark for reasonably large values of δ.

independent $X \in \{\pm 1\}^n$ can be sampled with seed length $O(k \cdot \log n)$ [2], yielding a seed length of $O(\log^2 n)$ for the setting of parameters above.

In this work, we ask whether the tail bound of Theorem 1 for k-wise independence is tight. That is, can we prove stronger tail bounds for k-wise independent X?

▶ **Question 2.** *How much independence is needed for X to satisfy a Hoeffding-like tail bound? That is, what is the minimum $k = k(n, \delta, T)$ for which any k-wise independent $X \in \{\pm 1\}^n$ satisfies*

$$\underset{X}{\mathbb{P}}\left[|X \cdot r| \geq T\right] \leq \delta$$

for all $r \in \{-1, 1\}^n$, where $\cdot$ denotes the inner product.

We remark that limited independence is not the only technique for derandomizing concentration bounds. Another construction which achieves seed length $O(\log n \cdot \log(1/\delta))$ is to sample X from a small-bias space [52]. Very recently, Gopalan et al. [27] constructed a much more sophisticated generator with seed length $\tilde{O}(\log(n/\delta))$, which is nearly optimal.

1.1.1 Our Results

Theorem 1 shows that $k(n, \delta, T) \leq O(\log(1/\delta))$ for $T = O(\sqrt{n \log(1/\delta)})$. In this work, we show that this is essentially tight:

▶ **Theorem 3.** *Let $c > 5$ be a constant and n sufficiently large. For $2^{-n^{o(1)}} \leq \delta \leq \frac{1}{\operatorname{poly}(n)}$ and $T = c\sqrt{n \log(1/\delta)}$, we have*

$$k(n, \delta, T) \geq \Omega\left(\frac{\log(1/\delta)}{\log(c)}\right).$$

This means there exists a k-wise independent distribution $X \in \{\pm 1\}^n$ such that

$$\mathbb{P}\left[\left|\sum_{i \in [n]} X_i\right| \geq T\right] > \delta,$$

for $k = k(n, \delta, T)$, n, δ, and T as above.

The only previous lower bound was

$$k(n, \delta, T) \geq \Omega\left(\frac{\log(1/\delta)}{\log n}\right),$$

which holds for any $T \leq n$ and is due to [64]. This is meaningful when $\delta < n^{-\omega(1)}$, but the lower bound is trivial for $\delta = 1/\operatorname{poly}(n)$. Thus our lower bound closes a large gap for the $\delta = 1/\operatorname{poly}(n)$ regime, which is of considerable interest [29, 48, 27].

The lower bound of [64] follows immediately from the fact that a random variable X that can be sampled with seed length s cannot satisfy a nontrivial tail bound with $\delta < 2^{-s}$, and that there exist k-wise independent distributions that can be sampled with seed length $s \leq O(k \cdot \log n)$. Indeed this lower bound holds for all distributions with small seed length and is not specific to k-wise independence.

The most natural way to prove Theorem 3 would be to construct a family of k-wise independent distributions that do not satisfy the required tail bound. However, we instead study the *dual* formulation of the problem (following [4, 20, 9]) and then use lower bound techniques from approximation theory. To the best of our knowledge, this indirect approach is novel for proving impossibility results for k-wise independence. Our results imply the existence of k-wise independent distributions with poor tail bounds, but give no immediate indication as to how to construct them!

We now describe the proof idea in slightly more detail. The answer to Question 2 can be posed in terms of the value of a certain linear program. The variables represent the probability distribution of the random variable X and the constraints force X to be k-wise independent. The objective of the linear program is maximize the tail probability $\mathbb{P}\left[|X \cdot r| \geq T\right]$. Thus, the value of the program is at most δ if and only if $k \geq k(n, \delta, T)$. Taking the dual of this linear program and appealing to strong duality yields an alternative characterization of $k(n, \delta, T)$. Namely, $k(n, \delta, T)$ is the smallest k for which the threshold function $F_T(x) = \mathbb{1}(|x| \geq T)$ admits an *upper sandwiching polynomial* of degree k and expectation at most δ. Here, an upper sandwiching polynomial is simply a polynomial p for which $p(x) \geq F_T(x)$ pointwise.

We then use ideas from weighted approximation theory to give a lower bound on k for which such sandwiching polynomials exist. In order to apply these ideas, we make a few symmetrization and approximation arguments to reduce the problem to a continuous one-dimensional problem: Find a degree lower bound for a univariate polynomial that is a good upper sandwich for the function $f_T(x) = \operatorname{sgn}(|x| - T)$, with respect to a Gaussian distribution. The solution to this problem appeals to a weighted Markov-type inequality. This inequality generalizes the classical Markov inequality for uniform approximations, which gives a bound on the derivative of a low-degree polynomial that is bounded on the unit interval:

▶ **Theorem 4** ([47]). *Let p be a polynomial of degree d with $|p(x)| \leq 1$ on the interval $[-1, 1]$. Then $|p'(x)| \leq d^2$ on $[-1, 1]$.*

The idea is that an upper sandwich for f_T must have a large jump at the threshold T, which is impossible for low-degree polynomials. The formal proof of this claim is based on a variant of an "infinite-finite range" inequality, which asserts that the weighted norm of a polynomial on the real line is bounded by its norm on a finite interval.

1.2 Agnostically Learning Halfspaces

Halfspaces are a fundamental concept class in machine learning, both in theory and in practice.[3] Their study dates back to the Perceptron algorithm of the 1950s. Halfspaces serve as building blocks in many applications, including boosting and kernel methods.

Halfspaces can be learned in the PAC model [67] either by solving a linear program, or via simple iterative update algorithms (e.g. the Perceptron algorithm). However, learning halfspaces with classification noise is a much more difficult problem, and often needs to be dealt with in practice.

In this work, we study a challenging model of *adversarial noise* – the agnostic learning model of Kearns et al. [38]. In this model, a learner has access to examples drawn from a distribution $\mathcal{D}$ on $X \times \{\pm 1\}$ and must output a hypothesis $h : X \to \{\pm 1\}$ such that

$$\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}}\left[h(x) \neq y\right] \leq \mathrm{opt} + \varepsilon,$$

where opt is the error of the best concept in the concept class – that is,

$$\mathrm{opt} = \min_{f \in \mathcal{C}} \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}}\left[f(x) \neq y\right].$$

[3] A halfspace is a function $f : \mathbb{R}^n \to \{\pm 1\}$ given by $f(x) = \operatorname{sgn}(w \cdot x - \theta)$ for $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$, where $\operatorname{sgn}(x) = 1$ if $x \geq 0$ and $\operatorname{sgn}(x) = -1$ otherwise.

The theory of agnostic learning is not well-understood, even in the case of halfspaces. Positive results for efficient agnostic learning of high-dimensional halfspaces are restricted to limited classes of distributions.[4] For instance, halfspaces can be learned under the uniform distribution over the hypercube or the unit sphere, or on any log-concave distribution [35, 39]. On the negative side, a variety of both computational and information-theoretic hardness results are known. For instance, proper agnostic learning of halfspaces (where the learner is required to output a hypothesis that is itself a halfspace) is known to be NP-hard [24, 30]. Moreover, agnostically learning halfspaces under arbitrary distributions is as hard as PAC learning DNFs [44], which is a longstanding open problem.

There is essentially only one known technique for agnostically learning high-dimensional halfspaces: the L_1 regression algorithm [35], which we discuss in more detail in Section 3.2. In its most general form, the algorithm selects a linear space of functions $\mathcal{H} \subset \{h : X \to \mathbb{R}\}$. After drawing a number of examples (x_i, y_i) from $\mathcal{D}$, it computes

$$h^* = \operatorname*{argmin}_{h \in \mathcal{H}} \sum_i |h(x_i) - y_i|.$$

The output of the algorithm is $\operatorname{sgn}(h^*(x) - t)$ for some t. We need to ensure that the minimisation can be computed efficiently (e.g. by linear programming) and that every concept $f \in \mathcal{C}$ can be approximated by some $h \in \mathcal{H}$ – that is $\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[|h(x) - f(x)|] \le \varepsilon$. If this is the case, then $\mathcal{C}$ is agnostically learnable in time $\operatorname{poly}(|\mathcal{H}|)$.

Kalai et al. (and most subsequent work on learning using L_1 regression, e.g. [40, 26, 11, 36, 25]) chose $\mathcal{H}$ to be the class of low-degree polynomials. They showed that under certain classes of distributions, every halfspace can be approximated by a polynomial of degree $O_\varepsilon(1)$, and hence halfspaces are agnostically learnable in time $n^{O_\varepsilon(1)}$.

Distributional assumptions arise because the L_1 approximation measure (namely $\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[|h(x) - f(x)|]$) depends on the underlying distribution. A distribution-independent approximation would require an L_∞ approximation, which is too much to hope for in many circumstances.

1.2.1 Our Results

In this work, we ask whether polynomial regression can be extended to work beyond the classes of distributions studied by Kalai et al. In particular, can polynomials provide good L_1 approximations to halfspaces under distributions with heavy tails? Such distributions, including power law distributions, arise naturally in many physical, biological, and networking contexts. Certain learning problems even require heavy tailed distributions on examples [51].

Our result addressing this question (Theorem 6) is a negative one. We show that polynomial approximations to halfspaces do not exist for a large class of distributions, namely:

▶ **Definition 5.** An absolutely continuous distribution $\mathcal{D}$ on $\mathbb{R}$ is a *log-superlinear (LSL) distribution* if there exist $C > 0$ and $\gamma \in (0, 1)$ such that the density w of $\mathcal{D}$ satisfies $w(x) \ge C \exp(-|x|^\gamma)$.[5]

[4] An efficient algorithm is one which runs in time polynomial in the dimension n for any constant $\varepsilon > 0$ – that is, time $n^{O_\varepsilon(1)}$.

[5] The name log-superlinear comes from the fact that the tails of the probability density function of a LSL distribution are heavier than that of the log-linear Laplace distribution.

▶ **Theorem 6.** *For any LSL distribution $\mathcal{D}$, there exists $\varepsilon > 0$ such that no polynomial (of any degree) can approximate the sign function with L_1 error less than ε with respect to $\mathcal{D}$.*

In particular, this implies that the polynomial regression algorithm is not able to agnostically learn thresholds on the real line to within arbitrarily small error. Note that this result does not rule out the possibility that halfspaces can be agnostically learned by other techniques. Indeed, the classic approach of empirical risk minimization (see [38] and the references therein) gives an efficient algorithm for learning thresholds (which are halfspaces in one dimension) under arbitrary distributions. Thus the problem of learning real thresholds under LSL distributions is an explicit example for which polynomial regression fails while other techniques can succeed.

If we were to take $\gamma \geq 1$, the probability density function $C(\gamma)e^{-|x|^\gamma}$ (where $C(\gamma)$ is a normalising constant) would give a log-concave distribution, in which case Kalai et al. [35] show that good polynomial approximations to halfspaces exist. Thus our result gives a threshold between where polynomial approximations to halfspaces exist and where they do not.

Our result for thresholds extends readily to an impossibility result for learning halfspaces over $\mathbb{R}^n$:

▶ **Theorem 7.** *For any product distribution $\mathcal{D}$ on $\mathbb{R}^n$ with a LSL marginal distribution on some coordinate, there exists $\varepsilon > 0$ and a halfspace h such that no polynomial can approximate h with L_1 error less than ε with respect to $\mathcal{D}$.*

As with Theorem 3, the proof of Theorem 6 relies on several Markov-type inequalities for weighted polynomial approximations. Early work on the approximate degree of boolean functions [56, 59] used Markov's inequality to get tight lower bounds on the degree of uniform approximations to symmetric functions. For weighted approximations under LSL distributions, we actually get a much stronger statement. It turns out that the derivative of a polynomial is bounded near the origin *independent of the degree* as long as that polynomial is absolutley bounded when integrated under a LSL distribution. With this powerful fact in hand, the proof of Theorem 6 is quite simple. Consider the threshold function $f(t) = \mathrm{sgn}(t)$. Since f has a "jump" at zero, any good polynomial approximation to f must be bounded and have a large derivative near zero. The higher quality the approximation, the larger a derivative we need. But since the derivative of any polynomial is bounded by a constant, we cannot get arbitrarily good approximations to f using polynomials.

We give the full proof in Section 3.4, and discuss the multivariate generalization in Section 3.5.

1.2.2 Related Work

Our result echoes prior work establishing the limits of *uniform* polynomial approximations for various concept classes. For instance, the seminal work of Minsky and Papert [49] showed that there is an *intersection* of two halfpsaces over $\mathbb{R}^n$ which cannot be represented as the sign of any polynomial. Building on work of Nisan and Szegedy [56], Paturi [59] gave tight lower bounds for uniform approximations to symmetric boolean functions. This, and subsequent work on lower bounds for approximate degree, immediately imply limitations for distribution-independent agnostic learning via polynomial regression. Klivans and Sherstov [42] also showed a strong generalization of Paturi's result to disjunctions, giving limitations on how well they can be approximated by linear combinations of arbitrary features. By contrast to all of these results, our work shows a strong limitation for certain *distribution-dependent* polynomial approximations.

In the distribution-dependent setting, Feldman and Kothari [25] showed that polynomial regression cannot be used to learn disjunctions with respect to symmetric distributions on the hypercube. Recent work of Daniely et al. [19] also uses ideas from approximation theory to show limitations on a broad class of regression and kernel-based methods for learning halfspaces, even under a margin assumption. While our results only apply to polynomial regression, they hold for approximations of arbitrarily high complexity (i.e. degree), and for a large class of natural distributions.

The limitations we prove for polynomial regression do not rule out the existence of other agnostic learning algorithms, including those using L_1 regression with different feature spaces. Wimmer [68] showed how to use a different family of basis functions to learn halfspaces over symmetric distributions on the hypercube. Subsequent work of Feldman and Kothari [25] improved the running time in the special case of disjunctions. We leave it as an intriguing open question to determine whether other basis functions can be used to learn halfspaces under LSL distributions.

2 Tail Bounds for Limited Independence

Our proof consists of three steps:

§2.1 First we reformulate the question of tail bounds for k-wise independent distributions using linear programming duality and symmetrisation. This reduces the problem to proving a degree lower bound on univariate polynomials. Namely we need to give a lower bound on the degree of a polynomial $p : \{0, 1, \cdots, n\} \to \mathbb{R}$ such that $p(i) \geq 0$ for all i, $p(i) \geq 1$ if $|i - n/2| \geq T$, and $\mathbb{E}[p(i)] \leq \delta$, where i is drawn from the binomial distribution.

§2.2 We then transform the problem from one about polynomials with a discrete domain to one about polynomials with a continuous domain. This amounts to showing that, since $\mathbb{E}[p(i)] \leq \delta$ with respect to the binomial distribution, we can bound $\mathbb{E}[p(x + n/2)]$ with respect to a truncated Gaussian distribution on x.

§2.3 Finally we can apply the tools of weighted approximation theory. We know that $p(x + n/2)$ is small for x near the origin, but $p(T + n/2) \geq 1$. We show that any low-degree polynomial that is bounded near the origin cannot grow too quickly. This implies that p must have high degree.

2.1 Dual Formulation

Question 2 from the introduction is equivalent to finding the smallest k for which the value of the following linear program is at most δ.

Linear Program Formulation of Question 2

$$\begin{aligned}
\max_{\psi} \quad & \sum_{x \in \{-1,1\}^n} \psi(x) F_T(x) & \\
\text{s.t.} \quad & \sum_{x \in \{-1,1\}^n} \psi(x) \chi_S(x) = 0 & \text{for all } |S| \leq k \\
& \sum_{x \in \{-1,1\}^n} \psi(x) = 1 & \\
& 0 \leq \psi(x) \leq 1 & \text{for all } x \in \{-1,1\}^n.
\end{aligned}$$

Here, $F_T(x) = 1$ if $|x| \geq T$ and is 0 otherwise, and $\chi_S(x)$ is the Fourier character corresponding to $S \subseteq [n]$.

If we set $\underset{X}{\mathbb{P}}[X = x] = \psi(x)$, then the constraints impose that X is a k-wise independent distribution, while the objective function is $\underset{X}{\mathbb{P}}\left[\left|\sum_{i\in[n]} X_i\right| \geq T\right]$. Thus the above linear program finds the k-wise independent distribution with the worst tail bound. If the value of the program is at most δ, then all k-wise independent distributions satisfy the tail bound, as required.

Taking the dual of the above linear program yields the following.

Dual Formulation of Question 2

$$\begin{aligned}
&\min_p \; 2^{-n} \sum_{x\in\{-1,1\}^n} p(x) && \\
&\text{s.t. } \deg(p) \leq k && \\
&\quad p(x) \geq F_T(x) && \text{for all } x \in \{-1,1\}^n.
\end{aligned}$$

By strong duality, the value of the dual linear program is the same as that of the primal.

The multilinear polynomial p is an "upper sandwich" of F_T – that is, $p \geq F_T$ and $\underset{X\in\{\pm 1\}^n}{\mathbb{E}}[p(X)]$ is minimal. Therefore, $k(n, \delta, T)$ is the smallest k for which F_T admits an upper sandwiching polynomial of degree k with expectation δ.

Consider the shifted univariate symmetrization of F_T

$$F'_T(x) = \begin{cases} 1 \text{ if } |x - n/2| \geq T \\ 0 \text{ otherwise.} \end{cases}$$

By applying the well-known Minsky-Papert symmetrization [49] to the dual formulation above, we get the following characterization.

▶ **Theorem 8.** *The quantity $k(n, \delta, T)$ from Question 2 is the smallest k for which there exists a degree-k univariate polynomial $p : \{0, \ldots, n\} \to \mathbb{R}$ such that*
1. *$p(i) \geq F'_T(i)$ for all $0 \leq i \leq n$ and*
2. *$2^{-n} \sum_{i=0}^{n} \binom{n}{i} p(i) \leq \delta$.*

The upper bound on $k(n, \delta, T)$ (Theorem 1) is proved (in the appendix) by showing that

$$p(i) = \left(\frac{i - n/2}{T}\right)^k$$

satisfies the requirements of Theorem 8 for an appropriate even k.[6] So this characterisation does in fact capture how upper bounds are proved. The fact that it is a tight characterisation allows us to prove that a barrier to the technique is in fact an impossibility result.

With this characterisation of our problem, we may move on to proving inapproximability results.

2.2 A Continuous Version

To apply techniques from the theory of weighted polynomial approximations, we move to polynomials on a continuous domain. We replace the binomial distribution upon which Theorem 8 evaluates p with a Gaussian distribution.

[6] While our results show that this polynomial is *asymptotically* optimal, numerical experiments have shown that it is not exactly optimal.

Define the probability density function

$$w(x) = \frac{1}{\sqrt{\pi}} e^{-x^2}.$$

We define the L_∞ norm with respect to the weight w:

$$\|g\|_{L_\infty(S)} = \sup_{x \in S} |g(x)| w(x).$$

Now we can give the continuous version of the problem:

▶ **Theorem 9.** *Let $T = c\sqrt{n \log(1/\delta)}$ for $c \geq 5$, and $d = k(n, \delta, T)$. Assume $n \geq (12c)^2(3\log(1/\delta))^3$. Then for $T' = 4cT/\sqrt{n}$, there is a degree d polynomial q such that*
1. *$q(T') = q(-T') \geq 1$ and*
2. *$\|q\|_{L_\infty[-\sqrt{d},\sqrt{d}]} \leq \delta^{0.9}(n+1)$.*

The following lemma is key to moving from the discrete to the continous setting. It shows that if a polynomial is bounded at evenly spaced points, then it must also be bounded between those points, assuming the number of points is sufficiently large relative to the degree.

▶ **Lemma 10** (adaptation of [23, 62, 56]). *Let q be a polynomial of degree d such that $|q(i)| \leq 1$ for $i = 0, 1, \ldots, m$, where $3d^2 \leq m$. Then $|q(x)| \leq \frac{3}{2}$ for all $x \in [0, m]$.*

Proof. Let $a = \max_{x \in [0,m]} |q'(x)|$. Then by the mean value theorem, $|q(x)| \leq 1 + a/2$ for $x \in [0, m]$. By Markov's inequality ([47], see also [17]),

$$a \leq \frac{2d^2(1 + a/2)}{m}.$$

Rearranging gives

$$\frac{a}{2 + a} \leq \frac{d^2}{m} \leq \frac{1}{3}.$$

Therefore, $a \leq 1$, and hence $|q(x)| \leq \frac{3}{2}$ for $x \in [0, m]$. ◀

We also require the following anti-concentration lemma.

▶ **Lemma 11.**

$$\binom{n}{n/2 + \alpha\sqrt{n}} \geq \frac{2^{n - 6\alpha^2}}{n + 1}.$$

Proof. It is well known via Stirling's approximation that $\binom{n}{k} \geq 2^{nH(k/n)}/(n+1)$, where $H(\cdot)$ denotes the binary entropy function. We estimate

$$\begin{aligned} H\left(\frac{1}{2} + \frac{\alpha}{\sqrt{n}}\right) &\geq \left(\frac{1}{2} + \frac{\alpha}{\sqrt{n}}\right)\left(1 - \frac{2\alpha}{(\log 2)\sqrt{n}}\right) + \left(\frac{1}{2} - \frac{\alpha}{\sqrt{n}}\right)\left(1 + \frac{2\alpha}{(\log 2)\sqrt{n}}\right) \\ &\geq 1 - \frac{4\alpha^2}{(\log 2)n}, \end{aligned}$$

which concludes the proof. ◀

Proof of Theorem 9. Let p be the polynomial promised by Theorem 8. By Theorem 1, we know that $d \le 3\log(1/\delta)$. Define

$$q(x) = p(x\sqrt{n}/4c + n/2).$$

Then $q(\pm T') = p(\pm T + n/2) \ge F'_T(\pm T + n/2) = 1$, dispensing with the first claim.

Now for all integers i in the interval $n/2 \pm \sqrt{nd}/4c$, we have

$$2^{-n}\binom{n}{i}|p(i)| \le \delta$$

and hence, by Lemma 11,

$$|p(i)| \le \frac{2^n\delta}{\binom{n}{n/2+\sqrt{nd}/4c}} \le (n+1)\delta 2^{6d/16c^2} \le (n+1)\delta^{1-18/16c^2} \le (n+1)\delta^{0.9}.$$

By Lemma 10, $|p(x)| \le \frac{3}{2}(n+1)\delta^{0.9}$ on the whole interval $n/2 \pm \sqrt{nd}/4c$. Thus $|q(x)| \le \frac{3}{2}(n+1)\delta^{0.9}$ on $[-\sqrt{d}, \sqrt{d}]$, completing the proof. ◀

2.3 The Lower Bound

Now we state the result we need from approximation theory. The following "infinite-finite range inequality" shows that the norm of weighted polynomial on the real line is determined by its norm on a finite interval around the origin. Thus, an upper bound on the magnitude of a polynomial near the origin yields a bound on its growth away from the origin. We will apply this to the polynomial given to us in Theorem 9.

▶ **Theorem 12.** *For any polynomial p of degree d and $B > 1$,*

$$\|p\|_{L_\infty(\mathbb{R}\setminus[-B\sqrt{d},B\sqrt{d}])} \le (2eB)^d \exp(-B^2 d)\|p\|_{L_\infty[-\sqrt{d},\sqrt{d}]}.$$

The proof follows [45, Theorem 6.1] and [54, Theorem 4.16.12].

Proof. Let $\tilde{p}$ be a polynomial of degree d. Let $T_d(x)$ denote the dth Chebyshev polynomial of the first kind [17]. By the extremal properties of T_d, we have

$$|\tilde{p}(x)| \le |T_d(x)|\left(\max_{t\in[-1,1]} |\tilde{p}(t)|\right) \le (2|x|)^d\left(\max_{t\in[-1,1]} |\tilde{p}(t)|\right)$$

for $|x| \ge 1$. Rescaling $p(x) = \tilde{p}(x/\sqrt{d})$ yields

$$|p(x)| \le \left(\frac{2|x|}{\sqrt{d}}\right)^d\left(\max_{t\in[-\sqrt{d},\sqrt{d}]} |p(t)|\right) \le \sqrt{\pi}e^d\left(\frac{2|x|}{\sqrt{d}}\right)^d \|p\|_{L_\infty[-\sqrt{d},\sqrt{d}]}$$

for $|x| \ge \sqrt{d}$. Now let $|x| = B\sqrt{d}$ for some $B > 1$. Then

$$|p(x)|w(x) \le e^d(2B)^d \exp(-B^2 d)\|p\|_{L_\infty[-\sqrt{d},\sqrt{d}]}.$$

Since the coefficient $(2eB)^d \exp(-B^2 d)$ is decreasing in B, this proves the claim. ◀

The above approximation theory result, combined with our continuous formulation Theorem 9, enables us to complete the proof.

▶ **Theorem 13.** *Let $T = c\sqrt{n\log(1/\delta)}$ for $c \ge 5$. Assume $n \ge (12c)^2(3\log(1/\delta))^3$ and $\delta \le 1/n^4$. Then $k(n,\delta,T) > \log(1/\delta)/9\log c$.*

Proof. Let q be the polynomial given by Theorem 9. Let $T' = 4cT/\sqrt{n}$, $d = \log(1/\delta)/9\log c$, and $B = T'/\sqrt{d} = 12c^2\sqrt{\log c}$. For the sake of contradiction, we suppose that q satisfies the conditions of Theorem 9, but $\deg(q) \leq d$. Then

$$\|q\|_{L_\infty(\mathbb{R}\setminus[-B\sqrt{d},B\sqrt{d}])} = \|q\|_{L_\infty(\mathbb{R}\setminus[-T',T'])} \geq \frac{\exp(-T'^2)}{\sqrt{\pi}}.$$

On the other hand, applying Theorem 12, gives

$$\|q\|_{L_\infty(\mathbb{R}\setminus[-B\sqrt{d},B\sqrt{d}])} \leq (2eB)^d \exp(-T'^2)\delta^{0.9}(n+1).$$

Combining the two inequalities gives

$$\frac{1}{\sqrt{\pi}} \leq (2eB)^d\delta^{0.9}(n+1) \leq \left(24ec^2\sqrt{\log(c)}\right)^{\log(1/\delta)/9\log(c)} \delta^{0.9}(n+1) \leq \delta^{1/3}(n+1),$$

which is a contradiction. ◀

Theorem 13 yields Theorem 3.

3 Agnostically Learning Halfspaces

The class of log-concave distributions over $\mathbb{R}^n$ (defined below) is essentially the broadest under which we know how to agnostically learn halfspaces. While many distributions used in machine learning are log-concave, such as the normal, Laplace, beta, and Dirichlet distributions, log-concave distributions do not capture everything. For instance, the log-normal distribution and heavier-tailed exponential power law distributions are not log-concave. The main motivating question for this section is whether we can relax the assumption of log-concavity for agnostically learning halfspaces. To this end, we show a negative result: for LSL distributions, agnostic learning of halfspaces will require new techniques.

3.1 Background

Our starting point is the work of Kalai et al. [35]. Among their results is the following.

▶ **Theorem 14** ([35]). *The concept class of halfspaces over $\mathbb{R}^n$ is agnostically learnable in time* $\mathrm{poly}(n^{O_\varepsilon(1)})$ *under log-concave distributions.*

A log-concave distribution is an absolutely continuous probability distribution such that the logarithm of the probability density function is concave. For example, the standard multivariate Gaussian distribution on $\mathbb{R}^n$ has the probability density function $x \mapsto e^{-||x||_2^2/2}/(2\pi)^{n/2}$. The natural logarithm of this is $-||x||_2^2/2 - n/2 \cdot \log(2\pi)$, which is concave. The class of log-concave distributions also includes the Laplace distribution and other natural distributions. However, it does not contain heavy-tailed distributions (such as power laws) nor non-smooth distributions (such as discrete probability distributions).

Kalai et al. also show that we can agnostically learn halfspaces under the uniform distribution over the hypercube $\{\pm1\}^n$ or over the unit sphere $\{x \in \mathbb{R}^n : ||x||_2 = 1\}$.

3.2 The L_1 Regression Algorithm

The results of Kalai et al. are based on the so-called L_1 regression algorithm, which relies on being able to approximate the concept class in question by a low-degree polynomial:

▶ **Theorem 15** ([35]). *Fix a distribution $\mathcal{D}$ on $X \times \{\pm 1\}$ and a concept class $\mathcal{C} \subset \{f : X \to \{\pm 1\}\}$.*[7] *Suppose that, for all $f \in \mathcal{C}$, there exists a polynomial $p : X \to \mathbb{R}$ of degree at most d such that $\mathop{\mathbb{E}}_{x \sim \mathcal{D}_X}[|p(x) - f(x)|] \le \varepsilon$, where $\mathcal{D}_X$ is the marginal distribution of $\mathcal{D}$ on X. Then, with probability $1 - \delta$ the L_1 regression algorithm outputs a hypothesis h such that*

$$\mathop{\mathbb{P}}_{(x,y) \sim \mathcal{D}}[h(x) \neq y] \le \min_{f \in \mathcal{C}} \mathop{\mathbb{P}}_{(x,y) \sim \mathcal{D}}[f(x) \neq y] + \varepsilon$$

in time $\mathrm{poly}(n^d, 1/\varepsilon, \log(1/\delta))$ with access only to examples drawn from $\mathcal{D}$.

The L_1 regression algorithm solves a linear program to find a polynomial p of degree at most d that minimises $\sum_i |p(x_i) - y_i|$, where (x_i, y_i) are the examples sampled from $\mathcal{D}$. The hypothesis is then $h(x) = \mathrm{sgn}(p(x) - t)$, where $t \in [-1, 1]$ is chosen to minimise the error of h on the examples.

Given Theorem 15, proving Theorem 14 reduces to showing that halfspaces can be approximated by low-degree polynomials under the distributions we are interested in. It is important to note that making assumptions on the distribution is necessary (barring a major breakthrough): Agnostically learning halfspaces under arbitrary distributions is at least as hard as PAC learning DNF formulas [44]. Moreover, proper learning of halfspaces under arbitrary distributions is known to be NP-hard [24].

In fact, we can reduce the task of approximating a halfspace to a one-dimensional problem. A halfspace is given by $f(x) = \mathrm{sgn}(w \cdot x - \theta)$ for some $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$. It suffices to find a univariate polynomial p of degree at most d such that $\mathop{\mathbb{E}}_{x \sim \mathcal{D}_{w,\theta}}[|p(x) - \mathrm{sgn}(x)|] \le \varepsilon$, where $\mathcal{D}_{w,\theta}$ is the distribution of $w \cdot x - \theta$ when x is drawn from $\mathcal{D}_X$. If $\mathcal{D}_X$ is log-concave, then so is $\mathcal{D}_{w,\theta}$.

3.3 On the Density of Polynomials

In this section, we give some intuition for why one might expect that polynomial approximations do not suffice for learning under LSL distributions. It turns out that under a LSL distribution w, polynomials actually fail to be dense in the space $C_0[w]$ of continuous functions vanishing at infinity when weighted by w. This is in stark contrast to the classical Weierstrass approximation theorem, which asserts that the polynomials are dense in C_0 under the uniform weight. These kinds of results address *Bernstein's approximation problem* [10], a precise statement of which is as follows.

▶ **Question 16.** *Let $w : \mathbb{R} \to [0, 1]$ be a measurable function. Let $C_0[w]$ denote the space of continuous functions f for which $\lim_{|x| \to \infty} f(x)w(x) = 0$. Under what conditions on w is it true that for every $f \in C_0[w]$, there is a sequence of polynomials $\{p_n\}_{n=1}^{\infty}$ for which*

$$\lim_{n \to \infty} \|(p_n - f)w\|_\infty = 0?$$

(The choice of the L_∞ norm here appears to make very little difference). If Bernstein's problem admits a positive resolution, we say that the polynomials are *dense* in $C_0[w]$. The excellent survey of Lubinsky [45] presents a number of criteria for when polynomials are dense. The one that is most readily applied was proved by Carleson [16] (but appears to be implicit in [34]):

[7] Here $X = \mathbb{R}^n$.

▶ **Theorem 17.** *Let w be even and positive with $\log(w(e^x))$ concave. Then the polynomials are dense in $C_0[w]$ iff*

$$\int_0^\infty \frac{\log w(x)}{1+x^2} \mathrm{d}x = -\infty.$$

This immediately yields the following dichotomy result for exponential power distributions:

▶ **Corollary 18.** *For $\gamma > 0$ and $w_\gamma(x) = \exp(-|x|^\gamma)$, the polynomials are dense in $C_0[w_\gamma]$ iff $\gamma \geq 1$.*

In particular, this justifies our assertion that the polynomials fail to be dense in the continuous functions under LSL distributions.

So what does this have to do with agnostically learning halfspaces? Recall that the analysis of the L_1-regression algorithm of Kalai et al. [35] reduces approximating a halfspace under a distribution $\mathcal{D}$ to the problem of approximating each threshold function $\operatorname{sgn}(x-\theta)$ under each marginal distribution of $\mathcal{D}$. So for the algorithm to work, we require $\mathcal{D}$ to have marginals w under which $\operatorname{sgn}(x-\theta)$ can be approximated arbitrarily well by polynomials. Now if the polynomials are dense in $C_0[w]$, then threshold functions can also be approximated arbitrarily well (since $C_0[w]$ is in turn dense in $L_1[w]$). Such an appeal to density actually underlies Kalai et al.'s proof of approximability under log-concave distributions. On the other hand, if the polynomials fail to be dense, then one might conjecture that thresholds cannot be arbitrarily well approximated.

Our result, presented in the next section, confirms the conjecture that even the *sign* function cannot be approximated arbitrarily well by polynomials under LSL distributions

3.4 Lower Bound for One Variable

Consider the LSL density function

$$w_\gamma(x) := C(\gamma)\exp(-|x|^\gamma)$$

on the reals for $\gamma \in (0,1)$, where $C(\gamma)$ is a normalizing constant. Define the sign function $\operatorname{sgn}(x) = 1$ if $x \geq 0$ and $\operatorname{sgn}(x) = -1$ otherwise. In this section, we show that for sufficiently small ε, the sign function does not have an L_1 approximation under the distribution w_γ. More formally,

▶ **Proposition 19.** *For any $\gamma \in (0,1)$, there exists an $\varepsilon = \varepsilon(\gamma)$ such that for any polynomial p,*

$$\int_{\mathbb{R}} |p(x) - \operatorname{sgn}(x)| w_\gamma(x)\, dx > \varepsilon.$$

The proof is based on the following Markov-type inequality, which roughly says that a bounded polynomial cannot have a large derivative (under the weight w_γ). This implies the claim, since the sign function we are trying to approximate has a large "jump" at the origin.

▶ **Lemma 20.** *For $\gamma \in (0,1)$ there is a constant $M(\gamma)$ such that*

$$\sup_{x\in\mathbb{R}}(|p'(x)|w_\gamma(x)) \leq M(\gamma) \int_{\mathbb{R}} |p(x)| w_\gamma(x)\, dx.$$

Proof. The lemma is a combination of a Markov-type inequality and a Nikolskii-type, available in a survey of Nevai [54]:

▶ **Theorem 21** ([55], [54, Theorem 4.17.4]). *There exists a constant $C_1(\gamma)$ such that for any polynomial p,*

$$\int_{\mathbb{R}} |p'(x)| w_\gamma(x)\ dx \le C_1(\gamma) \int_{\mathbb{R}} |p(x)| w_\gamma(x)\ dx.$$

▶ **Theorem 22** ([53], [54, Theorem 4.17.5]). *There exists a constant $C_2(\gamma)$ such that for any polynomial p,*

$$\sup_x (|p(x)| w_\gamma(x)) \le C_2(\gamma) \int_{\mathbb{R}} |p(x)| w_\gamma(x)\ dx.$$

◀

Proof of Proposition 19. Fix $\varepsilon \in (0,1)$ and suppose p is a polynomial satisfying

$$\int_{\mathbb{R}} |p(x) - \operatorname{sgn}(x)| w_\gamma(x)\ dx \le \varepsilon.$$

Since the absolute value of the sign function integrates to 1, this forces

$$\int_{\mathbb{R}} |p(x)| w_\gamma(x)\ dx \le 1 + \varepsilon \le 2.$$

Therefore, we have by Lemma 20 that $|p'(x)| w_\gamma(x) \le 2M(\gamma)$ for every x.

The idea is now to show that there is some x_0 for which $|p'(x_0)| w_\gamma(x_0) \ge \Omega(1/\varepsilon)$. To see this, let $\delta = 4\varepsilon / C(\gamma)$ and observe that there must exist some $x_+ \in [0, \delta]$ such that $p(x_+) \ge 1/2$. If this were not the case, then we would have

$$\int_{\mathbb{R}} |p(x) - \operatorname{sgn}(x)| w_\gamma(x)\ dx \ge \int_0^\delta \left(1 - \frac{1}{2}\right) C(\gamma) \exp(-x^\gamma)\ dx \ge \frac{\delta}{2} C(\gamma) \exp(-\delta^\gamma) \ge \varepsilon$$

for ε small enough, and hence δ small enough, to make $\exp(-\delta^\gamma) \ge 1/2$, yielding a contradiction. A similar argument shows that there is some $x_- \in [-\delta, 0]$ with $p(x_-) \le -1/2$. Therefore, by the mean value theorem, there is some $x_0 \in [x_-, x_+]$ with $p'(x_0) \ge 1/2\delta = C(\gamma)/8\varepsilon$. Moreover, because we took δ small enough, we also have $p'(x_0) w(x_0) \ge C(\gamma)/16\varepsilon$. This shows that no polynomial ε-approximates sgn as long as $\varepsilon < C/32M$. ◀

Moreover, the proposition shows that it is impossible to get arbitrarily close polynomial approximations to halfspaces under densities w for which there are constants C and $\gamma \in (0,1)$ with $w(x) \ge C \exp(-|x|^\gamma)$ for all $x \in \mathbb{R}$. This shows that LSL distributions on $\mathbb{R}$ do not enable arbitrarily close polynomial approximations to halfspaces.

3.5 Extending the Lower Bound to Multivariate Distributions

It is straightforward to extend the lower bound from the previous section to product distributions with LSL marginals.

▶ **Theorem 23.** *Let $X = (X_1, \ldots, X_n)$ be a random variable over $\mathbb{R}^n$ with density $f_X(x) = w(x_1) f(x_2, \ldots, x_n)$. Suppose the density w specifies a univariate γ-LSL distribution. Then there exists an $\varepsilon = \varepsilon(\gamma)$ such that for any polynomial p,*

$$\int_{\mathbb{R}^n} |p(x_1, \ldots, x_n) - \operatorname{sgn}(x_1)| f_X(x_1, \ldots, x_n)\ dx_1 dx_2 \ldots dx_n > \varepsilon.$$

That is, the linear threshold function $\operatorname{sgn}(x_1)$ *cannot be approximated arbitrarily well by polynomials.*

Proof. Let $p(x_1, \dots, x_n)$ be a polynomial, and define a univariate polynomial q by "averaging out" the variables $x_2, \dots, x_n$:

$$q(x_1) := \int_{\mathbb{R}^{n-1}} p(x_1, \dots, x_n) f(x_2, \dots, x_n)\ dx_2 \dots\ dx_n.$$

Then we have

$$\begin{aligned}
&\int_{\mathbb{R}} |q(x_1) - \operatorname{sgn}(x_1)| w(x_1)\ dx_1 \\
&\quad = \int_{\mathbb{R}} \left| \int_{\mathbb{R}^{n-1}} (p(x_1, \dots, x_n) - \operatorname{sgn}(x_1)) f(x_2, \dots, x_n)\ dx_2 \dots dx_n \right| w(x_1)\ dx_1 \\
&\quad \leq \int_{\mathbb{R}} \left(\int_{\mathbb{R}^{n-1}} |p(x_1, \dots, x_n) - \operatorname{sgn}(x_1)| f(x_2, \dots, x_n)\ dx_2 \dots dx_n \right) w(x_1)\ dx_1 \\
&\quad = \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} |p(x_1, \dots, x_n) - \operatorname{sgn}(x_1)| f_X(x_1, \dots, x_n)\ dx_1 dx_2 \dots dx_n.
\end{aligned}$$

By Proposition 19, the latter quantity must be at least $\varepsilon(\gamma)$. ◀

Let $w_\gamma^n(x) \propto \exp(-(|x_1|^\gamma + \cdots + |x_n|^\gamma))$ denote the density of the "prototypical" multivariate LSL distribution, with each marginal having the same exponential power law distribution. Our impossibility result holds uniformly for every distribution in the sequence $\{w_\gamma^n\}$. That is, for every $\gamma \in (0, 1)$, there exists $\varepsilon = \varepsilon(\gamma)$ for which halfspaces cannot be learned by polynomials under any of the distributions specified by $\{w_\gamma^n\}$.

As a consequence, we get inapproximability results for several natural classes of distributions that dominate $\{w_\gamma^n\}$ by constant factors (i.e. not growing with n).

1. Any power-law distribution, i.e. a distribution with density $\propto \|x\|^{-M}$ for some constant M, since such a distribution dominates every w_γ^n.
2. Multivariate generalizations of the log-normal distribution, i.e. any distribution with density $\propto \exp(-\operatorname{polylog}(\|x\|))$.
3. Multivariate exponential power distributions, which have densities $\propto \exp(-\|x\|^\gamma)$ for $\gamma \in (0, 1)$. These distributions dominate the prototypical w_γ^n by the inequality of ℓ_p-norms:

   $$\|x\|^\gamma \leq |x_1|^\gamma + \cdots + |x_n|^\gamma$$

   for every $0 \leq \gamma \leq 2$.

4 Further Work

Our negative results naturally suggest a number of directions for future work.

Are there other suitable derandomizations of concentration inequalities? In this work, we focused on understanding the limits of k-wise independent distributions. Gopalan et al. [27] gave a much more sophisticated generator with nearly optimal seed length. But could simple, natural pseudorandom distributions, such as small-bias spaces, give strong tail bounds themselves?

Are halfspaces agnostically learnable under LSL distributions? Our negative result does not even necessarily rule out the use of L_1 regression for this task: The polynomial regression algorithm of Kalai et al. [35] is in fact quite flexible. Nothing is really special about the basis of low-degree monomials, and the algorithm works equally well over any small, efficiently evaluable "feature space". That is, if we can show that halfspaces are well-approximated by linear combinations of features from a feature space $\mathcal{F}$ under a distribution $\mathcal{D}$, then we can agnostically learn halfspaces with respect to $\mathcal{D}$ in time proportional to $|\mathcal{F}|$. Could one hope for such approximations? Wimmer [68] and Feldman and Kothari [25] have shown how to use non-polynomial basis functions to obtain faster learning algorithms on the boolean hypercube. On the other hand, recent work of Dachman-Soled et al. [18] shows that, at least for product distributions on the hypercube, polynomials yield the best basis for L_1 regression.

Acknowledgements. We thank Varun Kanade, Scott Linderman, Raghu Meka, Jelani Nelson, Justin Thaler, Salil Vadhan, Les Valiant, and several anonymous reviewers for helpful discussions and comments.

References

1 Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *J. ACM*, 51(4):595–605, 2004.

2 Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986.

3 Noga Alon and Asaf Nussboim. K-wise independent random graphs. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 813–822. IEEE, 2008.

4 Louay M. J. Bazzi. Polylogarithmic independence can fool DNF formulas. *SIAM J. Comput.*, 38(6):2220–2272, March 2009.

5 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001.

6 Richard Beigel. The polynomial method in circuit complexity. In *Structure in Complexity Theory Conference*, pages 82–95, 1993.

7 Richard Beigel. Perceptrons, PP, and the polynomial hierarchy. *Computational Complexity*, 4:339–349, 1994.

8 M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *FOCS*, pages 276–287, Nov 1994.

9 Itai Benjamini, Ori Gurel-Gurevich, and Ron Peled. On k-wise independent distributions and boolean functions. *arXiv preprint arXiv:1201.3261*, 2012.

10 S. N. Bernstein. Le problème de l'approximation des fonctions continues sur tout l'axe réel et l'une de ses applications. *Bull. Math. Soc. France*, 52:399–410, 1924.

11 Eric Blais, Ryan O'Donnell, and Karl Wimmer. Polynomial regression under arbitrary product distributions. *Machine Learning*, 80(2-3):273–294, 2010.

12 Aline Bonami. Étude des coefficients de fourier des fonctions de $l^p(g)$. *Annales de l'institut Fourier*, 20(2):335–402, 1970.

13 Mark Braverman. Polylogarithmic independence fools AC0 circuits. *J. ACM*, 57(5):28:1–28:10, June 2008.

14 Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *FOCS*, pages 40–47, 2010.

15 Joshua Brody and Elad Verbin. The coin problem and pseudorandomness for branching programs. In *FOCS*, pages 30–39, 2010.

16 Lennart Carleson. Bernstein's approximation problem. *Proc. Amer. Math. Soc.*, 2:953–961, 1951.

17 E.W. Cheney. *Introduction to Approximation Theory.* AMS Chelsea Publishing Series. AMS Chelsea Pub., 1982.

18 Dana Dachman-Soled, Vitaly Feldman, Li-Yang Tan, Andrew Wan, and Karl Wimmer. Approximate resilience, monotonicity, and the complexity of agnostic learning. *CoRR*, abs/1405.5268, 2014. To appear in SODA 2015.

19 Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. The complexity of learning halfspaces using generalized linear methods. *CoRR*, abs/1211.0616, 2014.

20 Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. In Maria Serna, Ronen Shaltiel, Klaus Jansen, and Jose Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 6302 of *Lecture Notes in Computer Science*, pages 504–517, 2010.

21 Ilias Diakonikolas, Parikshit Gopalan, Ragesh Jaiswal, Rocco A. Servedio, and Emanuele Viola. Bounded independence fools halfspaces. In *In Proc. 50th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 171–180, 2009.

22 Ilias Diakonikolas, Rocco A. Servedio, Li-Yang Tan, and Andrew Wan. A regularity lemma, and low-weight approximators, for low-degree polynomial threshold functions. In *Proceedings of the 2010 IEEE 25th Annual Conference on Computational Complexity*, CCC'10, pages 211–222, Washington, DC, USA, 2010. IEEE Computer Society.

23 H. Ehlich and K. Zeller. Schwankung von polynomen zwischen gitterpunkten. *Mathematische Zeitschrift*, 86:41–44, 1964.

24 Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'06, pages 563–574, Washington, DC, USA, 2006. IEEE Computer Society.

25 Vitaly Feldman and Pravesh Kothari. Agnostic learning of disjunctions on symmetric distributions. *CoRR*, abs/1405.6791, 2014.

26 Parikshit Gopalan, Adam Tauman Kalai, and Adam R. Klivans. Agnostically learning decision trees. In *STOC*, pages 527–536, 2008.

27 Parikshit Gopalan, Daniel Kane, and Raghu Meka. Pseudorandomness for concentration bounds and signed majorities. *CoRR*, abs/1411.4584, 2014.

28 Parikshit Gopalan, Ryan O'Donnell, Yi Wu, and David Zuckerman. Fooling functions of halfspaces under product distributions. In *Proceedings of the 2010 IEEE 25th Annual Conference on Computational Complexity*, CCC'10, pages 223–234, Washington, DC, USA, 2010. IEEE Computer Society.

29 Parikshit Gopalan and Jaikumar Radhakrishnan. Finding duplicates in a data stream. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 402–411. Society for Industrial and Applied Mathematics, 2009.

30 V. Guruswami and P. Raghavendra. Hardness of learning halfspaces with noise. In *Proceedings of FOCS'06*, pages 543–552, 2006.

31 Nicholas J. A. Harvey, Jelani Nelson, and Krzysztof Onak. Sketching and streaming entropy via approximation theory. In *FOCS*, pages 489–498, 2008.

32 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):pp. 13–30, 1963.

33 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *STOC*, pages 356–364, 1994.

34 S. Izumi and T. Kawata. Quasi-analytic class and closure of $\{t^n\}$ in the interval $(-\infty, \infty)$. *Tohoku Math. J.*, 43:267–273, 1937.

35 Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM J. Comput.*, 37(6):1777–1805, 2008.

36 Daniel M. Kane, Adam Klivans, and Raghu Meka. Learning halfspaces under log-concave densities: Polynomial approximations and moment matching. In *COLT*, pages 522–545, 2013.

37 Daniel M Kane and Jelani Nelson. A derandomized sparse Johnson-Lindenstrauss transform. *arXiv preprint arXiv:1006.3585*, 2010.

38 Michael Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning. In *Machine Learning*, pages 341–352. ACM Press, 1994.

39 Adam R. Klivans, Philip M. Long, and Alex K. Tang. Baum's algorithm learns intersections of halfspaces with respect to log-concave distributions. In Irit Dinur, Klaus Jansen, Joseph Naor, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 5687 of *Lecture Notes in Computer Science*, pages 588–600. Springer Berlin Heidelberg, 2009.

40 Adam R. Klivans, Ryan O'Donnell, and Rocco A. Servedio. Learning geometric concepts via gaussian surface area. In *FOCS*, pages 541–550, 2008.

41 Adam R. Klivans and Rocco A. Servedio. Learning DNF in time $2^{\tilde{o}(n^{1/3})}$. *J. Comput. Syst. Sci.*, 68(2):303–318, 2004.

42 Adam R. Klivans and Alexander A. Sherstov. Lower bounds for agnostic learning via approximate rank. *Computational Complexity*, 19(4):581–604, 2010.

43 Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products. In *STOC*, pages 263–272, 2011.

44 Wee Sun Lee, Peter L. Bartlett, and Robert C. Williamson. On efficient agnostic learning of linear combinations of basis functions. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, COLT'95, pages 369–376, New York, NY, USA, 1995. ACM.

45 Doron Lubinsky. A survey of weighted polynomial approximation with exponential weights. *Surveys in Approximation Theory*, 3:1–105, 2007.

46 Michael Luby and Avi Wigderson. *Pairwise independence and derandomization.* Citeseer, 1995.

47 A. A. Markov. On a question of D. I. Mendeleev. *Zapiski Imperatorskoi Akademii Nauk,*, 62:1–24, 1890.

48 Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC'10, pages 427–436, New York, NY, USA, 2010. ACM.

49 Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge MA, 1972.

50 Michael Mitzenmacher and Salil Vadhan. Why simple hash functions work: Exploiting the entropy in a data stream. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'08, pages 746–755, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

51 Elchanan Mossel and Mesrob I Ohannessian. On the impossibility of learning the missing mass. *arXiv preprint arXiv:1503.03613*, 2015.

52 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Computing*, 22:838–856, 1993.

53 P. Nevai and V. Totik. Sharp Nikolskii inequalities with exponential weights. *Analysis Mathematica*, 13(4):261–267, 1987.

54 Paul Nevai. Géza Freud, orthogonal polynomials and Christoffel functions. A case study. *Journal of Approximation Theory*, 48(1):3–167, 1986.

55 Paul Nevai and Vilmos Totik. Weighted polynomial inequalities. *Constructive Approximation*, 2(1):113–127, 1986.

56 N. Nisan and M. Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994.

57 Noam Nisan. $\mathcal{RL} \subset \mathcal{SC}$. In *STOC*, pages 619–623, 1992.

58 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, New York, NY, USA, 2014.

59 Ramamohan Paturi. On the degree of polynomials that approximate symmetric boolean functions (preliminary version). In *STOC*, pages 468–474, 1992.

60 Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–17:24, September 2008.

61 Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via fourier analysis. In *APPROX-RANDOM*, pages 655–670, 2013.

62 T. J. Rivlin and E. W. Cheney. A comparison of uniform approximations on an interval and a finite subset thereof. *SIAM J. Numer. Anal.*, 3(2):311–320, 1966.

63 Sushant Sachdeva and Nisheeth K. Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends in Theoretical Computer Science*, 9(2):125–210, 2014.

64 J. Schmidt, A. Siegel, and A. Srinivasan. Chernoff–Hoeffding bounds for applications with limited independence. *SIAM J. Discrete Mathematics*, 8(2):223–250, 1995.

65 Alexander A. Sherstov. Communication lower bounds using dual polynomials. *Bulletin of the EATCS*, 95:59–93, 2008.

66 Alexander A. Sherstov. Separating AC^0 from depth-2 majority circuits. *SIAM J. Comput.*, 38(6):2113–2129, 2009.

67 Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

68 Karl Wimmer. Agnostically learning under permutation invariant distributions. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS'10, pages 113–122, Washington, DC, USA, 2010. IEEE Computer Society.

A Upper Bound for Limited Independence

Theorem 1 follows from the following well-known [64, 8] lemma, which we prove for completeness.

▶ **Lemma 24.** *Let $X \in \{\pm 1\}^n$ be uniform and $r \in \mathbb{R}^n$. For all even $k \geq 2$,*

$$\mathbb{E}\left[(X \cdot r)^k\right] \leq \left(e\,||r||_2^2\,k\right)^{k/2}.$$

An even stronger form of Lemma 24 follows immediately from the hypercontractivity theorem [12] [58, §9]: Letting $f(x) = x \cdot r$, we have

$$\mathbb{E}\left[(X \cdot r)^k\right] = ||f||_k^k \leq \left((k-1)^{\deg(f)/2}\,||f||_2\right)^k = \left(\sqrt{k-1}\,||r||_2\right)^k,$$

as required. A self-contained proof follows.

Proof. We start by bounding the moment generating function of $X \cdot r$: Let $t \in \mathbb{R}$ be fixed later. For any $i \in [n]$, we have

$$\mathbb{E}\left[e^{tr_iX_i}\right] = \frac{1}{2}\left(e^{tr_i} + e^{-tr_i}\right) = \sum_{k=0}^{\infty} \frac{(tr_i)^k + (-tr_i)^k}{2k!} = \sum_{k=0}^{\infty} \frac{(tr_i)^{2k}}{(2k)!} \leq \sum_{k=0}^{\infty} \frac{(t^2r_i^2)^k}{2^k k!} = e^{t^2r_i^2/2}.$$

By independence,

$$\mathbb{E}\left[e^{t(X\cdot r)}\right] = \prod_{i=1}^{n} \mathbb{E}\left[e^{tr_iX_i}\right] \leq \prod_{i=1}^{n} e^{t^2r_i^2/2} = e^{t^2||r||_2^2/2}.$$

We wish to bound a single moment, namely $\mathbb{E}\left[(X \cdot r)^{k_*}\right]$ for an even k_*. We do this by picking one term out of the taylor series of $\mathbb{E}\left[e^{t(X\cdot r)}\right]$. First we remove the odd terms:

$$\sum_{k \text{ even}} \frac{t^k}{k!}\mathbb{E}\left[(X \cdot r)^k\right] = \frac{1}{2}\left(\mathbb{E}\left[e^{t(X\cdot r)}\right] + \mathbb{E}\left[e^{-t(X\cdot r)}\right]\right) \leq e^{t^2||r||_2^2/2}$$

We have $\mathbb{E}\left[(X \cdot r)^k\right] \geq 0$ for even k, so we can remove terms from the above infinite sum without increasing it. Thus

$$\frac{t^{k_*}}{k_*!}\mathbb{E}\left[(X \cdot r)^{k_*}\right] \leq \sum_{k \text{ even}} \frac{t^k}{k!}\mathbb{E}\left[(X \cdot r)^k\right] \leq e^{t^2||r||_2^2/2}.$$

Rearranging and setting $t = \sqrt{k_*}/\,||r||_2$, we obtain

$$\mathbb{E}\left[(X \cdot r)^{k_*}\right] \leq \frac{k_*!}{t^{k_*}}e^{t^2||r||_2^2/2} = \frac{k_*!\,||r||_2^{k_*}\,e^{k_*/2}}{\sqrt{k_*}^{k_*}} \leq \left(\frac{k_*^2\,||r||_2^2\,e}{k_*}\right)^{k_*/2} = (e\,||r||_2^2\,k_*)^{k_*/2},$$

as required. ◀

Now we can prove the upper bound for k-wise independence using the connection between moment bounds and tail bounds [64].

Proof of Theorem 1. Note that, if $X \in \{\pm 1\}^n$ is k-wise independent, then

$$\mathbb{E}\left[(X \cdot r)^k\right] = \sum_{i_1 \cdots i_k \in [n]} \left(\prod_{j=1}^{k} r_{i_j}\right) \cdot \mathbb{E}\left[\prod_{j=1}^{k} X_{i_j}\right]$$

is the same as for uniform X, as this is the expectation of a degree-k polynomial. By Lemma 24 and Markov's inequality, we have (assuming k is even),

$$\mathbb{P}\left[|X \cdot r| \geq T\right] = \mathbb{P}\left[(X \cdot r)^k \geq T^k\right] \leq \frac{\mathbb{E}\left[(X \cdot r)^k\right]}{T^k} \leq \left(\frac{e\,||r||_2^2\,k}{T^2}\right)^{k/2}.$$

Substituting $k = 2\lceil \eta \log_e(1/\delta)\rceil$ and $T = e^{(\eta+1)/2\eta}\sqrt{k}\,||r||_2$, we have

$$\mathbb{P}\left[|X \cdot r| \geq T\right] \leq \left(\frac{e\,||r||_2^2\,k}{(e^{(\eta+1)/2\eta}\sqrt{k}\,||r||_2)^2}\right)^{\lceil \eta \log_e(1/\delta)\rceil} = e^{-\lceil \eta \log_e(1/\delta)\rceil/\eta} \leq \delta.$$

◀

Tighter Connections between Derandomization and Circuit Lower Bounds*

Marco L. Carmosino[1], Russell Impagliazzo[1], Valentine Kabanets[2], and Antonina Kolokolova[3]

1 **Department of Computer Science, University of California San Diego La Jolla, CA, USA**
`mcarmosi@cs.ucsd.edu`, `russell@eng.ucsd.edu`

2 **School of Computing Science, Simon Fraser University Burnaby, BC, Canada**
`kabanets@cs.sfu.ca`

3 **Department of Computer Science, Memorial University of Newfoundland St. John's, NL, Canada**
`kol@mun.ca`

Abstract

We tighten the connections between circuit lower bounds and derandomization for each of the following three types of derandomization:

- general derandomization of promise-BPP (connected to Boolean circuits),
- derandomization of Polynomial Identity Testing (PIT) over fixed finite fields (connected to arithmetic circuit lower bounds over the same field), and
- derandomization of PIT over the integers (connected to arithmetic circuit lower bounds over the integers).

We show how to make these connections *uniform equivalences*, although at the expense of using somewhat less common versions of complexity classes and for a less studied notion of inclusion.

Our main results are as follows:

1. We give the first proof that a non-trivial (nondeterministic subexponential-time) algorithm for PIT over a *fixed finite field* yields arithmetic circuit lower bounds.
2. We get a similar result for the case of PIT over the integers, strengthening a result of Jansen and Santhanam [13] (by removing the need for advice).
3. We derive a Boolean circuit lower bound for NEXP $\cap$ coNEXP from the assumption of sufficiently strong non-deterministic derandomization of promise-BPP (without advice), as well as from the assumed existence of an NP-computable non-empty property of Boolean functions useful for proving superpolynomial circuit lower bounds (in the sense of natural proofs of [20]); this strengthens the related results of [11].
4. Finally, we turn all of these implications into equivalences for appropriately defined promise classes and for a notion of robust inclusion/separation (inspired by [9]) that lies between the classical "almost everywhere" and "infinitely often" notions.

1998 ACM Subject Classification F.2.3 Tradeoffs between Complexity Measures

Keywords and phrases derandomization, circuit lower bounds, polynomial identity testing, promise BPP, hardness vs. randomness

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.645

* This work was partially supported by the Simons Foundation and NSF grant CCF-121351 (M. Carmosino, R. Impagliazzo) and by NSERC Discovery grant (V. Kabanets, A. Kolokolova)

1 Introduction

While randomness has become an indispensable tool for algorithm design, many (though not all) randomized algorithms have later been derandomized, i.e., shown to have equivalent, comparably fast deterministic algorithms. One highly successful method for derandomization of whole classes of algorithms has been the "hardness as randomness" paradigm [2, 7, 28, 19, 5, 12, 24, 22, 26]. In this paradigm, problems that are hard for the type of computation to be derandomized are converted into indistinguishable pseudo-random generators (PRGs) for the same class, which then replace the random choices of the randomized algorithm.

It has long been observed that this method seems to require hardness for the non-uniform version of the class (at least for worst-case derandomization). PRGs are a special case of "black-box" derandomization methods, where the algorithm to be accessed is only modified by replacing the random decisions with deterministic choices. It is relatively straightforward to show that "black-box" derandomization requires a circuit bound against the type of algorithm to be derandomized (see, e.g., [10]). Thus, via the hardness as randomness paradigm, strong circuit lower bounds can be proved *equivalent* to universal "black-box" derandomization.

More surprisingly, there are results that show the reverse direction, that derandomization requires strong circuit lower bounds, is true even for non-black-box algorithms [11]. In fact, even derandomizing a specific algorithm, the randomized polynomial identity test (PIT) of Schwartz and Zippel (also discovered by DeMillo and Lipton) [21, 29, 8], would imply strong lower bounds for arithmetic circuits [14, 13, 1]. However, the proofs of these statements are not direct reductions, but instead go through various complexity class collapses, and the conclusions in one direction are not usually exact matches for the other. So, unlike for "black-box" algorithms, we do not usually get a literal equivalence between a derandomization result and a circuit lower bound (one exception is [13]).

In this paper, we tighten the connections between circuit lower bounds and derandomization in several ways, for each of the three types of derandomization: general derandomization of promise-BPP (connected to Boolean circuits), derandomization of PIT over fixed finite fields (connected to arithmetic circuit lower bounds over the same field), and derandomization of PIT over the integers (connected to arithmetic circuit lower bounds over the integers). We show how to make these connections *equivalences*, although at the expense of using somewhat less common versions of complexity classes and a less studied notion of inclusion, the "robustly-often" inclusion introduced by [9]. Even for worst-case inclusion, we simplify and strengthen the known connections in several ways.

1.1 Our results

Following [13], let ml-NE denote the class of multilinear n-variate polynomials $F = \{f_n\}_{n\geq 0}$ over a domain $\mathcal{D}$ (for $\mathcal{D}$ the set of integers or a finite field) whose graph:

$$\{(a_1, \dots, a_n, f_n(a_1, \dots, a_n)) \mid n \geq 0,\ a_i \in \mathcal{D},\ 1 \leq i \leq n\}$$

is in the class $\mathsf{NE} = \mathsf{NTIME}[2^{O(n)}]$.

- For finite fields, we show that derandomization of PIT implies that some polynomial $F \in$ ml-NE does not have polynomial sized arithmetic circuits over the same field, nor does any polynomially bounded power of F.
  This is the first result getting circuit lower bounds from derandomization of a randomized PIT algorithm over fixed *finite* fields[1].

[1] Kabanets and Impagliazzo [14] prove a related but weaker version of the hardness-to-pseudorandomness

- We improve on the advice requirements for known connections between derandomization of PIT over the integers and arithmetic circuit lower bounds over the integers.
- Impagliazzo et al. [11] showed that (even nondeterministic) polynomial-time derandomization of promise-BPP would imply a Boolean circuit lower bound for NEXP. We strengthen this to the circuit lower bound for the smaller class NEXP $\cap$ coNEXP. This is the same class where sufficiently strong circuit lower bounds would imply that promise-BPP $\subseteq$ NP.
- [11] also showed that a Boolean circuit lower bound for NEXP would follow from the existence of an NP-computable non-empty property of Boolean functions that is useful for proving superpolynomial circuit lower bounds (in the sense of natural proofs of Razborov and Rudich [20]). We also strengthen this to the lower bound for the class NEXP $\cap$ coNEXP.
- The two notions of inclusion and hardness most frequently used in complexity are "every length" inclusion and hardness, and "infinitely often" inclusion and hardness. Unfortunately, the negation of everywhere inclusion is only infinitely often hardness, and vice versa. This prevents many of the connections above from being literal equivalences between lower bounds and derandomization. Using a third notion of inclusion, a variant of the "robustly often" inclusion of [9], we make all four of the above connections into literal equivalences[2].

1.2 Overview and related work

How can we argue that an upper bound on algorithmic complexity (efficient derandomization) yields a lower bound on algorithmic complexity (circuit lower bounds)? The various results in this area all follow the same general template (see, e.g., [3] for a formal treatment): We assume that we have both a circuit upper bound for some large class such as NEXP, and a general derandomization technique, and get a contradiction as follows:

Simulation: Meyer, Karp and Lipton [17] introduced techniques to give consequences of non-uniform (circuit) upper bounds for uniform classes. Using interactive proofs [6, 23], many of these can be extended to show that if a large class C such as EXP or PSPACE has small circuits, then C also has short (constant-round) interactive proofs (see, e.g., [5]).

Derandomization: Invoking the generic derandomization technique in the above simulation, we get that C can be simulated only with non-determinism.

Contradict a known lower bound: If C is NEXP itself, as in [11] and [27], the contradiction comes from some version of the non-deterministic time hierarchy theorem. An alternate method used by [1] is to pad up the non-trivial simulation of C in small non-deterministic time to show that some analogous class superpoly-C can be simulated in NEXP. The class superpoly-C will often be strong enough that a lower bound in that class can be shown by direct diagonalization (see, e.g., [16]). By simulation, NEXP will inherit the same lower bound.

result for finite fields: their arithmetic-complexity hardness assumption is not for a polynomial (defined over all extension fields), but rather for the function that *agrees* with the polynomial over a *particular* extension field.

2 [13] has an alternative method of getting an equivalence for the case of PIT over the *integers*, but at the expense of moving to versions of the classes with *advice*.

If both the circuit upper bound assumed and the derandomization assumed are worst-case, then the above template can be filled out to get a variety of trade-offs. However, when one or the other is only assumed to occur for infinitely many lengths of input, care needs to be taken to compare the input sizes where the simulation is possible, the corresponding lengths where derandomization is possible, and what non-worst-case versions of the known lower bounds are true.

While the general issue of connections between circuit complexity and derandomization has been the subject of intense research by many people, the papers most directly parallel to this work are [11, 14, 1, 13, 18]. Here, we briefly describe the main results and techniques of these five papers, and compare them to our results and techniques.

Promise-BPP. [11] considers the question of derandomizing $\mathsf{promise\text{-}BPP}$. While there are several other results in the paper, the main result for our purposes is an equivalence between a circuit lower bound for NEXP and a "non-trivial" simulation of $\mathsf{promise\text{-}BPP}$.

▶ **Theorem 1** ([11]). $\mathsf{NEXP} \not\subset \mathsf{P/poly}$ *if and only if* $\mathsf{promise\text{-}BPP} \subseteq \cap_{\epsilon>0}\mathsf{io\text{-}NTIME}[2^{n^\epsilon}]/n^\epsilon$.

While this is a strong result, and in fact an equivalence, there are some questions raised by the details in this statement. Since $\mathsf{promise\text{-}BPP} \subseteq \mathsf{P/poly}$, we know that advice can be powerful in this context. How significant is the small amount of advice allowed in the sub-exponential time algorithms for $\mathsf{promise\text{-}BPP}$? Is it really necessary or just a by-product of the proof? Can we get a stronger conclusion if we assume a Circuit Acceptance Probability Problem (estimating the acceptance probability of a size-n Boolean circuit to within an additive error of at most $1/n$; *CAPP*) algorithm with no advice? Also, [11] use the "Easy Witness Lemma" to obtain their result, which is itself derived by a somewhat convoluted indirect argument. Is this use of the Easy Witness Lemma necessary?

Here, we give some answers to these questions. Using arguments parallel to those in [1, 18] with respect to PIT, we remove the Easy Witness Lemma. Then (as [1] did for PIT) we show that *sufficiently strong nondeterministic algorithms for* $\mathsf{promise\text{-}BPP}$ *(without advice) imply the stronger conclusion that* $(\mathsf{NEXP} \cap \mathsf{coNEXP}) \not\subseteq \mathsf{P/poly}$ (Theorem 5). So there does seem to be a real difference between derandomizations with and without advice.

One use of advice in the original argument is that, if we have a different CAPP algorithm for every $\epsilon > 0$, then, for each length, the best value of ϵ and the algorithm that achieves that ϵ can be both given as advice. To remove this advice, we need to have a *single* algorithm that runs in nondeterministic sub-exponential time, $2^{n^{\epsilon(n)}}$ for some *computable* $\epsilon(n) \in o(1)$. This seems an equally natural definition of a problem being computable in sub-exponential time, and we adopt it throughout our paper. We prove some closure properties and normal forms for this notion that might be of independent interest.

Circuit lower bounds and nontrivial useful properties. Razborov and Rudich [20] defined an NP-constructive property useful against $\mathsf{P/poly}$ to be an NP-computable predicate on 2^n-bit inputs (the truth tables of n-variate Boolean functions) such that, whenever the predicate holds for infinitely many input lengths of a given family of Boolean functions $f = \{f_n\}_{n\geq 0}$, it follows that $f \notin \mathsf{P/poly}$. Call such a property *nontrivial* if there are infinitely many input lengths n such that at least one truth table of size 2^n satisfies the property.

It was shown in [11] that the existence of a nontrivial NP-constructive property useful against $\mathsf{P/poly}$ implies that $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$. We strengthen this to the circuit lower bound for $\mathsf{NEXP} \cap \mathsf{coNEXP}$ for nontrivial NP-constructive properties useful against computably superpolynomial circuit size. Moreover, we make this connection into an equivalence, using a variant of the "robustly often" notion of [9] (Theorem 7).

Polynomial identity testing over the rationals. [14] extends the connections between derandomization and circuit lower bounds to the arithmetic-complexity setting. In one direction, they show that if both $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ and the permanent function has polynomial-size arithmetic circuits over the rationals, then no nondeterministic sub-exponential time algorithm can solve PIT, even for infinitely many input sizes. In the other direction, either $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$ or the permanent requiring super-polynomial sized circuits gives some nontrivial algorithm for PIT, although not quite a direct converse.

This again raises some questions. Stating the result in the contrapositive, a nontrivial algorithm for PIT would yield either a Boolean circuit lower bound or an arithmetic circuit lower bound. Intuitively, Boolean circuit lower bounds should be more difficult, so can we get a similar result that only mentions arithmetic circuit lower bounds?

This question was addressed in [1, 18] and [13]. First, [1] and [18] give an alternate version of the final contradiction step that allows them to avoid the Easy Witness Lemma. This not only simplified the proof, but allowed them to replace the condition that $\mathsf{NEXP} \subset \mathsf{P/poly}$ with the weaker condition that $(\mathsf{NEXP} \cap \mathsf{coNEXP}) \subset \mathsf{P/poly}$. ([14] had also shown that $(\mathsf{NEXP} \cap \mathsf{coNEXP}) \subset \mathsf{P/poly}$ and the easiness of the permanent sufficed to show $\mathsf{PIT} \not\in \mathsf{NP}$.) [13] show that the Boolean and arithmetic lower bounds can be in some sense combined. They show that derandomizing a certain version of PIT, $\mathsf{low\text{-}PIT}$, in non-deterministic sub-exponential time is equivalent to proving a super-polynomial lower bound for circuits with restricted degrees on a multi-linear function whose values can be computed in $\mathsf{NEXP}/O(n)$. However, being determined by the correct advice, the functions where the arithmetic lower bound would hold in their result are somewhat non-constructive, and when the advice is incorrect, the corresponding functions might not be defined at all, so they cannot be combined into one universal function for each length.

We show that by using the [1, 18] type contradiction step, the two lower bounds can be replaced by a *single arithmetic circuit lower bound for a multilinear polynomial definable in* $\mathsf{NEXP} \cap \mathsf{coNEXP}$, thus removing this somewhat awkward non-uniformity.

Polynomial identity testing over a fixed finite field. The polynomial identity testing problem and arithmetic circuit complexity are also important over other fields, such as fixed finite fields. Here one fixes some constant-size finite field $\mathbb{F}$, and then asks for an efficient algorithm to decide if a given arithmetic circuit C over $\mathbb{F}$, defining a low-degree polynomial p (over $\mathbb{F}$ and any finite field extending $\mathbb{F}$), is such that $p \equiv 0$. The latter can be easily decided by a randomized polynomial-time Schwartz-Zippel algorithm evaluating p on random points from a sufficiently large (larger than the degree of p) extension field of $\mathbb{F}$.

Do similar connections between hardness and pseudorandomness hold for fixed finite fields? Before the work here, almost no such connections were known.

There are a number of obstacles to directly translating the [14] techniques to fixed finite fields. First, in the "derandomization of PIT to arithmetic hardness" direction, [14] use the clean algebraic recursive "expansion by minors" definition of the permanent, a problem complete for $\#\mathsf{P}$. While the same recursion holds for the permanent over finite fields, permanent over finite fields is not known to be $\#\mathsf{P}$ complete, or even NP-hard (under deterministic reductions).

Secondly, in the "arithmetic hardness to derandomization of PIT" direction, if the hitting set generator from [14] fails using a given f as the hard function over a finite field, it only follows that some power of f had a small arithmetic circuit, not the f itself. Though this power of f can be manipulated to obtain an arithmetic circuit that agrees with f over some particular extension field, the polynomial computed by this circuit would not be identically equivalent to f (see Section 2.5 for the difference between testing for identity and agreement of polynomials over a finite field). Therefore, a power of f could still have small arithmetic circuits and this would not contradict the hardness assumption used by [14] in the hardness to randomness direction. Thus, [14] has only a weak hardness-to-randomness result in the case of circuits over finite fields.

We give the first proof that *nondeterministic polynomial identity testing over a fixed finite field yields an arithmetic circuit lower bound* (Theorem 2). We avoid the permanent by showing a simulation lemma for an even larger class, PSPACE, by using a version of a PSPACE-complete function of [25] with a similar algebraic recursive definition. We observe that, not only would a small circuit for this function itself yield the required simulation, but also any circuit for a small power of the function would have a similar consequence. This matches the type of arithmetic lower bounds needed to derandomize polynomial identity testing using the hitting set generator of [14]. Thus, the correct connection is between derandomizing PIT and proving lower bounds on circuits that compute powers of polynomials.

Equivalence between circuit lower bounds and derandomization. While the results above come closer to showing that each of these three derandomization problems are equivalent to a corresponding lower bound, they are not literal equivalences. This is because of the distinction between infinitely often computation and worst-case computation. Making worst-case assumptions about algorithms is necessary to get even infinitely often hardness, but infinitely often hardness only suffices to get algorithms that work infinitely often.

To make versions of our results that are literal equivalences, we consider an intermediate notion between worst-case and infinitely-often computation, robustly-often computation introduced by [13]. Informally, an algorithm solves a given problem *robustly often* if there are infinitely many *intervals* of superpolynomially many input lengths where the algorithm is correct on each length in the interval.

While this, somewhat less standard, notion of inclusion (as well as the related notion of separation) is exactly what we need to make the connections between circuit lower bounds and derandomization into equivalences (Theorems 3, 4, and 6), we feel that this notion is quite natural and can be justified, e.g., by the following considerations. As technology improves, in general, all computational elements will improve somewhat. But computational elements will often get more diverse, so that say, the CPU in cell phones will not be improving its computational power as quickly as the top super-computer will. A reasonable model is to assume that there are two different Moore's laws, one for "high-end" technology and one for "low-end", with different periods of doubling. Thus, high-end and low-end technologies will be polynomially related in their resources, and the intermediate technologies will span the range between them. Thus, it is natural to look at polynomially related ranges of parameters, not just single values. Robust computation does just that, looking at whether computation is possible not just on infinitely many single input lengths, but whether there are infinitely many "technology levels" of unbounded polynomial reach where this computation is possible.

Remainder of the paper. The definitions are in Section 2. We formally state our main results with proof sketches in Section 3, see the full version for complete proofs.

2 Definitions

2.1 Arithmetic circuit complexity classes

For a finite field $\mathbb{F}$, define $\mathsf{ASize}_{\mathbb{F}}[s(n)]$ to be the class of all families of n-variate polynomials $\{f_n\}$ over $\mathbb{F}$ such that, for all sufficiently large n, the polynomial f_n is computed by some arithmetic circuit C_n of size at most $s(n)$ over $\mathbb{F}$ (i.e., the polynomials $f_n(x_1, \dots, x_n)$ and $C_n(x_1, \dots, x_n)$ are formally the same when viewed as the sums of monomials). Over the integers, the class $\mathsf{ASize}_{\mathbb{Z}}[s(n)]$ is defined analogously.

We denote by $\mathsf{ASizeDeg}_{\mathbb{F}}[s(n)]$ the subclass of $\mathsf{ASize}_{\mathbb{F}}[s(n)]$ of families of n-variate polynomials f_n that have degree at most $s(n)$. Over the integers, we denote by $\mathsf{ASizeDeg}_{\mathbb{Z}}[s(n)]$ the restricted class of polynomial families $\{f_n\}$ that have *formal degree*[3] at most $s(n)$.

"Easy" polynomials. Over the integers, we shall consider a polynomial f "easy" if some constant multiple $c \cdot f$ is computable by a "small" arithmetic circuit. More formally, we define $\mathsf{ASizeDegMul}[s(n)]$ as the class of polynomial families $\{f_n\}$ over $\mathbb{Z}$ of degree at most $s(n)$ such that, for some function $g : \mathbb{N} \to \mathbb{N}$ we have $\{g(n) \times f_n\} \in \mathsf{ASizeDeg}_{\mathbb{Z}}[s(n)]$ and g is computed by a family of variable-free $\mathsf{ASizeDeg}_{\mathbb{Z}}[s(n)]$ circuits. Thus, for each input length n, we could compute a *different* constant multiple of f. $\mathsf{ASizeDegMul}$ is equivalent to the class $ASIZEDEG'$ of [13].

Over finite fields, we shall consider a polynomial f "easy" if some bounded power f^d of f is computable by a "small" arithmetic circuit. More formally, over a finite field $\mathbb{F}$, we define the class $\mathsf{ASizeDegPow}_{\mathbb{F}}[s(n)]$ as the class of polynomial families $\{f_n\}$ over $\mathbb{F}$ of degree at most $s(n)$ such that, for some function $d : \mathbb{N} \to \mathbb{N}$ with $d(n) \leq s(n)$, we have $\{f_n^{d(n)}\} \in \mathsf{ASizeDeg}_{\mathbb{F}}[s(n)]$.

2.2 Polynomials computable in NE: The class ml-NE

Fix an arbitrary finite field $\mathbb{F} = \mathbb{F}_{p^k}$ of characteristic p. Let $f = \{f_n(x_1, \dots, x_n)\}_{n \geq 0}$ be an arbitrary family of polynomials over $\mathbb{F}$. For a polynomial f and a monomial m, denote by $\mathsf{coeff}(f, m)$ the coefficient of f at the monomial m (when f is written out as the sum of its monomials). Define the language$\textsc{monomial}(f) = \{(a_1, \dots, a_n, c) \mid a_1, \dots, a_n \in \{0, 1\},\ c \in \mathbb{F},\ \mathsf{coeff}(f_n, x_1^{a_1} \dots x_n^{a_n}) = c\}$. We denote by $\textsc{monomial}(f_n)$ the restriction of $\textsc{monomial}(f)$ to the case of f_n, i.e., $\textsc{monomial}(f_n)$ defines the coefficients of the n-variate polynomial f_n. For multilinear polynomial families f over the integers, the language $\textsc{monomial}(f)$ is defined similarly, with the natural change that the coefficient c be an integer in binary.

We say that a family of multilinear polynomials $f = \{f_n\}_{n \geq 0}$ over a finite field $\mathbb{F}$ or over integers $\mathbb{Z}$ is in the class $\mathsf{ml\text{-}NE}$ if $\textsc{monomial}(f) \in \mathsf{NE}$.

Observe that if $\textsc{monomial}(f) \in \mathsf{NE}$, then $\textsc{monomial}(f) \in \mathsf{coNE}$ also. Thus, we have $\textsc{monomial}(f) \in \mathsf{NE} \iff \textsc{monomial}(f) \in (\mathsf{NE} \cap \mathsf{coNE})$, and so $\mathsf{ml\text{-}NE} = \mathsf{ml\text{-}(NE \cap coNE)}$; the same equivalence holds also for the definition of $\mathsf{ml\text{-}NE}$ used by Jansen and Santhanam [13][4].

2.3 Computably subexponential and superpolynomial bounded classes

We call a function $\alpha \colon \mathbb{N} \to \mathbb{R}^{\geq 0}$ *computably super-constant*, denoted by $\alpha(n) \in \omega_c(1)$, if there exists computable, monotone non-decreasing function $\alpha' \colon \mathbb{N} \to \mathbb{R}^{\geq 0}$ with $\alpha'(n) \to \infty$ such that, for all sufficiently large $n \in \mathbb{N}$, $\alpha(n) \geq \alpha'(n)$. Without loss of generality, we may always assume that our computably super-constant functions $\alpha(n)$ are computable in time $\mathsf{poly}(n)$ (see full version for details).

[3] For input gates, regardless of their label, the formal degree is 1; an addition gate has the formal degree equal to the maximum of the formal degree of its input gates; a multiplication gate has the formal degree equal to the sum of the formal degrees of its input gates.

[4] The graph definition of $\mathsf{ml\text{-}NE}$ used by [13] and the monomial-coefficient definition of $\mathsf{ml\text{-}NE}$ given here are equivalent because of efficient interpolation. We use the coefficient definition in our work just for convenience, as it makes it obvious that one can evaluate polynomials over extensions of finite fields.

The standard definition of $\mathsf{NSUBEXP}$ is $\bigcap_{\epsilon>0} \mathsf{NTIME}[2^{n^{\epsilon}}]$. This is problematic when we need to run a padding argument for some language in this class, because the amount of padding required will depend on ϵ. There is no uniform way to produce the specific ϵ required for a particular amount of padding, so any padding argument must rely on advice to use the correct value of ϵ. We use our notion of computably super-constant functions to trade a more restricted class for freedom from advice, and define computably subexponential-time classes as follows: We say that a language L is in nondeterministic computably subexponential time, denoted $L \in \mathsf{NSUBEXP}_c$, if $L \in \mathsf{NTIME}[2^{n^{1/\alpha(n)}}]$ for some $\alpha(n) \in \omega_c(1)$. We denote by $\mathsf{superpoly}_c$ any function $n^{\alpha(n)}$ for some $\alpha(n) \in \omega_c(1)$.

2.4 Robustness

To establish equivalences between circuit lower bounds and derandomization, we need notions that are intermediate between infinitely-often and almost-everywhere. In the spirit of [9], we define "robust" notions of containment and separation for complexity classes. Our robust ranges are some fixed *computably super-polynomial* function in length, whereas the ranges of [9] are for every fixed polynomial function. The reason we require this alternative notion is because the simulation steps of our arguments use superpolynomial amounts of padding. The [9] definition handles fixed polynomial-time translations or translations with fixed polynomial advice, but not superpolynomial translations. Our definition is an attempt to make the minimal extention possible to [9] that can handle superpolynomial translations. Thus we do not expect our results to hold under the [9] robustness notions, without major technical innovation in hardness-randomness tradeoffs that dispenses with the necessity for superpolynomial padding.

For functions $l, r : \mathbb{N} \to \mathbb{N}$, called left and right "interval functions", define the (l, r)-*core* of a set $S \subseteq \mathbb{N}$ to be the set of intervals $[l(m), r(m)]$ that are entirely contained in S, i.e., $\mathsf{core}(S) = \{m \in \mathbb{N} \mid \forall n \in \mathbb{N}\ (l(m) \le n \le r(m) \implies n \in S)\}$. A set S is called (l, r)-*robust* if the (l, r)-core of S is infinite. Finally, we say that S is *computably robust* if S is $(m^{1/\alpha(m)}, m^{\alpha(m)})$-robust for some $\alpha \in \omega_c(1)$. For brevity, we shall call such a set S simply *α-robust.*

Robust inclusions. For a language L and a complexity class $\mathcal{C}$, we say that L is *uniform robustly often in* $\mathcal{C}$, denoted $L \in \mathsf{ro}_\star\text{-}\mathcal{C}$, if there is a language $N \in \mathcal{C}$ such that the set $S = \{n \in \mathbb{N} \mid L_n = N_n\}$ is computably robust, where $L_n = L \cap \{0,1\}^n$ is the nth slice of L. Our definition is "uniform" compared to the notion of [9] because the notion defined there has interval lengths that are defined by every fixed polynomial function – this is an infinite (but very regular) set of functions giving interval lengths. Our robust sets have a fixed pair of interval functions.

We say that a family $f = \{f_n\}$ of multilinear polynomials (over a finite field $\mathbb{F}$ or over integers $\mathbb{Z}$) is in $\mathsf{ro}_\star\text{-ml-NE}$, *robustly often* $\mathsf{ml\text{-}NE}$, if, for some NE machine M, the set $S = \{n \in \mathbb{N} \mid M$ correctly decides $\textsc{monomial}(f_n)\}$ is computably robust.

Robust promise classes. For a language L and a *semantic* complexity $\mathcal{C}$, we say that L is in *uniform robustly promise* $\mathcal{C}$, denoted $L \in \mathsf{rp}_\star\text{-}\mathcal{C}$, if there is a Turing machine M such that

$$S = \{n \in \mathbb{N} \mid \text{ for all } x \in \{0,1\}^n,\ M(x) \text{ is a } \mathcal{C}\text{-type machine and } M(x) \text{ decides if } x \in L_n\}$$

is computably robust.

▶ Remark. In general, $\mathsf{ro}_\star\text{-}\mathcal{C}$ and $\mathsf{rp}_\star\text{-}\mathcal{C}$ are different for a semantic class $\mathcal{C}$. For example, $L \in \mathsf{ro}_\star\text{-}(\mathsf{NE} \cap \mathsf{coNE})$ if there is a language $N \in (\mathsf{NE} \cap \mathsf{coNE})$ that robustly often agrees with L. That is, there is a pair of NE machines M_1 and M_2 such that, for every $x \in \{0,1\}^n$, exactly one of M_1 and M_2 accepts x, and $S = \{n \in \mathbb{N} \mid M_1 \text{ decides } L_n \text{ and } M_2 \text{ decides } \overline{L_n}\}$ is computably robust, where $\overline{L_n}$ is the complement of L_n.

In contrast, $L \in \mathsf{rp}_\star\text{-}(\mathsf{NE} \cap \mathsf{coNE})$ if there is a pair of NE machines M_1 and M_2 such that $S = \{n \in \mathbb{N} \mid M_1 \text{ decides } L_n \text{ and } M_2 \text{ decides } \overline{L_n}\}$ is computably robust. Note that, in the second case, the machines M_1 and M_2 may not be "complementary" on the input lengths outside of S, and so we may not have any language $N \in (\mathsf{NE} \cap \mathsf{coNE})$ that agrees with L robustly often: in the $\mathsf{rp}_\star$- case, the promise is not required to hold for slices outside the robust set.

Significant separations. To complement the inclusion types above, we define *uniform significant separations* denoted $\mathsf{ro}_\star\text{-}\mathcal{C} \not\subseteq \mathsf{SIZE}[s(n)]$. We write $\mathsf{ro}_\star\text{-}\mathcal{C} \not\subseteq \mathsf{SIZE}[s(n)]$ if there is a language $L \in \mathsf{ro}_\star\text{-}\mathcal{C}$ over computably robust set S such that L_n cannot be computed by a circuit of size $s(n)$ for infinitely many values $n \in \mathsf{core}(S)$.

Intuitively, a uniform significant separation means that we can always locate hard lengths for the separated class in the "middle" of large, computably robust intervals. This is different from the [9] notion, which says (intuitively) that hard lengths are never too far apart. Under our definition, if the robust set comes with a promise, this means that hard lengths are located in the center of large ranges where the promise holds. This is what our arguments for equivalence will hinge on. The generalization of this definition to arithmetic circuits and uniform robust promise classes is obvious. For example, we define two uniform significant separations below.

We say that $\mathsf{ro}_\star\text{-ml-NE} \not\subseteq \mathsf{ASize}[s(n)]$ if there is a polynomial family $f = \{f_n\} \in \mathsf{ro}_\star\text{-ml-NE}$, with MONOMIAL$(f)$ correctly decided by some NE machine on a computably robust set S, such that f_n cannot be computed by an arithmetic circuit of size $s(n)$ for infinitely many values $n \in \mathsf{core}(S)$. The case of other arithmetic circuit classes ($\mathsf{ASizeDeg}$ and $\mathsf{ASizeDegPow}$) is similar.

We say that $\mathsf{rp}_\star\text{-}(\mathsf{NE} \cap \mathsf{coNE}) \not\subseteq \mathsf{SIZE}[s(n)]$ if there is a promise problem $L \in \mathsf{rp}_\star\text{-}(\mathsf{NE} \cap \mathsf{coNE})$, with a pair of NE machines correctly deciding L and $\bar{L}$ over a computably robust set S of input lengths, such that L_n cannot be computed by a Boolean circuit of size $s(n)$ for infinitely many input lengths $n \in \mathsf{core}(S)$.

2.5 Derandomization of Polynomial Identity Testing

Let $\mathbb{F}$ be any finite field. For $D \in \{\mathbb{Z}, \mathbb{F}\}$, the Polynomial Identity Testing over D, denoted PIT_D, is the task to decide if a given arithmetic circuit C over D computes the identically zero polynomial (when the polynomial C is expanded as the sum of monomials[5]). The $\text{low-}\mathsf{PIT}_D$ variant is restricted to test only circuits that have degree (or, in the case of $\mathbb{Z}$, formal degree) less than their size.

A *hitting set generator* for PIT_D is a function $\mathcal{H}$, that on input 1^n, outputs a collection of t tuples $a_1, \dots, a_t \in D^n$ such that, for every arithmetic circuit $C(x_1, \dots, x_n)$ over D of

[5] Stating this definition using sums-of-monomials is crucial, because over a finite field $\mathbb{F}$, testing if a polynomial *agrees with* the zero polynomial for all inputs over $\mathbb{F}$ is actually coNP-complete. The sum-of-monomials definition that we use puts $\mathsf{PIT}_{\mathbb{F}} \in \mathsf{BPP}$, and thus it is meaningful to ask for a derandomization of this problem.

size at most n, if $C \not\equiv 0$, then there is at least one $i \in [t]$ where $C(a_i) \neq 0$. In case of a finite field $\mathbb{F}$, we allow the tuples $a_1, \ldots, a_t$ to come from $(\mathbb{F}')^n$ for some extension field $\mathbb{F}'$ of $\mathbb{F}$.

It was shown by [14] that a multilinear polynomial $f \notin \mathsf{ASizeDegPow}_{\mathbb{F}}[\mathsf{superpoly}_c]$ can be used to derandomize $\mathsf{low\text{-}PIT}_{\mathbb{F}}$ in subexponential time (by constructing a hitting set).

2.6 Derandomization of promise-BPP

Derandomizing promise-BPP is equivalent to getting an efficient algorithm for estimating the acceptance probability of a given Boolean circuit $C(x_1, \ldots, x_n)$ of size n to within an additive error $1/n$; the latter problem is called Circuit Acceptance Probability Problem (CAPP). We use the notation $\mathsf{promise\text{-}BPP} \subseteq \mathsf{NSUBEXP}_c$ to mean that there is a nondeterministic Turing machine that runs in computably subexponential time, and solves CAPP with an additive approximation error $1/n$. That is, on a given circuit $C(x_1, \ldots, x_n)$, the nondeterministic machine has at least one accepting computation, and every accepting computation yields a value r such that $|\mathbf{Pr}_{a\in\{0,1\}^n}[C(a) = 1] - r| \leq 1/n$.

We say that $\mathsf{promise\text{-}BPP} \subseteq \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c$ if a $\mathsf{NSUBEXP}_c$ algorithm correctly approximates CAPP only robustly often.

A collection $a_1, \ldots, a_t \in \{0,1\}^n$ is a *discrepancy set* of size t for circuits of size n if, for every Boolean circuit C of size n on n inputs, $\left|\mathbf{Pr}_{z\in\{0,1\}^n}[C(z) = 1] - \mathbf{Pr}_{i\in[t]}[C(a_i) = 1]\right| \leq 1/n$. It was shown by [5] that the truth table of a superpolynomial circuit-complexity Boolean function can be used to get a subexponential-size discrepancy set (in subexponential time).

3 Our results

3.1 PIT vs. arithmetic circuit lower bounds

▶ **Theorem 2.** *Fix an arbitrary finite field* $\mathbb{F}$*. We have*

1. $\mathsf{low\text{-}PIT} \in \mathsf{NSUBEXP}_c \Rightarrow \mathsf{ml\text{-}NE} \not\subseteq \mathsf{ASizeDegPow}[\mathsf{superpoly}_c]$.
2. $\mathsf{ml\text{-}NE} \not\subseteq \mathsf{ASizeDegPow}[\mathsf{superpoly}_c] \Rightarrow \mathsf{low\text{-}PIT} \in \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c$.

Proof sketch. (1) Assume a nondeterministic subexponential-time algorithm for low-PIT, but that ml-NE has "small" arithmetic circuits. We arithmetize TQBF to get a PSPACE-complete multilinear polynomial $f = \{f_n\}$ over $\mathbb{F}$. This polynomial f turns out to be computable in ml-NE, and so, by our assumption, some powers $f_n^{d_n}$, for "small" d_n, have small arithmetic circuits over $\mathbb{F}$.

For each n, we nondeterministically guess a small circuit and a small d_n. Using the ideas of the PSPACE $=$ IP proof, we then verify that the guessed circuit computes the polynomial $f_n^{d_n}$. This verification algorithm uses our assumed low-PIT algorithm, and runs in $\mathsf{NSUBEXP}_c$.

Computing powers $f_n^{d_n}$ is still PSPACE-complete, and so we get $\mathsf{PSPACE} \subseteq \mathsf{NSUBEXP}_c$. By padding, we obtain $\mathsf{SPACE}[\mathsf{superpoly}_c] \subseteq \mathsf{NE}$. By diagonalization, $\mathsf{SPACE}[\mathsf{superpoly}_c]$ contains a language L of some computably superpolynomial Boolean circuit complexity, almost everywhere. It follows that the multilinear extension of L over $\mathbb{F}$ requires arithmetic circuits of computably superpolynomial size, almost everywhere. On the other hand, each coefficient of this multilinear polynomial is computable in $\mathsf{SPACE}[\mathsf{superpoly}_c]$, and hence in NE.

(2) Assume that some family $g = \{g_n\}$ of polynomials in ml-NE is such that all powers $g_n^{d_n}$, for "small" d_n, require superpolynomial arithmetic circuit complexity for infinitely many input lengths n. By [14], we get that low-PIT is in $\mathsf{NSUBEXP}_c$ infinitely often. The input

lengths where low-PIT is easy (derandomized) correspond to the (smaller) input lengths where the polynomials g_n are actually hard.

To improve this "infinitely often" result to the "robustly often" one, we do the following: when asked to derandomize low-PIT for a certain input length n, we go to the related smaller length n', and consider polynomials over a superpolynomial interval of input lengths above n' as candidate hard polynomials; we use each such polynomial to construct a candidate hitting set by [14]. If the given interval above n' contains a length m such that g_m is hard, then our derandomization succeeds. Since there infinitely many intervals containing a length m where g_m is hard, there will be infinitely many intervals of superpolynomial length where our low-PIT algorithm is correct. ◀

▶ **Theorem 3.** *Fix an arbitrary finite field $\mathbb{F}$. We have*

$$\mathsf{low\text{-}PIT} \in \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c \;\Leftrightarrow\; \mathsf{rp}_\star\text{-}\mathsf{ml}\text{-}\mathsf{NE} \not\subseteq \mathsf{ASizeDegPow}[\mathsf{superpoly}_c].$$

Proof sketch. ($\Rightarrow$) We start as in the proof of Theorem 2, implication (1). We get a PSPACE-complete multilinear polynomial $f = \{f_n\}$ over $\mathbb{F}$ such that some powers $f_n^{d_n}$ are computable by small arithmetic circuits, for almost all input lengths n. Since our low-PIT algorithm is correct for infinitely many superpolynomial intervals of input lengths, we can guarantee the successful verification of an arithmetic circuit for $f_n^{d_n}$ for the corresponding superpolynomial intervals of input lengths only. This yields $\mathsf{PSPACE} \subseteq \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c$, and by padding, $\mathsf{SPACE}[\mathsf{superpoly}_c] \subseteq \mathsf{ro}_\star\text{-}\mathsf{NE}$. Finally, by diagonalization and multilinear extension, we get a family of multilinear polynomals g_n over $\mathbb{F}$ that require computably superpolynomial arithmetic circuit complexity almost everywhere, and yet we can compute the coefficients of g_n in NE for infinitely many superpolynomial intervals of input sizes n.

($\Leftarrow$) As in the proof of Theorem 2, implication (2), we will use hard polynomials to derandomize low-PIT by [14]. The difference now is that a given NE machine computes a valid polynomial only over some computably robust set S of input lengths n, and that this polynomial is hard only for infinitely many lengths $n \in \mathsf{core}(S)$. Still we can use this NE machine to construct a candidate hitting set for a given low-PIT instance so that, for infinitely many lengths $n \in \mathsf{core}(S)$, we get a correct hitting set, and so $\mathsf{low\text{-}PIT} \subseteq \mathsf{io}\text{-}\mathsf{NSUBEXP}_c$. To boost this to the robustly often inclusion, we employ a similar trick as before: use a superpolynomial-size interval of input lengths to get a collection of candidate hitting sets, and take their union. When all input lengths fall into an interval where our NE machine computes a valid polynomial, we get that the union of such candidate hitting sets is well-defined. If, in addition, the polynomial computed by our NE machine is actually hard on some length in this interval, then we get a correct hitting set. ◀

We have analogous results also for the case of integers $\mathbb{Z}$, with analogous proofs (using the analysis of [13] showing that Kaltofen's [15] polynomial factorization algorithm over integers respects formal degree). In particular, we have the following equivalence.

▶ **Theorem 4.** *Over $\mathbb{Z}$,*

$$\mathsf{low\text{-}PIT} \in \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c \;\Leftrightarrow\; \mathsf{rp}_\star\text{-}\mathsf{ml}\text{-}\mathsf{NE} \not\subseteq \mathsf{ASizeDegMul}[\mathsf{superpoly}_c].$$

3.2 Promise-BPP vs. Boolean circuit lower bounds

▶ **Theorem 5.** *We have*

1. $\mathsf{promise\text{-}BPP} \subseteq \mathsf{NSUBEXP}_c \;\Rightarrow\; (\mathsf{NE} \cap \mathsf{coNE}) \not\subseteq \mathsf{SIZE}[\mathsf{superpoly}_c]$.
2. $(\mathsf{NE} \cap \mathsf{coNE}) \not\subseteq \mathsf{SIZE}[\mathsf{superpoly}_c] \;\Rightarrow\; \mathsf{promise\text{-}BPP} \subseteq \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c$.

Proof sketch. (1) Assume that CAPP is correctly approximated by an $\mathsf{NSUBEXP}_c$ algorithm, but every language in $(\mathsf{NE} \cap \mathsf{coNE})$ has small superpolynomial-size Boolean circuits. The latter means that $\mathsf{E} \subseteq \mathsf{SIZE}[\mathsf{superpoly}_c]$, and hence, by [4], that $\mathsf{E} \subseteq \mathsf{MA}[\mathsf{superpoly}_c]$, for a related small superpolynomial time complexity. Using our CAPP algorithm, we get $\mathsf{MA}[\mathsf{superpoly}_c] \subseteq \mathsf{NSUBEXP}_c$. By padding the inclusion $\mathsf{E} \subseteq \mathsf{NSUBEXP}_c$, we get that $\mathsf{TIME}[2^{\mathsf{superpoly}_c}] \subseteq \mathsf{NE}$. Finally, by diagonalization, we get a language $L \in \mathsf{TIME}[2^{\mathsf{superpoly}_c}] \subseteq \mathsf{NE}$ that requires computably superpolynomial circuit complexity (almost everywhere). Since $\bar{L} \in \mathsf{TIME}[2^{\mathsf{superpoly}_c}] \subseteq \mathsf{NE}$, the conclusion follows.

(2) Assume we have a pair of NE machines that compute L and $\bar{L}$ for a language L requiring $\mathsf{superpoly}_c$-size circuits infinitely often. By the hardness-randomness tradeoff of [5], we get a $\mathsf{NSUBEXP}_c$ algorithm that correctly approximates CAPP infinitely often; the input lengths where the CAPP algorithm is correct correspond to the (smaller) input lengths where the language L is actually hard. To boost this to the desired $\mathsf{promise\text{-}BPP} \subseteq \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c$ inclusion, we use the same "interval trick" as in the arithmetic case (Theorem 2, (2)), to show that there is a $\mathsf{NSUBEXP}_c$ algorithm that, robustly often, produces a discrepancy set for circuits of size n. ◀

We extend the two implications of Theorem 5 to the equivalence, by carefully adapting the arguments above to the setting of robust inclusions and separations.

▶ **Theorem 6.** $\mathsf{promise\text{-}BPP} \subseteq \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c \Leftrightarrow \mathsf{rp}_\star\text{-}(\mathsf{NE} \cap \mathsf{coNE}) \not\subseteq \mathsf{SIZE}[\mathsf{superpoly}_c]$.

3.3 Robustly-often nontrivial useful properties

A *property* of Boolean functions is a family $\mathcal{P} = \{\mathcal{P}_n\}_{n\geq 0}$ of predicates $\mathcal{P}_n : \{0,1\}^{2^n} \to \{0,1\}$. For a function $s : \mathbb{N} \to \mathbb{N}$, we say that a property $\mathcal{P}$ is *useful against* $\mathsf{SIZE}[s]$ *at length* n if, whenever $\mathcal{P}_n$ accepts the truth table of a Boolean n-variate function $f_n : \{0,1\}^n \to \{0,1\}$, this means that f_n requires circuit size at least $s(n)$. We say that $\mathcal{P}$ is *nontrivial at length* n if $\mathcal{P}_n$ accepts at least one truth table of length 2^n. Finally, we say that $\mathcal{P}$ is *robustly-often nontrivially useful against* $\mathsf{SIZE}[\mathsf{superpoly}_c]$ *(denoted* $\mathsf{ro}_\star$*-useful)* if, for some $s(n) \in \mathsf{superpoly}_c$,

1. $S = \{n \in \mathbb{N} \mid \mathcal{P} \text{ is nontrivial at length } n\}$ is computably robust, and
2. $\mathcal{P}$ is useful against $\mathsf{SIZE}[s]$ at length n for infinitely many lengths $n \in \mathsf{core}(S)$.

As a corollary of Theorem 6, we get the following equivalence between circuit lower bounds for $\mathsf{NEXP} \cap \mathsf{coNEXP}$ and the existence of $\mathsf{ro}_\star$-useful properties.

▶ **Theorem 7.** *The following are equivalent:*

- $\mathsf{rp}_\star\text{-}(\mathsf{NE} \cap \mathsf{coNE}) \not\subseteq \mathsf{SIZE}[\mathsf{superpoly}_c] \iff$
- *there is a* NP*-computable* $\mathsf{ro}_\star$*-useful property*
- *there is a* P*-computable* $\mathsf{ro}_\star$*-useful property*

Proof. ($\Rightarrow$) Assume we have a pair of NE machines M_1 and M_0 that correctly compute L and $\bar{L}$, respectively, for some computably robust set S of input lengths, where L requires $\mathsf{superpoly}_c$ circuit size for infinitely many input lengths in $\mathsf{core}(S)$. Define a property $\mathcal{P}$ as follows:

"On input $T \in \{0,1\}^{2^n}$, guess 2^n candidate witnesses $a_1, \ldots, a_{2^n}$ of length $2^{O(n)}$ each, and check, for every $1 \leq i \leq 2^n$, if $T_i = b$, for $b \in \{0,1\}$, then $M_b(i)$ accepts a_i as a witness. If succeed for all i's, then accept. Otherwise, reject."

Clearly, $\mathcal{P}$ is NP-computable. Note that $\mathcal{P}_n$ accepts the truth table of L_n (and nothing else) for the computably robust set S of input lengths n, and so $\mathcal{P}$ is robustly-often nontrivial. Finally, since L requires circuit size $\mathsf{superpoly}_c$ for infinitely many input lengths $n \in \mathsf{core}(S)$, we get that $\mathcal{P}$ is useful against $\mathsf{SIZE}[\mathsf{superpoly}_c]$ at length n for infinitely many $n \in \mathsf{core}(S)$. Thus $\mathcal{P}$ is $\mathsf{ro}_\star$-useful.

($\Leftarrow$) Given an NP-computable property $\mathcal{P}$ that is nontrivial on a computably robust set S of input lengths n, and useful against $\mathsf{SIZE}[\mathsf{superpoly}_c]$ for infinitely many $n \in \mathsf{core}(S)$, we use $\mathcal{P}$ to nondeterministically guess a truth table T of length 2^n, nondeterministically verify that T is accepted by $\mathcal{P}_n$, and use T as a "hard" Boolean function to derandomize $\mathsf{promise\text{-}BPP}$ (using the hardness-randomness tradeoff of [5]).

We get that $\mathsf{promise\text{-}BPP} \subseteq \mathsf{io\text{-}NSUBEXP}_c$, since infinitely often we get a truth table T of superpolynomial circuit complexity. Using the "interval trick" (as in the proof of Theorem 5, implication (2)), we boost this inclusion to get $\mathsf{promise\text{-}BPP} \subseteq \mathsf{ro}_\star\text{-}\mathsf{NSUBEXP}_c$, which, by Theorem 6, implies $\mathsf{rp}_\star\text{-}(\mathsf{NE} \cap \mathsf{coNE}) \not\subseteq \mathsf{SIZE}[\mathsf{superpoly}_c]$. ◀

References

1. S. Aaronson and D. van Melkebeek. On circuit lower bounds from derandomization. *Theory of Computing*, 7(1):177–184, 2011.
2. M. Ajtai and A. Wigderson. Deterministic simulation of probabilistic constant depth circuits. In *Proceedings of the Twenty-Sixth Annual IEEE Symposium on Foundations of Computer Science*, pages 11–19, 1985.
3. B. Aydinlioglu and D. van Melkebeek. Nondeterministic circuit lower bounds from mildly derandomizing arthur-merlin games. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 269–279. IEEE, 2012.
4. L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.
5. L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.
6. L. Babai and S. Moran. Arthur-merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.
7. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13:850–864, 1984.
8. R.A. DeMillo and R.J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7:193–195, 1978.
9. L. Fortnow and R. Santhanam. Robust simulations and significant separations. In *Automata, Languages and Programming – 38th International Colloquium, ICALP, Proceedings, Part I*, pages 569–580, 2011.
10. J. Heintz and C.-P. Schnorr. Testing polynomials which are easy to compute. *L'Enseignement Mathématique*, 30:237–254, 1982.
11. R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.
12. R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *STOC'97*, pages 220–229, 1997.
13. M.J. Jansen and R. Santhanam. Stronger lower bounds and randomness-hardness tradeoffs using associated algebraic complexity classes. In Christoph Dürr and Thomas Wilke, editors, *STACS*, volume 14 of *LIPIcs*, pages 519–530. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2012.

14 V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1–2):1–46, 2004.

15 E. Kaltofen. Factorization of polynomials given by straight-line programs. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 375–412. JAI Press, Greenwich, CT, 1989.

16 R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1-3):40–56, 1982.

17 R.M. Karp and R.J. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(3-4):191–209, 1982.

18 J. Kinne, D. van Melkebeek, and R. Shaltiel. Pseudorandom generators, typically-correct derandomization, and circuit lower bounds. *Computational Complexity*, 21(1):3–61, 2012.

19 N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

20 Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

21 J.T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the Association for Computing Machinery*, 27(4):701–717, 1980.

22 R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the Association for Computing Machinery*, 52(2):172–216, 2005.

23 A. Shamir. IP=PSPACE. *Journal of the Association for Computing Machinery*, 39(4):869–877, 1992.

24 M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

25 L. Trevisan and S.P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

26 C. Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003.

27 R. Williams. Nonuniform acc circuit lower bounds. *Journal of the ACM (JACM)*, 61(1):2, 2014.

28 A.C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

29 R.E. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of an International Symposium on Symbolic and Algebraic Manipulation (EUROSAM'79)*, LNCS, pages 216–226, 1979.

Average Distance Queries through Weighted Samples in Graphs and Metric Spaces: High Scalability with Tight Statistical Guarantees

Shiri Chechik[1], Edith Cohen[1,2], and Haim Kaplan[1]

1 **Tel Aviv University**
Israel
{schechik,haimk}@cs.tau.ac.il
2 **Google Research**
Mountain View, CA, USA
edith@cohenwang.com

Abstract

The average distance from a node to all other nodes in a graph, or from a query point in a metric space to a set of points, is a fundamental quantity in data analysis. The inverse of the average distance, known as the (classic) closeness centrality of a node, is a popular importance measure in the study of social networks. We develop novel structural insights on the sparsifiability of the distance relation via weighted sampling. Based on that, we present highly practical algorithms with strong statistical guarantees for fundamental problems. We show that the average distance (and hence the centrality) for all nodes in a graph can be estimated using $O(\epsilon^{-2})$ single-source distance computations. For a set V of n points in a metric space, we show that after preprocessing which uses $O(n)$ distance computations we can compute a weighted sample $S \subset V$ of size $O(\epsilon^{-2})$ such that the average distance from any query point v to V can be estimated from the distances from v to S. Finally, we show that for a set of points V in a metric space, we can estimate the average pairwise distance using $O(n + \epsilon^{-2})$ distance computations. The estimate is based on a weighted sample of $O(\epsilon^{-2})$ pairs of points, which is computed using $O(n)$ distance computations. Our estimates are unbiased with normalized mean square error (NRMSE) of at most ϵ. Increasing the sample size by a $O(\log n)$ factor ensures that the probability that the relative error exceeds ϵ is polynomially small.

1998 ACM Subject Classification E.1 Data Structures, F.2.1 Numerical Algorithms and Problems, G.1.3 Numerical Linear Algebra, G.2.2 Graph Theory

Keywords and phrases Closeness Centrality, Average Distance, Metric Space, Weighted Sampling

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.659

1 Introduction

Measures of structural centrality based on shortest-paths distances, first studied by Bavelas [3], are classic tools in the analysis of social networks and other graph datasets. One natural measure of the importance of a node in a network is its *classic closeness centrality*, defined as the inverse of its average distance to all other nodes. This centrality measure, which is also termed *Bavelas closeness centrality* or the *Sabidussi Index* [13, 14, 24], was proposed by Bavelas [4], Beauchamp [5], and Sabidussi [20]. Formally, for a graph $G = (V, E)$ with $|V| = n$ nodes, the classic closeness centrality of $v \in V$ is

$$\textsc{cc}(v) = \frac{n-1}{\sum_{u \in V} \operatorname{dist}(u, v)} , \tag{1}$$

where $\mathrm{dist}(u, v)$ is the length of a shortest path between v and u in G and n is the number of nodes. Intuitively, this measure of centrality reflects the ability of a node to send goods to all other nodes.

In metric spaces, the average distance of a point z to a set V of n points, $\sum_{x\in V} \mathrm{dist}(z, x)/n$, is a fundamental component in some clustering and classification tasks. For clustering, the quality of a cluster can be measured by the sum of distances from a centroid (usually 1-median or the mean in Euclidean data). Consequently, the (potential) relevance of a query point to the cluster can be estimated by relating its average distance to the cluster points to that of the center or more generally, to the distribution of the average distance of each cluster point to all others. This classification method has the advantages of being non-parametric (making no distribution assumptions on the data), similarly to the popular k nearest neighbors [10] (kNN) classification. Average distance based classification complements kNN, in that it targets settings where the outliers in the labeled points do carry information that should be incorporated in the classifier. A recent study [16] demonstrated that this is the case for some data sets in the UCI repository, where average distance based classification is much more accurate than kNN classification.

These notions of centrality and average distance had been extensively used in the analysis of social networks and metric data sets. We aim here to provide better tools to facilitate the computation of these measures on very large data sets. In particular, we present estimators with tight statistical guarantees whose computation is highly scalable.

We consider inputs that are either in the form of an undirected graph (with nonnegative edge weights) or a set of points in a metric space. In case of graphs, distance of the underlying metric correspond to lengths of shortest paths. Our results also extend to inputs specified as directed strongly connected graphs where the distance are the round trip distances [6]. We use a unified notation where V is the set of nodes if the input is a graph, or the set of points in a metric space. We denote $|V| = n$. We use graph terminology, and mention metric spaces only when there is a difference between the two applications. We find it convenient to work with the sum of distances

$$\mathrm{W}(v) = \sum_{u\in V} \mathrm{dist}(v, u) \ .$$

Average distance is then simply $\mathrm{W}(v)/n$ and centrality is $\textsc{cc}(v) = (n - 1)/\,\mathrm{W}(v)$. Moreover, estimates $\hat{\mathrm{W}}(v)$ that are within a small relative error, that is $(1 - \epsilon)\,\mathrm{W}(u) \le \hat{\mathrm{W}}(u) \le (1 + \epsilon)\,\mathrm{W}(u)$, imply a small relative error on the average distance, by taking $\hat{\mathrm{W}}(v)/n$, and for centrality $\textsc{cc}(v)$, by taking $\hat{\textsc{cc}}(v) = (n - 1)/\hat{\mathrm{W}}(v)$.

We list the fundamental computational problems related to these measures.

- *All-nodes sums*: Compute $\mathrm{W}(v)$ of all $v \in V$.
- *Point queries (metric space):* Preprocess a set of points V in a metric space, such that given a query point v (any point in the metric space, not necessarily $v \in V$), we can quickly compute $\mathrm{W}(v)$.
- *1-median*: Compute the node u of maximum centrality or equivalently, minimum $\mathrm{W}(u)$.
- *All-pairs sum*: Compute the sum of the distances between all pairs, that is $\textsc{aps}(V) \equiv \frac{1}{2}\sum_{v\in V} \mathrm{W}(v)$.

In metric spaces, we seek algorithms that compute distances for a small number of pairs of points. In graphs, a distance computation between a specific pair of nodes u, v seems to be computationally equivalent in the worst-case to computing all distances from a single source node (one of the nodes) to all other nodes. Therefore, we seek algorithms that perform a small number of single-source shortest paths (SSSP) computations. An SSSP computation in a graph can be performed using Dijkstra's algorithm in time that is nearly linear in the number of edges [12]. To support parallel computation, it is also desirable to reduce dependencies between the distance or single-source distance computations.

The best known exact algorithms for the problems that we listed above do not scale well. To compute $\mathrm{W}(v)$ for all v, all-pairs sum, and 1-median, we need to compute the distances between all pairs of nodes, which in graphs is equivalent to an all-pairs shortest paths (APSP) computation. To answer point queries, we need to compute the distances from the query point to all points in V. In graphs, the hardness of some of these problems was formalized by the notion of subcubic equivalence [23]. Abboud et al [1] showed that exact 1-median is subcubic equivalent to APSP and therefore is unlikely to have a near linear time solution. We apply a similar technique and show (in Section 7) that the all-pairs sum problem is also subcubic equivalent to APSP. In general metric spaces, exact all pairs sum or 1-median clearly requires $\Omega(n^2)$ distance computations.[1]

Since exact computation does not scale to very large data sets, work in the area focused on approximations with small relative errors. We measure approximation quality by the normalized root mean square error (NRMSE), which is the square root of the expected (over randomization used in the algorithm) square difference between the estimate and the actual value, divided by the mean. When the estimator is unbiased (as with sample average), this is the ratio between the standard deviation and the mean, which is called the coefficient of variation (CV). Chebyshev's inequality implies that the probability that the estimator is within a relative error of η from its mean is at least $1-(CV)^2/(\eta)^2$. Therefore a CV of ϵ implies that the estimator is within a relative error of $\eta = c\epsilon$ from its mean with probability $\geq 1-1/c^2$.

The sampling based estimates that we consider are also well concentrated, meaning roughly that the probability of a larger error decreases exponentially. With concentration, by increasing the sample size by a factor of $O(\log n)$ we get that the probability that the relative error exceeds ϵ, for any one of polynomially many queries, is polynomially small. In particular, we can estimate the sum of the distances of the 1-median from all other nodes up to a relative error of ϵ with a polynomially small error probability.

Previous work

We review previous work on scalable approximation of 1-median, all-nodes sums, and all-pairs sum. These problems were studied in metric spaces and graphs. A natural approach to approximate the centrality of nodes is to take a uniform sample S of nodes, perform $|S|$ single source distance computations to determine all distances from every $v \in S$ to every $u \in V$, and then estimate $\mathrm{W}(v)$ by $\hat{\mathrm{W}}(v) = \frac{n}{|S|}\mathrm{W}_S(v)$, where $\mathrm{W}_S(v) = \sum_{a\in S}\mathrm{dist}(v,a)$ is the sum of the distances from v to the nodes of S. This approach was used by Indyk [18] to compute a $(1+\epsilon)$-approximate 1-median in a metric space using only $O(\epsilon^{-2}n)$ distance computations (See also [17] for a similar result with a weaker bound.). We discuss this uniform sampling approach in more detail in Section 6, where for completeness, we show how it can be applied to the all-nodes sums problem.

The sample average of a uniform sample was also used to estimate all-nodes centrality [11] (albeit with weaker, additive guarantees) and to experimentally identify the (approximate)

[1] Take a symmetric distance matrix with all entries in $(1-1/n, 1]$. To determine the 1-median we need to compute the exact sum of entries in each raw, that is, to exactly evaluate all entries in the raw. This is because an unread entry of 0 in any raw would determine the 1-median. Similarly, to compute the exact sum of distances we need to evaluate all entries. Deterministically, this amounts to $\binom{n}{2}$ distance computations.

top k centralities [19]. When the distance distribution is heavy-tailed, however, the sample average as an estimate of the true average can have a large relative error. This is because the sample may miss out on the few far nodes that dominate $\mathrm{W}(v)$.

Recently, Cohen et al [6] obtained ϵ NRMSE estimates for $\mathrm{W}(v)$ for any v, using single-source distance computations from each node in a uniform sample of ϵ^{-3} nodes. Estimates that are within a relative error of ϵ for all nodes were obtained using $\epsilon^{-3}\log n$ single-source computations. This approach applies in any metric space. The estimator for a point v is obtained by using the average of the distances from v to a uniform sample for nodes which are "close" to v and estimating distances to nodes "far" from v by their distance to the sampled node closest to v. The resulting estimate is biased, but obtains small relative errors using essentially the information of single-source distances from a uniform sample.

For the all-pairs sum problem in metric spaces, Indyk [17] showed that it can be estimated by scaling up the average of $\tilde{O}(n\epsilon^{-3.5})$ distances between pairs of points selected uniformly at random. The estimate has a relative error of at most ϵ with constant probability. Barhum, Goldreich, and Shraibman [2] improved Indyk's bound and showed that a uniform sample of $O(n\epsilon^{-2})$ distances suffices and also argued that this sample size is necessary (with uniform sampling). Barhum et al. also showed that in an Euclidean space a similar approximation can be obtained by projecting the points onto $O(1/\epsilon^2)$ random directions and averaging the distances between all pairwise projections. Goldreich and Ron [15] showed that in an unweighted graph $O(\epsilon^{-2}\sqrt{n})$ distances between random pairs of points suffice to estimate the sum of all pairwise distances, within a relative error of ϵ, with constant probability. They also showed that $O(\epsilon^{-2}\sqrt{n})$ distances from a fixed node s to random nodes v suffice to estimate $\mathrm{W}(v)$, within a relative error of ϵ, with constant probability. A difficulty with using this result, however, is that in graphs it is expensive to compute distances between random pairs of points in a scalable way: typically a single distance between a particular pair of nodes s and t is not easier to obtain than a complete single source shortest path tree from s.

Contributions and overview

Our design is based on computing a single *weighted* sample that provides estimates with statistical guarantees for all nodes/points. A sample of size $O(\epsilon^{-2})$ suffices to obtain estimates $\hat{\mathrm{W}}(z)$ with a CV of ϵ for any z. A sample of size $O(\epsilon^{-2}\log n)$ suffices for ensuring a relative error of at most ϵ for all nodes in a graph or for polynomially many queries in a metric space, with probability that is at least $1 - 1/poly(n)$.

The sampling algorithm is provided in Section 2. This algorithm computes a *coefficient* γ_v for each $v \in V$ such that $\sum_v \gamma_v = O(1)$. Then for a parameter k, we obtain sampling probabilities $p_u \equiv \min\{1, k\gamma_v\}$ for $u \in V$. Using the probabilities p_v, we can obtain a Poisson sample S of expected size $\sum_u p_u = O(k)$ or a VarOpt sample [8] that has exactly that size (rounded to an integer).

We present our estimators in Section 3. For each node u, the inverse probability estimator $\widehat{\mathrm{dist}}(z,u)$ is equal to $\mathrm{dist}(z,u)/p_u$ if u is sampled and is 0 otherwise. Our estimate of the sum $\mathrm{W}(z)$ is the sum of these estimates

$$\hat{\mathrm{W}}(z) = \sum_{u\in V}\widehat{\mathrm{dist}}(z,u) = \sum_{u\in S}\widehat{\mathrm{dist}}(z,u) = \sum_{u\in S}\frac{\mathrm{dist}(z,u)}{p_u} \ . \tag{2}$$

Since $p_u > 0$ for all u, the estimates $\widehat{\mathrm{dist}}(z,u)$ and hence the estimate $\hat{\mathrm{W}}(z)$ are unbiased.

We provide a detailed analysis in Section 4. We will show that our sampling probabilities provide the following guarantees. When choosing $k = O(\epsilon^{-2})$, $\hat{\mathrm{W}}(z)$ has CV ϵ. Moreover, the estimates have good concentration, so using a larger sample size of $O(\epsilon^{-2}\log n)$ we obtain that the relative error is at most ϵ for all nodes $v \in V$ with probability at least $1 - 1/poly(n)$.

In order to obtain a sample with such guarantees for some particular node z, the sampling probability of a node v should be (roughly) proportional to its distance $\mathrm{dist}(z, v)$ from z. Such a Probability Proportional to Size (PPS) sample of size $k = \epsilon^{-2}$ uses coefficients $\gamma_v = \mathrm{dist}(v, z)/\,\mathrm{W}(z)$ and has CV of ϵ. We will work with *approximate PPS* coefficients, which we define as satisfying $\gamma_v \geq c\,\mathrm{dist}(v, z)/\,\mathrm{W}(z)$ for some constant c. With approximate PPS we obtain a CV of ϵ with a sample of size $O(\epsilon^{-2})$. It is far from clear apriori, however, that there is a single set of *universal PPS* coefficients which are simultaneously (approximate) PPS for all nodes and are of size $\sum_v \gamma_v = O(1)$. That is, a single sample of size $O(\epsilon^{-2})$, which is independent of n and of the dimension of the space, would work for all nodes.

Beyond establishing the existence of universal PPS coefficients, we are interested in obtaining them, and the sample itself, using a near-linear computation. The dominant component of the computation of the sampling coefficients is performing $O(1)$ single-source distance computations. Therefore, it requires $O(m \log n)$ time in graphs and $O(n)$ pairwise distance queries in a metric space. A universal PPS sample of any given size k can then be computed in a single pass over the coefficients vector $\boldsymbol{\gamma}$ ($O(n)$ computation). We represent the sample S as a collection $\{(u, p_u)\}$ of nodes/points and their respective sampling probabilities. We can then use our sample for estimation using (2).

When the input is a graph, we compute single-source distances from each node in S to all other nodes in order to estimate $\mathrm{W}(v)$ of all $v \in V$. This requires $O(|S|m \log n)$ time and $O(n)$ space.

▶ **Theorem 1.** *All-nodes sums ($\mathrm{W}(v)$ for all $v \in V$) can be estimated unbiasedly as follows:*
- *With CV ϵ, using $O(\epsilon^{-2})$ single source distance computations.*
- *When using $O(\epsilon^{-2}\log n)$ single source distance computations, the probability that the maximum relative error, over the n nodes, exceeds ϵ is polynomially small.*

In a metric space, we can estimate $\mathrm{W}(x)$ for an arbitrary query point x, which is not necessarily a member of V, by computing the distances $\mathrm{dist}(x, v)$ for all $v \in S$ and applying the estimator (2). Thus, point queries in a metric space require $O(n)$ distance computations for preprocessing and $O(\epsilon^{-2})$ distance computations per query.

▶ **Theorem 2.** *We can preprocess a set of points V in a metric space using $O(n)$ time and $O(n)$ distance computations ($O(1)$ single source distance computations) to generate a weighted sample S of a desired size k. From the sample, we can unbiasedly estimate $\hat{\mathrm{W}}(z)$ using the distances between z and the points in S with the following guarantees:*
- *When $k = O(\epsilon^{-2})$, for any point query z, $\hat{\mathrm{W}}(z)$ has CV at most ϵ.*
- *When $k = O(\epsilon^{-2}\log n)$, the probability that the relative error of $\hat{\mathrm{W}}(z)$ exceeds ϵ for any of polynomially many queries z is polynomially small.*

We can also estimate all-pairs sum, using either primitive of single-source distances (for graphs) or distance computations (metric spaces).

▶ **Theorem 3.** *All-pairs sum can be estimated unbiasedly with the following statistical guarantees:*
- *CV of at most ϵ, using $O(\epsilon^{-2})$ single-source distance computations. With a relative error that exceeds ϵ with a polynomially small probability, using $O(\epsilon^{-2}\log n)$ single-source distance computations.*
- *With CV of at most ϵ, using $O(n + \epsilon^{-2})$ distance computations. With a relative error that exceeds ϵ with polynomially small probability, using $O(n + \epsilon^{-2}\log n)$ distance computations.*

The proof details are provided in Section 5. The part of the claim that uses single-source distance computations is established by using the estimate $\widehat{\text{APS}}(V) = \frac{1}{2}\sum_{z\in V}\hat{\text{W}}(z)$. When the estimates have CV of at most ϵ, even if correlated, so does the estimate $\widehat{\text{APS}}(V)$.[2] For the high probability claim, we use $O(\log n)$ single-source computations to ensure we obtain universal PPS coefficients with high probability (details are provided later), which imply that each estimate $\hat{W}(z)$, and hence the sum is concentrated.

For the second part that uses distance computations, we consider an approximate PPS distribution that is with respect to $\text{dist}(u,v)$, that is, the probability of sampling the pair (u,v) is at least $c\,\text{dist}(u,v)/\,\text{APS}(V)$ for some constant c. We show that we can compactly represent this distribution as the outer product of two probability vectors of size n. Using this representation we can draw $O(\epsilon^{-2})$ pairs independently in linear time, which we use for estimating the average.

Compared to the all-nodes sums algorithms of [6], our result here improves the dependency in ϵ from ϵ^{-3} to ϵ^{-2} (which is likely to be optimal for a sampling based approach), provides an unbiased estimates, and also facilitates approximate average distance oracles with very small storage in metric spaces (the approach of [6] would require the oracle to store a histogram of distances from each of ϵ^{-3} sampled nodes). For the all-pairs sum problem in graphs, we obtain an algorithm that uses $O(\epsilon^{-2})$ single source distance computations, which improves over an algorithm that does $O(\epsilon^{-3})$ single source distance computations implied by [6]. For the all pairs sum problem in a metric space, we obtain a CV of ϵ using $O(n+\epsilon^{-2})$ distance computation rather than $O(n\epsilon^{-2})$ distance computations required by the algorithms in [2, 17].

While our analysis does not optimize constants, our algorithms are very simple and we expect them to be effective in applications.

2 Constructing the sample

We present Algorithm 1 that computes a set of sampling probabilities associated with the nodes of an input graph G. We use graph terminology but the algorithm applies both in graphs and in metric spaces. The input to the algorithm is a set S_0 of base nodes and a parameter k (we discuss how to choose S_0 and k below). The algorithm consists of the following stages. We first compute a sampling coefficient γ_v for each node v such that $\sum_v \gamma_v = O(1)$. Then we use the parameter k and compute the sampling probabilities $p_v = \min\{1, k\gamma_v\}$. Finally we use the probabilities p_v to draw a sample of expected size $O(k)$, by choosing v with probability p_v. We usually apply the algorithm once with a pre-specified k to obtain a sample, but there are applications (see discussion in Section 8.4) in which we want to choose the sample size adaptively using the same coefficients.

Running time and sample size

The running time of this algorithm on a metric space is dominated by $|S_0|n$ distance computations. On a graph, the running time is $|S_0|m\log n$, and is dominated by the $|S_0|$ single-source shortest-paths computations. The expected size of the final sample S is $\sum_v p_v \le k\sum_v \gamma_v = O(k)$.

[2] In general if random variables X and Y have CV ϵ then so does their sum: $\frac{\mathsf{Var}(X+Y)}{(E(X+Y))^2} = \frac{\mathsf{Var}(X)+\mathsf{Var}(Y)+2\,\mathsf{Cov}(X,Y)}{(E(X+Y))^2} \le \frac{\mathsf{Var}(X)+\mathsf{Var}(Y)+2\sqrt{\mathsf{Var}(X)\,\mathsf{Var}(Y)}}{(E(X+Y))^2} \le \frac{\epsilon^2(E(X))^2+\epsilon^2(E(Y))^2+2\epsilon^2E(X)E(Y)}{(E(X+Y))^2} \le \epsilon^2$.

Algorithm 1 Compute universal PPS coefficients and sample

```
Input: Undirected graph with vertex set $V$ or a set of points $V$ in a metric space, base nodes $S_0$, parameter $k$
Output: A universal PPS sample $S$
// Compute sampling coefficients $\gamma_v$
foreach node $v$ do
    $\gamma_v \leftarrow 1/n$
foreach $u \in S_0$ do
    Compute shortest path distances $\mathrm{dist}(u,v)$ from $u$ to all other nodes $v \in V$
    $W \leftarrow \sum_v \mathrm{dist}(u,v)$
    foreach node $v \in V$ do
        $\gamma_v \leftarrow \max\{\gamma_v, \frac{\mathrm{dist}(u,v)}{W}\}$
foreach node $v \in V$ do // Compute sampling probabilities $p_v$
    $p_v \leftarrow \min\{1, k\gamma_v\}$
$S \leftarrow \emptyset$ // Initialize sample
foreach $v \in V$ do // Poisson sample according to $p_v$
    if $rand() < p_v$ then
        $S \leftarrow S \cup \{(v, p_v)\}$
return $S$
```

Choosing the base set S_0

We will show that in order to obtain the property that each estimate $\hat{\mathrm{W}}(v)$ has CV $O(\epsilon)$, it suffices that the base set S_0 includes a uniform sample of ≥ 2 nodes and we need to choose $k = \epsilon^{-2}$. Note that the CV is computed over the randomization in the choice of nodes to S_0 and of the sample we choose using the computed coeffcients. We will also introduce a notion of a *well positioned* node, which we precisely define in the sequel. We will see that when S_0 includes such a node, we also have CV of $O(\epsilon)$ with $k = \epsilon^{-2}$. This time using only the randomization in the selection of the sample. Moreover, if we choose $k = \epsilon^{-2} \log n$ and ensure that S_0 contains a well-positioned node with probability at least $1 - 1/poly(n)$ then we obtain that the probability that the relative error exceeds ϵ is polynomially small. We will see that most nodes are well positioned, and therefore, it is relatively simple to identify such a node with high probability.

3 Estimation

3.1 Centrality values for all nodes in a graph

For graphs, we compute estimates $\hat{\mathrm{W}}(v)$ for all nodes $v \in V$ as in Algorithm 2. We initialize all estimates to 0, and perform a SSSP computation from each node in $u \in S$. When scanning node v, during such SSSP computation, we add $\mathrm{dist}(u,v)/p_u$ to the estimate $\hat{\mathrm{W}}(v)$. The algorithms runs in $O(|S| m \log n)$ time, dominated by the $|S|$ SSSP computations from each node in the sample S.

3.2 Point queries (metric space)

For a query point z (which is not necessarily a member of V), we compute the distance $\mathrm{dist}(z,x)$ for all $x \in S$, and apply (2). This takes $|S|$ distance computations for each query.

Algorithm 2 Compute estimates $\hat{\mathrm{W}}(v)$ for all nodes v in the graph

```
Input: Weighted graph $G$, a sample $S = \{(u, p_u)\}$
foreach $v \in V$ do
    $\hat{\mathrm{W}}(v) \leftarrow 0$
foreach $u \in S$ do
    Perform a single-source shortest-paths computation from $u$.
    foreach scanned node $v \in V$ do
        $\hat{\mathrm{W}}(v) \leftarrow \hat{\mathrm{W}}(v) + \mathrm{dist}(u, v)/p_u$
return $(v, \hat{\mathrm{W}}(v))$ for $v \in V$
```

4 Correctness

We first show (Section 4.1) show that when $k = \epsilon^{-2}$, and S_0 includes either a uniform sample of size at least 2 then each estimate $\hat{\mathrm{W}}(v)$ has CV of $O(\epsilon)$. We then define well-positioned nodes in Section 4.2 and show that if S_0 contains a well positioned node we and sample size is $k = \epsilon^{-2}$ then the CV is $O(\epsilon)$ (Section 4.3) and when $k = O(\epsilon^{-2} \log n)$, the probability that the relative error exceeds ϵ is polynomially small (Section 4.5).

In Section 4.4 we establish an interesting property of our sampling coefficients: They can not grow too much even if the base set S_0 is very large. Clearly, $\sum_v \gamma_v \leq 1 + |S_0|$, but we will show that it is $O(1)$ regardless of the size of S_0.

We start with some useful lemmas.

▶ **Lemma 4.** *Suppose that S_0 contains a node u. Consider a node z such that u is the $(qn)^{th}$ closest node to z. Then for all nodes v,*

$$\gamma_v \geq \frac{1-q}{4} \cdot \frac{\mathrm{dist}(z, v)}{\mathrm{W}(z)} \ . \tag{3}$$

Proof. From the specification of Algorithm 1, the sampling coefficients γ_v satisfy

$$\gamma_v \geq \max\left\{\frac{1}{n}, \frac{\mathrm{dist}(u, v)}{\mathrm{W}(u)}\right\} \ . \tag{4}$$

Let $Q = \mathrm{dist}(z, u)$. Consider a classification of the nodes $v \in V$ to "close" nodes and "far" nodes according to their distance from z:

$$\begin{aligned} L &= \{v \in V \mid \mathrm{dist}(z, v) \leq 2Q\} \\ H &= \{v \in V \mid \mathrm{dist}(z, v) > 2Q\} \ . \end{aligned}$$

Since $\gamma_v \geq 1/n$, for $v \in L$ we have

$$\gamma_v \geq \frac{1}{n} \geq \left(\frac{1-q}{2}\right)\left(\frac{2}{1-q}\right)\frac{1}{n} = \left(\frac{1-q}{2}\right)\left(\frac{2Q}{(1-q)Q}\right)\frac{1}{n} \geq \left(\frac{1-q}{2}\right)\frac{\mathrm{dist}(z, v)}{\mathrm{W}(z)}, \tag{5}$$

where the last inequality holds since for $v \in L$ we have $\mathrm{dist}(z, v) \leq 2Q$, and since $\mathrm{W}(z) \geq (1-q)\, nQ$ if u is the (qn)th closest node to z.

For all v, we have that $\mathrm{dist}(u, v) \geq \mathrm{dist}(z, v) - Q$ by the triangle inequality. We also have $\mathrm{W}(u) \leq \mathrm{W}(z) + nQ$. Substituting into (4) we get that for every v

$$\gamma_v \geq \frac{\mathrm{dist}(u, v)}{\mathrm{W}(u)} \geq \frac{\mathrm{dist}(z, v) - Q}{\mathrm{W}(z) + nQ} \ . \tag{6}$$

In particular, for $v \in H$, we have

$$\mathrm{dist}(z,v) - Q \geq \frac{1}{2}\,\mathrm{dist}(z,v)\ . \tag{7}$$

As already mentioned, we also have $\mathrm{W}(z) \geq (1-q)\,nQ$ and thus

$$nQ \leq \frac{\mathrm{W}(z)}{1-q}\ , \tag{8}$$

and

$$\mathrm{W}(z) + nQ \leq \mathrm{W}(z)\left(1 + \frac{1}{1-q}\right) = W(z)\left(\frac{2-q}{1-q}\right)\ . \tag{9}$$

Substituting (9) and (7) in (6), we obtain that for $v \in H$,

$$\gamma_v \geq \frac{\mathrm{dist}(z,v) - Q}{\mathrm{W}(z) + nQ} \geq \frac{1}{2}\left(\frac{1-q}{2-q}\right)\frac{\mathrm{dist}(z,v)}{\mathrm{W}(z)}\ . \tag{10}$$

The lemma now follows from (5) and (10). ◀

▶ **Lemma 5.** *Consider a set of sampling coefficients γ_v such that for a node z, for all v and for some $c > 0$, $\gamma_v \geq c\frac{\mathrm{dist}(z,v)}{\mathrm{W}(z)}$. Let S be a sample obtained with probabilities $p_v = \min\{1, k\gamma_v\}$ (as in Algorithm 1), and let $\hat{\mathrm{W}}(z)$ be the inverse probability estimator as in (2). Then*

$$\mathsf{Var}[\hat{\mathrm{W}}(z)] \leq \frac{\mathrm{W}(z)^2}{k \cdot c}\ . \tag{11}$$

Proof. The variance of our estimator is

$$\begin{aligned}\mathsf{Var}[\hat{\mathrm{W}}(z)] &= \sum_v \left[p_v\left(\frac{\mathrm{dist}(z,v)}{p_v} - \mathrm{dist}(z,v)\right)^2 + (1-p_v)\,\mathrm{dist}(z,v)^2\right] \\ &= \sum_v \left(\frac{1}{p_v} - 1\right)\mathrm{dist}(z,v)^2\ . \end{aligned} \tag{12}$$

Note that nodes v for which $p_v = 1$ contribute 0 to the variance. For the other nodes we use the lower bound $p_v \geq ck\frac{\mathrm{dist}(z,v)}{\mathrm{W}(z)}$.

$$\begin{aligned}\sum_{v\in V}\left(\frac{1}{p_v} - 1\right)\mathrm{dist}(z,v)^2 &= \sum_{v\in V \mid p_v<1}\left(\frac{1}{p_v} - 1\right)\mathrm{dist}(z,v)^2 \\ &\leq \frac{\mathrm{W}(z)}{k\cdot c}\sum_{v\in V}\mathrm{dist}(z,v) \\ &\leq \frac{\mathrm{W}(z)^2}{k\cdot c}\ .\end{aligned}$$

◀

4.1 Base set containing a uniform sample

We now consider a situation where S_0 includes a uniform sample of nodes, and consider the corresponding expected approximation quality:

▶ **Lemma 6.** *Suppose that S_0 contains a uniform random sample of b nodes. Then for any node z,*

$$\mathsf{Var}[\hat{\mathrm{W}}(z)] \leq \frac{\mathrm{W}(z)^2}{k}\frac{4b}{b-1}\ . \tag{13}$$

Proof. We apply Lemma 5 with the bound on the coefficients as in Lemma 4 with u being the closest node to z in S_0. Assume that u is the xth closest node to z. By Lemma 4 and Lemma 5 we have

$$\mathsf{Var}[\hat{\mathrm{W}}(z) \mid x] \le \frac{\mathrm{W}(z)^2}{k}\frac{4}{1-x/n} \ . \tag{14}$$

Observe that x is a random variable which is the rank (= position in the sorted order of the nodes by distance from z) of the closest node to z in a uniform sample of size b. In particular x take values $\in \{1, 2, \ldots, n-b+1\}$ ($x = 1$ iff $u = z$). We have that the probability of rank x is

$$b\left(\frac{1}{n}\right)\left(\frac{n-x}{n-1}\right)\left(\frac{n-x-1}{n-2}\right)\cdots\left(\frac{n-x-b+2}{n-b+1}\right) \le b\left(1-\frac{x}{n}\right)^{b-1} \ .$$

(We choose the random subset of S_0 of b nodes without replacement, we split into b events according to the step in which the node of rank x is chosen. Other items should be chosen from the $n-x$ nodes of rank larger than x.) The variance $\mathsf{Var}[\hat{\mathrm{W}}(z)]$ is the expectation, over $x \in \{1, 2, \ldots, n-b+1\}$, of $\mathsf{Var}[\hat{\mathrm{W}}(z) \mid x]$. So from (14), we get

$$\begin{aligned}
\mathsf{Var}[\hat{\mathrm{W}}(z)] &\le \sum_{x=1}^{n-b+1} b\left(1-\frac{x}{n}\right)^{b-1}\left(\frac{\mathrm{W}(z)^2}{k}\frac{4}{(1-x/n)}\right)\\
&\le \frac{\mathrm{W}(z)^2}{k}4b\sum_{x=1}^{n-b+1}\left(1-\frac{x}{n}\right)^{b-2}\\
&\le \frac{\mathrm{W}(z)^2}{k}4b\int_0^1 (1-y)^{b-2}dy\\
&= \frac{\mathrm{W}(z)^2}{k}\frac{4b}{b-1} \ .
\end{aligned}$$

◀

It follows from Lemma 6 that if we choose $b \ge 2$ nodes uniformly into S_0 and $k = \epsilon^{-2}$, then for any node z, our estimator has $\mathsf{Var}[\hat{\mathrm{W}}(z)] = O(\epsilon^2\,\mathrm{W}(z)^2)$. This concludes the proof of the per-node (per-point) $O(\epsilon)$ bound on the CV of the estimator in the first part of Theorems 1 and 2 for a sample of size $O(\epsilon^{-2})$.

4.2 Well-positioned nodes

We provide a precise definition of a *well positioned* node. Let the *median distance* of a node u, denote by $m(u)$, be the distance between u and the $\lceil 1 + n/2\rceil$ closest node to u in V. Let $\textsc{MinMed} = \min_{v\in V} m(v)$ be the minimum median distance of any node $v \in V$. In a metric space, we can define $m(u)$ for any point u in the space (also for $u \notin V$), and accordingly, define $\textsc{MinMed}$ as the minimum $m(u)$ over all points u in the metric space.

We say that a node u is *well positioned* if $m(u) \le 2\,\textsc{MinMed}$, that is, $m(u)$, the median distance of u is within a factor of 2 of the minimum median distance. We now show that most nodes are well positioned.

▶ **Lemma 7.** *Let v be such that is $m(v) = \textsc{MinMed}$. Then all $\lceil 1 + n/2\rceil$ nodes in V that are closest to v are well positioned.*

Proof. Let u be one of the $\lceil 1 + n/2\rceil$ nodes closest to v. Then $\mathrm{dist}(u, v) \le \textsc{MinMed}$ and a ball of radius $2\,\textsc{MinMed}$ around u contains all the $\lceil 1 + n/2\rceil$ nodes closest to v. So $m(u) \le 2\,\textsc{MinMed}$. ◀

We are interested in well positioned nodes because of the following property:

▶ **Lemma 8.** *If u is a well positioned node, then for every node z we have that* $\mathrm{dist}(z,u) \leq 3m(z)$.

Proof. For every two nodes u and z we have that $\mathrm{dist}(u,z) \leq m(u) + m(z)$ since there must be at least one node x that is both within distance $m(u)$ from u and within distance $m(z)$ from z, and by the triangle inequality $\mathrm{dist}(u,z) \leq \mathrm{dist}(u,x) + \mathrm{dist}(x,z)$. The lemma follows since if u is well positioned then $m(u) \leq 2m(z)$. ◀

As we shall see, this means that sampling probabilities proportional to the distances from a well positioned node u approximate sampling probabilities proportional to the distances from any other node z, for nodes whose distance from z is substantially larger than $m(z)$.

4.3 Base set with a well-positioned node

We now consider the case where S_0 contains a well-positioned node. We show that in this case the coefficients γ_v satisfy what we call a *universal PPS* property:

▶ **Lemma 9.** *Suppose that S_0 contains a well-positioned node u. Then for all nodes v,*

$$\gamma_v \geq \frac{1}{18} \max_z \frac{\mathrm{dist}(z,v)}{\mathrm{W}(z)} \ . \tag{15}$$

Proof. We show that for any node z, $\gamma_v \geq \frac{1}{18}\frac{\mathrm{dist}(z,v)}{\mathrm{W}(z)}$ using a variation of the proof of Lemma 4.

We partition the nodes into two sets. A set L which contains the nodes v such that $\mathrm{dist}(z,v) \leq 6m(z)$ and a set H which contains the remaining nodes. By the definition of $m(z)$ we have that $\mathrm{W}(z) \geq m(z)(\lfloor \frac{n}{2} \rfloor - 1) \geq m(z)\frac{n}{3}$ (for $n \geq 9$). We obtain that for all $v \in L$,

$$\frac{\mathrm{dist}(v,z)}{\mathrm{W}(z)} \leq \frac{6m(z)}{m(z)\frac{n}{3}} = \frac{18}{n} \ .$$

Therefore,

$$\gamma_v \geq \frac{1}{n} \geq \frac{1}{18}\frac{\mathrm{dist}(v,z)}{\mathrm{W}(z)} \ .$$

We next consider $v \in H$. Since u is well positioned, by Lemma 8 we have that $\mathrm{dist}(z,u) \leq 3m(z)$. From the triangle inequality, $\mathrm{dist}(u,v) \geq \mathrm{dist}(z,v) - \mathrm{dist}(z,u) \geq \mathrm{dist}(z,v) - 3m(z) \geq \mathrm{dist}(z,v)/2$. We also have $\mathrm{W}(u) \leq \mathrm{W}(z) + n\,\mathrm{dist}(z,u) \leq \mathrm{W}(z) + 3nm(z) \leq 9\,\mathrm{W}(z)$. Therefore

$$\gamma_v \geq \frac{\mathrm{dist}(u,v)}{\mathrm{W}(u)} \geq \frac{(\mathrm{dist}(z,v)/2)}{9\,\mathrm{W}(z)} = \frac{1}{18}\frac{\mathrm{dist}(z,v)}{\mathrm{W}(z)} \ .$$ ◀

As a corollary, applying Lemma 5, we obtain:

▶ **Corollary 10.** *If S_0 contains a well-positioned node, then for any node z,* $\mathsf{Var}[\hat{\mathrm{W}}(z)] \leq 18\frac{\mathrm{W}(z)^2}{k}$.

4.4 Upper bound on the sum of the coefficients

One consequence of Lemma 9 is that the coefficients γ_u cannot grow too much even if the base set S_0 includes all nodes.

▶ **Corollary 11.** *Let*

$$\overline{\gamma}_v \equiv \max_z \frac{\operatorname{dist}(z,v)}{\mathrm{W}(z)} \ .$$

Then

$$\sum_v \overline{\gamma}_v = O(1) \ .$$

Proof. Consider the case where S_0 consists of a single well positioned node. By the definition of γ_v we have that $\sum_v \gamma_v \leq 2$. By Lemma 15 we have $\gamma_v \geq \frac{1}{18}\max_z \frac{\operatorname{dist}(z,v)}{\mathrm{W}(z)}$. Therefore $\sum_v \overline{\gamma}_v \leq 18\sum_v \gamma_v \leq 36$. ◀

4.5 High probability estimates

Lastly, we establish concentration of the estimates, which will conclude the proof of the very high probability claims in Theorem 1 and 2.

We need the following lemma:

▶ **Lemma 12.** *If our sampling coefficients are* approximate PPS *for a node z, that is, there is a constant c such that for all nodes v, $\gamma_v \geq c\frac{\operatorname{dist}(z,v)}{\mathrm{W}(z)}$, and we use $k = O(\epsilon^{-2}\log n)$, then*

$$\Pr\left[\frac{|\hat{\mathrm{W}}(z) - \mathrm{W}(z)|}{\mathrm{W}(z)} \geq \epsilon\right] = O(1/poly(n)) \ .$$

Proof. We apply the Chernoff-Hoeffding bound. Let $\tau = W(z)/(ck)$. We have

$$p_v \geq \min\{1, \operatorname{dist}(z,v)/\tau\} = \min\{1, ck\operatorname{dist}(z,v)/\mathrm{W}(z)\} \ . \tag{16}$$

The contribution of a node v to the estimate $\hat{\mathrm{W}}(z)$ is as follows. If $\operatorname{dist}(z,v) \geq \tau$, then the contribution is exactly $\operatorname{dist}(z,v)$. Otherwise, the contribution X_v of node v is $\operatorname{dist}(z,v)/p_v \leq \tau$ with probability p_v and 0 otherwise.

The contributions X_v of the nodes with $\operatorname{dist}(z,v) \leq \tau$ are thus independent random variables, each in the range $[0,\tau]$ with expectation $\operatorname{dist}(z,v)$. We complete the proof by applying the Chernoff-Hoeffding bound to bound the deviation of expectation of the sum of these random variables. We defer the details to the full version of the paper. ◀

We need the condition of Lemma 12 to hold for all nodes z with probability $1 - O(1/poly(n))$. Equivalently, we would like $\boldsymbol{\gamma}$ to be universal PPS with very high probability. If so, we apply a union bound to obtain that the estimates $\hat{\mathrm{W}}(z)$ for all nodes z have a relative error of at most ϵ with probability $1 - O(1/poly(n))$. The same argument applies to polynomially many queries in metric spaces.

It follows from Lemma 9 that we obtain the universal PPS property if S_0 includes a well positioned node. We would like this to happen with very high probability. We mention several ways to achieve this effect: (i) Since most nodes are well positioned (Lemma 7), taking a uniform random sample U of $O(\log n)$ nodes, and choosing the node $u = \arg\min_{u\in U} m(u)$ with minimum distance to its $\lceil n/2+1\rceil$ closest node, means that we are guaranteed with probability $1 - 1/poly(n)$ that u is well positioned. This identification step involves $O(\log n)$ single-source distance computations. (ii) Alternatively, we can ensure that S_0 contains a well positioned node (with a polynomially small error) by simply placing $O(\log n)$ uniformly selected nodes in S_0. The computation of the coefficients will then require $O(\log n)$ single-source distance computations. (iii) Lastly, if S_0 contains $O(\log n)$ uniformly selected nodes then we can apply a direct argument that with a polynomially small error for each node z, one of the $\lceil n/2+1 \rceil$ closest nodes to z is in S_0. This means we can apply Lemma 4 with $q \le 0.5$ to obtain that with a polynomially small error, the sampling probabilities are approximate PPS for all nodes and thus universal PPS with a polynomially small error.

To establish the second part of Theorem 2 in metric spaces, we would like to identify a well positioned node with a polynomially small ($O(1/poly(n))$) error using only $O(n)$ distance computations, which is more efficiently than by using $O(\log n)$ single-source distance computations.

To do so, we first provide a slightly relaxed definition of well positioned node and show that it retains the important properties. We will then show that a "relaxed" well positioned node can be identified with very high probability using only $O(\log^2 n)$ distance computations. When we identify such a node, we can use it in the base set S_0. This means we can use $O(n)$ distance computations in total to compute coefficients $\boldsymbol{\gamma}$ which are universal PPS with a polynomially small error. We then use $O(n)$ time to compute a sample of size $k = O(\epsilon^{-2} \log n)$, and use this sample to process point queries.

What remains is to introduce the relaxed definition of a well-positioned node and show that it has the claimed properties.

4.6 Relaxed well positioned points

For $Q \ge \lceil 1+n/2 \rceil$, we define the Q-quantile distance $m_Q(v)$ for a point v as the distance of the Qth closest point to v. We then define $\textsc{MinMed}_Q$ as the minimum Q-quantile distance over all points. Now, we define a point v to be Q well positioned if $m_{\lceil 1+n/2 \rceil}(v) \le 2\,\textsc{MinMed}_Q$.

Now observe that at least half the points have $m_Q(v) \le 2\,\textsc{MinMed}_Q$ and in particular are well positioned (extension of Lemma 7). Also observe that if z is Q well positioned then for any node u, $\text{dist}(z,u) \le 3m_Q(u)$ (extension of Lemma 8). We can also verify that for any $Q < 0.6n$ (any constant strictly smaller than 1 would do), a base set S_0 containing one Q well positioned point would also yield coefficients that satisfy the universal PPS property, albeit with a slightly larger constant.

We next show that we can identify a $0.6n$ well positioned point within a polynomially small error using very few distance computations:

▶ **Lemma 13.** *We can identify a $0.6n$ well positioned point with probability $1 - O(1/poly(n))$ using $O(\log^2 n)$ distance computations.*

Proof. We choose uniformly at random a set of points C of size $O(\log n)$. For each point in $v \in C$, we choose a uniform sample S_v of $O(\log n)$ points and compute the 0.55 quantile of $\{\text{dist}(v,u) \mid u \in S_v\}$. We then return the point $v \in C$ with the minimum sample 0.55 quantile.

We refer to C as the set of candidates. Note that since at least half the points $v \in V$ are such that $m_{0.6n}(v) \le 2\,\textsc{MinMed}_{0.6n}$, the set C contains such a point with probability $1 - O(1/poly(n))$.

The estimates are such that with probability $1 - O(1/poly(n))$, for all points in C, the sample 0.55 quantile is between the actual 0.5 and 0.6 quantiles. Therefore the point we returned (with a polynomially small error) has $m_{0.5n}$ at most the smallest $m_{0.6n}$ in C, which is at most $2\,\textsc{MinMed}_{0.6n}$. ◀

5 All-pairs sum

We now establish the claims of Theorem 3 for the all-pairs sum problem. We start with the first part of the claim, which is useful for graphs, estimates $\textsc{aps}(V)$ using single-source computations. To do so, we apply Algorithm 1 to compute sampling coefficients $\boldsymbol{\gamma}$ and then apply Algorithm 2 to compute estimates $\hat{\mathrm{W}}(v)$ for all v. Finally, we return the estimate $\widehat{\textsc{aps}}(V) = \frac{1}{2}\sum_{z\in V}\hat{\mathrm{W}}(z)$.

To obtain an estimate $\widehat{\textsc{aps}}(V)$ with CV of at most ϵ, we choose a base set S_0 that contains 2 uniformly sampled nodes when applying Algorithm 1. We then use sample size of $O(\epsilon^{-2})$ to ensure that the per-node estimates $\hat{\mathrm{W}}(z)$ have CV of at most ϵ. Note that the estimates of different nodes are correlated, as they all use the same sample, but the CV of the sum of estimates each with CV of at most ϵ must be at most ϵ. The total time amounts to $O(\epsilon^{-2})$ single-source distance computations.

To obtain universal PPS with polynomially small error we can identify a well positioned node with a polynomially small error, which can be done using $O(\log n)$ single-source computations. We then compute the sampling coefficients $\boldsymbol{\gamma}$ for a base set that contains this well-positioned node. (Which uses a single-source distance computation). The sampling coefficients we obtain have the universal PPS property and the sample-based estimates are concentrated. A sample size of size $O(\epsilon^{-2}\log n)$ would yield a relative error of at most ϵ with probability $1 - 1/poly(n)$, for each $\hat{\mathrm{W}}(z)$ and thus for the sum $\widehat{\textsc{aps}}(V)$. In total, we used $O(\epsilon^{-2}\log n)$ single-source computations.

The remaining part of this section treats the second part of the claim of Theorem 3, which applies to the all-pairs sum problem in metric spaces. We start with an overview of our approach. In order to obtain a good sample of pairs, we would like to sample pairs proportionally to $p_{ij} = \frac{\mathrm{dist}(i,j)}{\textsc{aps}(V)}$. The obvious difficulty we have to overcome is that the explicit computation of the probabilities p_{ij} requires a quadratic number of distance calculations.

Our first key observation is that we can obtain a sample with (nearly) the same statistical guarantees if we relax a little the sampling probabilities and the sample size: For some constant $c \geq 1$, we work with probabilities that satisfy $p_{ij} \geq c^{-1}\frac{\mathrm{dist}(i,j)}{\textsc{aps}(V)}$ and use a sample of size $k = c\epsilon^{-2}$.

We use independent sampling with replacement to compute a multiset S of pairs of points from $V \times V$. The estimator we use is the sample average inverse probability estimator:

$$\widehat{\textsc{aps}}(V) = \frac{1}{|S|}\sum_{(i,j)\in S}\mathrm{dist}(i,j)/p_{ij} \ .$$

This sample average is an unbiased estimate of $\textsc{aps}(V)$ and has CV of at most $\sqrt{k/c}$ which is ϵ when we use sample size $k = c\epsilon^{-2}$. Moreover, each summand is by definition at most $c\,\textsc{aps}(V)$ and therefore we obtain concentration by a direct application of Hoeffding's inequality: The probability of a relative error that is larger than ϵ when the sample size is k is at most $2e^{-2k\epsilon^2c^{-2}}$. In particular, if we take a sample size that is $O(\epsilon^{-2}\log n)$, we obtain that the probability that the relative error exceeds ϵ is polynomially small in n.

We next discuss how we facilitate such sampling efficiently. We would like to be able to sample with respect to relaxed p_{ij} and also have the sampling probabilities available for estimation. We show that we can express a set of relaxed probabilities (for some constant c) as the outer product of two probability distributions over points, $\boldsymbol{\gamma}\boldsymbol{\rho}^T$. The distribution $\boldsymbol{\gamma}$ has the universal PPS property with respect to some constant c'. The probability distribution $\boldsymbol{\rho}$ has the property that for some constant c'', for all v, $\rho_v \geq c''\frac{\mathrm{W}(v)}{\sum_u \mathrm{W}(u)}$. We now observe that for some constant $c = c'c''$, for all pairs u, v, $\rho_u\gamma_v \geq c\frac{\mathrm{dist}(u,v)}{\textsc{aps}(V)}$. That is, we can sample according to $p_{uv} = \rho_u \gamma_v$ and satisfy the relaxed conditions and obtain the desired statistical guarantees.

What remains is to provide details on (i) how we use the vectors $\boldsymbol{\gamma}$ and $\boldsymbol{\rho}$ to obtain a sample of pairs and (ii) how we compute such vectors that satisfy our conditions within a polynomially small error. These are addressed in the next two subsections.

5.1 Sampling pairs using the coefficient vectors

We show how we obtain k samples (v, u) from $\gamma_v \rho_u$ efficiently, using computation that is $O(n+k)$. Many sampling schemes (with or without replacement) will have the concentration properties we seek and the implementations are fairly standard. For completeness, we describe a scheme that computes independent samples with replacement. Our scheme obtains a sample from $V \times V$ by sampling independently a point i according to the probability distributions $\boldsymbol{\gamma}$ and a point j according to distribution ρ and returning (i, j).

What remains is to describe how we can obtain k independent samples with replacement from a probability vector $\boldsymbol{\gamma}$ in time $O(n + k)$.

We arbitrarily order the points, WLOG $i \in V$ is the ith point in the order. We compute $a_i = \sum_{h<i} \gamma_h$ and associate the intervals $[a_i, a_i + \gamma_i]$ with the point i.

To randomly draw a point $i \in V$ according to $\boldsymbol{\gamma}$, we can draw a random number $x \sim U[0, 1]$ and take the point $i \in V$ such that $x \in [a_i, a_i + \gamma_i)$. If we have k sorted random values, we can map all of them to points in V in $O(n)$ time using one pass on the sorted values and the sorted nodes. For completeness, we describe one way to obtain a sorted set of k independent random draws $x_1, \ldots, x_k \sim U[0, 1]$ using $O(k)$ operations: (i) We draw k values $y_1, \ldots y_k$ where $y_i \sim Exp[k + 1 - i]$ is exponentially distributed with parameters $k + 1 - i$. This can be done by drawing independent uniform $u_i \sim U[0, 1]$ and take $y_i = -\ln(u_i)/(k + 1 - i)$. (iii) Now observe that $x'_i \equiv \sum_{j \le i} y_j$ for $i \in [k]$ are k independent exponential random variables with parameter 1 which are sorted in increasing order. We can then transform x'_i to uniform random variables x_i using $x_i = 1 - \exp(-x'_i)$. Since the transformation is monotone, we obtain that x_i are sorted. Note that prefix sums of y_j and hence all x_i can be computed in $O(k)$ operations. Also note that we only need precision to the point needed to identify the point that each x_i maps into.

5.2 Computing the coefficient vectors

We recall that universal PPS coefficients can be computed using Algorithm 1 using n distance computations (and $O(n)$ additional computation), when our base set S_0 contains a well positioned point. The probability vector $\boldsymbol{\gamma}$ we work with is the universal PPS coefficients scaled to have a sum of 1.

We next discuss how we obtain the probability distribution $\boldsymbol{\rho}$. We show that given a $0.6n$ well positioned point (see Section 4.6), we can compute ρ_v that has the claimed properties with very high probability. From Lemma 13, we can identify a point that is $0.6n$ well positioned with probability at least $(1 - 1/poly(n))$, using only $O(\log^2 n)$ distance computations. We use the following lemma, which a variation of claim used for the pivoting upper bound estimate in [6]. What it roughly says is that for any node u and any node z that is within a constant times some quantile distance from u, we can get a constant factor approximation of $\mathrm{W}(u)$ from $\mathrm{W}(z)$ and $\mathrm{dist}(u, z)$.

▶ **Lemma 14.** *Consider a point u and a point z such that* $\mathrm{dist}(u, z)$ *is at most c times the distance of the $(qn)^{th}$ closest point to u. Then*

$$\mathrm{W}(u) \le n\,\mathrm{dist}(u, z) + \mathrm{W}(z) \le \left(1 + \frac{2c}{1-q}\right) \mathrm{W}(u) \ .$$

Proof. Left hand side is immediate from the triangle inequality. To establish the right hand side, first note that $(1-q)n$ of the points are at least as far as $\mathrm{dist}(z,u)/c$, thus $W(u) \geq \frac{(1-q)}{c} n\,\mathrm{dist}(u,z)$. From triangle inequality we have $\mathrm{W}(z) \leq \mathrm{W}(u) + n\,\mathrm{dist}(u,z)$. Combining we get:

$$\mathrm{W}(z) + n\,\mathrm{dist}(u,z) \leq \mathrm{W}(u) + 2n\,\mathrm{dist}(u,z) \leq (1 + \frac{2c}{1-q})\,\mathrm{W}(u)\ . \qquad \blacktriangleleft$$

Now consider a point z that is $0.6n$ well positioned and using the rough estimates

$$\hat{\mathrm{W}}'(u) = n\,\mathrm{dist}(u,z) + \mathrm{W}(z)$$

for all points u and accordingly the sampling probabilities

$$\rho_i = \frac{\hat{\mathrm{W}}'(i)}{\sum_j \hat{\mathrm{W}}'(j)}\ .$$

By definition, for all points u, the point z satisfies $\mathrm{dist}(u,z) \leq 3m_{0.6n}(u)$. We therefore can apply the lemma with $q = 0.6$ and $c = 3$ and obtain that for all v, $\rho_v \geq \frac{1-q}{1-q+2c}\frac{\mathrm{W}(v)}{\sum_j \mathrm{W}(j)}$. Note that given z, the vector $\boldsymbol{\rho}$ can be computed for all points using n distance computations, from z to all other points.

6 Uniform sampling based estimates

For completeness, we briefly present another solution for the all-points/nodes problem that is based on uniform sampling. The disadvantages over our weighted sampling approach is that it provides biased estimates and requires $\epsilon^{-2}\log n$ samples even when we are interested only in per-query guarantees.

To do so, we use a key lemma proved by Indyk [18, 17]. A proof of this lemma also appears in [22], and used to establish the correctness of his approximate 1-median algorithm.

▶ **Lemma 15.** *Let $Q \subset V$, $|Q| = k$ sampled uniformly at random (from all subsets of size k). Let u and v be two vertices such that $\mathrm{W}(v) \geq (1+\epsilon)\,\mathrm{W}(u)$. Then $\Pr(\mathrm{W}_Q(u) > \mathrm{W}_Q(v)) \leq e^{-\epsilon^2|Q|/64}$.*

Lemma (15) shows that if the average distance of two nodes differ by a factor larger than $1+\epsilon$, and we use a sample of size $\Omega(\epsilon^{-2})$ then the probability that the vertex of smaller average distance has larger average distance to the sample decays exponentially with the sample size. This lemma immediately implies that the 1-median with respect to a sample of size $O(\log n/\epsilon^2)$ is $(1+\epsilon)$-approximate 1-median with high probability.

To approximate all-pairs $\mathrm{W}(u)$, we use a uniform sample of size $O(\epsilon^{-2}\log n)$ and order the nodes according to the average distance to the sample. Using the lemma, and comparing to the ideal sorted order by exact $\mathrm{W}(v)$, two nodes v, u that are transposed have with high probability $\mathrm{W}(v)$ and $\mathrm{W}(u)$ within $1 \pm \epsilon$ from each other.

Recall however that the average distance to the uniform sample can be a very bad approximation of the average distance to the data set. We therefore perform adaptively another set of $O(\epsilon^{-1}\log n)$ single-source distance computations to compute exact $\mathrm{W}(v)$ of enough nodes in this nearly sorted order, so that the difference between exact $\mathrm{W}(v)$ of consecutive processed nodes is within $(1 \pm \epsilon)$.

We also mention here, for completeness, an improved approximate 1-median algorithm provided by Indyk. This algorithm only applies in metric spaces and computes a $(1+\epsilon)$-approximate 1-median with constant probability using only $O(n\epsilon^{-2})$ distance computations (eliminating the logarithmic factor). The algorithm works in iterations, where in each iteration a fraction of the points, those with largest average distance to the current sample, are excluded from further considerations. The sample size is then increased by a constant factor, obtaining more accurate estimates for the remaining points. The final sample size used is linear, but the set of remaining nodes is very small. This algorithm only applies in metric spaces because, as we mentioned in the introduction, arbitrary distance computations are not efficient in graphs. Indyk's approach can be extended to compute any approximate quantile of the distribution with similar probabilistic guarantees.

7 Hardness of Computing Sum of All-Pairs Distances

In this section we show that if there is a truly subcubic algorithm for computing $\textsc{aps}(V)$, the exact sum of all pairs distances then there is a truly subcubic algorithm for computing All Pairs Shortest Paths (APSP).

Williams and Williams [23] showed that APSP is subcubic equivalent to negative triangle detection. In the *negative triangle detection problem* we are given an undirected weighted graph $G = (V, E)$ with integer weights in $\{-M, ..., M\}$ and the goal is to determine if the graph contains a negative triangle, that is, a triangle whose edge weights sum up to a negative number. Therefore to show that a subcubic algorithm for $\textsc{aps}(V)$ implies a subcubic algorithm to APSP it suffices to give a subcubic reduction from the negative triangle detection problem to computing $\textsc{aps}(V)$. We show this by the following lemma.

▶ **Lemma 16.** *Given a $O(T(n, m))$ time algorithm for computing the sum of all distances ($\textsc{aps}(V)$) there is $O(T(n, m) + n^2)$ time algorithm for detecting a negative triangle.*

Proof. For an input instance $G = (V, E)$ for the negative triangle detection problem we construct a graph $G' = (V', E')$ for the sum of all distances problem. The vertex set V' is the union of three copies of V, that is $V' = V_1 \cup V_2 \cup V_3$ where vertex $u_i \in V_i$, $i = 1, 2, 3$, corresponds to vertex $u \in V$. We set $E' = \{(u, v) \mid u, v \in V'\}$, that is G' is a complete graph.

Let $\omega(e)$ denote the length of an edge $e \in E$. Recall that $\omega(e) \in \{-M, ..., M\}$. Let $N = 4M$. We define the length $\omega'(e)$ of an edge $e \in E'$ as follows. For every $(u, v) \in E$ we define $\omega'(u_1, v_2) = N + \omega(u, v)$, $\omega'(u_2, v_3) = N + \omega(u, v)$, and $\omega'(u_3, v_1) = 2N - \omega(u, v)$. We set $w(e) = 3N/2$ for any other edge $e \in E'$.

We claim that $\textsc{aps}(V') = \sum_{(u,v)\in E'} \omega'(u, v)$ if and only if G does not contain a negative triangle. In other words, we claim that either every edge in G' is a shortest path or G contains a negative cycle.

To see the first direction, assume G contains a negative triangle $(u, v), (u, x), (x, v)$. Now consider the path $P = (u_3, x_2), (x_2, v_1)$ from u_3 to v_1. Note that the length of this path is $\omega'(u_3, x_2) + \omega'(x_2, v_1) = N + \omega(u_3, x_2) + N + \omega(x_2, v_1) < 2N - \omega(u_3, v_1) = \omega'(u_3, v_1)$, where the strict inequality follows since $(u, v), (u, x), (x, v)$ is a negative triangle. If follows that $\textsc{aps}(V') < \sum_{(u,v)\in E'} \omega'(u, v)$.

To see the second direction, assume that $\textsc{aps}(V') < \sum_{(u,v)\in E'} \omega'(u, v)$. We need to show that G has a negative triangle.

We first claim that for every edge (u, v) which does not correspond to an edge in G (and hence $w(e) = 3N/2$) we have $\omega'(u, v) = \text{dist}_{G'}(u, v)$ (regardless if G has a negative triangle or not). To see this, note that $\omega'(u, v) = 3N/2 = 6M$ and that every path from u to v that consists of more than one edge is of weights at least $2N - 2M = 6M$. The same argument also holds for every edge from V_1 to V_2 and for every edge from V_2 to V_3.

It follows that only edges $(x, y) \in E'$ such that $x \in V_3$ and $y \in V_1$ may not be shortest paths. If $\textsc{aps}(V') < \sum_{(u,v)\in E'} \omega'(u, v)$ then there must be an edge $(u_3, v_1) \in E'$ such that

$u_3 \in V_3$ and $v_1 \in V_1$ and the edge (u_3, v_1) is not a shortest path. It is not hard to verify that only paths of the form $(u_3, x_2), (x_2, v_1)$ such that both edges (u_3, x_2) and (x_2, v_1) correspond to edges of G, could be shorter than the path (u_3, v_1). Let $(u_3, x_2), (x_2, v_1)$ be the shortest path from u_3 to v_1. We get that $N + \omega(u_3, x_2) + N + \omega(x_2, v_1) = \omega'(u_3, x_2) + \omega'(x_2, v_1) < \omega'(u_3, v_1) = 2N - \omega(u_3, v_1)$. So $\omega(u_3, x_2) + \omega(x_2, v_1) + \omega(u_3, v_1) < 0$ and G has a negative triangle. ◀

8 Extensions and Comments

8.1 The distribution of centrality values

What can we say about the centrality distribution? First we observe that the range of average distance $\mathrm{W}(v)/n$ values is between D/n to D, where D is the diameter (maximum distance between a pair of points in V). To see the upper bound, note that the average of values that are at most D, is at most D. For the lower bound, let u and v be nodes such that $\mathrm{dist}(u, v) = D$. Then for all $h \in V$, from triangle inequality, $\mathrm{dist}(u, h) + \mathrm{dist}(h, v) \geq D$, thus, $\mathrm{W}(h) \geq D$.

▶ **Lemma 17.** *The highest average distance value must satisfy*

$$\max_{v \in V} \mathrm{W}(v)/n \geq D/2 \ .$$

Proof. Consider the two nodes u and v such that $\mathrm{dist}(u, v) = D$. From triangle inequality, any point $h \in V$ has $\mathrm{dist}(u, h) + \mathrm{dist}(h, v) \geq D$. Summing over h we obtain that $\mathrm{W}(u) + \mathrm{W}(v) \geq nD$. Therefore, either $\mathrm{W}(u)$ or $\mathrm{W}(v)$ is at least $nD/2$. ◀

▶ **Lemma 18.** *If $z = \arg\min_{v \in V} \mathrm{W}(v)$ is the 1-median, then at least half the nodes satisfy $\mathrm{W}(v) \leq 3\,\mathrm{W}(z)$.*

Proof. Take the median distance $m(z)$ from z. Then the average distance from z is at least $m(z)/2$. Thus, $n \cdot m(z) \leq 2\,\mathrm{W}(z)$. Consider now a node v that is one of the $n/2$ closest to z. For any node u, $\mathrm{dist}(v, u) \leq \mathrm{dist}(z, u) + m(z)$. Therefore,

$$\mathrm{W}(v) = \sum_u \mathrm{dist}(v, u) \leq \sum_u \mathrm{dist}(z, u) + nm(z) \leq nm(z) + \mathrm{W}(z) \leq 3\,\mathrm{W}(z) \ . \quad ◀$$

Last we observe that it is easy to realize networks where there is a large spread of centrality values. At the extreme, consider a single point (node) that has distance D to a very tight cluster of $n - 1$ points. The points in the cluster have $\mathrm{W}(v) \approx D$ whereas the isolated point has $\mathrm{W}(v) \approx nD$. More generally, networks (or data sets) containing well separated clusters with different sizes would exhibit a spread in centrality values.

A side comment is that as a corollary of the proof of Lemma 17 we obtain that the all pairs sum in metric spaces can be estimated with CV ϵ and good concentration by the scaled average of distances of $O(n\epsilon^{-2})$ pairs sampled uniformly at random – as established in [2]. This is because there are at least $n - 1$ pairwise distances that are at least $D/2$, since each point that is not an endpoint of the diameter is of distance at least $D/2$ from at least one of the endpoints. Since the maximum distance is D, this immediately implies our claim. Recall, however, that when we are restricted to using $O(\epsilon^{-2})$ single-source distance computations from a uniform sample of nodes, the estimates can have large CV, but a similar bound can still be obtained using our weighted sampling approach (see Corollary 3).

8.2 Limitation to distances

We showed that any set of points V in any metric space can be "sparsified" in the sense that a weighted sample of size $O(\epsilon^{-2})$ allows us to estimate $\mathrm{W}(v)$ for any point v in the space. We refer to such a sample as a *universal PPS* sample, since it encapsulates a PPS sample of the entries in each row of the matrix. One can ask if we can obtain similar sparsification with respect to other nonnegative symmetric matrices. We first observe that in general, the size of a universal PPS sample may be $\Omega(n)$: Consider a matrix $A_{n\times n}$ so that for $i \in [n/2]$, $A_{2i-1,2i} \gg 0$ but all other entries are 0 (or close to 0). The average of each row is dominated by the other member of the pair $(2i-1, 2i)$, and therefore, any universal PPS sample must sample most points with probability close to 1.

Such a matrix can not be realized with distances, as it violates the triangle inequality, but it can be realized when entries correspond to (absolute value) of inner products of n vectors in n-dimensional Euclidean space $\mathbb{R}^n$. In this case, the sampling question we ask is a well studied embedding problem [21], for which it is known that the size of a universal PPS sample (the terminology *leverage scores* is used) can be of size $\Theta(d\epsilon^{-2})$, where d is the dimension [9, 21]. Intuitively, the gap between the universal PPS size between distances and inner products stems from the observation that being "far" (large distance) is something that usually applies with respect to many nodes, whereas being "close" (large inner product) is a local property.

8.3 Weighted centrality

Often different points v have different importance $\beta(v)$. In this case, we would like our centrality measure to reflect that by considering a weighted average of distances

$$\frac{\sum_i \beta(i)\,\mathrm{dist}(x_i, x_j)}{\sum_i \beta(i)} .$$

Our results, and in particular, the sampling construction extend to the weighted setting. First, instead of uniform base probabilities $1/n$, we use PPS probabilities according to $\beta(i)/\sum_j \beta(j)$ for node i. Second, when considering distances and probabilities from a base node, we use weight equal to the product of $\beta(v)\,\mathrm{dist}(u, v)$ (product of β and distance.). Third, in the analysis, we need to take quantiles/medians with respect to β mass and not just the number of points.

8.4 Adaptive (data dependent) sampling

We showed that the number of samples needed to determine an approximate 1-median on graphs is $O(\epsilon^{-2}\log n)$, where for each sample we perform a single-source distance computation. This bound is worst case which materializes when the 1-median z is such that all other points have $\mathrm{W}(u) = (1+\epsilon)\,\mathrm{W}(z)$. In this case, only the exact 1-median qualifies as an approximate 1-median and also, since there are so many other points, some are likely to have estimated $\hat{\mathrm{W}}(u) < \hat{\mathrm{W}}(z)$ if we use a smaller sample. On realistic instances, however, we would expect a larger separation between the 1-median and most other points. This would allow us to use fewer samples if we adaptively determine the sample size. Such an approach was proposed in [7] to identify a node with approximate maximum marginal influence and similarly can be applied here for the 1-median.

9 Conclusion

Weighted samples are often used as compact summaries of weighted data. With weighted sampling, even of very skewed data, a PPS sample of size ϵ^{-2} would provide us with good estimates with CV of $O(\epsilon)$ on the total sum of the population. The surprise factor of our result, which relies only on properties of metrics, is that we can design a single set of sampling probabilities, which we termed *universal PPS*, that forms a good weighted sample from the perspectives of *any* point in the metric space. Moreover, we do so in an almost lossless way in terms of the sample size to estimation quality tradeoffs. In particular, the sample size does not depend on the number of points n or the dimension of the space. Another perhaps surprising consequence of our results is that there is a rank-1 matrix that approximates the PPS probabilities of the full pairwise distances matrix. The approximation can be expressed as the outer product of two vectors, which can be computed using a linear number of distance computations.

References

1 A. Abboud, F. Grandoni, and V. Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *SODA*. ACM-SIAM, 2015.

2 K. Barhum, O. Goldreich, and A. Shraibman. On approximating the average distance between points. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 4627 of *Lecture Notes in Computer Science*. Springer, 2007.

3 A. Bavelas. A mathematical model for small group structures. *Human Organization*, 7:16–30, 1948.

4 A. Bavelas. Communication patterns in task oriented groups. *Journal of the Acoustical Society of America*, 22:271–282, 1950.

5 M. A. Beauchamp. An improved index of centrality. *Behavioral Science*, 10:161–163, 1965.

6 E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Computing classic closeness centrality, at scale. In *COSN*. ACM, 2014.

7 E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *CIKM*, 2014.

8 E. Cohen, N. Duffield, C. Lund, M. Thorup, and H. Kaplan. Efficient stream sampling for variance-optimal estimation of subset sums. *SIAM J. Comput.*, 40(5), 2011.

9 M. B. Cohen and R. Peng. ℓ_p row sampling by lewis weights. In *STOC*. ACM, 2015.

10 T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

11 D. Eppstein and J. Wang. Fast approximation of centrality. In *SODA*, pages 228–229, 2001.

12 M. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

13 L. C. Freeman. A set of measures of centrality based on betweeness. *Sociometry*, 40:35–41, 1977.

14 L. C. Freeman. Centrality in social networks: Conceptual clarification. *Social Networks*, 1, 1979.

15 O. Goldreich and D. Ron. Approximating average parameters of graphs. *Random Struct. Algorithms*, 32(4):473–493, 2008.

16 J. Hamidzadeh, R. Monsefi, and H. S. Yazdi. DDC: distance-based decision classifier. *Neural Computing and Applications*, 21(7), 2012.

17 P. Indyk. Sublinear time algorithms for metric space problems. In *STOC*. ACM, 1999.

18 P. Indyk. *High-dimensional Computational Geometry.* PhD thesis, Stanford University, 2000.
19 K. Okamoto, W. Chen, and X. Li. Ranking of closeness centrality for large-scale social networks. In *Proc. 2nd Annual International Workshop on Frontiers in Algorithmics*, FAW. Springer-Verlag, 2008.
20 G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.
21 M. Talagrand. Embedding subspaces of l_1 into l_1^n. *Proc. of the American Math. Society*, 108(2):363–369, 1990.
22 M. Thorup. Quick k-median, k-center, and facility location for sparse graphs. *SIAM J. Comput.*, 34(2):405–432, 2004.
23 V. Vassilevska Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *FOCS*. IEEE, 2010.
24 S. Wasserman and K. Faust, editors. *Social Network Analysis: Methods and Applications.* Cambridge University Press, 1994.

Two Structural Results for Low Degree Polynomials and Applications

Gil Cohen and Avishay Tal

Weizmann Institute of Science
Rehovot, Israel
{gil.cohen,avishay.tal}@weizmann.ac.il

Abstract

In this paper, two structural results concerning low degree polynomials over finite fields are given. The first states that over any finite field $\mathbb{F}$, for any polynomial f on n variables with degree $d \le \log(n)/10$, there exists a subspace of $\mathbb{F}^n$ with dimension $\Omega(d \cdot n^{1/(d-1)})$ on which f is constant. This result is shown to be tight. Stated differently, a degree d polynomial cannot compute an affine disperser for dimension smaller than $\Omega(d \cdot n^{1/(d-1)})$. Using a recursive argument, we obtain our second structural result, showing that any degree d polynomial f induces a partition of $\mathbb{F}^n$ to affine subspaces of dimension $\Omega(n^{1/(d-1)!})$, such that f is constant on each part.

We extend both structural results to more than one polynomial. We further prove an analog of the first structural result to sparse polynomials (with no restriction on the degree) and to functions that are close to low degree polynomials. We also consider the algorithmic aspect of the two structural results.

Our structural results have various applications, two of which are:

- Dvir [11] introduced the notion of extractors for varieties, and gave explicit constructions of such extractors over large fields. We show that over any finite field any affine extractor is also an extractor for varieties with related parameters. Our reduction also holds for dispersers, and we conclude that Shaltiel's affine disperser [26] is a disperser for varieties over $\mathbb{F}_2$.
- Ben-Sasson and Kopparty [6] proved that any degree 3 affine disperser over a prime field is also an affine extractor with related parameters. Using our structural results, and based on the work of Kaufman and Lovett [19] and Haramaty and Shpilka [17], we generalize this result to any constant degree.

1998 ACM Subject Classification F.1.0 Computation by Abstract Devices – General

Keywords and phrases low degree polynomials, affine extractors, affine dispersers, extractors for varieties, dispersers for varieties

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.680

1 Introduction

In this paper, we consider the following question concerning polynomials on n variables over the field with q elements, $\mathbb{F}_q$, where q is some prime power:

> What is the largest number $k = k_q(n, d)$, such that any polynomial on n variables over $\mathbb{F}_q$, with degree[1] at most d, is constant on some affine subspace of $\mathbb{F}_q^n$ with dimension k?

[1] Here, and throughout the paper, by degree we mean total degree.

A related question was studied by Tardos and Barrington ([28], Lemma 3) who proved that for any prime power q and for any degree d polynomial f on n variables over the *ring* $\mathbb{Z}_q$, there exists a "cube" with dimension $k = \Omega(n^{1/d})$, on which f is constant. That is, there exist linearly independent vectors $\Delta_1, \ldots, \Delta_k \in \mathbb{Z}_q^n$ such that for every $\alpha \in \{0,1\}^k$, $f(\sum_{i=1}^{k} \alpha_i \Delta_i) = f(0)$. Although the problem studied in [28] is different than the problem mentioned above in several respects, one can make use of the proof idea of Tardos and Barrington and show that $k_2(n,d) = \Omega(n^{1/(d-1)})$ for all n, d (see Appendix B).

The proof idea of [28] seems to be applicable to our problem only for $q = 2$, and new ideas are required for larger fields. For any q, the case $d = 1$ is trivial – $k_q(n,1) = n-1$. The case $d = 2$, at least over fields of characteristic 2, is also well understood. By Dickson's theorem ([9], Theorem 199), $k_q(n,2) \geq \lfloor n/2 \rfloor$ for fields of characteristic 2. This is tight, as can be seen by considering the inner product function $x_1x_2 + x_3x_4 + \cdots + x_{n-1}x_n$.

1.1 Our Results

Our first result is an asymptotically tight upper and lower bounds on $k_q(n,d)$ for any q and $d < \log(n)/10$. The following theorem gives a lower bound for $k_q(n,d)$. In fact, it has a stronger guarantee which is required by one of our applications (see Theorem 6). Informally, for any degree d polynomial f and a point $u_0 \in \mathbb{F}_q^n$, there exists a large subspace U such that f is constant on $u_0 + U$. Note that this is equivalent to saying that there exists a large *linear* subspace on which f is constant (namely, the affine shift is by the zero vector).

▶ **Theorem 1** (Structural Result I). *For any n, d, let k be the least integer such that*

$$n \leq k + (d+1) \cdot \sum_{j=0}^{d-1} (d-j) \cdot \binom{k+j-1}{j}. \tag{1}$$

Let q be a prime power. Let $f \colon \mathbb{F}_q^n \to \mathbb{F}_q$ be a degree d polynomial, and let $u_0 \in \mathbb{F}_q^n$. Then, there exists a subspace $U \subseteq \mathbb{F}_q^n$ of dimension k such that $f|_{u_0+U}$ is constant.

In particular, there exists a universal constant $c_1 \in (0,1)$ such that for all n, d, q, it holds that $k_q(n,d) \geq c_1 \cdot n^{1/(d-1)}$. Moreover, for $d \leq \log(n)/10$ it holds that $k_q(n,d) = \Omega(d \cdot n^{1/(d-1)})$.

Few remarks are in order:

Tightness. Theorem 1 is tight for $d \leq \log(n)/10$. Indeed, one can show that, with probability at most $q^{-\binom{k}{d}}$,[2] a random degree d polynomial on n variables over $\mathbb{F}_q$ is constant on any fixed affine subspace of dimension k. There are at most $q^{(k+1)n}$ affine subspaces of dimension k, so by the union bound, $k_q(n,d)$ must be smaller than any k such that $\binom{k}{d} > (k+1)n$. Hence, $k_q(n,d) < d^{1+1/(d-1)} \cdot n^{1/(d-1)}$. For $d \leq \log(n)/10$, the ratio between our upper and lower bound is $d^{O(1/d)} = 1 + O(\log(d)/d)$.

Low degree polynomials, affine dispersers and affine extractors. An *affine disperser* for dimension k is a function $f \colon \mathbb{F}_q^n \to \mathbb{F}_q$ with the following property. For every affine subspace $u_0 + U \subseteq \mathbb{F}_q^n$ of dimension k, f restricted to $u_0 + U$ is not constant [3]. Thus, in the language of pseudorandomness, Theorem 1 states that a degree $d \leq \log(n)/10$

[2] The expression $\binom{k}{d}$ in the exponent can be replaced by the number of solutions to the equation $r_1 + \cdots + r_k \leq d$, where $r_i \in \{0, \ldots, q-1\}$.

[3] An alternative definition requires that almost all field elements are obtained by f on $u_0 + U$.

polynomial is not an affine disperser for dimension $o(d \cdot n^{1/(d-1)})$, and in particular, polynomials with constant degree are not affine dispersers for sub-polynomial dimension. For the special case $q = 2$, based on the work of Ben-Eliezer *et al.* [2], one can say something stronger regarding the tightness of Theorem 1. A function $f\colon \mathbb{F}_q^n \to \mathbb{F}_q$ is called an *affine extractor* for dimension k with bias ε, if for every affine subspace $u_0 + U \subseteq \mathbb{F}_q^n$ of dimension k, it holds that $f(x)$, where x is sampled uniformly from $u_0 + U$, is ε-close in statistical distance, to the uniform distribution over $\mathbb{F}_q$. By [2] it holds that for every $d \geq 1$, there exists a degree d affine extractor $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ for any $k \geq \Omega(d \cdot n^{1/(d-1)})$, with $\varepsilon = 2^{-\Omega(k/d)}$(see Section 3.3).

The case of unbounded degree. Theorem 1 yields a non-trivial bound only for $d \leq O(\log n)$. When the degree of the polynomial is unbounded things behave differently. For example, it is considered a folklore that any function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is constant on some affine subspace with dimension $\Omega(\log n)$. Namely, $k_2(n, \infty) = \Omega(\log n)$ (this is, in fact, tight). On the other hand, Gabizon and Raz [12] noted that the polynomial $x_1^1 + x_2^2 + \cdots + x_n^n$ over the field with $n + 1$ elements is not constant on any dimension 1 affine subspace (see also [8]). Thus, $k_{n+1}(n, \infty) = 1$.

The independence of the field size. Note that the bound on $k_q(n, d)$ in Theorem 1 is independent of q. That is, when considering bounded degree polynomials, the field size does not affect $k_q(n, d)$. Throughout the paper we focus on low degree polynomials – polynomials of degree up to $\log(n)/10$. In this range of parameters, Theorem 1 and the fact that it is tight, allow us to suppress the field size and write $k(n, d)$ instead of $k_q(n, d)$, as we do from here on.

Partition of $\mathbb{F}^n$ to affine subspaces, induced by a low degree polynomial

Theorem 1 states that for any degree d polynomial f on n variables, there exists at least one large affine subspace, restricted to which, f is constant. However, for some of our applications we need a stronger structural result. More specifically, we ask what is the maximum number $\mathcal{K} = \mathcal{K}_q(n, d)$, such that any degree d polynomial on n variables over $\mathbb{F}_q$ induces a *partition* of $\mathbb{F}_q^n$ to dimension $\mathcal{K}$ affine subspaces, on each of which f is constant. Using Theorem 1, we show that $\mathcal{K}_q(n, d) = \Omega(n^{1/(d-1)!})$. That is, we obtain the following result.

▶ **Theorem 2** (Structural Result II). *There exists a universal constant $c_2 > 0$ such that the following holds. Let q be a prime power. Let $f\colon \mathbb{F}_q^n \to \mathbb{F}_q$ be a degree d polynomial. Then, there exists a partition of $\mathbb{F}_q^n$ to affine subspaces (not necessarily shifts of the same subspace), each of dimension $c_2 \cdot n^{1/(d-1)!}$, such that f is constant on each part.*

We do not know whether the lower bound in Theorem 2 for $\mathcal{K}_q(n, d)$ is tight or not for all d (note that it is tight for $d \leq 3$), and leave this as an open problem. More precisely, we ask what is the asymptotic behavior of $\mathcal{K}_q(n, d)$? Does it depend on q for, say, constant d?

Generalization of the structural results to many polynomials

Being a natural generalization and also necessary for some of our applications, we generalize the two structural results to the case of any number of polynomials (see Section 3.4). Let $f_1, \ldots, f_t\colon \mathbb{F}_q^n \to \mathbb{F}_q$ be polynomials of degree at most d. The generalization of the first structural result states that there exists an affine subspace of dimension $\Omega((n/t)^{1/(d-1)})$ on which *each* of the t polynomials is constant (see Theorem 19). By applying a probabilistic argument, one can show that the dependence in t is tight. For the second structural result, the guaranteed dimension in Theorem 2 is replaced by $\Omega(n^{1/(d-1)!}/t^e)$, where e is the base of the natural logarithm (see Theorem 20).

The algorithmic aspect

We further study the algorithmic aspect of the structural results (see Section 4). We devise a poly(n)-time deterministic algorithm (see Theorem 22), that given a degree d polynomial $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ as a black-box, performs poly(n) queries, and outputs a subspace of dimension $\Omega(k(n,d))$, restricted to which, f has degree at most $d-1$. By applying this algorithm recursively d times, one can efficiently obtain a subspace of dimension $\Omega(n^{1/(d-1)!})$ on which f is constant. Our algorithm only works for the binary field. Devising an algorithm for general fields is a natural problem.

Note that there is a gap between $k(n,d)$ and the dimension of the affine subspace that our algorithm produce. A natural open problem is whether this gap can be eliminated. Specifically, we ask whether there is a poly(n)-time algorithm that, given a black-box access to a degree d polynomial $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$, finds an affine subspace with dimension $k(n,d)$ on which f is constant?

Whether there exists an algorithm as in the problem above is not at all clear to us. Verifying that a degree d polynomial is constant on a given affine subspace with dimension $k(n,d)$ can be done in time $O(k(n,d)^d) \le O(n^2)$, and it might be the case that this problem is expressive enough to be **NP**-hard. We show that the latter scenario is unlikely, at least for constant d, by devising an $\exp(n^{1-\frac{1}{d-1}}) \cdot n^d$-time algorithm that outputs an affine subspace with dimension $\Omega(k(n,d))$ on which f is constant (see Theorem 24). We note that the naive algorithm iterates over all $\binom{2^n}{k(n,d)} = \exp(n^{1+\frac{1}{d-1}})$ affine subspaces with dimension $k(n,d)$. It is also worth mentioning that this algorithm works for all finite fields.

Sparse polynomials

We further give an analog of the first structural result to sparse polynomials (regardless of their degree) over any finite field. We have the following.

▶ **Theorem 3.** *Let q be a prime power. For any integer $c \ge 1$ the following holds. Let f be a polynomial on n variables over $\mathbb{F}_q$, with at most n^c monomials. Then, there exists an affine subspace of dimension $\Omega\left(n^{1/(4(q-1)c)}\right)$ on which f is constant.*

We note that unlike in the case of low degree polynomials, the field size q does affect the dimension of the affine subspace promised by Theorem 3. Some sort of dependency cannot be avoided. Indeed, as mentioned above, the polynomial $x_1^1 + x_2^2 + \cdots + x_n^n$ over the field with $n+1$ elements is not constant on any dimension 1 affine subspace, even though it has only n monomials. On the other hand, Theorem 3 gives no guarantee already for $q = \Omega(\log n)$, while the example above requires fields of size $\Omega(n)$. We leave open the problem of improving upon the dependence of Theorem 3 in the field size q, or proving that this dependence is optimal.

We note that for the special case $q = 2$, the lower bound in Theorem 3 is $\Omega\left(n^{1/(4c)}\right)$, which is essentially tight up to the constant 4 in the exponent, as implied by our tightness result for degree d polynomials. We do not know whether the constant 4 is necessary. Indeed, for degree d polynomials (which may have n^d monomials), the guarantee given by Theorem 1 is stronger, namely, $\Omega\left(n^{1/(d-1)}\right)$.

Functions that are close to low degree polynomials

Theorem 1 implies that any function that is close to a low degree polynomial, is constant on some large affine subspace.

▶ **Corollary 4.** *Let q be a prime power. Let $g : \mathbb{F}_q^n \to \mathbb{F}_q$ be a function that agrees with some degree d polynomial $f : \mathbb{F}_q^n \to \mathbb{F}_q$ on all points but for some subset $B \subseteq \mathbb{F}_q^n$. Then, there exists an affine subspace with dimension $\Omega((n - \log_q(|B|))^{1/(d-1)})$ on which g is constant.*

To see that, note that by averaging argument there is an affine subspace $w + W$ of dimension $n - \log_q(|B|) - 1$ on which f and g agrees. Applying Theorem 1 to $f|_{w+W}$ gives an affine subspace $u + U \subseteq w + W$ on which f, and thus g, is constant on. We suspect that better parameters can be achieved.

1.2 Applications

We now present several applications of our structural results.

Extractors and Dispersers for Varieties over all Finite Fields

Let $\mathbb{F}$ be some finite field. An affine subspace of $\mathbb{F}^n$ can be thought of as the set of common zeros of one or more degree 1 polynomials with coefficients in $\mathbb{F}$. Recall that an affine extractor over the field $\mathbb{F}$ is a function $f : \mathbb{F}^n \to \mathbb{F}$ that has small bias on every large enough affine subspace. In [11], the study of the following natural generalization was initiated: construct a function that has small bias on the set of common zeros of one or more degree $d > 1$ polynomials. In general, the set of common zeros of one or more polynomials is called a *variety*. For a set of polynomials $g_1, \dots, g_t$ on n variables over $\mathbb{F}$, we denote their variety by $\mathbf{V}(g_1, \dots, g_t) = \{x \in \mathbb{F}^n : g_1(x) = \cdots = g_t(x) = 0\}$. A function $f : \mathbb{F}^n \to \mathbb{F}$ as above is called an extractor for varieties.

In [11], two explicit constructions of extractors for varieties were given. For simplicity, we suppress here both the bias of the extractor and the number of output bits. Dvir's first construction works under no assumption on the variety size (more precisely, some assumption is made, but that assumption is necessary). The downside of this construction is that the underlining field is assumed to be quite large, more precisely, $|\mathbb{F}| > d^{\Omega(n^2)}$. The second construction works for fields with size as small as poly(d), however the construction is promised to work only for varieties with size at least $|\mathbb{F}|^{n/2}$. Dvir applies tools from algebraic geometry for his constructions.

Even the construction of affine extractors, which is a special case of extractors for varieties, is extremely challenging. Indeed, the (far from optimal) constructions known today use either very sophisticated exponential sum estimates [4, 33] or involved composition techniques [20], where the correctness relies, among other results, on deep structural results from additive combinatorics [30] and on XOR lemmas for low degree polynomials [32, 3]. The same can be said about the constructions of affine dispersers.

Given the difficulties in constructing affine extractors and dispersers, one may suspect that the construction of extractors and dispersers for varieties will be substantially more challenging, especially for small fields that seem to be immune against algebraic geometry based techniques. Nevertheless, based on our structural results, the following theorem states that any affine extractor is also an extractor for varieties with related parameters.

▶ **Theorem 5.** *Let q be a prime power. For any integers n, d, t the following holds. Let $f : \mathbb{F}_q^n \to \mathbb{F}_q$ be an affine extractor for dimension $\Omega(n^{1/(d-1)!}/t^e)$ with bias ε, where e is the base of the natural logarithm. Then, f is an extractor with bias ε for varieties that are the common zeros of any t polynomials, each of degree at most d.*

In fact, one can view Theorem 5 as an explanation for the difficulty of constructing affine extractors for dimension n^δ for constant $\delta < 1$.

We also obtain a reduction that does not depend on the number of polynomials defining the variety, but rather on the variety size (see Theorem 26). The proof idea in this case is to "approximate" the given variety by a variety induced by a small number of low degree polynomials, and then apply Theorem 5.

The state of the art explicit constructions of affine extractors for the extreme case $q = 2$, work only for dimension $\Omega(n/\sqrt{\log\log n})$ [4, 33, 20], and thus the reduction in Theorem 5 only gives an explicit construction of an extractor for varieties defined by quadratic polynomials (and in fact, up to $(\log\log n)^{1/(2e)}$ quadratic polynomials). However, a similar reduction to that in Theorem 5 also holds for dispersers.

▶ **Theorem 6.** *Let n, d, t be integers such that $d < \log(n/t)/10$. Let $f \colon \mathbb{F}_q^n \to \mathbb{F}_q$ be an affine disperser for dimension $\Omega(d \cdot (n/t)^{1/(d-1)})$. Then, f is a disperser for varieties that are the common zeros of any t polynomials of degree at most d.*

Over $\mathbb{F}_2$, an explicit construction of an affine disperser for dimension as small as $2^{\log^{0.9} n}$ is known [26]. Thus, we obtain the first disperser for varieties over $\mathbb{F}_2$.

▶ **Theorem 7.** *For any n, d, t such that $d < (1 - o_n(1)) \cdot \frac{\log(n/t)}{\log^{0.9} n}$, there exists an explicit construction of an affine disperser for varieties which are the common zeros of any t polynomials of degree at most d. In particular, when $t \le n^{\alpha}$ for some constant $\alpha < 1$, the requirement on the degree is $d < (1 - \alpha - o_n(1)) \cdot \log^{0.1} n$.*

A few words regarding the limitation of the reduction in Theorem 6 are in order. Note that even if f is an optimal affine disperser, that is, a disperser for dimension $O(\log n)$, Theorem 6 only guarantees that f is a disperser for varieties defined by degree $O(\log n)$ polynomials. One cannot expect much more from the reduction. Indeed, there exists a degree $O(\log n)$ polynomial that computes an optimal affine disperser (this can be proven via a probabilistic argument. See also Theorem 36). However, this affine disperser is clearly not a disperser for varieties defined by even a single degree $O(\log n)$ polynomial.

Thus, the reduction in Theorem 6 is useful only for varieties defined by degree $o(\log n)$ polynomials. A recent work of Hrubeš and Rao [16] shows that it would be challenging to construct an explicit f which is an extractor (or even a disperser) for varieties of size $2^{\rho n}$ defined by degree n^{ε} polynomials over $\mathbb{F}_2$, for any constants $0 < \varepsilon, \rho < 1$. Indeed, such a function would solve Valiant's problem [29], since f cannot be computed by Boolean circuits of logarithmic depth and linear size.

From Affine Dispersers to Affine Extractors

Constructing an affine disperser is, by definition, an easier task than constructing an affine extractor. Nevertheless, Ben-Sasson and Kopparty [6] proved (among other results) that any degree 3 affine disperser is also an affine extractor with comparable parameters. [4] Using the extension of Theorem 1 to many polynomials, we are able to generalize the reduction of Ben-Sasson and Kopparty, over prime fields, to any degree $d \ge 3$.

▶ **Theorem 8.** *Let p be a prime number. For all $d \ge 3$ and $\delta > 0$, there exists $c = c(d, \delta)$ such that the following holds. Let $f \colon \mathbb{F}_p^n \to \mathbb{F}_p$ be an affine disperser for dimension k, which has degree d as a polynomial over $\mathbb{F}_p$. Then, f is also an affine extractor for dimension $k' \triangleq c \cdot k^{d-2}$ with bias δ.*

[4] A reduction from "low rank" extractors to dispersers in the context of two sources was also obtained, by Ben-Sasson and Zewi [7], conditioned on the well-known Polynomial Freiman-Ruzsa conjecture from additive combinatorics.

Note that Theorem 8 is only interesting in the case where $k^{d-2} < n$. However, this case is achievable since a random polynomial of degree d is an affine disperser for dimension $O(d \cdot n^{1/(d-1)})$. In particular, Theorem 8 implies that an explicit construction of an optimal affine disperser that has a constant degree as a polynomial, suffices to break the current natural barrier in the construction of affine extractors, namely, constructing affine extractors for dimension $n^{1-\delta}$ for some constant $\delta > 0$ (here $\delta = 1/(d-1)$).

On top of Theorem 1, the key ingredient we use in the proof of Theorem 8 is the work of Kaufman and Lovett [19], generalizing a result by Green and Tao [13] (see Section 6). For $d = 4$, we get a better dependency between k and k' based on the work of Haramaty and Shpilka [17] (see Theorem 28).

$\mathsf{AC}^0[\oplus]$ Circuits and Affine Extractors / Dispersers

Constructing affine dispersers, and especially affine extractors, is a challenging task. As mentioned, the state of the art explicit constructions for affine extractors over $\mathbb{F}_2$ work only for dimension $\Omega(n/\sqrt{\log\log n})$. By a probabilistic argument however, one can show the existence of affine extractors for dimension $(1 + o(1)) \log n$ (see Lemma 31). Thus, there is an exponential gap between the non-explicit construction and the explicit ones.

It is therefore tempting to try and utilize this situation and prove circuit lower bounds for affine extractors. This idea works smoothly for AC^0 circuits. Indeed, by applying the work of Håstad [14], one can easily show that an AC^0 circuit on n inputs cannot compute an affine disperser for dimension $o(n/\text{polylog}(n))$ (see Corollary 30). However, strong lower bounds for AC^0 circuits are known, even for much simpler and more explicit functions such as Parity and Majority. Thus, it is far more interesting to prove lower bounds against circuit families for which the known lower bounds are modest. One example would be to show that a De Morgan formula of size $O(n^3)$ cannot compute a good affine extractor, improving upon the best known lower bound [15]. [5]

Somewhat surprisingly, we show that even depth 3 $\mathsf{AC}^0[\oplus]$ circuit (that is, AC^0 circuits with XOR gates) can compute an optimal affine extractor over $\mathbb{F}_2$. In fact, the same construction can also be realized by a polynomial-size De Morgan formula and has degree $(1 + o(1)) \log n$ as polynomial over $\mathbb{F}_2$ (see Theorem 36).

Theorem 36 is implicit in the works of [22, 24] who studied a similar problem in the context of bipartite Ramsey graphs (that is, two-source dispersers). We give an alternative proof in Appendix A, which can be extended to work also in the context of bipartite Ramsey graphs.

Given that depth 3 $\mathsf{AC}^0[\oplus]$ circuits exhibit the surprising computational power mentioned above, it is natural to ask whether depth 2 $\mathsf{AC}^0[\oplus]$ circuit can compute a good affine extractor. We stress that even depth 2 $\mathsf{AC}^0[\oplus]$ circuits should not be disregarded easily! For example, such circuits *can* compute, in a somewhat different setting, optimal Ramsey graphs (see [18], Section 11.7). Moreover, any degree d polynomial $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ can be computed by a depth 2 $\mathsf{AC}^0[\oplus]$ circuit with size n^d. Nevertheless, we complement the above result by showing that a depth 2 $\mathsf{AC}^0[\oplus]$ circuit cannot compute an affine disperser for sub-polynomial dimension. The proof is based on the following reduction.

▶ **Lemma 9.** *Let C be a depth 2 $\mathsf{AC}^0[\oplus]$ circuit on n inputs, with size n^c. Let $k < n/10 - c\log(n)$. If C computes an affine disperser for dimension k, then there exists a degree $2c$ polynomial over $\mathbb{F}_2$ on $\sqrt{n}/5$ variables which is an affine disperser for dimension k.*

[5] The property of being an affine extractor meets the largeness condition of the natural proof barrier [23]. However, it does not necessarily get in the way of improving existing polynomial lower bounds.

The proof of Lemma 9 uses ideas from our proof of the structural result for sparse polynomials (see Lemma 21). Lemma 9 together with Theorem 1 imply the following theorem.

▶ **Theorem 10.** *Let C be a depth 2 $\mathsf{AC}^0[\oplus]$ circuit on n inputs, with size n^c, which is an affine disperser for dimension k. Then, $k > k(\sqrt{n}/5, 2c) = \Omega(n^{1/4c})$.*

Good Affine Extractors are Hard to Approximate by Low Degree Polynomials

Using our second structural result, Theorem 2, we obtain an average-case hardness result, or in other words, correlation bounds for low degree polynomials. Namely, we show that any affine extractor with very good parameters cannot be approximated by low degree polynomials over $\mathbb{F}_2$.

▶ **Corollary 11.** *Let $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ be an affine extractor for dimension k with bias ε. Then, for any polynomial $g\colon \mathbb{F}_2^n \to \mathbb{F}_2$ of degree d such that $k = \Omega(n^{1/(d-1)!})$, it holds that*

$$\mathrm{Cor}(f,g) \triangleq \mathop{\mathbf{E}}_{x\sim\mathbb{F}_2^n}\left[(-1)^{f(x)}\cdot(-1)^{g(x)}\right] \le \varepsilon.$$

Proof. Let g be a degree d polynomial over $\mathbb{F}_2$ on n variables. By Theorem 2, there exists a partition of $\mathbb{F}_2^n$ to affine subspaces $P_1, P_2, \ldots, P_\ell$, each of dimension $k = \Omega(n^{1/(d-1)!})$, such that for all $i \in [\ell]$, $g|_{P_i}$ is some constant $g(P_i)$. Thus,

$$\begin{aligned}\mathrm{Cor}(f,g) &= \left|\mathop{\mathbf{E}}_{x\sim\mathbb{F}_2^n}[(-1)^{f(x)+g(x)}]\right| = \left|\mathop{\mathbf{E}}_{i\sim[\ell]}\mathop{\mathbf{E}}_{x\sim P_i}[(-1)^{f(x)+g(P_i)}]\right| \\ &\le \mathop{\mathbf{E}}_{i\sim[\ell]}\left|(-1)^{g(P_i)}\cdot\mathop{\mathbf{E}}_{x\sim P_i}[(-1)^{f(x)}]\right|,\end{aligned}$$

which is at most ε since f is an affine extractor for dimension k with bias ε. ◀

As mentioned, explicit constructions of affine extractors for dimension $\Omega(n/\sqrt{\log\log n})$ are known. Corollary 11 implies that these extractors cannot be approximated by quadratic polynomials. Corollary 11 also implies that for any constant $\beta \in (0,1)$, affine extractors for dimension $k \le 2^{(\log n)^\beta}$ with bias ε have correlation ε with degree $d \le O_\beta(\log\log n/\log\log\log n)$ polynomials.[6] Unfortunately, an explicit construction for extractors with such parameters has not yet been achieved.

We also note that stronger correlation bounds are known in the literature for explicit (and simple) functions (see [31] and references therein). Nevertheless, we find the fact that *any* affine extractor has small correlation with low degree polynomials interesting.

The Granularity of the Fourier Spectrum of Low-Degree Polynomials over $\mathbb{F}_2$

The bias of an arbitrary function $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ is clearly some integer multiplication of 2^{-n}. Theorem 2 readily implies that the bias of a degree d polynomial on n variables has a somewhat larger granularity – the bias is a multiplication of $2^{\Omega(n^{1/(d-1)!})}/2^n$ by some integer.[7]

[6] This is the best d we can guarantee for any k, and we gain nothing more by taking $k = O(\log n)$.
[7] Throughout the paper, for readability, we supress flooring and ceiling. In the last expression, however, it should be noted that we mean 2^{k-n}, where k is some integer such that $k = \Omega(n^{1/(d-1)!})$.

In fact, Theorem 2 implies that *all* Fourier coefficients of a low degree polynomial has this granularity. To see this, apply Theorem 2 to obtain a partition $P_1, \dots, P_\ell$ of $\mathbb{F}_2^n$ to affine subspaces of dimension $k = \Omega(n^{1/(d-1)!})$, such that for each $i \in [\ell]$, $f|_{P_i}$ is some constant $f(P_i)$. Let $\beta \in \mathbb{F}_2^n$. Then,

$$\begin{aligned}2^n \cdot \widehat{f}(\beta) &= \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \beta, x\rangle} \cdot (-1)^{f(x)} = \sum_{i=1}^{\ell} \sum_{x \in P_i} (-1)^{\langle \beta, x\rangle} \cdot (-1)^{f(x)} \\ &= \sum_{i=1}^{\ell} (-1)^{f(P_i)} \cdot \sum_{x \in P_i} (-1)^{\langle \beta, x\rangle}.\end{aligned}$$

The proof then follows as for all $i \in [\ell]$, the inner sum $\sum_{x \in P_i} (-1)^{\langle \beta, x\rangle}$ is either 0 or $\pm 2^k$.

1.3 Proof Overview

In this section we give proof sketches for some of our structural results. We start with Theorem 1, and consider first the special case $q = 2$. As mentioned, the proof for this special case follows the proof idea of [28]. We then consider general finite fields and present the new ideas required for this case.

We are given a point $u_0 \in \mathbb{F}_2^n$ and assume, without loss of generality, that $f(u_0) = 0$. We iteratively construct affine subspaces, restricted to which, f is zero. We start with affine subspaces of dimension 0, which are just the singletons $\{x\}$, where $x \in \mathbb{F}_2^n$ is such that $f(x) = 0$. Assume that we were able to find basis vectors $\Delta_1, \dots, \Delta_k$ for a subspace U such that f restricted $u_0 + U$ is constantly 0. Consider all cosets $x + U$, restricted to which f is constantly 0. We call such cosets *good*. Clearly the coset $u_0 + U$ is good. If at least one more good coset $x + U$ exists, then we can pick a new direction Δ_{k+1} to be $x + u_0$, and get that f is zero on $u_0 + \operatorname{span}\{\Delta_1, \dots, \Delta_{k+1}\}$, as indeed

$$\begin{aligned}u_0 + \operatorname{span}\{\Delta_1, \dots, \Delta_{k+1}\} &= (u_0 + \operatorname{span}\{\Delta_1, \dots, \Delta_k\}) \cup (u_0 + \Delta_{k+1} + \operatorname{span}\{\Delta_1, \dots, \Delta_k\}) \\ &= (u_0 + \operatorname{span}\{\Delta_1, \dots, \Delta_k\}) \cup (x + \operatorname{span}\{\Delta_1, \dots, \Delta_k\}).\end{aligned}$$

The main observation used to derive Theorem 1 is the following. Given $\Delta_1, \dots, \Delta_k$, there exists a degree $D \le d^2 \cdot k^{d-1}$ polynomial $t : \mathbb{F}_2^n \to \mathbb{F}_2$, such that $x + U$ is a good coset if and only if $t(x) = 1$. Since we know that t is not the constant 0 function (as $t(u_0) = 1$), the DeMillo-Lipton-Schwartz-Zippel lemma (see Lemma 13) implies that there are at least 2^{n-D} x's such that $t(x) = 1$, namely, 2^{n-D} good cosets. So in each iteration, by our choice of Δ_{k+1}, we ensure that one coset in the next iteration is good, and then use DeMillo-Lipton-Schwartz-Zippel to claim that many other cosets are good as well. One can continue expanding the subspace U until $n \le D$, which completes the proof.

For a general finite field, $\mathbb{F}_q$, we similarly define a polynomial $t(x)$ over $\mathbb{F}_q$ that attains only the values 0 and 1, and whose 1's capture the good cosets. The polynomial $t(x)$ is of degree at most $(q-1) \cdot d^2 \cdot k^{d-1}$. We wish to find a new direction Δ_{k+1}, linearly independent of $\Delta_1, \dots, \Delta_k$, such that all cosets along the line $\{u_0 + \Delta_{k+1} \cdot a\}_{a \in \mathbb{F}_q}$, i.e. $\{u_0 + \Delta_{k+1} \cdot a + U\}_{a \in \mathbb{F}_q}$, are good. Over $\mathbb{F}_2$ this task was easy since $u_0 + U$ and $x + U$ define such a line.

The main new idea needed over $\mathbb{F}_q$ is to consider a polynomial

$$s(y) = \prod_{a \in \mathbb{F}_q} t(u_0 + y \cdot a),$$

whose variable represents a direction in $\mathbb{F}_q^n$ rather than a point. Note that $s(y)$ has degree at most $q \cdot \deg(t)$ and that $s(y) = 1$ if and only if $t(u_0 + y \cdot a) = 1$ for all $a \in \mathbb{F}_q$. Thus, $s(y) = 1$ iff f is zero on all cosets $\{u_0 + y \cdot a + U\}_{a \in \mathbb{F}_q}$, whose union is a dimension $k+1$ affine subspace as long as $y \notin U$. As before, since $s(0) = 1$, by a generalized DeMillo-Lipton-Schwartz-Zippel lemma, it holds that $s(\cdot)$ has many 1's, and as long as $k \ll n^{1/(d-1)}$ there is some $y \in s^{-1}(1)$ such that $y \notin U$. We can now pick such a y as Δ_{k+1}. A slightly more careful argument shows that actually there is no dependency of the dimension k in the field size q.

The proof of the second structural result (Theorem 2) can be described informally as follows. Consider a degree d polynomial f. Theorem 1 implies the existence of an affine subspace $u_0 + U$ with dimension $\Omega(n^{1/(d-1)})$ on which f is constant. One can then show (see Lemma 16) that restricting f to any affine shift of U yields a degree (at most) $d-1$ polynomial. Thus, one can partition each such affine subspace recursively to obtain a partition of $\mathbb{F}_q^n$ to affine subspaces (not necessarily shifts of one another), such that f is constant on each one of them.

In fact, to prove Theorem 2, one is not required to find an affine subspace on which f is constant, and it suffices to find an affine subspace on which the degree of f decreases. In order to obtain the first algorithmic result (Theorem 22), we devise an algorithm that finds such an affine subspace and proceed similarly to the proof of Theorem 2. To obtain the second algorithmic result (Theorem 24), we observe that the polynomial t described above has many linear factors. This structure of t allows us to save on the running time.

The generalization of Theorems 1 and 2 to more than one polynomial is quite straightforward.

2 Preliminaries

We shall denote prime numbers with the letter p and prime powers with q. The set $\{1, \dots, n\}$ is denoted by $[n]$. We denote by $\log(\cdot)$ the logarithm to the base 2. Throughout the paper, for readability sake, we suppress flooring and ceiling. For $x, y \in \mathbb{F}_q^n$ we denote by $\langle x, y \rangle$ their scalar product over $\mathbb{F}_q$, i.e., $\langle x, y \rangle = \sum_{i=1}^n x_i \cdot y_i$. The vector e_i is the unit vector defined as having 1 in the i^{th} entry and 0 elsewhere. For a set $T \subseteq [n]$, we denote by $\mathbf{1}_T$ the indicating vector of T with 1 in the i^{th} entry if $i \in T$ and 0 otherwise. For a vector $\alpha \in \mathbb{N}^m$, we denote its *weight* by $\mathrm{wt}(\alpha) \triangleq \sum_i \alpha_i$.

The statistical distance between two random variables X, Y, over the same domain D, denoted by $\mathrm{SD}(X, Y)$, is defined as $\mathrm{SD}(X, Y) = \max_{A \subseteq D} |\Pr[X \in A] - \Pr[Y \in A]|$. It is known that $\mathrm{SD}(X, Y)$ is a metric. More precisely, it is (up to a multiplicative constant factor of 2) the ℓ_1 norm of the vector $(\Pr[d \in X] - \Pr[d \in Y])_{d \in D} \in \mathbb{R}^{|D|}$. In particular, we have the triangle inequality: for X, Y, Z over D, $\mathrm{SD}(X, Z) \le \mathrm{SD}(X, Y) + \mathrm{SD}(Y, Z)$. Moreover, if X can be written as a convex combination of two random variables Y, Z as follows $X = (1 - \gamma) \cdot Y + \gamma \cdot Z$, where $\gamma \in [0, 1]$, then $\mathrm{SD}(X, Y) \le \gamma$. We sometimes abuse notation, and for a set $S \subseteq D$, consider S also as the random variable that is uniformly distributed over the set S.

Restriction to an affine subspace

Let $f\colon \mathbb{F}_q^n \to \mathbb{F}_q$ be a function, $U \subseteq \mathbb{F}_q^n$ a subspace of dimension k and $u_0 \in \mathbb{F}_q^n$ some vector. We denote by $f|_{u_0+U} : (u_0 + U) \to \mathbb{F}_q$ the restriction of f to $u_0 + U$. The degree of $f|_{u_0+U}$ is defined as the minimal degree of a polynomial (from $\mathbb{F}_q^n$ to $\mathbb{F}_q$) that agrees with f on $u_0 + U$. For recursive arguments, it will be very useful to fix some basis $u_1, \dots, u_k$ for U and to consider the function $g : \mathbb{F}_q^k \to \mathbb{F}_q$ defined by

$$g(x_1, \ldots, x_k) = f\left(u_0 + \sum_{i=1}^{k} x_i \cdot u_i\right).$$

Note that the $\deg(g) = \deg(f|_{u_0+U})$ regardless of the choice for the basis.

Polynomials

We review some definitions and known facts about polynomials that we use.

The degree of a function $f : \mathbb{F}_q^n \to \mathbb{F}_q$, denoted by $\deg(f)$, is the degree of the unique multivariate polynomial over $\mathbb{F}_q$, where each individual degree is at most $q-1$, which agrees with f on $\mathbb{F}_q^n$. In the special case $q = 2$, such polynomials are called multi-linear. We will abuse notation and interchange between a function and its unique polynomial over $\mathbb{F}_q$ that agrees with f on $\mathbb{F}_q^n$.

▶ **Definition 12.** Let $f\colon \mathbb{F}_q^n \to \mathbb{F}_q$ be a polynomial of degree d, and let $\Delta \in \mathbb{F}_q^n$. The polynomial $\frac{\partial f}{\partial \Delta}(x) \triangleq f(x+\Delta) - f(x)$ is called the *derivative of f in direction Δ*.

It is easy to verify that $\deg\left(\frac{\partial f}{\partial \Delta}\right) \leq \deg(f) - 1$. Let $\Delta_1, \ldots, \Delta_k \in \mathbb{F}_q^n$ then

$$\frac{\partial^k f}{\partial \Delta_1 \ldots \partial \Delta_k}(x) = \sum_{S \subseteq [k]} (-1)^{1+|S|} \cdot f\left(x + \sum_{i \in S} \Delta_i\right)$$

is a degree $\leq \deg(f) - k$ polynomial.

The following lemma is a variant of the well-known DeMillo-Lipton-Schwartz-Zippel lemma [10, 25, 34].

▶ **Lemma 13** (DeMillo-Lipton-Schwartz-Zippel). *Let q be a prime power. Let $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ be a degree d non-zero polynomial. Then, $\Pr_{x \sim \mathbb{F}_q^n}[f(x_1, \ldots, x_n) \neq 0] \geq q^{-d/(q-1)}$.*

For completeness, we give the proof of Lemma 13 in Appendix C. The following folklore fact about polynomials over $\mathbb{F}_2$ is easy to verify.

▶ **Lemma 14** (Möbius inversion formula). *Let $f(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} a_S \cdot \prod_{i \in S} x_i$ be a polynomial over $\mathbb{F}_2$. Then, its coefficients are given by the formula: $a_S = \sum_{T \subseteq S} f(\mathbf{1}_T)$.*

Circuits

A Boolean circuit is an unbounded fan-in circuit composed of OR and AND gates, and literals x_i, $\neg x_i$. The size of such a circuit is the number of gates in it. A Boolean formula is a Boolean circuit such that every OR and AND gate has fan-out 1. De Morgan formula is a Boolean formula where each gate has fan-in at most 2. We recall that an AC^0 circuit is a Boolean circuit of polynomial size and constant depth. An $\mathsf{AC}^0[\oplus]$ circuit is an AC^0 circuit with unbounded fan-in XOR gates as well.

3 Structural Results

This section contains the proofs of all the structural results in this paper. In Section 3.1 we give a proof for Theorem 1. Section 3.2 contains the proof for Theorem 2. The tightness of the first structural result is given in Section 3.3. In Section 3.4 we describe the generalization of the two structural results to many polynomials. In Section 3.5 we prove Theorem 3.

3.1 Proof of Theorem 1

In this section we prove Theorem 1. For a slightly simpler proof, for the special case $q = 2$, we refer the reader to Appendix B. The proof of Theorem 1 is based on the following lemma.

▶ **Lemma 15.** *Let $f: \mathbb{F}_q^n \to \mathbb{F}_q$ be some function, and let U be a subspace of $\mathbb{F}_q^n$ with basis vectors $\Delta_1, \dots, \Delta_k$. Then, there exist polynomials $(f_\alpha)_{\alpha \in \{0,1,\dots,q-1\}^k}$ such that*
1. $\deg(f_\alpha) \le \deg(f) - \mathrm{wt}(\alpha)$ *for all* $\alpha \in \{0,1,\dots,q-1\}^k$.
2. *Let $x \in \mathbb{F}_q^n$, then $f|_{x+U} \equiv 0$ if and only if $f_\alpha(x) = 0$ for all $\alpha \in \{0,1,\dots,q-1\}^k$.*

Proof. Complete $\Delta_1, \dots, \Delta_k$ into a basis of $\mathbb{F}_q^n$ by picking vectors $\Delta_{k+1}, \dots, \Delta_n \in \mathbb{F}_q^n$. Let A be the linear transformation which maps the standard basis into $\Delta_1, \dots, \Delta_n$, and let $g(y) := f(Ay)$ (alternatively, $f(x) = g(A^{-1}x)$). Write g as a polynomial over $\mathbb{F}_q$:

$$g(y) = \sum_{\gamma \in \{0,1,\dots,q-1\}^n} c_\gamma \cdot \prod_{i=1}^{n} y_i^{\gamma_i} .$$

Since both f and g can be obtained from one another by applying a linear transformation to the inputs, we have $\deg(f) = \deg(g)$. Think of the input to g as a concatenation of two parts $y = z \circ w$, where $z \in \mathbb{F}_q^k$, $w \in \mathbb{F}_q^{n-k}$. Let $P_z : \mathbb{F}_q^n \to \mathbb{F}_q^k$ be the projection of a vector of length n to the first k coordinates and let $P_w : \mathbb{F}_q^n \to \mathbb{F}_q^{n-k}$ be the projection to the last $n-k$ coordinates. We may rewrite g as

$$g(z \circ w) = \sum_{\alpha \in \{0,1,\dots,q-1\}^k} \sum_{\beta \in \{0,1,\dots,q-1\}^{n-k}} c_{\alpha\circ\beta} \cdot \prod_{i=1}^{k} z_i^{\alpha_i} \cdot \prod_{i=1}^{n-k} w_i^{\beta_i} .$$

By reordering the summations we get

$$g(z \circ w) = \sum_{\alpha \in \{0,1,\dots,q-1\}^k} g_\alpha(w) \cdot \prod_{i=1}^{k} z_i^{\alpha_i} ,$$

where

$$g_\alpha(w) = \sum_{\beta \in \{0,1,\dots,q-1\}^{n-k}} c_{\alpha\circ\beta} \cdot \prod_{i=1}^{n-k} w_i^{\beta_i} .$$

Note that $\deg(g_\alpha) \le \deg(g) - \mathrm{wt}(\alpha)$. We have

$$\begin{aligned} f|_{x+U} \equiv 0 \quad &\iff \quad g|_{A^{-1}x + A^{-1}U} \equiv 0 \\ &\iff \quad g|_{A^{-1}x + \mathrm{span}\{e_1,\dots,e_k\}} \equiv 0 \ (*) . \end{aligned}$$

Writing $(z, w) = (P_z(A^{-1}x), P_w(A^{-1}x))$ gives

$$\begin{aligned} (*) \quad &\iff \quad \forall z' \in \mathbb{F}_q^k : g(z' \circ w) = 0 \\ &\iff \quad \forall \alpha : g_\alpha(w) = 0 \\ &\iff \quad \forall \alpha : g_\alpha(P_w(A^{-1}x)) = 0 . \end{aligned}$$

Taking f_α to be the composition $g_\alpha \circ P_w \circ A^{-1}$ we obtain Item 2. As $P_w \circ A^{-1}$ is simply a linear transformation, it is clear that $\deg(f_\alpha) \le \deg(g_\alpha) \le \deg(g) - \mathrm{wt}(\alpha) \le \deg(f) - \mathrm{wt}(\alpha)$, which completes the proof. ◀

Proof of Theorem 1. Assume without loss of generality that $f(u_0) = 0$, as otherwise we can look at the polynomial $g(x) = f(x) - f(u_0)$ which is of the same degree. The proof is by induction. Let k be such that

$$n > k + (d+1) \cdot \sum_{j=0}^{d-1} (d-j) \cdot \binom{k+j-1}{j} . \tag{2}$$

We assume by induction that there exists an affine subspace $u_0 + \mathrm{span}\{\Delta_1, \ldots, \Delta_k\} \subseteq \mathbb{F}_q^n$, where the Δ_i's are linearly independent vectors, on which f evaluates to 0. Assuming Equation 2 holds, we show there exists a vector Δ_{k+1}, linearly independent of $\Delta_1, \ldots, \Delta_k$, such that $f \equiv 0$ on $u_0 + \mathrm{span}\{\Delta_1, \ldots, \Delta_{k+1}\}$. To this aim, consider the set

$$A = \left\{ x \in \mathbb{F}_q^n \;\middle|\; f|_{x+\mathrm{span}\{\Delta_1,\ldots,\Delta_k\}} \equiv 0 \right\} .$$

By the induction hypothesis, $u_0 \in A$. By Lemma 15, for any $x \in \mathbb{F}_q^n$,

$$f|_{x+\mathrm{span}\{\Delta_1,\ldots,\Delta_k\}} \equiv 0 \quad \Longleftrightarrow \quad \forall \alpha \in \{0,1,\ldots,q-1\}^k : f_\alpha(x) = 0 ,$$

where f_α is of degree at most $d - \mathrm{wt}(\alpha)$. Thus $f_\alpha \equiv 0$ for $\mathrm{wt}(\alpha) > d$, and we may write A as

$$A = \left\{ x \in \mathbb{F}_q^n \mid \forall \alpha : \mathrm{wt}(\alpha) \le d,\ f_\alpha(x) = 0 \right\} .$$

Hence, A is the set of solutions to a system of $\le \binom{k+d}{d}$ polynomial equations, where there are at most $\binom{k+j-1}{j}$ equations which correspond to α's of weight j and thus to degree (at most) $d-j$ polynomials. One can also write A as the set of non-zeros to the single polynomial

$$t(x) := \prod_{\alpha : \mathrm{wt}(\alpha) \le d} \left(1 - f_\alpha(x)^{q-1}\right) ,$$

which is of degree

$$\deg(t) \le (q-1) \cdot \sum_{j=0}^{d-1} (d-j) \cdot \binom{k+j-1}{j} .$$

Note that $t(x)$ obtains only the values 0 and 1. Let $R \subseteq \mathbb{F}_q$ be an arbitrary subset of $\mathbb{F}_q$ with size $|R| = \min(q, d+1)$. Define a polynomial $s(y) := \prod_{r \in R} t(u_0 + r \cdot y)$. We claim that any non-zero of s not in the span of $\{\Delta_1, \ldots, \Delta_k\}$ can be taken to be the desired Δ_{k+1}. Indeed, if y is such that $s(y) = 1$, then $t(u_0 + r \cdot y) = 1$ for all $r \in R$. That is, for every $z \in \mathrm{span}(\Delta_1, \ldots, \Delta_k)$ and any $r \in R$ it follows that $f(u_0 + z + r \cdot y) = 0$. Namely, f obtains $|R|$ roots on the affine line with offset $u_0 + z$ and direction y. If $R = \mathbb{F}_q$ then clearly this implies that f is the zero function restricted to the line. Otherwise, $|R| = d+1$ and thus f, which is a degree d polynomial, obtains $d+1$ zeros on the line. Thus, again f is the zero function on this line. Hence, $f(u_0 + z + r \cdot y) = 0$ for all $r \in \mathbb{F}_q$.

Thus, we just have to show that there exists some non-zero of s which is linearly independent of $\{\Delta_1, \ldots, \Delta_k\}$. Since the trivial solution $y = 0$ is a non-zero of s, we get that s is not the constant 0 function. Thus, by Lemma 13 it holds that $\Pr[s(y) \neq 0] \ge q^{-\deg(s)/(q-1)}$. The above equation implies that s has at least $q^{n-\deg(s)/(q-1)}$ ones. Since we need to avoid q^k linear combinations of the previous $\Delta_1, \ldots, \Delta_k$, it is enough to have

$$n - \frac{\deg(s)}{q-1} > k . \tag{3}$$

Since

$$\deg(s) \le (d+1)\cdot(q-1)\cdot\sum_{j=0}^{d-1}(d-j)\cdot\binom{k+j-1}{j}$$

and by the assumption on k in Equation (2) we have that Equation (3) holds. ◀

3.2 Proof of Theorem 2

In this section we prove Theorem 2. To this end we use the following lemma.

▶ **Lemma 16.** *Let q be a prime power. Let $f\colon \mathbb{F}_q^n \to \mathbb{F}_q$ be a degree d polynomial. Assume there exists an affine subspace u_0+U, restricted to which f has degree at most $d-1$. Then, the degree of f restricted to any affine shift of U is at most $d-1$.*

Proof. Fix $u_1 \in \mathbb{F}_q^n$. Now, for any $u \in U$

$$f(u_1+u) = f(u_1+u) - f(u_0+u) + f(u_0+u) = \frac{\partial f}{\partial(u_1-u_0)}(u_0+u) + f(u_0+u).$$

Since the degree of the partial derivative of f is at most $d-1$ and the degree of $f|_{u_0+U}$ is also at most $d-1$, we get that $f|_{u_1+U}$ has degree at most $d-1$. ◀

Proof of Theorem 2. Let $c_1 \in (0,1)$ be the constant from Theorem 1. Define the sequence $\{\beta_d\}_{d=1}^{\infty}$ as follows.

$$\beta_d = \begin{cases} 1/2, & d=1; \\ \beta_{d-1}\cdot c_1^{\frac{1}{(d-2)!}}, & d>1. \end{cases}$$

We will prove by induction on d, the degree of a given polynomial f, that there exists a partition of $\mathbb{F}_q^n$ to affine subspaces of dimension $\ge \beta_d \cdot n^{1/(d-1)!}$, such that f restricted to each part is constant. The proof then follows by noting that for all $d \ge 1$,

$$\beta_d = \frac{1}{2}\cdot c_1^{\frac{1}{(d-2)!}+\cdots+\frac{1}{1!}+\frac{1}{0!}} \ge \frac{c_1^e}{2},$$

and thus one can take $c_2 = c_1^e/2$ to be the constant in the theorem statement.

The base case of the induction, namely $d=1$, trivially follows as f is an affine function, and we can partition $\mathbb{F}_q^n$ to q affine subspaces of dimension $n-1 \ge n/2 = \beta_1 n$, such that on each of which f is constant. Assume now that f is a degree $d>1$ polynomial. By Theorem 1 and Lemma 16, there exists a partition of $\mathbb{F}_q^n$ to affine subspaces of dimension $k \ge c_1\cdot n^{1/(d-1)}$, such that f restricted to any affine subspace in the partition has degree at most $d-1$. Fix some affine subspace u_0+U in this partition, and apply the induction hypothesis to the polynomial $f' = f|_{u_0+U}$, which has degree $d' \le d-1$. [8] By the induction hypothesis, we obtain a partition of u_0+U such that f is constant on each part. Moreover, the dimension of each such part is at least

$$\beta_{d'}\cdot k^{\frac{1}{(d'-1)!}} \ge \beta_{d-1}\cdot k^{\frac{1}{(d-2)!}} \ge \beta_{d-1}\cdot\left(c_1\cdot n^{\frac{1}{d-1}}\right)^{\frac{1}{(d-2)!}} = \beta_{d-1}\cdot c_1^{\frac{1}{(d-2)!}}\cdot n^{\frac{1}{(d-1)!}} = \beta_d\cdot n^{\frac{1}{(d-1)!}},$$

where the first inequality follows since $\{\beta_d\}_{d=1}^{\infty}$ is monotonically decreasing and $d' \le d-1$, and the last equality follows by the definitions of the β_d's. ◀

[8] We may apply the induction because there exists a linear bijection from U to $\mathbb{F}_q^{\dim U}$. More precisely, if A is an $n\times k$ matrix over $\mathbb{F}_q$ that maps U to $\mathbb{F}_q^k$ bijectively, then one can apply the induction to the polynomial $f''(x) = f'(u_0+Ax)$, defined on k variables, and then induce a partition of u_0+U from the partition of $\mathbb{F}_q^k$ obtained by the induction. The induction can be carried on f'' since $\deg f'' \le \deg f' \le d-1$, where the first inequality holds because the variables of f'' are linear combinations of the variables of f'.

3.3 On the Tightness of Structural Result I

Roughly speaking, Theorem 1 states that for any prime power q, a degree d polynomial over $\mathbb{F}_q$ in n variables is not an affine disperser for dimension $k = \Omega(n^{1/(d-1)})$. We mentioned that this result is tight in the sense that by increasing k a bit, there exists a degree d polynomial which is an affine disperser. In this section we show, that in the special case $q = 2$, a stronger claim can be proven. Namely, by increasing k a bit, there exists a degree d polynomial which is an affine extractor.

▶ **Theorem 17.** *There exists a constant c such that the following holds. Let n, d be such that $d < n/2$. There exists a degree d polynomial $f : \mathbb{F}_2^n \to \mathbb{F}_2$, such that for every affine subspace $u_0 + U \subseteq \mathbb{F}_2^n$ of dimension $k \geq cd \cdot n^{1/(d-1)}$, $\operatorname{bias}(f|_{u_0+U}) \leq 2^{-\Omega(k/d)}$.*

To prove Theorem 17 we apply the following lemma due to Ben-Eliezer, Hod and Lovett [2].

▶ **Lemma 18** ([2], Lemma 2)**.** *Fix $\varepsilon > 0$ and let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a random degree d polynomial*[9] *for $d \leq (1-\varepsilon)n$. Then,*

$$\Pr_f\left[\operatorname{bias}(f) > 2^{-c_1 n/d}\right] \leq 2^{-c_2\binom{n}{\leq d}},$$

where $0 < c_1, c_2 < 1$ are constants depending only on ε.

Proof of Theorem 17. Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a random polynomial of degree at most d. Fix an affine subspace $u_0 + U \subseteq \mathbb{F}_2^n$ of dimension k. One can easily show that $f|_{u_0+U}$ is equidistributed as a random polynomial on k variables, of degree at most d. Therefore, by Lemma 18,

$$\Pr_f\left[\operatorname{bias}(f|_{u_0+U}) > 2^{-c_1 k/d}\right] \leq 2^{-c_2\binom{k}{\leq d}},$$

where c_1, c_2 are the constants from Lemma 18 suitable for the (somewhat arbitrary) choice $\varepsilon = 1/2$. By taking the union bound over all $\leq 2^n \cdot \binom{2^n}{k}$ affine subspaces of $\mathbb{F}_2^n$ of dimension k, it is enough to require that

$$2^{-c_2\binom{k}{\leq d}} \cdot 2^n \cdot \binom{2^n}{k} < 1$$

so to conclude the proof of the theorem. It is easy to verify that one can choose c, as a function of c_2, such that the above equation does hold for k as defined in the theorem statement. ◀

3.4 Generalization of the Structural Results to Many Polynomials

▶ **Theorem 19** (Structural Result I for many polynomials)**.** *Let q be a prime power. Let $f_1, \ldots, f_t : \mathbb{F}_q^n \to \mathbb{F}_q$ be polynomials of degree $d_1, \ldots, d_t$ respectively. Let k be the least integer satisfying the inequality*

$$n \leq k + \sum_{i=1}^{t} (d_i + 1) \cdot \sum_{j=0}^{d_i - 1} (d_i - j) \cdot \binom{k+j-1}{j}.$$

Then, for every $u_0 \in \mathbb{F}_q^n$ there exists a subspace $U \subseteq \mathbb{F}_q^n$ of dimension k, such that for all $i \in [t]$, f_i restricted to $u_0 + U$ is a constant function. In particular, if $d_1, \ldots, d_t \leq d$ then $k = \Omega((n/t)^{1/(d-1)})$. Moreover, for $d \leq \log(n/t)/10$, $k = \Omega(d \cdot (n/t)^{1/(d-1)})$.

[9] That is, every monomial of degree at most d appears in f with probability 1/2, independently of all other monomials.

Before proving Theorem 19 we note that by applying a probabilistic argument, it can be shown that the theorem is tight. In particular, it has the right dependency in the number of polynomials t.

Proof. The proof is very similar to that of Theorem 1, so we only highlight the differences. As in the proof of Theorem 1, we may assume that $f_1, \dots, f_t$ evaluate to 0 at u_0. We build by induction an affine subspace $u_0 + U$ on which all the t polynomials evaluate to 0. Given we already picked basis vectors $\Delta_1, \dots, \Delta_k$, we consider the set A to be the following:

$$A = \left\{ x \in \mathbb{F}_q^n \;\middle|\; \forall i \in t,\;\; f_i|_{x+\mathrm{span}\{\Delta_1,\dots,\Delta_k\}} \equiv 0 \right\}.$$

As in the proof of Theorem 1, A can be written as the set of solutions to a single polynomial equation $t(x) = 1$, where

$$\deg(t) \le (q-1) \cdot \sum_{i=1}^{t} (d_i + 1) \sum_{j=0}^{d_i - 1} (d_i - j) \cdot \binom{k+j-1}{j},$$

Similarly to Theorem 1, the polynomial s is now defined, where $\deg(s) \le (d+1) \cdot \deg(t)$ and such that any non-zero of s, that is independent of $\Delta_1, \dots, \Delta_k$, can be taken to be Δ_{k+1}. By DeMillo-Lipton-Schwartz-Zippel lemma, it follows that as long k is not too large, such a root can be found. ◀

Similarly to the way we deduced Theorem 2 from Theorem 1, one can deduce the following theorem from Theorem 19. We omit the proof.

▶ **Theorem 20** (Structural Result II for many polynomials). *Let q be a prime power. Let $f_1, \dots, f_t : \mathbb{F}_q^n \to \mathbb{F}_q$ be polynomials of degree at most d. Then, there exists a partition of $\mathbb{F}_q^n$ to affine subspaces, each of dimension $\Omega(n^{1/(d-1)!}/t^e)$, such that $f_1, \dots, f_t$ are all constant on each part.*

3.5 Sparse Polynomials

In this section we prove Theorem 3. To this end, we prove the following lemma.

▶ **Lemma 21.** *Let f be a polynomial on n variables over $\mathbb{F}_q$, with n^c monomials. If f is an affine disperser for dimension k, then there exists a subspace U of dimension $\Omega(\sqrt{n})$ on which $f|_U$ is of degree at most $2(q-1)c$.*

Lemma 21 implies Theorem 3. Indeed, the above lemma states that for any polynomial f on n variables and n^c monomials over $\mathbb{F}_q$, there exists an affine subspace of $\mathbb{F}_q^n$, with dimension $k(\Omega(\sqrt{n}), 2(q-1)c)$, on which f is constant. By Theorem 1, $k(\Omega(\sqrt{n}), 2(q-1)c) = \Omega(n^{1/(4(q-1)c)})$, as desired.

Proof of Lemma 21. We perform a random restriction to all variables $x_1, \dots, x_n$. For each $i \in [n]$, independently, with probability $1 - (2 \cdot n^c)^{-1/(2c)}$, we set x_i to 0. Consider a monomial that has at least $2c$ distinct variables. The probability that such a monomial survives the restriction is at most $1/(2 \cdot n^c)$. Thus, by the union bound, with probability at least 1/2, no monomial with more than $2c$ distinct variables survived the restriction. Restricting ourselves to this event, since we may assume that the individual degree of each variable in the original polynomial is at most $q-1$, any surviving monomial has degree at most $2(q-1)c$.

The expected number of variables that survived the random restriction is $n \cdot (2 \cdot n^c)^{-1/(2c)} = \Omega(\sqrt{n})$. Thus, by the Chernoff bound, with probability at least, say, 3/4, the number of surviving variables is $\Omega(\sqrt{n})$.

Thus, there exists a restriction of the variables that keeps $\Omega(\sqrt{n})$ of them alive, and such that the resulting polynomial has degree at most $2(q-1)c$. ◀

4 The Algorithmic Aspect

4.1 Efficient Algorithm for Finding a Somewhat Large Subspace

▶ **Theorem 22.** *Let $f \colon \mathbb{F}_2^n \to \mathbb{F}_2$ be a polynomial of degree $d \leq \log(n)/3$ given as a black-box. Then, there exists an algorithm that makes $\mathrm{poly}(n)$ queries to f, runs in time $\mathrm{poly}(n)$, and finds an affine subspace U of dimension $\Omega(d \cdot n^{1/(d-1)})$ such that $\deg(f|_U) \leq d-1$.*

The proof of Theorem 22 is deferred to Appendix B.1 as it relies on notations and ideas from the proof of the first structural result for the binary field, which can be found in Appendix B. We advise the reader to look at the latter section before reading the proof of Theorem 22.

Theorem 22 yields the following corollary.

▶ **Corollary 23.** *There exists an algorithm that given a degree d polynomial $f : \mathbb{F}_2^n \to \mathbb{F}_2$ as a black box, runs in $\mathrm{poly}(n)$-time and finds an affine subspace of dimension $\Omega(n^{1/(d-1)!})$ on which f is constant.*

4.2 Subexponential-Time Algorithm for Finding an Optimal Subspace

▶ **Theorem 24.** *There exists a constant $\beta > 0$ such that the following holds. There is an algorithm that given $f \colon \mathbb{F}_q^n \to \mathbb{F}_q$, a degree d polynomial (as a list of monomials), where $3 \leq d \leq \log(n)/10$, and $u_0 \in \mathbb{F}_q^n$ as inputs, finds an affine subspace $u_0 + U$ of dimension $\Omega(k(n,d))$, restricted to which f is constant. The algorithm runs in time $q^{\beta \cdot n^{(d-2)/(d-1)}} \cdot \mathrm{poly}(n^d)$, and uses $\mathrm{poly}(n^d, \log q)$ space.*

We obtain the following corollary.

▶ **Corollary 25.** *There exists a $q^{n-k} \cdot \mathrm{poly}(n^d)$-time $\mathrm{poly}(n^d, \log q)$-space algorithm that given $f \colon \mathbb{F}_q^n \to \mathbb{F}_q$, a degree d polynomial, partitions $\mathbb{F}_q^n$ to affine subspace of dimension k on each of which f is constant, where $k = \Omega(n^{1/(d-1)!})$.*

In particular, one can compute the number of satisfying assignments for f using Corollary 25.

Proof. We follow the proof of Theorem 1. Again, we may assume $f(u_0) = 0$. Given the previously chosen vectors $\Delta_1, \dots, \Delta_k$ such that f is the constant 0 on $u_0 + \mathrm{span}\{\Delta_1, \dots, \Delta_k\}$, we show how to find a new vector Δ_{k+1} which is linearly independent of $\Delta_1, \dots, \Delta_k$, such that f is constantly zero on $u_0 + \mathrm{span}\{\Delta_1, \dots, \Delta_{k+1}\}$. The set A is the set of solutions to the following set of polynomial equations:

$$\{f_\alpha(x) = 0 \ : \ \alpha \in \{0, 1, \dots, q-1\}^k, \mathrm{wt}(\alpha) \leq d-1\} \ ,$$

and by our assumptions, u_0 is a solution to all of these equations. By treating the polynomial f as a formal sum of monomials we can calculate each f_α in $\mathrm{poly}(n^d)$ time. Let R be some arbitrary subset of $\mathbb{F}_q$ of size $\min(q, d+1)$ then any solution y to the following set of equations which is linearly independent of $\Delta_1, \dots, \Delta_k$ can be the new direction Δ_{k+1}:

$$\{f_\alpha(u_0 + r \cdot y) = 0 \ : \ \alpha \in \{0, 1, \dots, q-1\}^k, \mathrm{wt}(\alpha) \leq d-1, \ r \in R\} \ .$$

It is therefore enough to find more than q^k different solutions to this set of equations, in order to guarantee that one of them will be linearly independent of the previous Δ_i's. In order to do so, we partition the set of equations into the set of linear equations and the set of non-linear equations:

$$L = \{f_\alpha(u_0 + r \cdot y) = 0 \ : \ \alpha \in \{0,1,\ldots,q-1\}^k, \mathrm{wt}(\alpha) \le d-1, \deg(f_\alpha) = 1, \ r \in R\} \,.$$
$$NL = \{f_\alpha(u_0 + r \cdot y) = 0 \ : \ \alpha \in \{0,1,\ldots,q-1\}^k, \mathrm{wt}(\alpha) \le d-1, \deg(f_\alpha) > 1, \ r \in R\} \,.$$

Let $m = \sum_{f_\alpha \in NL} \deg(f_\alpha)$. Since 0^n is a solution to all equations in $L \cup NL$, we can impose new linear equations which hold for 0^n, keeping the system consistent. More specifically, we define a new set L', which initially is equal to L, and iteratively add equations of the form $\{y_i = 0\}$ to L' until $\dim(L') = n - m - k - 1$. [10]

The set of solutions to both L' and NL is non-empty as it contains the all zeros vector. Furthermore, the sum of the degrees of equations in $L' \cup NL$ is exactly $(n-m-k-1)+m = n-k-1$. Therefore, by Lemma 13, there are at least q^{k+1} solutions to the equations in $L' \cup NL$, which guarantees that one of the solutions is linearly independent of $\Delta_1, \ldots, \Delta_k$.

Next, we show how to find all solutions to the equations in $L' \cup NL$. We find a basis for the set of solutions to L' using Gaussian elimination, and iterate over all vectors in the affine subspace this basis spans. For each vector y in this affine subspace we verify that all the equations in NL are satisfied by y. The running time of this process is $O(q^{n-\dim(L')} \cdot |NL| \cdot n^d)$, which is $O(q^{m+k+1} \cdot n \cdot n^d)$.

As $m \le \min(d+1, q) \cdot \sum_{i=0}^{d-2} (d-i) \cdot \binom{k+i-1}{i}$, an elementary calculation shows that for $k \le \frac{d}{10e} \cdot n^{1/(d-1)}$ and $3 \le d \le \log(n)/10$ we have $m + k \le \beta \cdot n^{(d-2)/(d-1)}$ for some universal constant β. Thus, the total running time of the algorithm is $q^{\beta \cdot n^{(d-2)/(d-1)}} \cdot \mathrm{poly}(n^d)$. The algorithm uses $O((|NL|+|L|) \cdot n^d \cdot \mathrm{polylog}(q))$ space to store and manipulate the polynomials f_α. In addition, $O(n^2 \cdot \mathrm{polylog}(q))$ space is used to perform the Gaussian elimination. Overall the space used by the algorithm is $O(n^{d+1} \cdot \mathrm{polylog}(q))$. ◀

5 Extractors and Dispersers for Varieties

We start this section by proving Theorem 5.

Proof of Theorem 5. Let $g_1, \ldots, g_t \colon \mathbb{F}_q^n \to \mathbb{F}_q$ be degree d polynomials. By Theorem 20, there exists a partition of $\mathbb{F}_q^n$ to affine subspaces $P_1, \ldots, P_\ell$, each of dimension $\Omega(n^{1/(d-1)!}/t^e)$, such that $g_j|_{P_i}$ is constant for all $i \in [\ell]$ and $j \in [t]$. Since f is an affine extractor for such dimension, with bias ε, then for all $i \in [\ell]$ it holds that $\mathrm{SD}(f(P_i), \mathbb{F}_q) \le \varepsilon$.

Let $I \subseteq [\ell]$ be the set of indices of affine subspaces in the partition such that $i \in I$ if and only if $g_j|_{P_i} = 0$ for all $j \in [t]$. In other words, we consider the partition of $\mathbf{V}(g_1, \ldots, g_t)$ to affine subspaces, induced by the partition of $\mathbb{F}_q^n$ to $P_1, \ldots, P_\ell$. Since the P_i's are disjoint, the random variable $f(\mathbf{V}(g_1, \ldots, g_t)) = f(\cup_{i \in I} P_i)$ is a convex combination of the random variables $\{f(P_i)\}_{i \in I}$. Thus, $\mathrm{SD}(f(\mathbf{V}(g_1, \ldots, g_t)), \mathbb{F}_q) \le \max_{i \in I} \mathrm{SD}(f(P_i), \mathbb{F}_q) \le \varepsilon$. ◀

We now give a formal statement and proof for the reduction from extractors for varieties to affine extractors, which does not depend on the number of polynomials defining the variety, but rather on the variety size.

[10] We add these constraints as concentrating at finding a solution of this form (that is, a solution that satisfies all equations in $L' \cup NL$ rather than only the equations in $L \cup NL$) is easier from the computational aspect.

▶ **Theorem 26.** *For every $d \in \mathbb{N}$ and $\delta, \rho \in (0,1)$ the following holds. Let $f\colon \mathbb{F}_q^n \to \mathbb{F}_q$ be an affine extractor for dimension $\Omega(n^{1/(d-1)!}/\ell^e)$ with bias ε, where $\ell = \log_q(1/(\rho\delta))$. Then, f is an extractor with bias $\varepsilon + \delta$ for varieties with density at least ρ (i.e., size at least $\rho \cdot q^n$), that are the common zeros of any degree (at most) d polynomials.*

Proof. Let $g_1, \ldots, g_t\colon \mathbb{F}_q^n \to \mathbb{F}_q$ be degree (at most) d polynomials. First, we prove the existence of ℓ polynomials $h_1, \ldots, h_\ell\colon \mathbb{F}_q^n \to \mathbb{F}_q$, each of degree at most d, with a variety that approximates $\mathbf{V}(g_1, \ldots, g_t)$. More precisely, we will have

$$\mathbf{V}(g_1, \ldots, g_t) \subseteq \mathbf{V}(h_1, \ldots, h_\ell) \quad \text{and} \quad \Pr_{x\sim\mathbb{F}_q^n}[x \in \mathbf{V}(h_1, \ldots, h_\ell) \setminus \mathbf{V}(g_1, \ldots, g_t)] \le q^{-\ell}, \quad (4)$$

The proof of this claim follows by a standard argument, like the one that appears in [21, 27]: Let $\alpha_1, \ldots, \alpha_\ell$ be random vectors, sampled uniformly and independently from $\mathbb{F}_q^t$. For each $i \in [\ell]$, define the (random) polynomial

$$H_i(x) = \sum_{j=1}^{t} (\alpha_i)_j \cdot g_j(x),$$

where the summation and multiplications are taken over $\mathbb{F}_q$. Clearly, if $x \in \mathbf{V}(g_1, \ldots, g_t)$ then $H_i(x) = 0$ with probability 1 (where the probability is taken over $\alpha_1, \ldots, \alpha_\ell$). Otherwise, for each $i \in [\ell]$, $\Pr[H_i(x) = 0] = 1/q$. By an averaging argument, one can fix $\alpha_1, \ldots, \alpha_\ell$ and obtain fixed polynomials $h_1, \ldots, h_\ell$, of degree at most d, that satisfy the conditions in Equation (4).

Since f is an affine extractor with bias ε for dimension $\Omega(n^{1/(d-1)!}/\ell^e)$, Theorem 5 implies that $\mathrm{SD}(f(\mathbf{V}(h_1, \ldots, h_\ell)), \mathbb{F}_q) \le \varepsilon$. To conclude the proof, we show that

$$\mathrm{SD}(f(\mathbf{V}(h_1, \ldots, h_\ell)), f(\mathbf{V}(g_1, \ldots, g_t))) \le \delta.$$

To see this, observe that $\mathbf{V}(h_1, \ldots, h_\ell)$ can be written as a convex combination

$$\mathbf{V}(h_1, \ldots, h_\ell) = \frac{|\mathbf{V}(g_1, \ldots, g_t)|}{|\mathbf{V}(h_1, \ldots, h_\ell)|} \cdot \mathbf{V}(g_1, \ldots, g_t) + \left(1 - \frac{|\mathbf{V}(g_1, \ldots, g_t)|}{|\mathbf{V}(h_1, \ldots, h_\ell)|}\right) \cdot \mathcal{E},$$

where $\mathcal{E}$ is some random variable over $\mathbb{F}_q$. Thus, by Equation (4),

$$\mathrm{SD}(\mathbf{V}(h_1, \ldots, h_\ell), \mathbf{V}(g_1, \ldots, g_t)) \le 1 - \frac{|\mathbf{V}(g_1, \ldots, g_t)|}{|\mathbf{V}(h_1, \ldots, h_\ell)|} \le \frac{q^{-\ell}}{\rho} = \delta.$$

This implies that $\mathrm{SD}(f(\mathbf{V}(h_1, \ldots, h_\ell)), f(\mathbf{V}(g_1, \ldots, g_t))) \le \delta$, as claimed. ◀

Next, we prove Theorem 6 which gives an analog reduction from dispersers for varieties to affine dispersers.

Proof of Theorem 6. Let $g_1, \ldots, g_t\colon \mathbb{F}_q^n \to \mathbb{F}_q$ be degree (at most) d polynomials. Let $u_0 \in \mathbf{V}(g_1, \ldots, g_t)$ (if $\mathbf{V}(g_1, \ldots, g_t) = \emptyset$, there is nothing to prove). By Theorem 19, there exists a subspace U of dimension $\Omega(d \cdot (n/t)^{1/(d-1)})$ such that $u_0 + U \subseteq \mathbf{V}(g_1, \ldots, g_t)$. The proof then follows as f is an affine disperser for dimension $\Omega(d \cdot (n/t)^{1/(d-1)})$. ◀

6 From Affine Dispersers to Affine Extractors

To prove Theorem 8, we use the following theorem of Kaufman and Lovett [19].

▶ **Theorem 27** ([19]). *Let p be a prime number and let $f \colon \mathbb{F}_p^n \to \mathbb{F}_p$ be a degree (at most) d polynomial with* $\text{bias}(f) \geq \delta$. *Then, there exist $c = c(d, \delta)$ polynomials $f_1, \ldots, f_c$ of degree at most $d-1$ such that $f = G(f_1, \ldots, f_c)$, for some function $G : \mathbb{F}_p^c \to \mathbb{F}_p$. Moreover, $f_1, \ldots, f_c$ are derivatives of the form $\frac{\partial f}{\partial y}$ where $y \in \mathbb{F}_p^n$.*

Proof of Theorem 8. We show by a counter-positive argument that if f is not an affine extractor for dimension k' with bias δ, then f is not an affine disperser for dimension k. Let $f : \mathbb{F}_p^n \to \mathbb{F}_p$ be a function which is not an affine extractor for dimension k' with bias δ. Then, there exists an affine subspace $u_0 + U$, with $\dim(U) = k'$ such that $\text{bias}(f|_{u_0+U}) > \delta$. Let $u_1, \ldots, u_{k'}$ be a basis for U and let $g : \mathbb{F}_p^{k'} \to \mathbb{F}_p$ be the function defined by $g(y_1, \ldots, y_{k'}) = f(u_0 + \sum_{i=1}^{k'} u_i \cdot y_i)$. Then, g is a δ-biased polynomial of degree $\leq d$. Applying Theorem 27 to g, we can write it as $G(g_1, \ldots, g_c)$, where the g_i's are of degree at most $d-1$, and $c = c(d, \delta)$ as defined in Theorem 27.

By Theorem 19, there is an affine subspace W of $\mathbb{F}_p^{k'}$ with dimension $c_1 \cdot (k'/c)^{1/(d-2)}$ for which all the g_i's are constant, for some constant $c_1 > 0$. In particular $g|_W$ is constant, which implies that there exists a subspace of $\mathbb{F}_p^n$, with the same dimension, on which the original function f is constant. Taking $k' = k^{d-2} \cdot \frac{c(d,\delta)}{c_1^{d-2}}$ completes the proof. ◀

For degree 3 and 4, we rely on stronger results from [17]. Although degree 3 was treated in [6], we present it here for completeness.

▶ **Theorem 28.** *Let $f \colon \mathbb{F}_p^n \to \mathbb{F}_p$ be an affine disperser for dimension k of degree d. If $d = 3$ then f is an affine extractor for dimension $k' = k + O(\log(1/\delta)^2)$ with bias δ. If $d = 4$ then f is an affine extractor for dimension $k' = k \cdot \text{poly}(1/\delta)$ with bias δ.*

Proof. As in the proof of Theorem 8, it is enough to show that if g is a degree 3 or 4 polynomial over $\mathbb{F}_p$ with k' variables and bias $\geq \delta$ then there exists a subspace of dimension k on which g is constant. We consider the two cases $\deg(f) = 3, 4$ separately.

Cubic ($\deg(g) = 3$)

Implicit in [17], any polynomial of degree 3 with bias $\geq \delta$, in particular g, can be represented as $\sum_{i=1}^{r} \ell_i(x) \cdot q_i(x) + q_0(x)$, where the ℓ_i's are linearly independent linear functions (with no constant term), $\deg(q_i) \leq 2$ and $r = O(\log^2(1/\delta))$. Restricting to the subspace W defined by $\{x : \ell_i(x) = 0\}$ reduces the degree of g to at most 2, and by Lemma 16, this is also true for any coset of this subspace. By averaging, there is a coset on which $\text{bias}(g|_{w+W}) \geq \delta$. By Dickson's theorem [9], there is an affine subspace $w' + W'$ of $w + W$ of co-dimension $O(\log(1/\delta))$ on which g is constant. Setting $k' = k + O(\log^2(1/\delta))$ ensures that $\dim(W')$ is at least k.

Quartic ($\deg(g) = 4$)

Theorem 4 in [17] states that any polynomial of degree 4 with bias $\geq \delta$, in particular g, can be represented as

$$\sum_{i=1}^{r} \ell_i(x) \cdot g_i(x) + \sum_{i=1}^{r} q_i(x) \cdot q_i'(x) + g_0(x),$$

where $\deg(\ell_i) \leq 1, \deg(q_i) \leq 2, \deg(q_i') \leq 2, \deg(g_i) \leq 3$ and $r = \text{poly}(1/\delta)$. By Theorem 19, there exists a subspace W of dimension $\Omega(n/r)$ on which all ℓ_i's, q_i's and q_i''s are constants. By Lemma 16, in any coset of W the degrees of ℓ_i, q_i and q_i' for $i = 1, \ldots, r$ are decreased by at least 1, hence $g|_{w+W}$ is of degree at most 3 for any coset $w + W$. Since $\text{bias}(g) \geq \delta$, by averaging there is a coset on which $\text{bias}(g|_{w+W}) \geq \delta$. Using the earlier case of biased cubic polynomials, there is an affine subspace $w' + W'$ of dimension $\Omega(n/r) - O(\log^2(1/\delta))$ on which g is constant. Setting $k' = k \cdot \text{poly}(1/\delta)$ ensures that the dimension of W' is at least k. ◀

Remark

It may be tempting to think that the polynomial loss of parameters in our reduction from affine extractors to affine dispersers, $k' = O_{\delta,d}(k^{d-2})$, is not necessary. Indeed, Theorem 28 shows that for degree 3 and 4 one can take the dimension k' of the affine extractor (for a constant error, say) to be linear in k – the dimension of the affine disperser. However, this linear dependency breaks for $d \geq 6$, as pointed up to us by Shachar Lovett. To see this, take $f : \mathbb{F}_2^n \to \mathbb{F}_2$ to be the product of two random degree 3 polynomials. It is easy to check that, with high probability, f is an affine disperser for dimension $\Theta(\sqrt{n})$, whereas $\Pr[f = 1] = 1/4 + o(1)$. Namely, f is not even an (n, n) affine extractor.

Nonetheless, a better polynomial dependency may still be possible. Perhaps $k' = O_{\delta,d}(k^{(d-2)/2})$ (which is not ruled out by similar counterexamples).

7 $\mathsf{AC}^0[\oplus]$ Circuits and Affine Extractors / Dispersers

In Section 7.1 we (easily) derive lower bounds on the dimension for which an AC^0 circuit can be affine disperser. In Section 7.2 we prove that a depth 2 $\mathsf{AC}^0[\oplus]$ circuit on n inputs cannot compute an affine disperser for dimension $n^{o(1)}$. We do so by a reduction to Theorem 1.

7.1 AC^0 Circuits Cannot Compute Affine Dispersers for Dimension $o(n/\text{polylog}(n))$

The next lemma, following Håstad's work [14], appears in [5].

▶ **Lemma 29** ([5], Corollary 3.7, restated)**.** *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function computable by a depth d and size s Boolean circuit. Then, there is a restriction ρ leaving $\frac{n}{10(10\log(s))^{d-2}} - \log(s)$ variables alive, under which $f|_\rho$ is constant.*

Lemma 29 readily implies the following corollary.

▶ **Corollary 30.** *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function computable by a Boolean circuit of depth d and size s. Then, f cannot be a bit fixing disperser (and, in particular, f cannot be an affine disperser) for min-entropy $k < \frac{n}{10(10\log(s))^{d-2}} - \log(s)$.*

7.2 Depth 2 $\mathsf{AC}^0[\oplus]$ Circuits Cannot Compute Good Affine Dispersers

As mentioned in the introduction, to prove Theorem 10, one only needs to prove Lemma 9.

Proof of Lemma 9. During the proof we will exploit the fact that if a function f on n inputs is an affine disperser for dimension k, then fixing the values of m inputs or even the values of m linear functions on the inputs, one gets an affine disperser on $n - m$ inputs for the same dimension k.

We assume that the top gate is an XOR gate. Afterwards we justify this assumption by showing that if the top gate is not an XOR gate, then the circuit C could not have computed an affine disperser with the claimed parameters to begin with.

Note that one might as well assume that there are no XOR gates at the bottom level. Indeed, assume there are t XOR gates at the bottom level, and denote by $\ell_1, \ldots, \ell_t$ the linear functions computed by these gates, respectively. Define the linear function $\ell = \ell_1 \oplus \cdots \oplus \ell_t$. Note that if ℓ is the constant 1 then by removing all the t gates from C and wiring the constant 1 as an input to the top gate, one gets an equivalent circuit with no XOR gates at the bottom layer. Assume therefore that ℓ is not the constant 1. Then, by removing all the XOR gates at the bottom layer, we get a circuit, with no XOR gates at the bottom layer, that is equivalent to the original circuit on the affine subspace $\{x : \ell(x) = 0\}$. Hence, the resulting circuit is an affine disperser on $n-1$ inputs for dimension k.

We perform a random restriction to all variables, leaving a variable alive with probability $p = \frac{1}{4\sqrt{n}}$ and otherwise setting the value of a variable uniformly and independently at random. We show that the restriction shrinks all OR, AND gates to have fan-in smaller than $2c$ with positive probability. We consider AND gates, but our arguments may be carried to OR gates similarly. The restriction shrinks every AND gate in the following way: if one of the literals which is an input to the AND gate is false under the restriction, the AND gate is eliminated. Otherwise, the AND gate shrinks to be the AND of all the remaining live variables. We wish to bound the probability that each AND gate is of fan-in greater than $2c$ after the restriction. Let m be the fan-in of the AND gate before the restriction, and m' its fan-in afterwards. We have

$$\begin{aligned}\Pr[m' \geq 2c] &= \sum_{i=2c}^{m} \binom{m}{i} \cdot p^i \cdot \left(\frac{1-p}{2}\right)^{m-i} \leq \sum_{i=2c}^{m} \binom{m}{i} \cdot p^i \cdot (1/2)^{m-i} \\ &= (1/2)^m \cdot \sum_{i=2c}^{m} \binom{m}{i} \cdot (2p)^i .\end{aligned}$$

Since $2p$ is smaller than 1, the right hand side of the above inequality is at most $(1/2)^m \cdot 2^m \cdot (2p)^{2c} = (2p)^{2c}$. Thus, $\Pr[m' \geq 2c] \leq (2p)^{2c}$. By our choice of parameter p, this is at most $1/(4n)^c$. By union bound over all $\leq n^c$ AND and OR gates, with probability at least $1 - 1/4^c \geq 3/4$ over the random restrictions, the fan-in of all AND and OR gates, under the restriction, is smaller than $2c$. Furthermore, by Chernoff bound, with probability greater than $1/2$ over the random restrictions, the number of surviving variables is at least $\sqrt{n}/5$. Therefore, there exists a restriction where the number of surviving variables is $\sqrt{n}/5$ and all AND and OR gates in the resulting circuit, under the restriction, have fan-in smaller than $2c$. Expressing the resulting circuit as a polynomial over $\mathbb{F}_2$ we get a polynomial on at least $\sqrt{n}/5$ variables with degree at most $2c$ which is an affine disperser for dimension k.

We are left to justify the assumption that the top gate must be an XOR gate. For contradiction, assume that the top gate is an OR gate. The case where the top gate is an AND gate is handled similarly. If there is an XOR gate at the bottom layer of C, we choose such gate and consider the affine subspace of co-dimension 1 on which this XOR gate outputs 1. Since the top gate is an OR gate, the circuit C is the constant 1 on an affine subspace of co-dimension 1. This stands in contradiction as k is (much) smaller than $n-1$. Thus, we obtain a depth 2 AC^0 circuit with size $s = n^c$. However, under the assumption that $k < n/10 - \log(s)$ this is a contradiction to Corollary 30. ◀

Acknowledgement. We wish to thank our advisor Ran Raz for many helpful discussions and for his encouragement. We thank Chaim Even Zohar, Elad Haramaty, Noam Lifshitz and Amir Shpilka for helpful discussions regarding this work. We thank the user who goes by the name david from Stack Exchange for pointing out [2]. We thank the anonymous referees for pointing out [28] and for many helpful comments.

References

1 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009.

2 I. Ben-Eliezer, R. Hod, and S. Lovett. Random low degree polynomials are hard to approximate. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 366–377. Springer, 2009.

3 A. Bhattacharyya, S. Kopparty, G. Schoenebeck, M. Sudan, and D. Zuckerman. Optimal testing of reed-muller codes. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 488–497. IEEE, 2010.

4 J. Bourgain. On the construction of affine extractors. *GAFA Geometric And Functional Analysis*, 17(1):33–57, 2007.

5 R. B. Boppana and M. Sipser. The complexity of finite functions. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pages 757–804. 1990.

6 E. Ben-Sasson and S. Kopparty. Affine dispersers from subspace polynomials. *SIAM Journal on Computing*, 41(4):880–914, 2012.

7 E. Ben-Sasson and N. Zewi. From affine to two-source extractors via approximate duality. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 177–186. ACM, 2011.

8 M. DeVos and A. Gabizon. Simple affine extractors using dimension expansion. In *Computational Complexity (CCC), 2010 IEEE 25th Annual Conference on*, pages 50–57. IEEE, 2010.

9 L. E. Dickson. *Linear groups with an exposition of the Galois field theory.* B.G Teubner's Sammlung von Lehrbuchern auf dem Gebiete der mathematischen Wissenschaften mit Einschluss ihrer Anwendungen. B.G. Teubner, 1901.

10 R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978.

11 Z. Dvir. Extractors for varieties. *computational complexity*, 21(4):515–572, 2012.

12 A. Gabizon and R. Raz. Deterministic extractors for affine sources over large fields. *Combinatorica*, 28(4):415–440, 2008.

13 B. Green and T. Tao. The distribution of polynomials over finite fields, with applications to the gowers norms. *Contributions to Discrete Mathematics*, 4(2), 2009.

14 J. Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, pages 6–20, 1986.

15 J. Håstad. The shrinkage exponent of de Morgan formulas is 2. *SIAM Journal on Computing*, 27(1):48–64, 1998.

16 P. Hrubeš and A. Rao. Circuits with medium fan-in. *Electronic Colloquium on Computational Complexity (ECCC)*, 20, 2014.

17 E. Haramaty and A. Shpilka. On the structure of cubic and quartic polynomials. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 331–340. ACM, 2010.

18 S. Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springerverlag Berlin Heidelberg, 2012.

19 T. Kaufman and S. Lovett. Worst case to average case reductions for polynomials. In *Foundations of Computer Science (FOCS), 2008 49th Annual IEEE Symposium on*, pages 166–175. IEEE, 2008.

20 X. Li. A new approach to affine extractors and dispersers. In *Computational Complexity (CCC), 2011 IEEE 26th Annual Conference on*, pages 137–147. IEEE, 2011.

21 A. Razborov. Lower bounds on the size of bounded depth networks over a complete basis with logical addition (Russian). *Matematicheskie Zametki*, 41(4):598–607, 1987.

22 A. Razborov. Bounded-depth formulas over $\{\wedge, \oplus\}$ and some combinatorial problems. *Complexity of Algorithms and Applied Mathematical Logic (in Russian). Ser. Voprosy Kibernetiky (Problems in Cybernetics), S. I. Adian, Ed., Moscow*, pages 149–166, 1988.
23 A. Razborov and S. Rudich. Natural proofs. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 204–213. ACM, 1994.
24 P. Savickỳ. Improved Boolean formulas for the Ramsey graphs. *Random Structures & Algorithms*, 6(4):407–415, 1995.
25 J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.
26 R. Shaltiel. Dispersers for affine sources with sub-polynomial entropy. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 247–256. IEEE, 2011.
27 R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC'87, pages 77–82, New York, NY, USA, 1987. ACM.
28 G. Tardos and D. A. M. Barrington. A lower bound on the mod 6 degree of the or function. *Computational Complexity*, 7(2):99–108, 1998.
29 L. G. Valiant. Graph-theoretic arguments in low-level complexity. In Jozef Gruska, editor, *MFCS*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 1977.
30 L. A. Vinh. The szemerédi–trotter type theorem and the sum-product estimate in finite fields. *European Journal of Combinatorics*, 32(8):1177–1181, 2011.
31 E. Viola. Guest column: correlation bounds for polynomials over {0,1}. *ACM SIGACT News*, 40(1):27–44, 2009.
32 E. Viola and A. Wigderson. Norms, XOR lemmas, and lower bounds for GF(2) polynomials and multiparty protocols. In *Computational Complexity, 2007. CCC'07. Twenty-Second Annual IEEE Conference on*, pages 141–154. IEEE, 2007.
33 A. Yehudayoff. Affine extractors over prime fields. *Combinatorica*, 31(2):245–256, 2011.
34 R. Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *EUROSAM*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.

A Depth 3 $\mathrm{AC}^0[\oplus]$ Circuits Can Compute Optimal Affine Extractors

We start this section by giving a proof for the following folklore lemma. We bother doing so because afterwards we argue that the proof implies, in fact, something stronger, which we make use of.

▶ **Lemma 31.** *There exist universal constants n_0, c such that the following holds. For every $\varepsilon > 0$ and $n > n_0$ there exists an affine extractor for dimension k with bias ε, $f : \mathbb{F}_2^n \to \mathbb{F}_2$, where $k = \log \frac{n}{\varepsilon^2} + \log\log \frac{n}{\varepsilon^2} + c$.*

The proof of Lemma 31 makes use of Hoeffding bound.

▶ **Theorem 32** (Hoeffding Bound). *Let $X_1, \ldots, X_n$ be independent random variables for which $X_i \in [a_i, b_i]$. Define $X = \frac{1}{n} \cdot \sum_{i=1}^n X_i$, and let $\mu = \mathbb{E}[X]$. Then,*

$$\Pr[|X - \mu| \geq \varepsilon] \leq 2 \cdot \exp\left(-\frac{2n^2\varepsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

Proof of Lemma 31. Let $F : \mathbb{F}_2^n \to \mathbb{F}_2$ be a random function, that is, $\{F(x)\}_{x \in \mathbb{F}_2^n}$ are independent random bits. Fix an affine subspace $u_0 + U \subseteq \mathbb{F}_2^n$ of dimension k as defined above. By Hoeffding Bound (Theorem 32),

$$\Pr\left[\frac{1}{2^k}\left|\sum_{u\in u_0+U}(-1)^{F(u)}\right|\geq\varepsilon\right]\leq 2\cdot\exp\left(-\frac{2^k\varepsilon^2}{2}\right).$$

The number of affine subspaces of dimension k is bounded by $2^n\binom{2^n}{k}\leq 2^{(k+1)n}$. Hence, by union bound over all affine subspaces, if $2^{(k+1)n}\cdot 2e^{-2^k\varepsilon^2/2}<1$ then there exists a function $f:\mathbb{F}_2^n\to\mathbb{F}_2$ that is an affine extractor for dimension k with bias ε. It is a simple calculation to show that our choice of k suffices for the above equation to hold. ◀

For the proof of Theorem 36, we introduce the following notion.

▶ **Definition 33.** An (n,k,d) *linear injector* with size m is a family of $d\times n$ matrices $\{A_1,\dots,A_m\}$ over $\mathbb{F}_2$ with the following property: for every subspace $U\subseteq\mathbb{F}_2^n$ of dimension k, there exists an $i\in[m]$ such that $\ker(A_i)\cap U=\{0\}$.

▶ **Lemma 34.** *For every n,k such that $2\leq k\leq n$, there exists an $(n,k,k+1)$ linear injector with size $m=nk$.*

Proof. Fix a subspace $U\subseteq\mathbb{F}_2^n$ of dimension k. Let A be a $d\times n$ matrix such that every entry of A is sampled from $\mathbb{F}_2$ uniformly and independently at random. For every $u\in U\setminus\{0\}$ it holds that $\Pr[Au=0]=2^{-d}$. By taking the union bound over all elements in $U\setminus\{0\}$, we get that $\Pr[\ker(A)\cap U\neq\{0\}]\leq 2^{k-d}$. Let $A_1,\dots,A_m$ be $d\times n$ matrices such that the entry of each of the matrices is sampled from $\mathbb{F}_2$ uniformly and independently at random. By the above equation, it holds that $\Pr[\forall i\in[m]\ \ker(A_i)\cap U\neq\{0\}]\leq 2^{m(k-d)}$. The number of linear subspaces of dimension k is bounded above by $\binom{2^n}{k}$, which is bounded above by 2^{nk-1} for $k\geq 2$. Thus, if $2^{nk-1}\cdot 2^{m(k-d)}<1$ there exists an (n,k,d) linear injector with size m. The latter equation holds for $d=k+1$ and $m=nk$. ◀

▶ **Lemma 35.** *Let n_0,c be the constants from Lemma 31. Let $n>n_0$ and let k,ε be such that $k=\log\frac{n}{\varepsilon^2}+\log\log\frac{n}{\varepsilon^2}+c$. Let $\{A_1,\dots,A_m\}$ be an (n,k,d) linear injector with size m. Then, there exist functions $f_1,\dots,f_m:\mathbb{F}_2^d\to\mathbb{F}_2$ such that the function $f:\mathbb{F}_2^n\to\mathbb{F}_2$ defined by*

$$f(x)=\bigoplus_{i=1}^m f_i(A_ix)\tag{5}$$

is an affine extractor for dimension k with bias ε.

Proof. Recall that in the proof of Lemma 31, we took F to be a random function. We observe however, that the proof did not use the full independence offered by a uniformly sampled random function. In fact, the proof required only that for every affine subspace $u_0+U\subseteq\mathbb{F}_2^n$ of dimension k, $\{f(u)\}_{u\in u_0+U}$ are independent random bits.

Let $F_1,\dots,F_m:\mathbb{F}_2^d\to\mathbb{F}_2$ be independent random functions, that is, the random bits $\{F_i(x)\}_{i\in[m],x\in\mathbb{F}_2^d}$ are independent. Define the random function $F:\mathbb{F}_2^n\to\mathbb{F}_2$ as follows

$$F(x)=\bigoplus_{i=1}^m F_i(A_ix).$$

We claim that for every affine subspace $u_0+U\subseteq\mathbb{F}_2^n$ of dimension k, the random bits $\{F(u)\}_{u\in u_0+U}$ are independent. By the observation above, proving this will conclude the proof. Let $u_0+U\subseteq\mathbb{F}_2^n$ be an affine subspace of dimension k. As $\{A_1,\dots,A_m\}$ is an (n,k,d)

linear injector, there exists an $i \in [m]$ such that $\ker(A_i) \cap U = \{0\}$. This implies that for every two distinct elements $u, v \in U$ it holds that $A_i(u_0 + u) \neq A_i(u_0 + v)$. Otherwise $A_i(u + v) = 0$ and thus $u + v$, a non-zero vector in U, lies in $\ker(A_i)$. This stands in contradiction to the choice of i. Recall that F_i is a random function, and from the above it follows that A_i behaves as an injection to the domain $u_0 + U$. Hence, the random bits $\{F_i(A_i u)\}_{u \in u_0 + U}$ are independent. Since $F(x)$ is defined to be the XOR of $F_i(A_i x)$ with $m - 1$ other *independent* random variables, we get that $\{F(u)\}_{u \in u_0 + U}$ are also independent random bits, as claimed. ◀

▶ **Theorem 36.** *Let f be the function from Equation* (5)*, where $\{A_1, \ldots, A_m\}$ is the (n, k, d) linear injector from Lemma 34 (that is, $m = nk$ and $d = k+1$). Then, f is an affine extractor for dimension k and bias ε, where $k = \log(n/\varepsilon^2) + \log\log(n/\varepsilon^2) + O(1)$. Moreover,*
1. $\deg(f) = \log(n/\varepsilon^2) + \log\log(n/\varepsilon^2) + O(1)$.
2. *f can be realized by an* XOR−AND−XOR *circuit of size $O((n/\varepsilon)^2 \cdot \log^3(n/\varepsilon))$.*
3. *f can be realized by a De Morgan formula of size $O((n^5/\varepsilon^2) \cdot \log^3(n/\varepsilon))$.*

Proof. To prove the first item, we note that each of the f_i's is a function on $d = k+1$ inputs, and thus can be computed by a polynomial with degree at most $k+1$. The proof then follows as in the computation of f, each f_i is composed with linear functions of the variables, and f is the XOR of the f_i's.

To prove the second item, we show an XOR−AND−XOR circuit C with the desired size, that computes the function f. Since each of the functions f_i are degree d polynomials on d inputs, each of them can be computed by an XOR − AND circuit, where the fan-in of the top XOR gate is bounded above by 2^d and the fan-in of each AND gate is at most d. Thus, for $i \in [m]$, each of the functions $f_i(A_i x)$ on n inputs is computable by an XOR − AND − XOR circuit.

By its definition, f is the XOR of these functions and so one can collapse this XOR together with the top m XOR gates. This yields an XOR − AND − XOR circuit C that computes f.

The size of the circuit C is $O(m \cdot d \cdot 2^d)$ as each of the m functions $f_i(A_i x)$ applies 2^d AND gates, each on d XOR gates (whom in turn compute the linear injector). Since $m = nk$ and $d = k + 1$, $\text{size}(C) = O((n/\varepsilon)^2 \cdot \log^3(n/\varepsilon))$ as stated.

As for the third item, we show a De Morgan formula with the desired size, that computes f. Since each of the functions f_i are on d inputs, each of them can be computed by a De Morgan formula of size $O(2^d)$. Moreover, every XOR operation needed for the computation of the linear injector $\{A_1, \ldots, A_m\}$ can be implemented in size $O(n^2)$. Replacing each leaf in the formula for f_i with the relevant formula computing the corresponding bit of $A_i x$ (or its negation), results in an $O(2^d n^2)$ size De Morgan formula computing $f_i(A_i x)$. Again, since the XOR of bits $y_1, \ldots, y_m$ can be computed by a De Morgan formula of size $O(m^2)$, and one can replace each leaf marked by y_i (or $\neg y_i$) with the formula computing $f_i(A_i x)$ (or its negation), one gets a De Morgan formula computing f of size

$$O(m^2 \cdot 2^d \cdot n^2) = O((nk)^2 \cdot 2^k \cdot n^2) = O((n^5/\varepsilon^2) \cdot \log^3(n/\varepsilon)),$$

as desired. ◀

B A Slightly Simpler Proof of the First Structural Result for $\mathbb{F}_2$

In this section we give a slightly simpler proof for Theorem 1, for the special case $q = 2$, based on ideas in [28]. We prove the following:

▶ **Theorem 37** (Structural Result I for the Binary Field). *Let k be the smallest integer such that*

$$n \le k + \sum_{j=0}^{d-1} (d-j) \cdot \binom{k}{j} .$$

Let $f \colon \mathbb{F}_2^n \to \mathbb{F}_2$ be a degree d polynomial, and let $u_0 \in \mathbb{F}_2^n$. Then, there exists a subspace $U \subset \mathbb{F}_2^n$ of dimension k such that $f|_{u_0+U}$ is constant.

Proof. Fix $u_0 \in \mathbb{F}_2^n$. We assume without loss of generality that $f(u_0) = 0$, as otherwise we can look at the polynomial $g(x) = f(x) - f(u_0)$ which is of the same degree. The proof is by induction. Let k be such that

$$n > k + \sum_{j=0}^{d-1} (d-j) \cdot \binom{k}{j} . \tag{6}$$

We assume by induction that there exists an affine subspace $u_0 + \operatorname{span}\{\Delta_1, \dots, \Delta_k\} \subseteq \mathbb{F}_2^n$, where the Δ_i's are linearly independent vectors on which f evaluates to 0. Assuming Equation 6 holds, we show there exists a vector Δ_{k+1}, linearly independent of $\Delta_1, \dots, \Delta_k$, such that $f \equiv 0$ on $u_0 + \operatorname{span}\{\Delta_1, \dots, \Delta_{k+1}\}$. To this aim, consider the set

$$A = \left\{ x \in \mathbb{F}_2^n \;\middle|\; \forall S \subseteq [k],\ f\left(x + \sum_{i \in S} \Delta_i\right) = 0 \right\} .$$

By the induction hypothesis, $u_0 \in A$. It can be verified that for any $x \in \mathbb{F}_2^n$

$$\forall S \subseteq [k] : f\left(x + \sum_{i\in S} \Delta_i\right) = 0 \quad \Leftrightarrow \quad \forall S \subseteq [k] : f_S(x) = 0 \ ,$$

where f_S is defined by

$$f_S(x) \triangleq \sum_{T \subseteq S} f\left(x + \sum_{i \in T} \Delta_i\right).$$

Namely, f_S is the derivative of f in directions $\{\Delta_i\}_{i \in S}$. In particular, $\deg(f_S) \le d - |S|$. Thus $f_S \equiv 0$ for $|S| > d$, and we may write A as

$$A = \{x \in \mathbb{F}_2^n \ \mid \ \forall S \subseteq [k] : |S| \le d,\ f_S(x) = 0\} .$$

Hence, A is the set of solutions to a system of $\binom{k}{\le d}$ polynomial equations, where there are $\binom{k}{j}$ equations which correspond to sets S of size j and thus to degree (at most) $d - j$ polynomials. [11] One can also write A as the set of solutions to the single polynomial equation

$$\prod_{S \subseteq [k] : |S| \le d} (1 - f_S(x)) \ = 1,$$

[11] In particular, equations that correspond to sets S of size d are of the form $c_S = 0$ for some constant $c_S \in \mathbb{F}_2$. Since A is non-empty, the constants c_S must be 0, making those equations tautologies $0 = 0$ that does not depend on x. Moreover, most of the remaining equations correspond to sets S of size $d - 1$, and are therefore either linear equations or tautologies.

which is of degree $D \le \sum_{j=0}^{d-1} (d-j) \cdot \binom{k}{j}$. Since A is non-empty, by DeMillo-Lipton-Schwartz-Zippel lemma (Lemma 13, for $q = 2$) we have that

$$|A| \ge 2^{n-D} \ge 2^{n-\sum_{j=0}^{d-1}(d-j)\cdot\binom{k}{j}}. \tag{7}$$

This, together with Equation (6) implies that $|A| > 2^k$. Hence, there exists a point $y \in A$ such that $y - u_0 \notin \mathrm{span}\{\Delta_1, \Delta_2, \dots, \Delta_k\}$. Pick such a point u arbitrarily and denote by $\Delta_{k+1} \triangleq u - u_0$. Since both u_0 and u are in A we have that $f \equiv 0$ on $u_0 + \mathrm{span}\{\Delta_1, \dots, \Delta_{k+1}\}$. The inductive proof shows that there exists a subspace U of dimension k such that f is constant on $u_0 + U$ and

$$n \le k + \sum_{j=0}^{d-1} (d-j) \cdot \binom{k}{j} , \tag{8}$$

since otherwise we could have continue this process and pick a bigger subspace U'. ◀

B.1 Proof of Theorem 22

The proof of Theorem 22 uses the following lemma.

▶ **Lemma 38.** *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a degree d polynomial, and let U be a linear subspace with basis $\Delta_1, \dots, \Delta_k$. Then, $\deg(f|_U) \le d-1$ if and only if $f_S(0) = 0$ for all $S \subseteq [k]$ of size d, where $f_S(x) := \sum_{T \subseteq S} f\left(x + \sum_{i \in T} \Delta_i\right)$.*

Proof of Lemma 38. As noted in the Preliminaries section, the degree of $f|_U$ is equal to the degree of $g : \mathbb{F}_2^k \to \mathbb{F}_2$ defined as $g(y_1, \dots, y_k) = f(\sum_{i=1}^k y_i \Delta_i)$. Since $\deg(g) \le d$, we may write $g(y) = \sum_{S \subseteq [k], |S| \le d} a_S \cdot \prod_{i \in S} y_i$, where $a_S \in \mathbb{F}_2$ are constants. By Möbius inversion formula (Fact 14), $a_S = \sum_{T \subseteq S} g(\mathbf{1}_T)$. By the definition of g, we establish the relation $a_S = \sum_{T \subseteq S} f(\sum_{i \in T} \Delta_i) = f_S(0)$. Hence,

$$\begin{aligned} \deg(f|_U) \le d-1 \quad &\iff \quad \deg(g) \le d-1 \\ &\iff \quad \forall S \subseteq [k] \quad \text{s.t. } |S| = d, a_S = 0 \\ &\iff \quad \forall S \subseteq [k] \quad \text{s.t. } |S| = d, f_S(0) = 0, \end{aligned}$$

which completes the proof. ◀

Proof of Theorem 22. Similarly to the proof of Theorem 37, we find by induction basis vectors $\Delta_1, \dots, \Delta_k$ for the subspace U. We assume by induction that $\deg(f|_U) \le d-1$, and we wish to find a new vector Δ_{k+1}, linearly independent of $\Delta_1, \dots, \Delta_k$, for which $\deg(f|_{U'}) \le d-1$, where $U' = \mathrm{span}\{\Delta_1, \dots, \Delta_{k+1}\}$. We continue doing so as long as $\binom{k}{d-1} + k < n$.[12]

By Lemma 38, for any set $S \subseteq [k]$ of size d, $f_S(0) = 0$. We wish to find a new vector Δ_{k+1} such that for all $S \subseteq [k+1]$ of size d, $f_S(0) = 0$. It suffices to consider sets S of size d that contains $k+1$, since the correctness for all other sets is implied by the induction hypothesis.

For sets S of size $d-1$, $f_S(x)$ is an affine function and can be written as $f_S(x) = \langle \ell_S, x \rangle + c_S$, where $\ell_S \in \mathbb{F}_2^n$ and $c_S \in \mathbb{F}_2$. Let W be the linear subspace of $\mathbb{F}_2^n$ spanned by $\{\ell_S : S \subseteq [k], |S| = d-1\}$. Let Δ_{k+1} be any vector orthogonal to W, and linearly independent of $\Delta_1, \Delta_2, \dots, \Delta_k$. Since, $\dim(W^\perp) = n - \dim(W) \ge n - \binom{k}{d-1}$, which by our

[12] Note that this is slightly better than the expression we had in Theorem 37.

assumption is strictly bigger than k, such a vector Δ_{k+1} exists. Let $S \subseteq [k+1]$ be a set of size d that contains $k+1$ and let $S' = S \cap [k]$, then

$$f_S(0) = f_{S'}(0) + f_{S'}(\Delta_{k+1}) = \langle \ell_{S'}, 0\rangle + c_{S'} + \langle \ell_{S'}, \Delta_{k+1}\rangle + c_{S'} = 0 \ ,$$

where in the first equality we used the definitions of f_S and $f_{S'}$, and in the last equality we used the fact that Δ_{k+1} is orthogonal to $\ell_{S'}$. Using Lemma 38 we have shown that our choice of Δ_{k+1} gives a linear subspace $U' = \mathrm{span}\{\Delta_1, \dots, \Delta_{k+1}\}$ for which $f|_{U'}$ is of degree $\leq d-1$.

We now explain how to find, for any set S of size $d-1$, the affine function $f_S(x)$ (that is, ℓ_S and c_S) by performing $2^{d-1} \cdot (n+1)$ queries to f. As f_S is affine, knowing the values of f_S on the inputs $0, e_1, e_2, \dots, e_n$ determines ℓ_S and c_S: $c_S = f_S(0)$ and $(\ell_S)_i = c_S + f_S(e_i)$ for $i \in [n]$. Each one of the values $f_S(0), f_S(e_1), \dots, f_S(e_n)$ can be computed using 2^{d-1} queries to f, by the definition of f_S.

We now describe how can one efficiently find the vector Δ_{k+1} given $\Delta_1, \dots, \Delta_k$. Using Gaussian elimination we find a basis for $W^{\perp}$. We check for each basis vector if it is not in the span of $\Delta_1, \dots, \Delta_k$; after checking $k+1$ vectors we are promised to find such a vector. Next, we analyze the dimension of the subspace returned by the algorithm, the number of queries it makes to f, and the total running time.

Dimension of subspace

We abuse notation and denote by k the number of rounds in our algorithm, which is also the dimension of the subspace the algorithm returns. Since the algorithm stopped, we know that $\binom{k}{d-1} + k \geq n$. By a simple calculation, under the assumption that $d \leq \log(n)/3$ we get that $k = \Theta(d \cdot n^{1/(d-1)})$.

Number of queries

Overall through the k rounds of the algorithm we query f on all vectors of the form $v + \sum_{i \in T} \Delta_i$ for $v \in \{0, e_1, \dots, e_n\}$ and $T \subseteq [k]$ of size $\leq d-1$. Hence, if we make sure not to query f more than once on the same point, the number of queries is $(n+1) \cdot \binom{k}{\leq d-1}$ which is at most $O(n^2)$ for $d \leq \log(n)/3$.

Running time

The total running time per round is $O(n^3)$ since we perform Gaussian elimination to calculate the basis for $W^{\perp}$, and another Gaussian elimination to check which of the first $k+1$ vectors of this basis is not in $\mathrm{span}\{\Delta_1, \dots, \Delta_{k+1}\}$. In addition, in each round we calculate the linear functions ℓ_S, but this only takes $O(n^2 \cdot 2^d)$ time, which is negligible compared to $O(n^3)$ under the assumption that $d \leq \log(n)/3$. Therefore, the total running time is $O(n^3 \cdot k)$. ◀

C Proof of DeMillo-Lipton-Schwartz-Zippel Variant

In this section we provide a proof for Lemma 13. Our proof is adapted from the proof of Lemma A.36 in the book of Arora and Barak [1].

Proof of Lemma 13. Since we only care about the values the polynomial take on $\mathbb{F}_q^n$, we may assume without loss of generality that the individual degree of each variable is at most $q-1$, since $a^q = a$ for all $a \in \mathbb{F}_q$.

We use induction on n. If $n = 1$ then f is a univariate polynomial of degree d for some $d \le q-1$, since we assumed each individual degree is at most $q-1$. We have $\Pr[f(x_1) \neq 0] \ge 1 - d/q \ge q^{-d/(q-1)}$, where the first inequality follows since a univariate degree d polynomial over a field obtains at most d roots, and the last inequality can be verified for any $d \le q-1$ using basic calculus. Suppose the statement is true when the number of variables is at most $n-1$. Then f can be written as

$$f(x_1, \dots, x_n) = \sum_{i=0}^{\min(d,q-1)} x_1^i \cdot f_i(x_2, \dots, x_n)$$

where f_i is of total degree at most $d-i$. Let k be the largest i such that f_i is a non-zero polynomial. By conditioning we have,

$$\Pr[f(x_1, \dots, x_n) \neq 0] \ge \Pr[f_k(x_2, \dots, x_n) \neq 0] \cdot \Pr[f(x_1, \dots, x_n) \neq 0 \mid f_k(x_2, \dots, x_n) \neq 0] \ .$$

By the induction hypothesis, the first multiplicand is at least $q^{-(d-k)/(q-1)}$. As for the second multiplicand, for any fixed $(x_2, \dots, x_n) = (a_2, \dots, a_n)$ such that $f_k(a_2, \dots, a_n) \neq 0$, we get that $f(x_1, a_2, \dots, a_n)$ is a non-zero univariate polynomial, in the variable x_1, of degree k. Hence, $\Pr_{x_1 \sim \mathbb{F}_q}[f(x_1, a_2, \dots, a_n) \neq 0] \ge q^{-k/(q-1)}$ from the base case. Overall we get

$$\Pr[f(x_1, \dots, x_n) \neq 0] \ge q^{-(d-k)/(q-1)} q^{-k/(q-1)} = q^{-d/(q-1)} \ . \qquad ◀$$

The Minimum Bisection in the Planted Bisection Model*

Amin Coja-Oghlan[1], Oliver Cooley[2], Mihyun Kang[2], and Kathrin Skubch[1]

1 **Goethe University, Mathematics Institute**
10 Robert Mayer St, Frankfurt 60325, Germany
`{acoghlan,skubch}@math.uni-frankfurt.de`

2 **Graz University of Technology, Institute of Optimization and Discrete Mathematics (Math B)**
Steyrergasse 30, 8010 Graz, Austria
`{cooley,kang}@math.tugraz.at`

Abstract

In the *planted bisection* model a random graph $\boldsymbol{G}(n, p_+, p_-)$ with n vertices is created by partitioning the vertices randomly into two classes of equal size (up to ± 1). Any two vertices that belong to the same class are linked by an edge with probability p_+ and any two that belong to different classes with probability $p_- < p_+$ independently. The planted bisection model has been used extensively to benchmark graph partitioning algorithms. If $p_\pm = 2d_\pm/n$ for numbers $0 \le d_- < d_+$ that remain fixed as $n \to \infty$, then w.h.p. the "planted" bisection (the one used to construct the graph) will not be a minimum bisection. In this paper we derive an asymptotic formula for the minimum bisection width under the assumption that $d_+ - d_- > c\sqrt{d_+ \ln d_+}$ for a certain constant $c > 0$.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases Random graphs, minimum bisection, planted bisection, belief propagation

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.710

1 Introduction

1.1 Background and motivation

Since the early days of computational complexity graph partitioning problems have played a central role in computer science [18, 24]. Over the years they have inspired some of the most important algorithmic techniques that we have at our disposal today, such as network flows or semidefinite programming [3, 17, 19, 25, 38].

In the context of the probabilistic analysis of algorithms, it is hard to think of a more intensely studied problem than the *planted bisection model.* In this model a random graph $\boldsymbol{G} = \boldsymbol{G}(n, p_{+1}, p_{-1})$ on $[n] = \{1, \ldots, n\}$ is created by choosing a map $\boldsymbol{\sigma} : V \to \{-1, 1\}$ uniformly at random subject to $||\boldsymbol{\sigma}^{-1}(1)| - |\boldsymbol{\sigma}^{-1}(-1)|| \le 1$ and connecting any two vertices

* The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 278857–PTCC. Supported by Austrian Science Fund (FWF): P26826 and W1230, Doctoral Program "Discrete Mathematics".

$v \neq w$ with probability $p_{\boldsymbol{\sigma}(v)\boldsymbol{\sigma}(w)}$ independently, where $0 \leq p_{-1} < p_{+1} \leq 1$. To ease notation, we often write p_+ for p_{+1} and p_- for p_{-1}, and handle subscripts similarly for other parameters.

Given the random graph $\boldsymbol{G}$ (but not the planted bisection $\boldsymbol{\sigma}$), the task is to find a *minimum bisection* of $\boldsymbol{G}$, i.e., to partition the vertices into two disjoint sets $S, \bar{S} = [n] \setminus S$ whose sizes satisfy $||S| - |\bar{S}|| \leq 1$ such that the number of S-$\bar{S}$-edges is minimum. The planted bisection model has been employed to gauge algorithms based on spectral, semidefinite programming, flow and local search techniques, to name but a few [5, 6, 7, 8, 9, 11, 14, 15, 16, 22, 23, 27, 31, 29].

Remarkably, for a long time the algorithm with the widest range of $n, p_\pm$ for which a minimum bisection can be found efficiently was one of the earliest ones, namely Boppana's spectral algorithm [6]. It succeeds if

$$n(p_+ - p_-) \geq c\sqrt{np_+ \ln n}$$

for a certain constant $c > 0$. Under this assumption the planted bisection is minimum w.h.p. In fact, recently the critical value $c^* > 0$ for which this statement is true was identified explicitly [36]. In particular, for $n(p_+ - p_-) > c^*\sqrt{np_+ \ln n}$ the minimum bisection width simply equals $(\frac{1}{4} + o(1))n^2 p_-$ w.h.p.

But if $n(p_+ - p_-) < c^*\sqrt{np_+ \ln n}$, then the minimum bisection width will be strictly smaller than the width of the planted bisection w.h.p. Yet there is another spectral algorithm [9] that finds a minimum bisection w.h.p. under the weaker assumption that

$$n(p_+ - p_-) \geq c\sqrt{np_+ \ln(np_+)}, \tag{1.1}$$

for a certain constant $c > 0$, and even certifies the optimality of its solution. However, [9] does not answer what is arguably the most immediate question: what is the *typical* value of the minimum bisection width?

In this paper we derive the value to which the (suitably scaled) minimum bisection width converges in probability. We confine ourselves to the case that $\frac{n}{2}p_\pm = d_\pm$ remain fixed as $n \to \infty$. Hence, the random graph $\boldsymbol{G}$ has bounded average degree. This is arguably the most interesting case because the discrepancy between the planted and the minimum bisection gets larger as the graphs get sparser. In fact, it is easy to see that in the case of fixed $\frac{n}{2}p_\pm = d_\pm$ the difference between the planted and the minimum bisection width is $\Theta(n)$ as the planted bisection is not even locally optimal w.h.p.

Although we build upon some of the insights from [9], it seems difficult to prove our main result by tracing the fairly complicated algorithm from that paper. Instead, our main tool is an elegant message passing algorithm called *Warning Propagation* that plays an important role in the study of random constraint satisfaction problems via ideas from statistical physics [32]. Running Warning Propagation on $\boldsymbol{G}$ naturally corresponds to a fixed point problem on the 2-simplex, and the minimum bisection width can be cast as a function of the fixed point.

1.2 The main result

To state the fixed point problem, we consider the functions

$$\psi : \mathbb{R} \to \mathbb{R}, \quad x \mapsto \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \leq x \leq 1 \\ 1 & \text{if } x > 1, \end{cases} \qquad \tilde{\psi} : \mathbb{R} \to \mathbb{R}, \quad x \mapsto \begin{cases} -1 & \text{if } x \leq -1 \\ 1 & \text{if } x > -1. \end{cases}$$

Let $\mathcal{P}(\{-1,0,1\})$ be the set of probability measures on $\{-1,0,1\}$. Clearly, we can identify $\mathcal{P}(\{-1,0,1\})$ with the set of all maps $p : \{-1,0,1\} \to [0,1]$ such that $p(-1)+p(0)+p(1) = 1$, i.e., the 2-simplex. Further, let us define a map

$$\mathcal{T}_{d_+,d_-} : \mathcal{P}(\{-1,0,1\}) \to \mathcal{P}(\{-1,0,1\}) \tag{1.2}$$

as follows. Given $p \in \mathcal{P}(\{-1,0,1\})$, let $(\eta_{p,i})_{i\geq 1}$ be a family of i.i.d. $\{-1,0,1\}$-valued random variables with distribution p. Moreover, let $\gamma_\pm = \mathrm{Po}(d_\pm)$ be Poisson variables that are independent of each other and of the $\eta_{p,i}$. Let

$$Z_{p,d_+,d_-} := \sum_{i=1}^{\gamma_+} \eta_{p,i} - \sum_{i=\gamma_++1}^{\gamma_++\gamma_-} \eta_{p,i}. \tag{1.3}$$

Then we let $\mathcal{T}_{d_+,d_-}(p) \in \mathcal{P}(\{-1,0,1\})$ be the distribution of $\psi(Z_{p,d_+,d_-})$. Further, with $(\eta_{p,i})_{i\geq 1}$ and $\gamma_\pm$ as before, let

$$\begin{aligned}\varphi_{d_+,d_-} :&\mathcal{P}(\{-1,0,1\}) \to \mathbb{R},\\ &p \mapsto \frac{1}{2}\mathbb{E}\left[\sum_{i=1}^{\gamma_+} \mathbf{1}\{\eta_{p,i} = -\tilde{\psi}(Z_{p,d_+,d_-})\} + \sum_{i=\gamma_++1}^{\gamma_++\gamma_-} \mathbf{1}\{\eta_{p,i} = \tilde{\psi}(Z_{p,d_+,d_-})\}\right].\end{aligned}$$

Moreover, let us call $p \in \mathcal{P}(\{-1,0,1\})$ *skewed* if $p(1) \geq 1 - d_+^{-10}$. Finally, we denote the minimum bisection width of a graph G by $\mathrm{bis}(G)$.

▶ **Theorem 1.1.** *There exists a constant $c > 0$ such that for any $d_\pm > 0$ satisfying $d_+ \geq 2$ and $d_+ - d_- \geq c\sqrt{d_+ \ln d_+}$ the map $\mathcal{T}_{d_+,d_-}$ has a unique skewed fixed point p^* and $n^{-1}\mathrm{bis}(\boldsymbol{G})$ converges in probability to $\varphi_{d_+,d_-}(p^*)$.*

Note that $\mathcal{T}_{d_+,d_-}$ may have further fixed points besides p^*, but p^* is the only fixed point which is skewed. We also note that the condition $d_+ \geq 2$ is not optimised – any constant larger than 1 would do as a lower bound, but then in any case the condition $d_+ \geq 2$ follows from the lower bound on $d_+ - d_-$ for sufficiently large c.

In the following sections we will use that the assumptions of Theorem 1.1 allow us to assume that also d_+ is sufficiently large.

1.3 Further related work

Determining the minimum bisection width of a graph is NP-hard [18] and there is evidence that the problem does not even admit a PTAS [26]. On the positive side, it is possible to approximate the minimum bisection width within a factor of $O(\ln n)$ for graphs on n vertices in polynomial time [38].

The planted bisection model has been studied in statistics under the name "stochastic block model" [20]. However, in the context of statistical inference the aim is to recover the planted partition $\boldsymbol{\sigma}$ as best as possible given $\boldsymbol{G}$ rather than to determine the minimum bisection width. Recently there has been a lot of progress, much of it inspired by non-rigorous work [12], on the statistical inference problem. The current status of the problem is that matching upper and lower bounds are known for the values of $d_\pm$ for which it is possible to obtain a partition that is non-trivially correlated with $\boldsymbol{\sigma}$ [30, 33, 35]. Furthermore, there are algorithms that recover a best possible approximation to $\boldsymbol{\sigma}$ under certain conditions on $d_\pm$ [1, 34, 36]. But since our objective is different, the methods employed in the present paper are somewhat different and, indeed, rather simpler.

Finally, there has been recent progress on determining the minimum bisection width on the Erdős-Rényi random graph. Although its precise asymptotics remain unknown in the case of bounded average degrees d, it was proved in [13] that the main correction term corresponds to the "Parisi formula" in the Sherrington-Kirkpartrick model [39]. Additionally, regarding the case of very sparse random graphs (i.e. with constant average degree), there is a sharp threshold (at $np = \ln 4$) for the minimum bisection width to be linear in n [28].

Generally speaking, the approach that we pursue is somewhat related to the notion of "local weak convergence" of graph sequences as it was used in [2]. More specifically, we are going to argue that the minimum bisection width of $\boldsymbol{G}$ is governed by the "limiting local structure" of the graph, which is a two-type Galton-Watson tree. The fixed point problem in Theorem 1.1 mirrors the execution of a message passing algorithm on the Galton-Watson tree. The study of this fixed point problem, for which we use the *contraction method* [37], is the key technical ingredient of our proof. We believe that this strategy provides an elegant framework for tackling many other problems in the theory of random graphs as well. In fact, in a recent paper [10] we combined Warning Propagation with a fixed point analysis on Galton-Watson trees to the k-core problem, and in [4] Warning Propagation was applied to the random graph coloring problem.

2 Outline

From here on we keep the notation and the assumptions of Theorem 1.1. In particular, we assume that $d_+ - d_- \geq c\sqrt{d_+ \ln d_+}$ for a large enough constant $c > 0$ and that $d_\pm$ remain fixed as $n \to \infty$. Furthermore we assume that d_+ is bounded from below by a large enough constant. Throughout the paper all graphs will be locally finite and of countable size.

Three main insights enable the proof of Theorem 1.1. The first one, which we borrow from [9], is that w.h.p. $\boldsymbol{G}$ features a fairly large set $\mathcal{C}$ of vertices such that for any two optimal bisections τ_1, τ_2 of $\boldsymbol{G}$ (i.e. maps $\tau_1, \tau_2 : V(\boldsymbol{G}) \to \{\pm 1\}$), we either have $\tau_1(v) = \tau_2(v)$ for all $v \in \mathcal{C}$ or $\tau_1(v) = -\tau_2(v)$ for all $v \in \mathcal{C}$. In the language of random constraint satisfaction problems, the vertices in $\mathcal{C}$ are "frozen". While there remain $\Omega(n)$ unfrozen vertices, the subgraph that they induce is subcritical, i.e., all components are of size $O(\ln n)$ and indeed most are of bounded size.

The second main ingredient is an efficient message passing algorithm called *Warning Propagation*, (cf. [32, Chapter 19]). We will show that a bounded number of Warning Propagation iterations suffice to arrange almost all of the unfrozen vertices optimally (i.e. to assign almost all of the vertices to two classes such that there is a minimum bisection respecting this assignment) and thus to obtain a very good approximation to the minimum bisection w.h.p. (Proposition 2.2). This insight reduces our task to tracing Warning Propagation for a bounded number of rounds.

This last problem can be solved by studying Warning Propagation on a suitable Galton-Watson tree, because $\boldsymbol{G}$ only contains a negligible number of short cycles w.h.p. (Lemma 2.3). Thus, the analysis of Warning Propagation on the random tree is the third main ingredient of the proof. This task will turn out to be equivalent to studying the fixed point problem from Section 1.2 (Proposition 2.5). We proceed to outline the three main components of the proof.

2.1 The core

Given a vertex u of a graph G let $\partial_G u$ denote the neighbourhood of u in G. We sometimes omit the subscript G when the graph is clear from the context. More particularly, in the random graph $\boldsymbol{G}$, let $\partial_\pm u$ denote the set of all neighbours w of u in $\boldsymbol{G}$ with $\boldsymbol{\sigma}(w)\boldsymbol{\sigma}(v) = \pm 1$. Following [9], we define $\mathcal{C}$ as the largest subset $U \subset [n]$ such that

$$||\partial_\pm u| - d_\pm| \le \frac{c}{4}\sqrt{d_+ \ln d_+} \quad \text{and} \quad |\partial u \setminus U| \le 100 \quad \text{for all } u \in U. \tag{2.1}$$

Clearly, the set $\mathcal{C}$, which we call the *core*, is uniquely defined because any union of sets U that satisfy (2.1) also has the property. Let $\boldsymbol{\sigma}_\mathcal{C} : \mathcal{C} \to \{\pm 1\}$, $v \mapsto \boldsymbol{\sigma}(v)$ be the restriction of the "planted assignment" to $\mathcal{C}$.

Furthermore, for a graph G, a set $U \subset V(G)$ and a map $\sigma : U \to \{-1, 1\}$ we let

$$\operatorname{cut}(G,\sigma) := \min\left\{ \sum_{\{v,w\}\in E(G)} \frac{1-\tau(v)\tau(w)}{2} \;\middle|\; \tau : V(G) \to \{\pm 1\} \text{ satisfies } \tau(v) = \sigma(v) \text{ for all } v \in U \right\}.$$

In words, $\operatorname{cut}(G,\sigma)$ is the smallest number of edges in a cut of G that separates the vertices in $U \cap \sigma^{-1}(-1)$ from those in $U \cap \sigma^{-1}(1)$. In particular, $\operatorname{cut}(G,\sigma_\mathcal{C})$ is the smallest cut of G that separates the vertices in the core $\mathcal{C}$ that are frozen to -1 from those that are frozen to 1.

Finally, for any vertex v we define a set $\mathcal{C}_v = \mathcal{C}_v(G,\sigma)$ of vertices via the following process.

C1 Let $\mathcal{C}_v^{(0)} = \{v\} \cup \partial_G v$.

C2 Inductively, let $\mathcal{C}_v^{(t+1)} = \mathcal{C}_v^{(t)} \cup \bigcup_{u \in \mathcal{C}_v^{(t)} \setminus \mathcal{C}} \partial_G u$ and let $\mathcal{C}_v = \bigcup_{t \ge 0} \mathcal{C}_v^{(t)}$.

▶ **Lemma 2.1** ([9], Proposition 19 and Section 3.6). *We have* $\operatorname{bis}(\boldsymbol{G}) = \operatorname{cut}(\boldsymbol{G}, \boldsymbol{\sigma}_\mathcal{C})$ *and* $|\mathcal{C}| \ge n(1 - d_+^{-100})$ *w.h.p. Furthermore, for any* $\varepsilon > 0$ *there exists* $\omega > 0$ *such that w.h.p.* $\sum_{v\in[n]} |\mathcal{C}_v| \cdot \mathbf{1}\{|\mathcal{C}_v| \ge \omega\} \le \varepsilon n$.

2.2 Warning Propagation

To calculate $\operatorname{cut}(\boldsymbol{G}, \boldsymbol{\sigma}_\mathcal{C})$ we adopt the *Warning Propagation* ("WP") message passing algorithm[1]. Let us first introduce WP for a generic graph $G = (V(G), E(G))$ and a map $\sigma : U \subset V(G) \to \{-1, 1\}$. At each time $t \ge 0$, WP sends a "message" $\mu_{v\to w}(t|G,\sigma) \in \{-1, 0, 1\}$ from v to w for any edge $\{v,w\} \in E(G)$. The messages are directed objects, i.e., $\mu_{v\to w}(t|G,\sigma)$ and $\mu_{w\to v}(t|G,\sigma)$ may differ. They are defined inductively by

$$\mu_{v\to w}(0|G,\sigma) := \begin{cases} \sigma(v) & \text{if } v \in U, \\ 0 & \text{otherwise,} \end{cases} \qquad \mu_{v\to w}(t+1|G,\sigma) := \psi\left(\sum_{u\in\partial v\setminus w} \mu_{u\to v}(t|G,\sigma)\right). \tag{2.2}$$

Note that U does not appear explicitly in the notation $\mu_{v\to w}(t|G,\sigma)$ despite being integral to the definition – it is however implicit in the notation since U is the domain of σ.

Thus, the WP messages are initialised according to σ. Subsequently, v sends message ± 1 to w if it receives more ± 1 than ∓ 1 messages from its neighbours $u \ne w$. If there is a tie, v

[1] A discussion of Warning Propagation in the context of the "cavity method" from statistical physics can be found in [32].

sends out 0. Finally, for $t \geq 0$ define

$$\mu_v(t|G,\sigma) := \sum_{w \in \partial v} \mu_{w \to v}(t|G,\sigma).$$

The intuition is that the message $\mu_{v \to w}$ which v sends to w indicates which class v is most likely to be in based on the current local information it receives from its other neighbours. To minimise the cut, we would like to place v into the class in which most of its neighbours lie. The initialisation is given by the set U, which we will choose to be the core.

▶ **Proposition 2.2.** *For any $\varepsilon > 0$ there exists $t_0 = t_0(\varepsilon, d_+, d_-)$ such that for all $t \geq t_0$ w.h.p.*

$$\left| \mathrm{cut}(\boldsymbol{G}, \boldsymbol{\sigma}_{\mathcal{C}}) - \frac{1}{2} \sum_{v \in [n]} \sum_{w \in \partial v} \mathbf{1}\{\mu_{w \to v}(t|\boldsymbol{G},\boldsymbol{\sigma}) = -\tilde{\psi}\left(\mu_v(t|\boldsymbol{G},\boldsymbol{\sigma})\right)\} \right| \leq \varepsilon n.$$

We defer the proof of Proposition 2.2 to Section 3.

2.3 The local structure

Proposition 2.2 shows that w.h.p. in order to approximate $\mathrm{cut}(\boldsymbol{G}, \boldsymbol{\sigma}_{\mathcal{C}})$ up to a small error of εn we merely need to run WP for a number t_0 of rounds that is bounded in terms of ε. The upshot is that the WP messages $\mu_{w \to v}(t|\boldsymbol{G},\boldsymbol{\sigma})$ that are required to figure out the minimum bisection width are determined by the *local* structure of $\boldsymbol{G}$. We show that the local structure of $\boldsymbol{G}$ "converges to" a suitable Galton-Watson tree. For this purpose, for simplicity we always say that the number of potential neighbours of any vertex in each class is $n/2$. This ignores the fact that if n is odd the classes do not have quite this size and the fact that a vertex cannot be adjacent to itself. However, ignoring these difficulties will not affect our calculations in any significant way.

Our task boils down to studying WP on that Galton-Watson tree. Specifically, let $\boldsymbol{T} = \boldsymbol{T}_{d_+,d_-}$ be the Galton-Watson tree with two types $+1, -1$ and offspring matrix

$$\begin{pmatrix} \mathrm{Po}(d_+) & \mathrm{Po}(d_-) \\ \mathrm{Po}(d_-) & \mathrm{Po}(d_+) \end{pmatrix}. \tag{2.3}$$

Hence, a vertex of type ± 1 spawns $\mathrm{Po}(d_+)$ vertices of type ± 1 and independently $\mathrm{Po}(d_-)$ vertices of type ∓ 1. Moreover, the type of the root vertex $r_{\boldsymbol{T}}$ is chosen uniformly at random. Let $\boldsymbol{\tau} = \boldsymbol{\tau}_{d_+,d_-} : V(\boldsymbol{T}) \to \{\pm 1\}$ assign each vertex of $\boldsymbol{T}$ its type.

The random graph $(\boldsymbol{G}, \boldsymbol{\sigma})$ "converges to" $(\boldsymbol{T}, \boldsymbol{\tau})$ in the following sense. For two triples (G, r, σ), (G', r', σ') of graphs G, G', root vertices $r \in V(G)$, $r' \in V(G')$ and maps $\sigma : V(G) \to \{\pm 1\}$, $\sigma' : V(G') \to \{\pm 1\}$ we write $(G, \sigma) \cong (G', \sigma')$ if there is a graph isomorphism $\varphi : G \to G'$ such that $\varphi(r) = r'$ and $\sigma = \sigma' \circ \varphi$. Further, we denote by $\partial^t(G, r, \sigma)$ the rooted graph obtained from (G, r) by deleting all vertices at distance greater than t from r together with the restriction of σ to this subgraph. The following lemma characterises the local structure of $(\boldsymbol{G}, \boldsymbol{\sigma})$.

▶ **Lemma 2.3.** *Let $t > 0$ be an integer and let T be any tree with root r and map $\tau : V(T) \to \{\pm 1\}$. Then*

$$\frac{1}{n} \sum_{v \in [n]} \mathbf{1}\left\{\partial^t(\boldsymbol{G}, v, \boldsymbol{\sigma}) \cong \partial^t(T, r, \tau)\right\} \quad \overset{n \to \infty}{\to} \quad \mathbb{P}\left[\partial^t(\boldsymbol{T}, r_{\boldsymbol{T}}, \boldsymbol{\tau}) \cong \partial^t(T, r, \tau)\right] \quad \textit{in probability.}$$

Furthermore, w.h.p. $\boldsymbol{G}$ does not contain more than $\ln n$ vertices v such that $\partial^t(\boldsymbol{G}, v, \boldsymbol{\sigma})$ contains a cycle.

Proof. Given a tree T with root r and map $\tau : V(T) \to \{\pm 1\}$, let

$$X_t = X_t(T, r, \tau) = \frac{1}{n} \sum_{v \in [n]} \mathbf{1}\left\{\partial^t(\boldsymbol{G}, v, \boldsymbol{\sigma}) \cong \partial^t(T, r, \tau)\right\}$$

and

$$p_t = p_t(T, r, \tau) = \mathbb{P}\left[\partial^t(\boldsymbol{T}, r_{\boldsymbol{T}}, \boldsymbol{\tau}) \cong \partial^t(T, r, \tau)\right].$$

The proof proceeds by induction on t. If $t = 0$, pick a vertex $\boldsymbol{v} \in [n]$ uniformly at random, then $X_0 = \mathbb{P}_{\boldsymbol{v}}\left(\boldsymbol{\sigma}(\boldsymbol{v}) = \tau(r)\right) = \frac{1}{2}$ and $p_0 = \mathbb{P}_{\boldsymbol{T}}\left(\boldsymbol{\tau}(r_{\boldsymbol{T}}) = \tau(r)\right) = \frac{1}{2}$ for any $\tau(r) \in \{\pm 1\}$. To proceed from t to $t+1$, let d denote the number of children $v_1, \ldots, v_d$ of r in T. For each $i = 1, \ldots, d$, let T_i denote the tree rooted at v_i in the forest obtained from T by removing r and let $\tau_i : V(T_i) \to \{\pm 1\}$ denote the restriction of τ to the vertex set of T_i. Finally, let $C_1, \ldots, C_{\tilde{d}}$ for some $\tilde{d} \leq d$ denote the distinct isomorphism classes among $\{\partial^t(T_i, v_i, \tau_i) : i = 1, \ldots, d\}$, and let $c_j = |\{i : \partial^t(T_i, v_i, \tau_i) \in C_j\}|$. Let $v \in [n]$ be an arbitrary vertex in $\boldsymbol{G}$. Our aim is to determine the probability of the event $\{\partial^{t+1}(\boldsymbol{G}, v, \boldsymbol{\sigma}) \cong \partial^{t+1}(T, r, \tau)\}$. Therefore, we think of $\boldsymbol{G}$ as being created in three rounds. First, partition $[n]$ in two classes. Second, randomly insert edges between vertices in $[n] \setminus \{v\}$ according to their planted sign. Finally, reveal the neighbours of v. For the above event to happen, v must have d neighbours in $\boldsymbol{G}$. Since $|\partial_\pm v|$ are independent binomially distributed random variables with parameters $\frac{n}{2}$ and $p_\pm$ and because $\frac{n}{2} p_\pm = d_\pm$, we may approximate $|\partial_\pm v|$ with a poisson distribution, and v has degree d with probability

$$\frac{(d_+ + d_-)^d}{d! \exp(d_+ + d_-)} + \mathrm{o}(1).$$

Conditioned on v having degree d, by induction v is adjacent to precisely c_j vertices with neighbourhood isomorphic to $\partial^t(T_i, v_i, \tau_i) \in C_j$ with probability

$$\binom{d}{c_1 \ldots c_{\tilde{d}}} \prod_{j=1}^{\tilde{d}} p_t(C_j) + \mathrm{o}(1).$$

The number of cycles of length $\ell \leq 2t + 3$ in $\boldsymbol{G}$ is stochastically bounded by the number of such cycles in $\boldsymbol{G}(n, d_+/n)$ (the standard 1-type binomial random graph). For each ℓ, this number tends in distribution to a poisson variable with bounded mean (see e.g. Theorem 3.19 in [21]) and so the total number of such cycles is bounded w.h.p. Thus all the pairwise distances (in $\boldsymbol{G} - v$) between neighbours of v are at least $2t + 1$ w.h.p. (and in particular this proves the second part of the lemma). Therefore

$$\mathbb{E}_{\boldsymbol{G}}[X_{t+1}] = \frac{(d_+ + d_-)^d}{d! \exp(d_+ + d_-)} \binom{d}{c_1 \ldots c_{\tilde{d}}} \prod_{j=1}^{\tilde{d}} p_t(C_j) + \mathrm{o}(1).$$

By definition of $\boldsymbol{T}$, we obtain $\mathbb{E}[X_{t+1}] = p_{t+1} + \mathrm{o}(1)$. To apply Chebyshev's inequality, it remains to determine $\mathbb{E}[X_{t+1}^2]$. Let $\boldsymbol{v}, \boldsymbol{w} \in [n]$ be two randomly choosen vertices. Then w.h.p. $\boldsymbol{v}$ and $\boldsymbol{w}$ have distance at least $2t + 3$ in $\boldsymbol{G}$, conditioned on which $\partial^{t+1}(\boldsymbol{G}, \boldsymbol{v}, \boldsymbol{\sigma})$ and $\partial^{t+1}(\boldsymbol{G}, \boldsymbol{w}, \boldsymbol{\sigma})$ are independent. Therefore we obtain

$$\begin{aligned}
&\mathbb{P}_{\boldsymbol{v}, \boldsymbol{w}}\left(\partial^{t+1}(\boldsymbol{G}, \boldsymbol{v}, \boldsymbol{\sigma}) \cong \partial^{t+1}(T, r, \tau) \wedge \partial^{t+1}(\boldsymbol{G}, \boldsymbol{w}, \boldsymbol{\sigma}) \cong \partial^{t+1}(T, r, \tau)\right) \\
&\quad = \mathbb{P}_{\boldsymbol{v}}\left(\partial^{t+1}(\boldsymbol{G}, \boldsymbol{v}, \boldsymbol{\sigma}) \cong \partial^{t+1}(T, r, \tau)\right) \mathbb{P}_{\boldsymbol{w}}\left(\partial^{t+1}(\boldsymbol{G}, \boldsymbol{w}, \boldsymbol{\sigma}) \cong \partial^{t+1}(T, r, \tau)\right) + \mathrm{o}(1)
\end{aligned}$$

And finally

$$\begin{aligned}\mathbb{E}_{\boldsymbol{G}}[X_{t+1}^2] =&\mathbb{E}_{\boldsymbol{G}}\left[\mathbb{P}_{\boldsymbol{v}}\left(\partial^{t+1}(\boldsymbol{G},\boldsymbol{v},\boldsymbol{\sigma})\cong\partial^{t+1}(T,r,\tau)\right)\mathbb{P}_{\boldsymbol{w}}\left(\partial^{t+1}(\boldsymbol{G},\boldsymbol{w},\boldsymbol{\sigma})\cong\partial^{t+1}(T,r,\tau)\right)\right]\\ &+\frac{1}{n}\mathbb{E}_{\boldsymbol{G}}[X_{t+1}]+\mathrm{o}(1)\\ =&\mathbb{E}_{\boldsymbol{G}}[X_{t+1}]^2+\mathrm{o}(1).\end{aligned}$$

The first assertion follows from Chebyshev's inequality. ◀

2.4 The fixed point

Let (T,r,τ) be a rooted tree together with a map $\tau : V(T)\to\{\pm1\}$. Then for any pair v,w of adjacent vertices we have the WP messages $\mu_{v\to w}(t|T,\tau)$, $t\geq 0$, as defined in (2.2). Since we are going to be particularly interested in the messages directed towards the root, we introduce the following notation. Given the root r, any vertex $v\neq r$ of T has a unique parent vertex w (the neighbour of v on the unique path from v to r). Initially, let

$$\mu_{v\uparrow}(0|T,r,\tau)=\tau(v) \tag{2.4}$$

and define

$$\mu_{v\uparrow}(t|T,r,\tau)=\mu_{v\to w}(t|T,\tau) \tag{2.5}$$

for $t>0$. In addition, set $\mu_{r\uparrow}(0|T,r,\tau)=\tau(r)$ and let

$$\mu_{r\uparrow}(t+1|T,r,\tau)=\psi\left(\sum_{v\in\partial_T r}\mu_{v\uparrow}(t|T,r,\tau)\right)\qquad(t\geq 0) \tag{2.6}$$

be the message that r would send to its parent if there was one.

For $p=(p(-1),p(0),p(1))\in\mathcal{P}(\{-1,0,1\})$ we let $\bar{p}=(p(1),p(0),p(-1))$. Remembering the map

$$\mathcal{T}=\mathcal{T}_{d_+,d_-}:\mathcal{P}(\{-1,0,1\})\to\mathcal{P}(\{-1,0,1\})$$

from Section 1.2 and writing $\mathcal{T}^t$ for its t-fold iteration, we observe the following.

▶ **Lemma 2.4.** *Let $p_t=\mathcal{T}^t(0,0,1)$.*
1. *Given that $\boldsymbol{\tau}(r_{\boldsymbol{T}})=+1$, the message $\mu_{r_{\boldsymbol{T}}\uparrow}(t|\boldsymbol{T},r_{\boldsymbol{T}},\boldsymbol{\tau})$ has distribution p_t.*
2. *Given that $\boldsymbol{\tau}(r_{\boldsymbol{T}})=-1$, the message $\mu_{r_{\boldsymbol{T}}\uparrow}(t|\boldsymbol{T},r_{\boldsymbol{T}},\boldsymbol{\tau})$ has distribution $\bar{p}_t$.*

Proof. The proof is by induction on t. In the case $t=0$ the assertion holds because $\mu_{r_{\boldsymbol{T}}\uparrow}(0|\boldsymbol{T},r_{\boldsymbol{T}},\boldsymbol{\tau})=\boldsymbol{\tau}(r_{\boldsymbol{T}})$. Now, assume that the assertion holds for t. To prove it for $t+1$, let $C_\pm$ be the set of all children v of $r_{\boldsymbol{T}}$ with $\boldsymbol{\tau}(r_{\boldsymbol{T}})\boldsymbol{\tau}(v)=\pm1$. By construction, $|C_\pm|$ has distribution $\mathrm{Po}(d_\pm)$. Furthermore, let $(\boldsymbol{T}_v,v,\tau_v)$ signify the subtree pending on a child v of $r_{\boldsymbol{T}}$. Because $\boldsymbol{T}$ is a Galton-Watson tree, the random subtrees $\boldsymbol{T}_v$ are mutually independent. Moreover, each $\boldsymbol{T}_v$ is distributed as a Galton-Watson tree with offspring matrix (2.3) and a root vertex of type $\pm\boldsymbol{\tau}(r_{\boldsymbol{T}})$ for each $v\in C_\pm$. Therefore, by induction the message $\mu_{v\uparrow}(t|\boldsymbol{T}_v,v,\boldsymbol{\tau}_v)$ has distribution p_t if $\boldsymbol{\tau}(v)=1$ resp. $\bar{p}_t$ if $\boldsymbol{\tau}(v)=-1$. As a consequence,

$$\mu_{r_{\boldsymbol{T}}\uparrow}(t+1|\boldsymbol{T},r_{\boldsymbol{T}},\boldsymbol{\tau})=\psi\left(\sum_{v\in C_+}\mu_{v\uparrow}(t|\boldsymbol{T}_v,v,\boldsymbol{\tau}_v)+\sum_{v\in C_-}\mu_{v\uparrow}(t|\boldsymbol{T}_v,v,\boldsymbol{\tau}_v)\right)$$

has distribution p_{t+1} if $\boldsymbol{\tau}(r_{\boldsymbol{T}})=1$ and $\bar{p}_{t+1}$ otherwise. ◀

Lemma 2.4 shows that the operator $\mathcal{T}$ mimics WP on the Galton-Watson tree $(\boldsymbol{T}, r_{\boldsymbol{T}}, \boldsymbol{\tau})$. Hence, to understand the behaviour of WP after a large enough number of iterations we need to investigate the fixed point to which $\mathcal{T}^t(0,0,1)$ converges as $t \to \infty$. In Section 4 we will establish the following.

▶ **Proposition 2.5.** *The operator $\mathcal{T}$ has a unique skewed fixed point p^* and $\lim_{t\to\infty} \mathcal{T}^t(0,0,1) = p^*$.*

Proof of Theorem 1.1. Consider the random variables

$$X_n := \frac{1}{n}\mathrm{bis}(\boldsymbol{G}), \qquad Y_n^{(t)} := \frac{1}{2}\frac{1}{n}\sum_{v\in[n]}\sum_{w\in\partial_{\boldsymbol{G}}v} \mathbf{1}\{\mu_{w\to v}(t|\boldsymbol{G},\boldsymbol{\sigma}) = -\tilde{\psi}\left(\mu_v(t|\boldsymbol{G},\boldsymbol{\sigma})\right)\}.$$

Then Lemma 2.1 and Proposition 2.2 imply that for any $\varepsilon > 0$,

$$\lim_{t\to\infty}\lim_{n\to\infty} \mathbb{P}\left[|X_n - Y_n^{(t)}| > \varepsilon\right] = 0. \tag{2.7}$$

By Definition (2.2), $\mu_{w\to v}(t|\boldsymbol{G},\boldsymbol{\sigma})$ and $\mu_v(t|\boldsymbol{G},\boldsymbol{\sigma})$ are determined by $\partial^t_{\boldsymbol{G}}v$ and the initialisation $\mu_{u\to w}(0|\boldsymbol{G},\boldsymbol{\sigma})$ for all $u,w \in \partial^t_{\boldsymbol{G}}v$, $\{u,w\}\in E(\boldsymbol{G})$. Since (2.5) and (2.6) match the recursive definition (2.2) of $\mu_{w\to v}(t|\boldsymbol{G},\boldsymbol{\sigma})$ and $\mu_v(t|\boldsymbol{G},\boldsymbol{\sigma})$, Lemma 2.3 implies that for any fixed $t > 0$ (as n tends to infinity),

$$Y_n^{(t)} \overset{n\to\infty}{\to} x^{(t)} := \frac{1}{2}\mathbb{E}\left[\sum_{w\in\partial_{\boldsymbol{T}}r_{\boldsymbol{T}}} \mathbf{1}\{\mu_{w\uparrow}(t|\boldsymbol{T},r_{\boldsymbol{T}},\boldsymbol{\tau}) = -\psi(\mu_{r_{\boldsymbol{T}}}(t|\boldsymbol{T},r_{\boldsymbol{T}},\boldsymbol{\tau}))\}\right]$$
$$\text{in probability.} \tag{2.8}$$

Now let p^* denote the unique skewed fixed point of $\mathcal{T}$ guaranteed by Proposition 2.5. Since each child of $r_{\boldsymbol{T}}$ can be considered a root of an independent instance of $\boldsymbol{T}$ to which we can apply Lemma 2.4, we obtain that given $(\boldsymbol{\tau}(w))_{w\in\partial r_{\boldsymbol{T}}}$ the sequence $(\mu_{w\uparrow}(t|\boldsymbol{T},r_{\boldsymbol{T}},\boldsymbol{\tau}))_{w\in\partial r_{\boldsymbol{T}}}$ converges to a sequence of independent random variables $(\eta_w)_{w\in\partial r_{\boldsymbol{T}}}$ with distribution p^* (if $\boldsymbol{\tau}(w) = 1$) and $\bar{p}^*$ (if $\boldsymbol{\tau}(w) = -1$). By definition $\mu_{r_{\boldsymbol{T}}}(t|\boldsymbol{T},r_{\boldsymbol{T}},\boldsymbol{\tau})$ converges to $\sum_{w\in\partial r_{\boldsymbol{T}},\boldsymbol{\tau}(w)=1}\eta_w + \sum_{w\in\partial r_{\boldsymbol{T}},\boldsymbol{\tau}(w)=-1}\eta_w$. Considering the offspring distributions of $r_{\boldsymbol{T}}$ in both cases, i.e. $\boldsymbol{\tau}(r_{\boldsymbol{T}}) = \pm1$, we obtain from $\varphi_{d_+,d_-}(p) = \varphi_{d_+,d_-}(\bar{p})$ for all $p \in \mathcal{P}(\{-1,0,1\})$ that

$$\lim_{t\to\infty} x^{(t)} = \varphi_{d_+,d_-}(p^*). \tag{2.9}$$

Finally, combining (2.7)–(2.9) completes the proof. ◀

3 Proof of Proposition 2.2

▶ **Lemma 3.1.** *If $v \in \mathcal{C}$ and $w \in \partial_{\boldsymbol{G}}v$, then $\mu_{v\to w}(t|\boldsymbol{G},\boldsymbol{\sigma}) = \boldsymbol{\sigma}(v) = \mu_{v\to w}(t|\boldsymbol{G},\boldsymbol{\sigma}_{\mathcal{C}})$ for all $t \geq 0$.*

Proof. We proceed by induction on t. For $t = 0$ the assertion is immediate from the initialisation of the messages. To go from t to $t+1$, consider $v \in \mathcal{C}$ and $w \in \partial_{\boldsymbol{G}}v$. We may assume without loss of generality that $\boldsymbol{\sigma}(v) = 1$. By the definition of the WP message,

$$\mu_{v\to w}(t+1|\boldsymbol{G},\boldsymbol{\sigma}) = \psi\left(\sum_{u\in\partial_{\boldsymbol{G}}v\setminus\{w\}} \mu_{u\to v}(t|\boldsymbol{G},\boldsymbol{\sigma})\right) = \psi\left(S_+ + S_- + S_0\right) \tag{3.1}$$

where

$$S_+ := \sum_{u \in \mathcal{C} \cap \boldsymbol{\sigma}^{-1}(+1) \cap \partial_{\boldsymbol{G}} v \setminus \{w\}} \mu_{u \to v}(t|\boldsymbol{G}, \boldsymbol{\sigma}),$$

$$S_- := \sum_{u \in \mathcal{C} \cap \boldsymbol{\sigma}^{-1}(-1) \cap \partial_{\boldsymbol{G}} v \setminus \{w\}} \mu_{u \to v}(t|\boldsymbol{G}, \boldsymbol{\sigma}),$$

$$S_0 := \sum_{u \in \partial_{\boldsymbol{G}} v \setminus (\mathcal{C} \cup \{w\})} \mu_{u \to v}(t|\boldsymbol{G}, \boldsymbol{\sigma}).$$

Now, (2.1) ensures that

$$S_+ \geq d_+ - \frac{c}{4}\sqrt{d_+ \ln d_+}, \quad S_- \geq -d_- - \frac{c}{4}\sqrt{d_+ \ln d_+}, \quad S_0 \leq 100 \leq \frac{c}{4}\sqrt{d_+ \ln d_+}, \tag{3.2}$$

provided that the constant $c > 0$ is chosen large enough. Combining (3.1) and (3.2), we see that $S_+ + S_- + S_0 \geq 1$ and thus $\mu_{v \to w}(t+1|\boldsymbol{G}, \boldsymbol{\sigma}) = 1$. The exact same argument works for $\mu_{v \to w}(t+1|\boldsymbol{G}, \boldsymbol{\sigma}_{\mathcal{C}}) = 1$. ◀

Let $\boldsymbol{G}_v$ denote the subgraph of $\boldsymbol{G}$ induced on $\mathcal{C}_v$. To prove Proposition 2.2, fix $s > 0$ large enough. Let $\mathcal{S} = \mathcal{S}(s)$ be the set of all vertices such that either $|\mathcal{C}_v| > \sqrt{s}$ or $\boldsymbol{G}_v$ is cyclic. Then Lemma 2.1 (with slightly smaller ε) and Lemma 2.3 imply that $|\mathcal{S}| \leq \varepsilon n$ w.h.p. For the rest of this section, let $v \notin \mathcal{S}$ be fixed.

For $w \in \mathcal{C}_v \setminus \{v\}$ we let $w_{\uparrow v}$ be the neighbour of w on the path from w to v. We define $\boldsymbol{G}_{w \to v}$ as the component of w in the graph obtained from $\boldsymbol{G}_v$ by removing the edge $\{w, w_{\uparrow v}\}$. The vertex set of $\boldsymbol{G}_{w \to v}$ will be denoted by $\mathcal{C}_{w \to v}$. Further, $h_{w \to v}$ is the maximum distance between w and any other vertex in $\boldsymbol{G}_{w \to v}$. Additionally, h_v is the maximum distance between v and any other vertex in $\boldsymbol{G}_v$. Finally, let $\boldsymbol{\sigma}_v : \mathcal{C}_v \to \{\pm 1\}$, $w \mapsto \boldsymbol{\sigma}(w)$ and let $\boldsymbol{\sigma}_{\mathcal{C},v} : \mathcal{C}_v \cap \mathcal{C} \to \{\pm 1\}$, $w \mapsto \boldsymbol{\sigma}_{\mathcal{C}}(w)$.

▶ **Lemma 3.2.**

1. *For any $w \in \mathcal{C}_v \setminus \{v\}$ and any $t > h_{w \to v}$ we have* $\mu_{w \to w_{\uparrow v}}(t|\boldsymbol{G}, \boldsymbol{\sigma}) = \mu_{w \to w_{\uparrow v}}(h_{w \to v} + 1|\boldsymbol{G}, \boldsymbol{\sigma}) = \mu_{w \to w_{\uparrow v}}(t|\boldsymbol{G}, \boldsymbol{\sigma}_{\mathcal{C}})$.
2. *For any $t \geq h_v$ we have $\mu_v(t|\boldsymbol{G}, \boldsymbol{\sigma}) = \mu_v(h_v + 1|\boldsymbol{G}, \boldsymbol{\sigma}) = \mu_v(t|\boldsymbol{G}, \boldsymbol{\sigma}_{\mathcal{C}})$.*

Proof. The proof of (1) proceeds by induction on $h_{w \to v}$. The construction **C1**–**C2** of $\mathcal{C}_v$ ensures that any $w \in \mathcal{C}_v$ with $h_{w \to v} = 0$ either belongs to $\mathcal{C}$ or has no neighbour besides $w_{\uparrow v}$. Hence for the first case the assumption follows from Lemma 3.1. If $\partial_{\boldsymbol{G}} w \setminus \{w_{\uparrow v}\} = \emptyset$ we obtain that $\mu_{w \to w_{\uparrow v}}(t|\boldsymbol{G}, \boldsymbol{\sigma}) = \mu_{w \to w_{\uparrow v}}(t|\boldsymbol{G}, \boldsymbol{\sigma}_{\mathcal{C}}) = 0$ for all $t \geq 1$ by the definition of the WP messages. Now, assume that $h_{w \to v} > 0$ and let $t > h_{w \to v}$. Then all neighbours $u \neq w_{\uparrow v}$ of w in $\boldsymbol{G}_{w \to v}$ satisfy $h_{u \to v} < h_{w \to v}$. Thus, by induction

$$\begin{aligned}\mu_{w \to w_{\uparrow v}}(t|\boldsymbol{G}, \boldsymbol{\sigma}) &= \psi\left(\sum_{u \in \partial_{\boldsymbol{G}} w \setminus \{w_{\uparrow v}\}} \mu_{u \to w}(t-1|\boldsymbol{G}, \boldsymbol{\sigma})\right) \\ &= \psi\left(\sum_{u \in \partial_{\boldsymbol{G}} w \setminus \{w_{\uparrow v}\}} \mu_{u \to w}(h_{u \to v} + 1|\boldsymbol{G}, \boldsymbol{\sigma})\right) = \mu_{w \to w_{\uparrow v}}(h_{w \to v} + 1|\boldsymbol{G}, \boldsymbol{\sigma}).\end{aligned}$$

An analogous argument applies to $\mu_{w \to w_{\uparrow v}}(t|\boldsymbol{G}, \boldsymbol{\sigma}_{\mathcal{C}})$. The proof of (2) is similar. ◀

For each vertex $w \in \mathcal{C}_v$, $w \neq v$, let $\mu^*_{w \to v} = \mu_{w \to w_{\uparrow v}}(s|\boldsymbol{G}, \boldsymbol{\sigma})$. Further, let $\mu^*_w = \mu_w(s|\boldsymbol{G}, \boldsymbol{\sigma})$. In addition, for $z \in \{\pm 1\}$ let

$$\boldsymbol{\sigma}^z_{w \to v} : \mathcal{C}_{w \to v} \cap (\{w\} \cup \mathcal{C}) \to \{\pm 1\}, \qquad u \mapsto \begin{cases} z & \text{if } u = w, \\ \boldsymbol{\sigma}(u) & \text{otherwise.} \end{cases}$$

In words, $\boldsymbol{\sigma}^z_{w\to v}$ freezes w to z and all other $u \in \mathcal{C}_{w\to v}$ that belong to the core to $\boldsymbol{\sigma}(u)$. Analogously, let

$$\boldsymbol{\sigma}^z_v : \mathcal{C}_v \cap (\{v\} \cup \mathcal{C}) \to \{\pm 1\}, \qquad u \mapsto \begin{cases} z & \text{if } u = v, \\ \boldsymbol{\sigma}(u) & \text{otherwise.} \end{cases}$$

▶ **Lemma 3.3.** *Suppose that $u \in \mathcal{C}_v \setminus \{v\}$, such that $h_{u\to v} \geq 1$.*

1. *If $z = \mu^*_{u\to v} \in \{-1, 1\}$, then*

$$\operatorname{cut}(\boldsymbol{G}_{u\to v}, \boldsymbol{\sigma}^z_{u\to v}) < \operatorname{cut}(\boldsymbol{G}_{u\to v}, \boldsymbol{\sigma}^{-z}_{u\to v}). \tag{3.3}$$

Similarly, if $z = \psi(\mu^*_v) \in \{-1, 1\}$, then

$$\operatorname{cut}(\boldsymbol{G}_v, \boldsymbol{\sigma}^z_v) < \operatorname{cut}(\boldsymbol{G}_v, \boldsymbol{\sigma}^{-z}_v). \tag{3.4}$$

2. *If $\mu^*_{u\to v} = 0$, then*

$$\operatorname{cut}(\boldsymbol{G}_{u\to v}, \boldsymbol{\sigma}^{+1}_{u\to v}) = \operatorname{cut}(\boldsymbol{G}_{u\to v}, \boldsymbol{\sigma}^{-1}_{u\to v}). \tag{3.5}$$

Similarly, if $\mu^*_v = 0$, then

$$\operatorname{cut}(\boldsymbol{G}_v, \boldsymbol{\sigma}^{+1}_v) = \operatorname{cut}(\boldsymbol{G}_v, \boldsymbol{\sigma}^{-1}_v). \tag{3.6}$$

Proof. We prove (3.3) and (3.5) by induction on $h_{u\to v}$. If $h_{u\to v} = 1$ then we have that all neighbours $w \in \partial_{\mathcal{C}_{u\to v}} u$ of u with $\mu^*_{u\to v} \neq 0$ are in $\mathcal{C}$, i.e. fixed under $\boldsymbol{\sigma}^z_{u\to v}$. Since $\mathcal{C}_{u\to v} = \partial_{\boldsymbol{G}} u \setminus \{u_{\uparrow v}\} \cup \{u\}$, we obtain

$$\operatorname{cut}(\mathcal{C}_{u\to v}, \boldsymbol{\sigma}^{-z}_{u\to v}) - \operatorname{cut}(\mathcal{C}_{u\to v}, \boldsymbol{\sigma}^z_{u\to v}) = \left| \sum_{w \in \partial_{\boldsymbol{G}} u \setminus \{u_{\uparrow v}\}} \mu^*_{w\to v} \right| \tag{3.7}$$

by definition of z. By the induction hypothesis and because $\boldsymbol{G}_{u\to v}$ is a tree (as $v \notin \mathcal{S}$) we have that (3.7) holds for $h_{u\to v} > 1$ as well. A similar argument yields (3.4) and (3.6). ◀

Now, let $\mathcal{U}_v$ be the set of all $w \in \mathcal{C}_v$ such that $\mu^*_{w\to v} \neq 0$. Furthermore, let

$$\boldsymbol{\sigma}_{\uparrow v} : \mathcal{U}_v \cup \{v\} \to \{-1, +1\}, \qquad w \mapsto \begin{cases} \tilde{\psi}(\mu^*_v) & \text{if } w = v, \\ \mu^*_{w\to v} & \text{otherwise.} \end{cases}$$

Thus, $\boldsymbol{\sigma}_{\uparrow v}$ sets all $w \in \mathcal{C}_v \cap \mathcal{C} \setminus \{v\}$ to their planted sign and all $w \in \mathcal{U}_v \setminus \mathcal{C}$ to $\mu^*_{w\to v}$. Moreover, $\boldsymbol{\sigma}_{\uparrow v}$ sets v to $\psi(\mu^*_v)$ if $\psi(\mu^*_v) \neq 0$ and to 1 if there is a tie.

▶ **Corollary 3.4.** *We have* $\operatorname{cut}(\boldsymbol{G}_v, \boldsymbol{\sigma}_{\mathcal{C}}) = \operatorname{cut}(\boldsymbol{G}_v, \boldsymbol{\sigma}_{\uparrow v})$.

Proof. This is immediate from Lemma 3.3. ◀

Hence, in order to determine an optimal cut of $\boldsymbol{G}_v$ we merely need to figure out the assignment of the vertices in $\mathcal{C}_v \setminus (\{v\} \cup \mathcal{U}_v)$. Suppose that $\boldsymbol{\sigma}^*_{\uparrow v} : \mathcal{C}_v \to \{\pm 1\}$ is an optimal extension of $\boldsymbol{\sigma}_{\uparrow v}$ to a cut of $\boldsymbol{G}_v$, i.e.,

$$\operatorname{cut}(\boldsymbol{G}_v, \boldsymbol{\sigma}_{\uparrow v}) = \sum_{\{u,w\} \in E(\boldsymbol{G}_v)} \frac{1}{2}(1 - \boldsymbol{\sigma}^*_{v\uparrow}(u)\boldsymbol{\sigma}^*_{v\uparrow}(w)).$$

▶ **Corollary 3.5.** *It holds that* $\sum_{w \in \partial_{\boldsymbol{G}} v} \frac{1}{2}(1 - \boldsymbol{\sigma}^*_{v\uparrow}(v)\boldsymbol{\sigma}^*_{v\uparrow}(w)) = \sum_{w \in \partial_{\boldsymbol{G}} v} \mathbf{1}\{\mu^*_{w\to v} = -\tilde{\psi}(\mu_v)\}$.

Proof. Part (2) of Lemma 3.3 implies that $\boldsymbol{\sigma}^*_{v\uparrow}(v)\boldsymbol{\sigma}^*_{v\uparrow}(w) = 1$ for all $w \in \partial_{\boldsymbol{G}} v$ such that $\mu^*_{w\to v} = 0$. ◀

Proof of Proposition 2.2. Given $\varepsilon > 0$ choose $\delta = \delta(\varepsilon, d_+, d_-)$ sufficiently small and $s = s(\varepsilon, \delta, d_+, d_-) > 0$ sufficiently large. In particular, pick s large enough so that

$$\mathbb{P}(|\mathcal{S}| \geq \delta n) < \varepsilon. \tag{3.8}$$

Provided that δ is suitable small, the Chernoff bound implies that for large n

$$\mathbb{P}\left(\frac{1}{2}\sum_{v\in\mathcal{S}} |\partial_{\boldsymbol{G}} v| \geq \varepsilon n \,\middle|\, |\mathcal{S}| < \delta n\right) < \varepsilon. \tag{3.9}$$

Now, suppose that $\boldsymbol{\sigma}^*_{\mathcal{C}}$ is an optimal extension of $\boldsymbol{\sigma}_{\mathcal{C}}$ to a cut of $\boldsymbol{G}$ and let $v \notin \mathcal{S}$. Then using the definition of $\mathcal{C}_v$, Corollary 3.4 implies that

$$\sum_{w\in\partial_{\boldsymbol{G}} v} (1 - \boldsymbol{\sigma}^*_{\mathcal{C}}(v)\boldsymbol{\sigma}^*_{\mathcal{C}}(w)) = \sum_{w\in\partial_{\boldsymbol{G}} v} (1 - \boldsymbol{\sigma}^*_{v\uparrow}(v)\boldsymbol{\sigma}^*_{v\uparrow}(w)).$$

Therefore, we obtain

$$\mathbb{P}\left(\left|\mathrm{cut}(\boldsymbol{G}, \boldsymbol{\sigma}_{\mathcal{C}}) - \frac{1}{2}\sum_{v\notin\mathcal{S}}\sum_{w\in\partial_{\boldsymbol{G}} v} (1 - \boldsymbol{\sigma}^*_{v\uparrow}(v)\boldsymbol{\sigma}^*_{v\uparrow}(w))\right| \geq \varepsilon n\right) \leq \mathbb{P}\left(\frac{1}{2}\sum_{v\in\mathcal{S}} |\partial_{\boldsymbol{G}} v| \geq \varepsilon n\right) \leq 2\varepsilon.$$

The assertion follows from Lemma 3.2 for $t \geq s$. ◀

4 Proof of Proposition 2.5

We continue to denote the set of probability measures on $\mathcal{X} \subset \mathbb{R}^k$ by $\mathcal{P}(\mathcal{X})$. For a $\mathcal{X}$-valued random variable X we denote by $\mathcal{L}(X) \in \mathcal{P}(\mathcal{X})$ the distribution of X. Furthermore, if $p, q \in \mathcal{P}(\mathcal{X})$, then $\mathcal{P}_{p,q}(\mathcal{X})$ denotes the set of all probability measures μ on $\mathcal{X} \times \mathcal{X}$ such that the marginal distribution of the first (resp. second) component coincides with p (resp. q). The space $\mathcal{P}(\{-1, 0, 1\})$ is complete with respect to (any and in particular) the L_1-Wasserstein metric, defined by

$$\ell_1(p, q) = \inf\left\{\mathbb{E}|X - Y| : X, Y \text{ are random variables with } \mathcal{L}(X, Y) \in \mathcal{P}_{p,q}(\{-1, 0, 1\})\right\}.$$

In words, the infimum of $\mathbb{E}|X - Y|$ is over all couplings (X, Y) of the distributions p, q. Such a coupling (X, Y) is *optimal* if $\ell_1(p, q) = \mathbb{E}|X - Y|$. Finally, let $\mathcal{P}^*(\{-1, 0, 1\})$ be the set of all skewed probability measures on $\{-1, 0, 1\}$. Being a closed subset of $\mathcal{P}(\{-1, 0, 1\})$, $\mathcal{P}^*(\{-1, 0, 1\})$ is complete with respect to $\ell_1(\,\cdot\,,\,\cdot\,)$.

As in the definition (1.2)-(1.3) of the operator $\mathcal{T} = \mathcal{T}_{d_+,d_-}$ for $p \in \mathcal{P}(\{-1, 0, 1\})$ we let $(\eta_{p,i})_{i\geq 1}$ be a family of independent random variables with distribution p. Further, let $\gamma_\pm = \mathrm{Po}(d_\pm)$ be independent of each other and of the $(\eta_{p,i})_{i\geq 1}$. We introduce the shorthands

$$Z_p = Z_{p,d_+,d_-}, \qquad Z_{p,+} = \sum_{i=1}^{\gamma_+} \eta_{p,i}, \qquad Z_{p,-} = \sum_{i=\gamma_++1}^{\gamma_++\gamma_-} \eta_{p,i} \quad \text{so that} \quad Z_p = Z_{p,+} - Z_{p,-}.$$

Also set $\lambda = c\sqrt{d_+ \ln d_+}$ and recall that $c > 0$ is a constant that we assume to be sufficiently large.

▶ **Lemma 4.1.** *The operator $\mathcal{T}$ maps $\mathcal{P}^*(\{-1, 0, 1\})$ into itself.*

Proof. Suppose that $p \in \mathcal{P}(\{-1,0,1\})$ is skewed. Then

$$\mathbb{P}\left(Z_p < 1\right) \le \mathbb{P}\left(Z_{p,+} \le d_+ - \frac{\lambda-1}{2}\right) + \mathbb{P}\left(Z_{p,-} \ge d_- + \frac{\lambda-1}{2}\right). \tag{4.1}$$

Since $|\eta_{p,i}| \le 1$ for all i, we can bound the second summand from above by invoking the Chernoff bound on a binomial approximation of the Poisson distribution to obtain

$$\mathbb{P}\left(\gamma_- \ge d_- + \frac{c}{2}\sqrt{d_+ \ln d_+} - \frac{1}{2}\right) < \frac{1}{3}d_+^{-10}, \tag{4.2}$$

provided c is large enough. To bound the other summand from above we use that $(\eta_{p,i})_{i\ge1}$ is a sequence of independent *skewed* random variables, whence by the Chernoff bound

$$\begin{aligned}
&\mathbb{P}\left(Z_{p,+} \le d_+ - \frac{\lambda-1}{2}\right)\\
&\le \mathbb{P}\left(|\gamma_+ - d_+| > \lambda/8\right) + \mathbb{P}\left(Z_{p,-} \le d_+ - \frac{\lambda-1}{2}\middle|\gamma_+ \ge d_+ - \lambda/8\right)\\
&\le \frac{1}{3}d_+^{-10} + \mathbb{P}\left[\mathrm{Bin}(d_+ - \lambda/8, 1-d_+^{-10}) \le d_+ - \lambda/7\right] < \frac{2}{3}d_+^{-10},
\end{aligned} \tag{4.3}$$

provided that c is sufficiently big. Combining (4.1)–(4.3) completes the proof. ◀

▶ **Lemma 4.2.** *The operator $\mathcal{T}$ is ℓ_1-contracting on $\mathcal{P}^*(\{-1,0,1\})$.*

Proof. Let $p,q \in \mathcal{P}^*(\{-1,0,1\})$. We aim to show that $\ell_1(\mathcal{T}(p),\mathcal{T}(q)) \le \frac{1}{2}\ell_1(p,q)$. To this end, we let $(\eta_{p,i},\eta_{q,i})_{i\ge1}$ be a family of random variables with distribution p resp. q such that $(\eta_{p,i})_{i\ge1}$ are independent and $(\eta_{q,i})_{i\ge1}$ are independent but such that the pair $(\eta_{p,i},\eta_{q,i})$ is an optimal coupling for every i. Then by the definition of $\ell_1(\,\cdot\,,\,\cdot\,)$,

$$\ell_1(\mathcal{T}(p),\mathcal{T}(q)) \le \mathbb{E}\left|\psi(Z_p) - \psi(Z_q)\right|. \tag{4.4}$$

To estimate the r.h.s., let $\tilde\eta_{p,i} = \mathbf{1}\{\eta_{p,i} = 1\}$, $\tilde\eta_{q,i} = \mathbf{1}\{\eta_{q,i} = 1\}$. Further, let $\mathfrak{F}_i$ be the σ-algebra generated by $\tilde\eta_{p,i},\tilde\eta_{q,i}$ and let $\mathfrak{F}$ be the σ-algebra generated by γ_+,γ_- and the random variables $(\tilde\eta_{p,i},\tilde\eta_{q,i})_{i\ge1}$. Additionally, let $\gamma = \gamma_+ + \gamma_-$ and consider the three events

$$\mathfrak{A}_1 = \left\{\sum_{i=1}^{\gamma}\tilde\eta_{p,i}\tilde\eta_{q,i} \ge \gamma - 10\right\}, \qquad \mathfrak{A}_2 = \{\gamma \ge 2d_+\}, \qquad \mathfrak{A}_3 = \{\gamma_+ - \gamma_- \le 20\}.$$

We are going to bound $|\psi(Z_p) - \psi(Z_q)|$ on $\mathfrak{A}_1 \setminus (\mathfrak{A}_2 \cup \mathfrak{A}_3)$, $\overline{\mathfrak{A}_1 \cup \mathfrak{A}_2 \cup \mathfrak{A}_3}$, $\mathfrak{A}_2$ and $\mathfrak{A}_3 \setminus \mathfrak{A}_2$ separately. The bound on the first event is immediate: if $\mathfrak{A}_1 \setminus (\mathfrak{A}_2 \cup \mathfrak{A}_3)$ occurs, then $\psi(Z_p) = \psi(Z_q) = 1$ with certainty. Hence,

$$\mathbb{E}\left[|\psi(Z_p) - \psi(Z_q)| \cdot \mathbf{1}_{\mathfrak{A}_1\setminus(\mathfrak{A}_2\cup\mathfrak{A}_3)}\right] = 0. \tag{4.5}$$

Let us turn to the second event $\overline{\mathfrak{A}_1 \cup \mathfrak{A}_2 \cup \mathfrak{A}_3}$. Because the pairs $(\eta_{p,i},\eta_{q,i})_{i\ge1}$ are mutually independent, we find

$$\mathbb{E}\left[|\eta_{p,i} - \eta_{q,i}|\,|\,\mathfrak{F}\right] = \mathbb{E}\left[|\eta_{p,i} - \eta_{q,i}|\,|\,\mathfrak{F}_i\right] \qquad \text{for all } i \ge 1. \tag{4.6}$$

Clearly, if $\tilde\eta_{p,i}\tilde\eta_{q,i} = 1$, then $\eta_{p,i} - \eta_{q,i} = 0$. Consequently,

$$\mathbb{E}\left[|\eta_{p,i} - \eta_{q,i}|\,|\,\mathfrak{F}_i\right] \le \frac{\mathbb{E}|\eta_{p,i} - \eta_{q,i}|}{\mathbb{P}[\tilde\eta_{p,i}\tilde\eta_{q,i} = 0]} = \frac{\mathbb{E}|\eta_{p,1} - \eta_{q,1}|}{\mathbb{P}[\tilde\eta_{p,1}\tilde\eta_{q,1} = 0]}. \tag{4.7}$$

Since the events $\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3$ are $\mathfrak{F}$-measurable and because $\bar{\mathfrak{A}}_2$ ensures that $\gamma < 2d_+$, (4.6) and (4.7) yield

$$\mathbb{E}[|\psi(Z_p) - \psi(Z_q)|\,|\mathfrak{F}]\mathbf{1}_{\overline{\mathfrak{A}_1 \cup \mathfrak{A}_2 \cup \mathfrak{A}_3}} \leq \frac{2d_+ \mathbb{E}|\eta_{p,1} - \eta_{q,1}|}{\mathbb{P}[\tilde{\eta}_{p,1}\tilde{\eta}_{q,1} = 0]} \cdot \mathbf{1}_{\overline{\mathfrak{A}_1 \cup \mathfrak{A}_2 \cup \mathfrak{A}_3}}. \tag{4.8}$$

Further, because the pairs $(\eta_{p,i}, \eta_{q,i})_{i \geq 1}$ are independent and because p, q are skewed,

$$\mathbb{P}\left(\overline{\mathfrak{A}_1 \cup \mathfrak{A}_2 \cup \mathfrak{A}_3}\right) \leq \mathbb{P}\left(\gamma \leq 2d_+, \sum_{i=1}^{\gamma} \tilde{\eta}_{p,i}\tilde{\eta}_{q,i} \leq \gamma - 10\right) \leq \left(2d_+ \mathbb{P}\left(\tilde{\eta}_{p,1}\tilde{\eta}_{q,1} = 0\right)\right)^{10}. \tag{4.9}$$

Combining (4.8) and (4.9), we obtain

$$\mathbb{E}\left[\mathbb{E}\left[|\psi(Z_p) - \psi(Z_q)||\mathfrak{F}\right]\mathbf{1}_{\overline{\mathfrak{A}_1 \cup \mathfrak{A}_2 \cup \mathfrak{A}_3}}\right] \leq (2d_+)^{11}\,\mathbb{P}\left(\tilde{\eta}_{p,1}\tilde{\eta}_{q,1} = 0\right)^9 \mathbb{E}|\eta_{p,1} - \eta_{q,1}|. \tag{4.10}$$

Since p, q are skewed, we furthermore obtain $\mathbb{P}\left(\tilde{\eta}_{p,1}\tilde{\eta}_{q,1} = 0\right) \leq 2d_+^{-10}$. Therefore

$$\begin{aligned}\mathbb{E}\left[|\psi(Z_p) - \psi(Z_q)|\mathbf{1}_{\overline{\mathfrak{A}_1 \cup \mathfrak{A}_2 \cup \mathfrak{A}_3}}\right] &= \mathbb{E}\left[\mathbb{E}\left[|\psi(Z_p) - \psi(Z_q)||\mathfrak{F}\right]\mathbf{1}_{\overline{\mathfrak{A}_1 \cup \mathfrak{A}_2 \cup \mathfrak{A}_3}}\right] \\ &\leq 2^{20} d_+^{-79}\mathbb{E}|\eta_{p,1} - \eta_{q,1}|.\end{aligned}$$

With respect to $\mathfrak{A}_2$, the triangle inequality yields

$$\mathbb{E}[|\psi(Z_p) - \psi(Z_q)|\mathbf{1}_{\mathfrak{A}_2}] \leq 2\mathbb{E}|\eta_{p,1} - \eta_{q,1}| \cdot \mathbb{E}[\gamma \mathbf{1}_{\mathfrak{A}_2}]. \tag{4.11}$$

Further, since $\gamma = \mathrm{Po}(d_+ + d_-)$, the Chernoff bound entails that $\mathbb{E}[\gamma \mathbf{1}_{\mathfrak{A}_2}] \leq d_+^{-1}$ if the constant c is chosen large enough. Combining this estimate with (4.11), we get

$$\mathbb{E}[|\psi(Z_p) - \psi(Z_q)|\mathbf{1}_{\mathfrak{A}_2}] \leq 2d_+^{-1}\mathbb{E}|\eta_{p,1} - \eta_{q,1}|. \tag{4.12}$$

Finally, on $\mathfrak{A}_3 \setminus \mathfrak{A}_2$ we have

$$\mathbb{E}[|\psi(Z_p) - \psi(Z_q)|\mathbf{1}_{\mathfrak{A}_3 \setminus \mathfrak{A}_2}] \leq 4d_+\mathbb{E}|\eta_{p,1} - \eta_{q,1}|\,\mathbb{P}\left[\gamma_+ - \gamma_- \leq 20\right]. \tag{4.13}$$

Since $\gamma_\pm = \mathrm{Po}(d_\pm)$ and $d_+ - d_- \geq \lambda$, the Chernoff bound yields $\mathbb{P}\left[\gamma_+ - \gamma_- \leq 20\right] \leq d_+^{-2}$, if c is large enough. Hence, (4.13) implies

$$\mathbb{E}[|\psi(Z_p) - \psi(Z_q)|\mathbf{1}_{\mathfrak{A}_3 \setminus \mathfrak{A}_2}] \leq 4d_+^{-1}\mathbb{E}|\eta_{p,1} - \eta_{q,1}|. \tag{4.14}$$

Finally, the assertion follows from (4.4), (4.5), (4.10), (4.12) and (4.14). ◀

Proof of Proposition 2.5. The assertion follows from Lemmas 4.1 and 4.2 and the Banach fixed point theorem. ◀

Acknowledgements. We would like to thank an anonymous referee for detailed comments on an earlier draft of the paper.

References

1 E. Abbe, A. Bandeira, and G. Hall. Exact recovery in the stochastic block model. *arXiv preprint arXiv:1405.3267*, 2014.

2 D. Aldous and J. Steele. The objective method: probabilistic combinatorial optimization and local weak convergence. In *Probability on discrete structures*, pages 1–72. Springer, 2004.

3 S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56:5, 2009.

4 V. Bapst, A. Coja-Oghlan, S. Hetterich, F. Rassmann, and D. Vilenchik. The condensation phase transition in random graph coloring. *arXiv preprint arXiv:1404.5513*, 2014.

5 B. Bollobás and A. Scott. Max cut for random graphs with a planted partition. *Combinatorics, Probability and Computing*, 13:451–474, 2004.

6 R. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *Proc. 28th Foundations of Computer Science*, pages 280–285. IEEE, 1987.

7 T. Bui, S. Chaudhuri, T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7:171–191, 1987.

8 T. Carson and R. Impagliazzo. Hill-climbing finds random planted bisections. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 903–909. Society for Industrial and Applied Mathematics, 2001.

9 A. Coja-Oghlan. A spectral heuristic for bisecting random graphs. In *Random Structures and Algorithms*, pages 351–398, 2006.

10 A. Coja-Oghlan, O. Cooley, M. Kang, and K. Skubch. How does the core sit inside the mantle? *arXiv preprint arXiv:1503.09030*, 2015.

11 A. Condon and R. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18:116–140, 2001.

12 A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84:066106, 2011.

13 A. Dembo, A. Montanari, and S. Sen. Extremal cuts of sparse random graphs. *arXiv preprint arXiv:1503.03923*, 2015.

14 T. Dimitriou and R. Impagliazzo. Go with the winners for graph bisection. In *Proc. 9th SODA*, pages 510–520. ACM/SIAM, 1998.

15 M. Dyer and A. Frieze. The solution of some random np-hard problems in polynomial expected time. *Journal of Algorithms*, 10:451–489, 1989.

16 U. Feige and J. Kilian. Heuristics for semirandom graph problems. *Journal of Computer and System Sciences*, 63:639–671, 2001.

17 U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM Journal on Computing*, 31:1090–1118, 2002.

18 M. Garey, D. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical computer science*, 1:237–267, 1976.

19 M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42:1115–1145, 1995.

20 P. Holland, K. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5:109–137, 1983.

21 S. Janson, T. Łuczak, and A. Ruciński. *Random graphs.* John Wiley & Sons, 2000.

22 M. Jerrum and G. Sorkin. The metropolis algorithm for graph bisection. *Discrete Applied Mathematics*, 82:155–175, 1998.

23 A. Juels. *Topics in black-box combinatorial function optimization.* PhD thesis, UC Berkeley, 1996.

24 R. Karp. *Reducibility among combinatorial problems.* Springer, 1972.

25 M. Karpinski. Approximability of the minimum bisection problem: An algorithmic challenge. In *Proc. 27th Mathematical Foundations of Computer Science*, pages 59–67. Springer, 2002.

26 S. Khot. Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing*, 36:1025–1071, 2006.

27 L. Kučera. Expected complexity of graph partitioning problems. *Discrete Applied Mathematics*, 57:193–212, 1995.

28 M. Luczak and C. McDiarmid. Bisecting sparse random graphs. *Random Structures and Algorithms*, 18:31–38, 2001.

29 K. Makarychev, Y. Makarychev, and A. Vijayaraghavan. Approximation algorithms for semi-random partitioning problems. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 367–384. ACM, 2012.

30 L. Massoulié. Community detection thresholds and the weak ramanujan property. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 694–703. ACM, 2014.

31 F. McSherry. Spectral partitioning of random graphs. In *Proc. 42nd Foundations of Computer Science*, pages 529–537. IEEE, 2001.

32 M. Mézard and A. Montanari. *Information, physics, and computation.* Oxford University Press, 2009.

33 E. Mossel, J. Neeman, and A. Sly. Stochastic block models and reconstruction. *arXiv preprint arXiv:1202.1499*, 2012.

34 E. Mossel, J. Neeman, and A. Sly. Belief propagation, robust reconstruction, and optimal recovery of block models. *arXiv preprint arXiv:1309.1380*, 2013.

35 E. Mossel, J. Neeman, and A. Sly. A proof of the block model threshold conjecture. *arXiv preprint arXiv:1311.4115*, 2013.

36 E. Mossel, J. Neeman, and A. Sly. Consistency thresholds for the planted bisection model. *arXiv preprint arXiv:1407.1591 v2*, 2014.

37 R. Neininger and L. Rüschendorf. A general limit theorem for recursive algorithms and combinatorial structures. *The Annals of Applied Probability*, 14:378–418, 2004.

38 H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proc. 40th ACM symposium on Theory of computing*, pages 255–264. ACM, 2008.

39 M. Talagrand. The parisi formula. *Annals of Mathematics*, 163:221–263, 2006.

Local Convergence of Random Graph Colorings

Amin Coja-Oghlan[*1], **Charilaos Efthymiou**[†2], **and Nor Jaafari**[1]

1 **Mathematics Institute, Goethe University of Frankfurt**
10 Robert Mayer St, Frankfurt 60325, Germany
`{acoghlan,jaafari}@math.uni-frankfurt.de`

2 **College of Computing, Georgia Institute of Technology**
266 Ferst Drive, Atlanta, 30332, USA
`efthymiou@gmail.com`

Abstract

Let $\mathbf{G} = \mathbf{G}(n, m)$ be a random graph whose average degree $d = 2m/n$ is below the k-colorability threshold. If we sample a k-coloring σ of $\mathbf{G}$ uniformly at random, what can we say about the correlations between the colors assigned to vertices that are far apart? According to a prediction from statistical physics, for average degrees below the so-called *condensation threshold* $d_{k,\mathrm{cond}}$, the colors assigned to far away vertices are asymptotically independent [Krzakala et al: PNAS 2007]. We prove this conjecture for k exceeding a certain constant k_0. More generally, we determine the joint distribution of the k-colorings that σ induces locally on the bounded-depth neighborhoods of a fixed number of vertices.

1998 ACM Subject Classification G.2 Discrete Mathematics, G.3 Probability and Statistics, G.2.2. Graph Theory, F.2.2, Computations on discrete structures

Keywords and phrases Random graph, Galton-Watson tree, phase transitions, graph coloring, Gibbs distribution, convergence

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.726

1 Introduction

We let $\mathbf{G} = \mathbf{G}(n, m)$ *denote the random graph on the vertex set* $[n] = \{1, \ldots, n\}$ *with precisely* m *edges. Unless specified otherwise, we assume that* $m = m(n) = \lceil dn/2 \rceil$ *for a fixed number* $d > 0$.

Going back to the seminal paper of Erdős and Rényi [18], the problem of coloring the random graph $\mathbf{G}$ remains one of the longest-standing challenges in probabilistic combinatorics. Over the past half-century, efforts have been devoted to problems ranging from determining the likely value of the chromatic number [4, 9, 26, 29] and its concentration [5, 27, 36] to algorithmic ones such as constructing or sampling colorings of the random graph [3, 12, 14, 16, 23, 24, 38].

The last few years have witnessed substantial progress w.r.t. estimating $\chi(\mathbf{G})$ accurately. Achlioptas and Friedgut [2] proved that for any number $k \geq 3$ of colors there exists a *sharp threshold sequence* $d_{k-\mathrm{col}} = d_{k-\mathrm{col}}(n)$ such that for any fixed $\varepsilon > 0$, $\mathbf{G}$ is k-colorable w.h.p. if $2m/n < d_{k-\mathrm{col}}(n) - \varepsilon$, while $\chi(\mathbf{G}) > k$ w.h.p. if $2m/n > d_{k-\mathrm{col}}(n) + \varepsilon$. Furthermore, starting

[*] The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 278857–PTCC.
[†] Research is supported by ARC GaTech.

from [4] there is a sequence of results which improve on the bounds for $d_{k-\mathrm{col}}(n)$. The best current bounds read

$$(2k-1)\ln k - 2\ln 2 + \delta_k \le \liminf_{n\to\infty} d_{k-\mathrm{col}}(n) \le \limsup_{n\to\infty} d_{k-\mathrm{col}}(n) \le (2k-1)\ln k - 1 + \delta_k, \quad (1)$$

where $\lim_{k\to\infty}\delta_k = 0$, see [10, 11]. In addition, the understanding of the geometry of the set of k-colorings has advanced significantly [1, 32], too.

Much of the recent work has been inspired by ideas from statistical physics. Indeed, based on a systematic but non-rigorous approach, the "cavity method" [30, 31], physicists have derived "predictions" on a wide range of problems in combinatorics. More specifically, from the statistical mechanics viewpoint, problems such as random graph coloring are examples of "disordered systems", and the principal interest is in the associated *Gibbs distributions*. For instance, if $k \ge 3$ is an integer and $G = (V, E)$ is a k-colorable graph, this is a probability distribution on $[k]^V$, namely the uniform distribution $\mu_{G,k}$ on the set $\mathcal{S}_k(G)$ of all k-colorings of G. The fact that graph coloring is computationally hard suggests that $\mu_{G,k}$ is a quite a complex object to study, and so it is. Not only does the Gibbs distribution reflect structural properties of the problem such as the geometry of the space of colorings. But it is also expected that its *spatial mixing properties*, i.e. the nature of correlations under the Gibbs distribution, have a substantial impact on the performance of algorithms both for finding and sampling (counting) colorings [13, 16, 28, 38]. Moreover, the existence of long range correlation phenomena under the Gibbs distribution have been related to the hardness of certain computational problems, e.g. see [37, 19, 20].

The usual setting where we study spatial mixing under the Gibbs distribution is as follows: Considering a small region of vertices $\Lambda \subset V(G)$, we analyze how the coloring of Λ is correlated with that of vertices at some distance ω around Λ, as ω increases. It turns out that spatial mixing is the most fundamental variant in understanding the behavior of Gibbs distribution.

As far as the case where $\mathbf{G} = G(n, m)$ is regarded, it seems that the local treelike structure of the graph plays a prominent role in the study of spatial mixing. It is well known that the neighborhood structure within a fixed radius around some vertex $v \in \mathbf{G}$ is asymptotically distributed as a Galton-Watson tree with offspring distribution $Po(d)$, where $d = 2m/n$. This observation motivates the question of whether it is possible to study the spatial mixing properties of the Gibbs distribution on $\mathbf{G}$ by means of the Gibbs distribution on Galton-Watson trees with $Po(d)$ offspring. For certain questions like the so-called Gibbs uniqueness (e.g. see [21]) the reduction from $\mathbf{G}$ to a Galton-Watson tree is trivially true. However, for different kinds of spatial mixing this reduction turns out to be far from trivial, e.g. the reconstructibility we consider here.

In this work we study the above question in its full generality. We investigate the asymptotic relation between the Gibbs distribution over the k-colorings of a $Po(d)$ Galton-Watson tree and the marginal of the Gibbs distribution over the k-colorings of $\mathbf{G}$ on a fixed size neighborhood around some vertex v. We show that these two distributions converge to each other in a very specific way, provided that the average degree d is smaller than the so-called *condensation threshold*

$$d_{k,\mathrm{cond}} = \inf\left\{ d > 0 : \limsup_{n\to\infty} \mathbb{E}[Z_k(G(n,m))^{1/n}] < k(1-1/k)^{d/2} \right\}.$$

The precise value of $d_{k,\mathrm{cond}}$ has recently been determined rigorously (for k exceeding a certain constant k_0) [7]. An asymptotic expansion of that precise formula yields the explicit expression

$$d_{k,\mathrm{cond}} = (2k-1)\ln k - 2\ln 2 + \delta_k \qquad \text{with } \delta_k \to 0,$$

matching the best current lower bound on $d_{k-\mathrm{col}}$ (cf. (1)). Based on non-rigorous considerations from statistical physics, we do not expect that the aforementioned convergence is true for $d > d_{k,\mathrm{cond}}$ (see further discussion in Section 3).

As a matter of fact we show that the above convergence does hold even if we consider a fixed number of neighborhoods jointly. That is, for some fixed r, we consider r many vertices in $\mathbf{G}$ and a small radius neighborhood around each vertex. We show that the joint distribution of the k-colorings of these neighborhoods under $\mu_{\mathbf{G},k}$ converges to the product of r Gibbs distributions over the k-colourings of a $Po(d)$ Galton-Watson tree.

All the above imply that the reduction from $\mathbf{G}$ to the Galton-Watson tree is indeed correct for studying any generic spatial mixing condition as long as $d < d_{k,\mathrm{cond}}$. A direct corollary of our results is to verify the existence of a certain kind of decorrelation between the color assignments of distant vertices, that was predicted by physicists using the cavity method, see [25]. Furthermore, there is an impressing consequence that comes from the observation that $d_{k,\mathrm{cond}}$ is asymptotically equal to the k-colorability threshold for $\mathbf{G}$. Our result shows that the long-range effects that drive up the chromatic number of $\mathbf{G}$ are not noticeable at a "local scale" (see further discussion in Section 3).

The challenging task in our analysis is that even though we speak about the local behavior of the Gibbs distribution of $\mathbf{G}$, we still need to argue about its global properties. To this end we make a novel use of the "planted trick", introduced in [1]. In particular, we generalize the planted model by introducing what we call the "planted replica model". The planting, now, involves two independent k-colorings rather than just one. Our formalization of the problem as well as the use of planting are quite generic. For this reason, we expect that our approach can be extended to other models on random (hyper)graphs such as the hard-core model, hypergraph two coloring, random k-SAT, etc.

2 Results

For a k-colorable graph $G = (V, E)$, a vertex v and an integer $\omega \geq 0$ we let $\partial^\omega(G, v)$ signify the depth-ω neighborhood of v, i.e., the graph obtained from G by deleting all vertices at a distance greater than ω from v. Additionally, for a set $U \subset V$ we let $\mu_{k,G|U}$ denote the projection of $\mu_{k,G}$ onto $[k]^U$, i.e.,

$$\mu_{k,G|U}(\sigma_0) = \mu_{k,G}\left(\left\{\sigma \in [k]^V : \forall u \in U : \sigma(u) = \sigma_0(u)\right\}\right) \qquad (\sigma_0 \in [k]^U).$$

If H is a subgraph of G we briefly write $\mu_{k,G|H} = \mu_{k,G|V(H)}$. Additionally, given a graph G, we let $\mathbf{v}_1, \mathbf{v}_2, \ldots$ denote vertices of G that are chosen uniformly and independently at random. Finally, let $\|\cdot\|_{\mathrm{TV}}$ be the total variation norm. The main result of this paper is

▶ **Theorem 1.** *There is a constant $k_0 > 0$ such that for any $k \geq k_0$, $d < d_{k,\mathrm{cond}}$, $l \geq 1$, $\omega \geq 0$ we have*

$$\lim_{n\to\infty} \mathbb{E}\left\|\mu_{k,\mathbf{G}|\partial^\omega(\mathbf{G},\mathbf{v}_1)\cup\cdots\cup\partial^\omega(\mathbf{G},\mathbf{v}_l)} - \bigotimes_{i=1}^{l} \mu_{k,\partial^\omega(\mathbf{G},\mathbf{v}_i)}\right\|_{\mathrm{TV}} = 0. \tag{2}$$

In words, suppose that $k \geq k_0$ is not too small, let $d < d_{k,\mathrm{cond}}$ and fix integers l, ω. Choose a random graph $\mathbf{G}$ and pick l vertices $\mathbf{v}_1, \ldots, \mathbf{v}_l$ uniformly and independently at random. Standard properties of the random graph ensure that w.h.p. their depth-ω neighborhoods

$\partial^\omega(\mathbf{G},\mathbf{v}_1),\ldots,\partial^\omega(\mathbf{G},\mathbf{v}_l)$ are pairwise disjoint and acyclic. Hence, each $\partial^\omega(\mathbf{G},\mathbf{v}_i)$ is a tree w.h.p. However, w.h.p. in $\mathbf{G}$ there are paths of length $\Omega(\log n)$ joining (most of) the vertices at distance precisely ω from the randomly choosen "roots" $\mathbf{v}_i$. Now, (2) states that w.h.p. the total variation distance of the following two distributions tends to 0 as $n \to \infty$. Under the first distribution, choose a k-coloring σ of the *entire* random graph $\mathbf{G}$ uniformly at random and consider its projection onto the forest $\partial^\omega(\mathbf{G},\mathbf{v}_1)\cup\cdots\cup\partial^\omega(\mathbf{G},\mathbf{v}_l)$. In particular, σ has to respect the constraints imposed by the "long paths" that join the different components $\partial^\omega(\mathbf{G},\mathbf{v}_i)$. Under the second distribution, ignore these long-range effects and obtain a k-coloring of $\partial^\omega(\mathbf{G},\mathbf{v}_1)\cup\cdots\cup\partial^\omega(\mathbf{G},\mathbf{v}_l)$ simply by picking a k-coloring of each tree $\partial^\omega(\mathbf{G},\mathbf{v}_i)$ independently and uniformly at random (a task that can be performed efficiently by dynamic programming).

Setting $\omega = 0$ in Theorem 1 yields the following statement, which is of interest in its own right.

▶ **Corollary 2.** *There is a number $k_0 > 0$ such that for all $k \geq k_0$, $d < d_{k,\mathrm{cond}}$ and any integer $l > 0$ we have*

$$\lim_{n\to\infty} \mathbb{E}\left\|\mu_{k,\mathbf{G}|\{\mathbf{v}_1,\ldots,\mathbf{v}_l\}} - \bigotimes_{i=1}^{l}\mu_{k,\mathbf{G}|\{\mathbf{v}_i\}}\right\|_{\mathrm{TV}} = 0. \tag{3}$$

By the symmetry of the colors, for each vertex v the "marginal distribution" $\mu_{k,\mathbf{G}|\{v\}}$ is just the uniform distribution on $[k]$ for every vertex v. Hence, Corollary 2 states that for $d < d_{k,\mathrm{cond}}$ w.h.p. in the random graph $\mathbf{G}$ for randomly chosen vertices $\mathbf{v}_1,\ldots,\mathbf{v}_l$ the following is true: if we choose a k-coloring σ of $\mathbf{G}$ at random, then $(\sigma(\mathbf{v}_1),\ldots,\sigma(\mathbf{v}_l)) \in [k]^l$ is asymptotically uniformly distributed. Prior results of Montanari and Gershenfeld [22] and of Montanari, Restrepo and Tetali [33] imply that (3) holds for $d < 2(k-1)\ln(k-1)$, about an additive $\ln k$ below $d_{k,\mathrm{cond}}$.

Another interesting special case occurs if we set $l = 1$ in Theorem 1. In this case we obtain a result about the well-known *reconstruction problem*. Suppose we draw a k-coloring σ of $\mathbf{G}$ at random and we reveal the assignment of a vertex v. The reconstruction problem amounts to studying how the information about the assignment at v biases the distribution of assignments of other vertices in $\mathbf{G}$, i.e. point to set correlation. It is straightforward that the assignments at the neighbors of v are correlated with that of v since they must be distinct. More generally, the reconstruction problem considers the bias on the assignments of vertices on a *fixed* "radius" ω from v. If this correlation persists as $\omega \to \infty$ we say that *reconstruction is possible* on $\mathbf{G}$. Otherwise we say that *reconstruction is impossible.* A similar notion can be defined on a random Galton-Watson tree $\mathbf{T}(k,d)$ with offspring distribution $\mathrm{Po}(d)$. That is, the reconstruction problem considers how the color assignment at the root biases the distribution of the assignments of the vertices at level ω in a random k-colouring of $\mathbf{T}(k,d)$. [1]

▶ **Corollary 3.** *There is a number $k_0 > 0$ such that for all $k \geq k_0$ and $d < d_{k,\mathrm{cond}}$ the following is true.*

Reconstruction is possible on $\mathbf{G} \Leftrightarrow$ *tree reconstruction is possible on* $T(k,d)$. (4)

The point of Corollary 3 is that it reduces the reconstruction problem on a combinatorially extremely intricate object, namely the random graph $\mathbf{G}$, to the same problem on a much

[1] Formal definitions can be found in [22].

simpler structure, namely the Galton-Watson tree. That said, the reconstruction problem on $\mathbf{T}(d)$ is far from trivial. The best current bounds show that there exists a sequence $(\delta_k)_k \to 0$ such that non-reconstruction holds in $\mathbf{T}(d)$ if $d < (1-\delta_k)k \ln k$, while reconstruction occurs if $d > (1+\delta_k)k \ln k$ [15].

Montanari, Restrepo and Tetali [33] proved (4) for $d < 2(k-1)\ln(k-1)$, about an additive $\ln k$ below $d_{k,\mathrm{cond}}$. This gap could alternatively be plugged by invoking recent results on the geometry of the set of k-colorings [6, 10, 32]. However, Corollary 3 is an immediate consequence of Theorem 1.

3 Discussion and related work

Erdős' observation [17] that the random graph $\mathbf{G}$ provides an example of a graph with high girth and high chromatic number is, quite literally, a textbook application of the "probabilistic method". One possible proof is to combine (1) with the well-known observation that for any $\ell \geq 3$ the probability that $\mathbf{G}$ fails to contain a cycle of length at most ℓ remains bounded away from 0 as $n \to \infty$. The difficulty of coming up with a deterministic construction of such a graph highlights the extent to which the phenomenon baffles intuition [34].

Theorem 1 goes one step further. It shows that the long-range effects that drive up the chromatic number of $\mathbf{G}$ are "locally elusive". Indeed, suppose we project a random k-coloring of $\mathbf{G}$ to the depth-ω neighborhood of a bounded number of vertices. Then Theorem 1 shows that asymptotically no traces of the long-range effects that drive up the chromatic number remain as the induced coloring is distributed as though no such effects were present. In particular, w.h.p. *any* k-coloring of the neighborhood of a given vertex v extends in asymptotically the same number of ways to the entire graph. Needless to say, we are unaware of any deterministic construction that exhibits this property.

It is instructive to discuss the relationship between Theorem 1 and the geometry of the set $\mathcal{S}_k(\mathbf{G})$ of k-colorings. For $(1+\eta_k)k \ln k < d < d_{k,\mathrm{cond}}$, where $\lim_{k\to\infty} \eta_k = 0$, w.h.p. the set $\mathcal{S}_k(\mathbf{G})$ shatters into an exponential number of disjoint "clusters" $\mathcal{C}_1, \ldots, \mathcal{C}_N$, each containing merely an exponentially small fraction of the set $\mathcal{S}_k(\mathbf{G})$ [1]. Additionally, w.h.p. a randomly chosen coloring σ of $\mathbf{G}$ belongs to a cluster $\mathcal{C}_i$ that is "frozen" [1, 32]. That is, there is a number $\Omega(n)$ of vertices v that receive the same color under *all* the k-colorings in $\mathcal{C}_i$. Conversely, the cluster $\mathcal{C}_i$ can be characterised (almost) completely by a map $\tau : V \to [k] \cup \{*\}$ that assigns all frozen vertices their color and that sets all other variables to the "joker color" $*$ [10, 32]. In effect, the *internal* structure of a typical cluster is subject to strong long-range correlations. For instance, if we consider the projection of a random k-coloring τ_i chosen from the cluster $\mathcal{C}_i$ to a bounded number $\mathbf{v}_1, \ldots, \mathbf{v}_l$ of randomly chosen vertices, then the distribution of the resulting color vector $(\tau_i(\mathbf{v}_1), \ldots, \tau_i(\mathbf{v}_l))$ would be far from uniform. What Theorem 2 and Corollary 2 show is that these long range correlations "cancel out perfectly" due to the large overall number of clusters.

No such cancellation is expected to occur for d beyond $d_{k,\mathrm{cond}}$ but below the k-colorability threshold [25]. In fact, non-rigorous but sophisticated arguments from physics predict that in this regime, the so-called *condensed phase*, the set $\mathcal{S}_k(\mathbf{G})$ is dominated by a *bounded* number of frozen clusters. Hence, the joint distribution $(\sigma(\mathbf{v}_1), \ldots, \sigma(\mathbf{v}_l))$ will be a mixture of a bounded number of "frozen" distributions $(\tau_i(\mathbf{v}_1), \ldots, \tau_i(\mathbf{v}_l))$. Consequently, we expect Theorem 1 to be best-possible in terms of the range of d.

Theorem 1 deals with the absence of "long-range correlations" in the random graph coloring problem. It makes sense to raise similar questions in a wide variety of other problems as well. In fact, the method that we develop in this paper is rather generic, and we expect that it will generalise to various other problems; the following section will provide a detailed discussion. Immediate examples that spring to mind include random hypergraph coloring or models from physics such as the Potts model. Another potential candidate may be the hardcore model on the random regular graph, which was studied by Bhatnagar, Sly and Tetali via a rather different approach [8], and its generalisations [35].

4 Outline

Throughout this section we assume that $k \geq k_0$ and that $d < d_{k,\mathrm{cond}}$.

4.1 Spatial mixing via replicas

The proof of Theorem 1 consists of three components. The first part is to reduce the proof of Theorem 1 to studying the distribution called *random replica model.* This is a distribution over triples $(\mathbf{G}, \sigma_1, \sigma_2)$ such that $\mathbf{G}$ is a k-colorable graph and $\sigma_1, \sigma_2 \in \mathcal{S}_k(\mathbf{G})$. In particular, it is induced by the following experiment.

RR1 Choose a random graph $\mathbf{G} = \mathbf{G}(n, m)$ subject to the condition that $\mathbf{G}$ is k-colorable.

RR2 Sample two k-colorings σ_1, σ_2 of $\mathbf{G}$ uniformly and independently.

In an analogous manner we define the distribution $\mathbf{T}^{\otimes}(k, d)$ as follows: The distribution is over triples $(\mathbf{T}, \tau_1, \tau_2)$, where $\mathbf{T}$ is an instance of the Galton-Watson tree with offspring distribution $\mathrm{Po}(d)$ and τ_1, τ_2 are two independent random k-colorings of $\mathbf{T}$.

So as to proceed with our arguments we need a bit of terminology. A *rooted graph* is a graph $G = (V, E)$ with vertex set $V \subset \mathbb{R}$ together with a distinguished vertex $v_0 \in V$, the *root*. Further, a *dicolored rooted graph* is a k-colorable rooted graph (G, v_0) together with two k-colorings $\sigma_1, \sigma_2 \in \mathcal{S}_k(G)$. An *isomorphism* between two dicolored rooted graphs $(G, v_0, \sigma_1, \sigma_2)$, $(G', v_0', \sigma_1', \sigma_2')$ is a graph isomorphism $\varphi : G \to G'$ such that $\varphi(v_0) = v_0'$, $\sigma_1 = \sigma_1' \circ \varphi$, $\sigma_2 = \sigma_2' \circ \varphi$ and $\varphi(v) < \varphi(w)$ iff $v < w$ for all vertices v, w of G. In words, φ preserves the roots, both colorings and the order of the vertices (which are real numbers). For $\omega \geq 0$ we denote by $\partial^{\omega}(G, v, \sigma_1, \sigma_2)$ the rooted dicolored graph obtained from $(G, v, \sigma_1, \sigma_2)$ by deleting all vertices whose distance from v exceeds ω.

Given Γ a rooted dicolored graph and $\omega \geq 0$, for some graph $G = (V, E)$, $\sigma_1, \sigma_2 \in \mathcal{S}_k(G)$, we let

$$Q_{\Gamma,\omega}(G) = \frac{1}{|V|} \sum_{v \in V} \mathbf{1}\left\{\partial^{\omega}(G, v, \sigma_1, \sigma_2) \cong \partial^{\omega}\Gamma\right\}.$$

That is, $Q_{\Gamma,\omega}(G)$ is the fraction of vertices of G whose depth-ω neighbourhood, dicolored as in (σ_1, σ_2), are isomorphic to $\partial^{\omega}\Gamma$. The following proposition reduces the proof of Theorem 1 to studying the quantity $Q_{\Gamma,\omega}$ under the random replica model. In particular, it holds

▶ **Proposition 4.** *Assume that for any dicolored rooted tree θ and any integer $\omega \geq 0$, we have*

$$Q_{\theta,\omega}(\mathbf{G}, \sigma_1, \sigma_2) \xrightarrow{\mathbb{P}} \mathbb{P}\left[\partial^{\omega}\mathbf{T}^{\otimes}(d, k) \cong \partial^{\omega}\theta\right]. \tag{5}$$

Then (2) *holds for any $\omega \geq 0$, $l \geq 0$.*

The proof of Proposition 4 is based on an averaging argument that generalises a very elegant argument from [22]. More details can be found in Section 5.

4.2 Planting replicas

The above implies that we need to study the random variable $Q_{\theta,\omega}(\mathbf{G},\sigma_1,\sigma_2)$ with $(\mathbf{G},\sigma_1,\sigma_2)$ chosen from the random replica model. It turns out that studying the random replica model is a formidable task. E.g. for half the range of d we consider there is not even a practical way of finding a k-coloring not to mention generating one at random (cf. the discussion in [1]). For this reason we study the random replica model by means of the *planted replica model.* The planted replica model, is a probability distribution that is easy both to implement and to analyse.

For two maps $\sigma_1,\sigma_2 : [n] \to [k]$ let

$$\mathcal{F}(\sigma_1,\sigma_2) = \sum_{i=1}^{k}\left[\binom{|\sigma_1^{-1}(i)|}{2} + \binom{|\sigma_2^{-1}(i)|}{2}\right] - \sum_{i,j=1}^{k}\binom{|\sigma_1^{-1}(i)\cap\sigma_2^{-1}(j)|}{2},$$

i.e. $\mathcal{F}(\sigma_1,\sigma_2)$ is the number of edges of the complete graph on $[n]$ that are monochromatic in at least one of σ_1 or σ_2.

The planted replica model is induced by the following experiment.

PR1 Sample two maps $\hat\sigma_1,\hat\sigma_2 : [n]\to[k]$ independently and uniformly at random subject to the condition that $\mathcal{F}(\hat\sigma_1,\hat\sigma_2) \le \binom{n}{2} - m$.

PR2 Choose a graph $\hat{\mathbf{G}}$ on $[n]$ with precisely m edges uniformly at random, subject to the condition that both $\hat\sigma_1,\hat\sigma_2$ are proper k-colorings of $\hat{\mathbf{G}}$.

The above experiment is easy to get a handle on. Indeed, for large enough n the conditioning in **PR1** is essentially void as $\mathbb{E}[\mathcal{F}(\hat\sigma_1,\hat\sigma_2)] \sim (1-1/k)^2 n$ and $m = O(n)$. In addition, **PR2** amounts to simply choosing a random set of m edges out of the $\binom{n}{2} - \mathcal{F}(\hat\sigma_1,\hat\sigma_2)$ "allowed" edges of the complete graph.

The random replica model and the planted replica model are a priori two different distributions. However, we show that they are closely related in the following sense.

▶ **Proposition 5.** *For any sequence $(\mathcal{A}_n)_n$ of events we have*

$$\lim_{n\to\infty}\mathbb{P}\left[(\hat{\mathbf{G}},\hat\sigma_1,\hat\sigma_2)\in\mathcal{A}_n\right] = 0 \quad\Rightarrow\quad \lim_{n\to\infty}\mathbb{P}\left[(\mathbf{G},\sigma_1,\sigma_2)\in\mathcal{A}_n\right] = 0.$$

In probability jargon, Proposition 5 states that the random replica model is contiguous with respect to the planted replica model. That is, triples (G,σ_1,σ_2) that are typical w.r.t. the first distribution are typical w.r.t. the second distribution, too. The above proposition generalises a result from [6]. For more details about Proposition 5, see Section 6.

4.3 A coupling argument

The following proposition summarises the third and last ingredient to the proof of Theorem 1.

▶ **Proposition 6.** *Let θ be a dicolored rooted tree θ and let $\omega \ge 0$. Then $Q_{\theta,\omega}(\hat{\mathbf{G}},\hat\sigma_1,\hat\sigma_2)$ converges in probability to $\mathbb{P}[\partial^\omega\mathbf{T}^\otimes(k,d)\cong\partial^\omega\theta]$ as $n\to\infty$.*

The proof of Proposition 6 is based on a coupling argument that illustrates how convenient it is to work with the planted replica model. Namely, to prove Proposition 6 it is merely necessary to couple the distribution of the breadth first search tree from a random vertex v in $(\hat{\mathbf{G}},\hat\sigma_1,\hat\sigma_2)$ up to depth ω with the truncated random tree $\partial^\omega\mathbf{T}^\otimes(k,d)$. The coupling is rather immediate from the definitions of these two distributions.

To complete the proof of Theorem 1, we combine Propositions 5 and 6 to conclude that for any $\theta,\omega,$. $Q_{\theta,\omega}(\mathbf{G},\sigma_1,\sigma_2)$ converges in probability to $\mathbb{P}[\partial^\omega\mathbf{T}^\otimes(d,k)\cong\partial^\omega\theta]$. Hence, the assertion follows from Proposition 4.

5 Proof of Proposition 4

Let G be a graph and $Z_k(G) = |\mathcal{S}_k(G)|$. For some $X : \mathcal{S}_k(G) \to \mathbb{R}$, we write

$$\langle X(\sigma)\rangle_G = \frac{1}{Z_k(G)} \sum_{\sigma \in \mathcal{S}_k(G)} X(\sigma).$$

That is, $\langle X(\sigma)\rangle_G$ denotes the expectation of X over the choice of random colorings of G.

Let some integers $l, \omega \geq 0$, let $\theta_1, \ldots, \theta_l$ be rooted trees and let $\tau_1 \in \mathcal{S}_k(\theta_1), \ldots, \tau_l \in \mathcal{S}_k(\theta_l)$. For a graph G, let $\mathcal{U} = \mathcal{U}(G)$ denote the number of vertex sequences $v_1, \ldots, v_l$ such that $\partial^\omega(G, v_i) \cong \theta_i$ for each $i \in [l]$. Let $Y(G)$ denote the set of vertex sequences $v_1, \ldots, v_l$ that in addition to $\partial^\omega(G, v_i) \cong \theta_i$ for each $i \in [l]$, also satisfy

$$\left| \left\langle \prod_{i=1}^{l} \mathbf{1}\{\partial^\omega(G, v_i, \sigma) \cong (\theta_i, \tau_i)\} \right\rangle_G - \prod_{i=1}^{l} Z_k(\theta_i)^{-1} \right| > \delta,$$

for fixed $\delta > 0$. Conditional on the convergence condition in (5) we show that

$$n^{-l}|Y(\mathbf{G})| \xrightarrow{\mathbb{P}} 0. \tag{6}$$

Then, the proposition follows by using (6) and noting that the random graph converges locally to the Galton-Watson tree with offspring distribution Po(d), i.e. w.h.p. we have that

$$n^{-l}|\mathcal{U}| = o(1) + \prod_{i=1}^{l} \mathbb{P}[\mathbf{T}(d) \cong \theta_i].$$

For (6), we extend an argument from [22, Proposition 3.2]. Given a sequence $\varepsilon = \varepsilon(n)$, we let $\mathcal{X}_{\theta_1,\ldots,\theta_l}(G, [l], \omega)$ be the set of all vertex sequences $u_1, \ldots, u_l$ such that $\partial^\omega(G, u_i) \cong \theta_i$ while

$$\left| \left\langle \prod_{i \in [l]} \left(\mathbf{1}\{\partial^\omega(G, u_i, \sigma) \cong (\theta_i, \tau_i)\} - \frac{1}{Z_k(\theta_i)} \right) \right\rangle_G \right| > \varepsilon.$$

Conditional on the convergence assumption in (5), we show that there is a sequence $\varepsilon = \varepsilon(n) = o(1)$ such that w.h.p. (over the graph instances) it holds that $|\mathcal{X}_{\theta_1,\ldots,\theta_l}(\mathbf{G}, [l], \omega)| \leq \varepsilon n^l$. In particular, let $z_i = Z_k(\theta_i)$ and $t_i(v, \sigma) = \mathbf{1}\{\partial^\omega(\mathbf{G}, v, \sigma) \cong (\theta_i, \tau_i)\}$. Moreover, set

$$Q_i(v) = \mathbf{1}\{\partial^\omega(\mathbf{G}, v) \cong \theta_i\} \cdot \left\langle (t_i(v, \sigma_1) - z_i^{-1})(t_i(v, \sigma_2) - z_i^{-1}) \right\rangle_{\mathbf{G}}, \quad Q_i = \frac{1}{n} \sum_{v \in [n]} Q_i(v).$$

The convergence assumption in (5) implies that there exists $\varepsilon = \varepsilon(n) = o(1)$ such that $\sum_{i \in [l]} Q_i \leq \varepsilon^3$. Then fixing an arbitrary $i_0 \in [l]$ we get that

$$\begin{aligned}
\frac{\varepsilon^2}{n^l} |\mathcal{X}_{\theta_1,\ldots,\theta_l}(\mathbf{G}, [l], \omega)| &\leq \frac{1}{n^l} \sum_{u_1,\ldots,u_l \in [n]} \left\langle \prod_{i \in [l]} (t_i(u_i, \sigma) - z_i^{-1}) \right\rangle_{\mathbf{G}}^2 \prod_{i=1}^{l} \mathbf{1}\{\partial^\omega(\mathbf{G}, u_i) \cong \theta_i\} \\
&\leq \frac{1}{n^l} \sum_{u_1,\ldots,u_l \in [n]} \left\langle (t_{i_0}(u_{i_0}, \sigma_1) - z_{i_0}^{-1})(t_{i_0}(u_{i_0}, \sigma_2) - z_{i_0}^{-1}) \right\rangle_{\mathbf{G}} \prod_{i=1}^{l} \mathbf{1}\{\partial^\omega(\mathbf{G}, u_i) \cong \theta_i\} \\
&\qquad \text{[as } \sigma_1, \sigma_2 \text{ are independent]} \\
&\leq \frac{1}{n^l} \sum_{u_1,\ldots,u_l \in [n]} Q_{i_0}(u_{i_0}) = Q_{i_0} \leq \varepsilon^3,
\end{aligned} \tag{7}$$

which implies that w.h.p. $|\mathcal{X}_{\theta_1,\ldots,\theta_l}(\mathbf{G},[l],\omega)| \leq \varepsilon n^l$. Now, we consider the sequences of l vertices which do not belong to $\mathcal{X}_{\theta_1,\ldots,\theta_l}(\mathbf{G},[l],\omega)$, i.e. the majority of the l-tuples. We show that all of them are, somehow, well behaved.

▶ **Claim 7.** *Let $\mathcal{E}_{\theta_1,\ldots,\theta_l}$ be the set of all l-tuples $(v_1,\ldots,v_l)$ of distinct vertices such that $\partial^\omega(\mathbf{G},v_i) \cong \theta_i$ for all $i \in [l]$. Under the assumption in (5) the following is true: There is a number $C > 0$ such that for all $(v_1,\ldots,v_l) \in \mathcal{E}_{\theta_1,\ldots,\theta_l} \setminus \mathcal{X}_{\theta_1,\ldots,\theta_l}(\mathbf{G},[l],\omega)$*

$$\left|\left\langle \prod_{i\in[l]} \mathbf{1}\{\partial^\omega(\mathbf{G},v_i,\sigma) \cong (\theta_i,\tau_i)\}\right\rangle_{\mathbf{G}} - \prod_{i\in[l]} z_i^{-1}\right| \leq C\varepsilon^{1/2}.$$

Then we get (6) from (7) and Claim 7. The proposition follows.

6 Proof of Proposition 5

Before proving the proposition, we consider the following: Given two maps $\sigma_1,\sigma_2 : [n] \to [k]$, we let the *overlap* matrix $\rho(\sigma_1,\sigma_2)$ be a $k\times k$ matrix such that $\rho_{ij}(\sigma_1,\sigma_2) = \frac{1}{n}\left|\sigma_1^{-1}(i) \cap \sigma_2^{-1}(j)\right|$.

▶ **Claim 8.** *Let $\bar\rho$ be the uniform distribution on $[k]^2$. Then, there is $k_0 > 0$ such that for all $k \geq k_0$ and all $d < d_{k,\mathrm{cond}}$, it holds that $\mathbb{E}[\langle\|\rho(\sigma_1,\sigma_2) - \bar\rho\|_F\rangle_{\mathbf{G}}] = o(1)$.*

In words, the above claim asserts that for typical instances of $\mathbf{G}$ the expectation over the choice of the random graph $\mathbf{G}$(the outer $\mathbb{E}$) of the average ℓ_2-distance of the overlap of two randomly chosen k-colorings of $\mathbf{G}$ from $\bar\rho$ goes to 0 as $n \to \infty$. The $d < 2(k-1)\ln(k-1)$ case of Claim 8 was previously proved in [33] by way of the second moment analysis from [4]. As it turns out, the regime $2(k-1)\ln(k-1) < d < d_{k,\mathrm{cond}}$ requires a somewhat more sophisticated argument.

In addition to Claim 8, we need the following concentration result from [6].

▶ **Theorem 9** ([6])**.** *There is $k_0 > 0$ such that for all $k \geq k_0$ and all $d < d_{k,\mathrm{cond}}$ we have*

$$\lim_{\omega\to\infty}\lim_{n\to\infty} \mathbb{P}\left[|\ln Z_k(\mathbf{G}) - \ln \mathbb{E}[Z_k(\mathbf{G})]| \leq \omega\right] = 1.$$

Proof. The proof of the proposition is by contradiction. Assume that $(\mathcal{A}'_n)_{n\geq 1}$ is a sequence of events such that for some fixed number $\varepsilon > 0$ we have

$$\lim_{n\to\infty} \mathbb{P}\left[(\hat{\mathbf{G}},\hat\sigma_1,\hat\sigma_2) \in \mathcal{A}'_n\right] = 0 \quad \text{while} \quad \limsup_{n\to\infty} \mathbb{P}\left[(\mathbf{G},\sigma_1,\sigma_2) \in \mathcal{A}'_n\right] > 2\varepsilon. \tag{8}$$

Setting $\omega(n) = \ln\ln\left(1/\mathbb{P}\left[(\hat{\mathbf{G}},\hat\sigma_1,\hat\sigma_2) \in \mathcal{A}'_n\right]\right)$, we let $\mathcal{B}_n$ be the set of all pairs (σ_1,σ_2) of maps $[n] \to [k]$ such that $\|\rho(\sigma_1,\sigma_2) - \bar\rho\|_2 \leq \sqrt{\omega/n}$. Also we let

$$\mathcal{A}_n = \{(G,\sigma_1,\sigma_2) \in \mathcal{A}'_n : (\sigma_1,\sigma_2) \in \mathcal{B}_n\}.$$

We observe that

$$\omega(n) = \ln\ln\left(1/\mathbb{P}\left[(\hat{\mathbf{G}},\hat\sigma_1,\hat\sigma_2) \in \mathcal{A}_n\right]\right) \to \infty. \tag{9}$$

Then Claim 8 and (8) imply that

$$\lim_{n\to\infty} \mathbb{P}\left[(\hat{\mathbf{G}},\hat\sigma_1,\hat\sigma_2) \in \mathcal{A}_n\right] = 0 \quad \text{while} \quad \limsup_{n\to\infty} \mathbb{P}\left[(\mathbf{G},\sigma_1,\sigma_2) \in \mathcal{A}_n\right] > \varepsilon.$$

The assumption that $\lim_{n\to\infty}\mathbb{P}\left[(\hat{\mathbf{G}},\hat{\sigma}_1,\hat{\sigma}_2)\in\mathcal{A}_n\right]=0$, implies that

$$\begin{aligned}\mathbb{E}[Z_k(\mathbf{G})^2\mathbf{1}\{\mathcal{A}_n\}] &= \sum_{(\sigma_1,\sigma_2)\in\mathcal{B}_n}\mathbb{P}\left[\sigma_1,\sigma_2\in\mathcal{S}_k(\mathbf{G}),(\mathbf{G},\sigma_1,\sigma_2)\in\mathcal{A}_n\right]\\ &= \sum_{(\sigma_1,\sigma_2)\in\mathcal{B}_n}\mathbb{P}\left[(\mathbf{G},\sigma_1,\sigma_2)\in\mathcal{A}_n|\sigma_1,\sigma_2\in\mathcal{S}_k(\mathbf{G})\right]\mathbb{P}\left[\sigma_1,\sigma_2\in\mathcal{S}_k(\mathbf{G})\right]\\ &\le q_n\sum_{(\sigma_1,\sigma_2)\in\mathcal{B}_n}\mathbb{P}\left[(\mathbf{G},\sigma_1,\sigma_2)\in\mathcal{A}_n|\sigma_1,\sigma_2\in\mathcal{S}_k(\mathbf{G})\right],\end{aligned}\tag{10}$$

where $q_n=\max\{\mathbb{P}\left[\sigma_1,\sigma_2\in\mathcal{S}_k(\mathbf{G})\right]:(\sigma_1,\sigma_2)\in\mathcal{B}_n\}$. Using the definition of the planted replica model, (10) writes as follows:

$$\mathbb{E}[Z_k(\mathbf{G})^2\mathbf{1}\{\mathcal{A}_n\}] \le k^{2n}q_n\,\mathbb{P}\left[(\hat{\mathbf{G}},\hat{\sigma}_1,\hat{\sigma}_2)\in\mathcal{A}_n\right].\tag{11}$$

Furthermore, since $\sum_{j=1}^k\rho_{ij}(\sigma_1,\sigma_2)^2,\sum_{i=1}^k\rho_{ij}(\sigma_1,\sigma_2)^2\ge 1/k$ for all $i,j\in[k]$, inclusion/exclusion principle implies that

$$\begin{aligned}\frac{1}{n}\ln\mathbb{P}\left[\sigma_1,\sigma_2\in\mathcal{S}_k(\mathbf{G})\right] &\le \frac{d}{2}\ln\left(1-\frac{2}{k}+\|\rho(\sigma_1,\sigma_2)\|_2^2\right)+O(1/n)\\ &= d\ln(1-1/k)+O(\omega/n) \qquad\qquad \text{for all } (\sigma_1,\sigma_2)\in\mathcal{B}_n.\end{aligned}$$

Hence, $q_n\le(1-1/k)^{2m}\exp(O(\omega))$. Plugging this bound into (11) and setting $\bar{z}=\mathbb{E}[Z_k(\mathbf{G})]$, we get that

$$\begin{aligned}\mathbb{E}[Z_k(\mathbf{G})^2\mathbf{1}\{\mathcal{A}_n\}] &\le k^{2n}(1-1/k)^{2m}\exp(O(\omega))\,\mathbb{P}\left[(\hat{\mathbf{G}},\hat{\sigma}_1,\hat{\sigma}_2)\in\mathcal{A}_n\right]\\ &= \bar{z}^2\exp(O(\omega))\,\mathbb{P}\left[(\hat{\mathbf{G}},\hat{\sigma}_1,\hat{\sigma}_2)\in\mathcal{A}_n\right].\end{aligned}\tag{12}$$

On the other hand, if $\mathbb{P}\left[(\mathbf{G},\sigma_1,\sigma_2)\in\mathcal{A}'_n\right]>\varepsilon$, then Theorem 9 implies that

$$\mathbb{P}\left[(\mathbf{G},\sigma_1,\sigma_2)\in\mathcal{A}'_n\cap\{Z_k(\mathbf{G})\ge\bar{z}/\omega\}\right]>\varepsilon/2.$$

Hence, the distribution of the random replica model yields

$$\mathbb{E}[Z_k(\mathbf{G})^2\mathbf{1}\{\mathcal{A}_n\}]\ge\frac{\varepsilon}{2}\left(\frac{\bar{z}}{\omega}\right)^2.\tag{13}$$

But due to (9), (13) contradicts (12). The proposition follows. ◀

Acknowledgement. We thank Ralph Neininger for helpful discussions.

References

1 Dimitris Achlioptas and Amin Coja-Oghlan. Algorithmic barriers from phase transitions. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 793–802. IEEE, 2008.

2 Dimitris Achlioptas and Ehud Friedgut. A sharp threshold for k-colorability. *Random Structures Algorithms*, 14(1):63–70, 1999.

3 Dimitris Achlioptas and Michael Molloy. The analysis of a list-coloring algorithm on a random graph. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 204–212. IEEE, 1997.

4 Dimitris Achlioptas and Assaf Naor. The two possible values of the chromatic number of a random graph. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 587–593. ACM, 2004.

5 Noga Alon and Michael Krivelevich. The concentration of the chromatic number of random graphs. *Combinatorica*, 17(3):303–313, 1997.

6 Victor Bapst, Amin Coja-Oghlan, and Charilaos Efthymiou. Planting colourings silently. *arXiv preprint arXiv:1411.0610*, 2014.

7 Victor Bapst, Amin Coja-Oghlan, Samuel Hetterich, Felicia Raßmann, and Dan Vilenchik. The condensation phase transition in random graph coloring. *arXiv preprint arXiv:1404.5513*, 2014.

8 Nayantara Bhatnagar, Allan Sly, and Prasad Tetali. Decay of correlations for the hardcore model on the d-regular random graph. *arXiv preprint arXiv:1405.6160*, 2014.

9 Béla Bollobás. The chromatic number of random graphs. *Combinatorica*, 8(1):49–55, 1988.

10 Amin Coja-Oghlan. Upper-bounding the k-colorability threshold by counting covers. *arXiv preprint arXiv:1305.0177*, 2013.

11 Amin Coja-Oghlan and Dan Vilenchik. Chasing the k-colorability threshold. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 380–389. IEEE, 2013.

12 Martin Dyer and Alan Frieze. Randomly coloring random graphs. *Random Structures & Algorithms*, 36(3):251–272, 2010.

13 Martin Dyer, Alistair Sinclair, Eric Vigoda, and Dror Weitz. Mixing in time and space for lattice spin systems: A combinatorial view. *Random Structures & Algorithms*, 24(4):461–479, 2004.

14 Charilaos Efthymiou. MCMC sampling colourings and independent sets of $G(n, d/n)$ near uniqueness threshold. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 305–316. SIAM, 2014.

15 Charilaos Efthymiou. Reconstruction/non-reconstruction thresholds for colourings of general Galton-watson trees. *arXiv preprint arXiv:1406.3617*, 2014.

16 Charilaos Efthymiou. Switching colouring of $G(n, d/n)$ for sampling up to Gibbs uniqueness threshold. In *Algorithms-ESA 2014*, pages 371–381. Springer, 2014.

17 Paul Erdős. Graph theory and probability. *canad. J. Math*, 11:34G38, 1959.

18 Paul Erdős and A Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, 5:17–61, 1960.

19 Andreas Galanis, Qi Ge, Daniel Štefankovič, Eric Vigoda, and Linji Yang. Improved inapproximability results for counting independent sets in the hard-core model. *Random Structures & Algorithms*, 45(1):78–110, 2014.

20 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability for antiferromagnetic spin systems in the tree non-uniqueness region. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 823–831. ACM, 2014.

21 Hans-Otto Georgii. *Gibbs measures and phase transitions*, volume 9. Walter de Gruyter, 2011.

22 Antoine Gerschenfeld and Andrea Montanari. Reconstruction for models on random graphs. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 194–204. IEEE, 2007.

23 Geoffrey R Grimmett and Colin JH McDiarmid. On colouring random graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 77, pages 313–324. Cambridge Univ Press, 1975.

24 Michael Krivelevich and Benny Sudakov. Coloring random graphs. *Information Processing Letters*, 67(2):71–74, 1998.

25 Florent Krzakała, Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *Proceedings of the National Academy of Sciences*, 104(25):10318–10323, 2007.

26 Tomasz Łuczak. The chromatic number of random graphs. *Combinatorica*, 11(1):45–54, 1991.

27 Tomasz Łuczak. A note on the sharp concentration of the chromatic number of random graphs. *Combinatorica*, 11(3):295–297, 1991.

28 Fabio Martinelli, Alistair Sinclair, and Dror Weitz. Fast mixing for independent sets, colorings, and other models on trees. *Random Structures & Algorithms*, 31(2):134–172, 2007.

29 David W Matula. Expose-and-merge exploration and the chromatic number of a random graph. *Combinatorica*, 7(3):275–284, 1987.

30 Marc Mézard and Andrea Montanari. *Information, physics, and computation.* Oxford University Press, 2009.

31 Marc Mézard, Giorgio Parisi, and Riccardo Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002.

32 Michael Molloy. The freezing threshold for k-colourings of a random graph. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 921–930. ACM, 2012.

33 Andrea Montanari, Ricardo Restrepo, and Prasad Tetali. Reconstruction and clustering in random constraint satisfaction problems. *SIAM Journal on Discrete Mathematics*, 25(2):771–808, 2011.

34 Jaroslav Nešetřil. A combinatorial classic—sparse graphs with high chromatic number. In *Erdős Centennial*, pages 383–407. Springer, 2013.

35 Olivier Rivoire, Giulio Biroli, Olivier C Martin, and Marc Mézard. Glass models on bethe lattices. *The European Physical Journal B-Condensed Matter and Complex Systems*, 37(1):55–78, 2004.

36 Eli Shamir and Joel Spencer. Sharp concentration of the chromatic number on random graphs $G(n, p)$. *Combinatorica*, 7(1):121–129, 1987.

37 Allan Sly. Computational transition at the uniqueness threshold. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 287–296. IEEE, 2010.

38 Yitong Yin and Chihao Zhang. Sampling colorings almost uniformly in sparse random graphs. *arXiv preprint arXiv:1503.03351*, 2015.

Towards Resistance Sparsifiers

Michael Dinitz[*1], **Robert Krauthgamer**[†2], **and Tal Wagner**[3]

1 **Johns Hopkins University, Baltimore, MD, USA**
`mdinitz@cs.jhu.edu`
2 **Weizmann Institute of Science, Rehovot, Israel**
`robert.krauthgamer@weizmann.ac.il`
3 **Massachussetts Institute of Technology, Cambridge, MA, USA**
`talw@mit.edu`

Abstract

We study *resistance sparsification* of graphs, in which the goal is to find a sparse subgraph (with reweighted edges) that approximately preserves the effective resistances between every pair of nodes. We show that every dense regular expander admits a $(1+\varepsilon)$-resistance sparsifier of size $\tilde{O}(n/\varepsilon)$, and conjecture this bound holds for all graphs on n nodes. In comparison, spectral sparsification is a strictly stronger notion and requires $\Omega(n/\varepsilon^2)$ edges even on the complete graph.

Our approach leads to the following structural question on graphs: Does every dense regular expander contain a sparse regular expander as a subgraph? Our main technical contribution, which may of independent interest, is a positive answer to this question in a certain setting of parameters. Combining this with a recent result of von Luxburg, Radl, and Hein [16] leads to the aforementioned resistance sparsifiers.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases edge sparsification, spectral sparsifier, graph expansion, effective resistance, commute time

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.738

1 Introduction

Compact representations of discrete structures are of fundamental importance, both from an applications point of view and from a purely mathematical perspective. Graph sparsification is perhaps one of the simplest examples: given a graph $G(V,E)$, is there a subgraph that represents G truthfully, say up to a small approximation? This notion has had different names in different contexts, depending on the property that is being preserved: preserving distances is known as a *graph spanner* [11], preserving the size of cuts is known as a *cut sparsifier* [3], while preserving spectral properties is known as a *spectral sparsifier* [13]. These concepts are known to be related, for example, every spectral sparsifier is clearly also a cut sparsifier, and spectral sparsifiers can be constructed by an appropriate sample of spanners [6].

Our work is concerned with sparsification that preserves *effective resistances*. We define this in Section 1.1, but informally the effective resistance between two nodes u and v is the voltage differential between them when we regard the graph as an electrical network of resistors with one unit of current injected at u and extracted at v. Effective resistances are very useful in many applications that seek to cluster nodes in a network (see [16] and

[*] Work supported in part by NSF award #1464239.
[†] Work supported in part by a US-Israel BSF grant #2010418 and an Israel Science Foundation grant #897/13.

references therein for a comprehensive list), and are also of fundamental mathematical interest. For example, they have deep connections to random walks on graphs (see [10] for an excellent overview of this connection). Most famously, the commute time between two nodes u and v (the expected time for a random walk starting at u to hit v plus the expected time for a random walk starting at v to hit u) is exactly $2m$ times the effective resistance between u and v, where throughout $n := |V|$ and $m := |E|$. Hence, we are concerned with sparsification which preserves commute times.

We ask whether graphs admit a good *resistance sparsifier*: a reweighted subgraph $G'(V, E', w')$ in which the effective resistances are equal, up to a $(1+\varepsilon)$-factor, to those in the original graph. The short answer is yes, because every $(1+\varepsilon)$-spectral sparsifier is also a $(1+\varepsilon)$-resistance sparsifier. Using the spectral-sparsifiers of [2], we immediately conclude that every graph admits a $(1+\varepsilon)$-resistance sparsifier with $O(n/\varepsilon^2)$ edges.

Interestingly, the same $1/\varepsilon^2$ factor loss appears even when we interpret "sparsification" far more broadly. For example, a natural approach to compressing the effective resistances is to use a metric embedding (instead of looking for a subgraph): map the nodes into some metric, and use the metric's distances as our resistance estimates. This approach is particularly attractive since it is well-known that effective resistances form a metric space which embeds isometrically into ℓ_2-squared (i.e., the metric is of negative type, see e.g. [5]). Hence, using the Johnson-Lindenstrauss dimension reduction lemma, we can represent effective resistances up to a distortion of $(1+\varepsilon)$ using vectors of dimension $O(\varepsilon^{-2} \log n)$, i.e., using total space $\tilde{O}(n/\varepsilon^2)$. In fact, this very approach was used by [14] to quickly compute effective resistance estimates, which were then used to construct a spectral sparsifier.

Since a $1/\varepsilon^2$ term appears in both of these natural ways to compactly represent effective resistances, an obvious question is whether this is *necessary.* For the stronger requirement of spectral sparsification, we know the answer is yes – every spectral sparsifier of the complete graph requires $\Omega(n/\varepsilon^2)$ edges [2, Section 4] (see also [1]). However, it is currently unknown whether such a bound holds also for resistance sparsifiers, and the starting point of our work is the observation (based on [16]) that for the complete graph, every $O(1/\varepsilon)$-regular expander is a $(1+\varepsilon)$-resistance sparsifier, despite not being a $(1+\varepsilon)$-spectral sparsifier! We thus put forward the following conjecture.

▶ **Conjecture 1.** *Every graph admits a $(1+\varepsilon)$-resistance sparsifier with $\tilde{O}(n/\varepsilon)$ edges.*

We make the first step in this direction by proving the special case of dense regular expanders (which directly generalize the complete graph). Even this very special case turns out to be nontrivial, and in fact leads us to another beautiful problem which is interesting in its own right.

▶ **Question 2.** *Does every dense regular expander contain a sparse regular expander as a subgraph?*

Our positive answer to this question (for a certain definition of expanders) forms the bulk of our technical work (Sections 2 and 3), and is then used to find good resistance sparsifiers for dense regular expanders (Section 4).

1.1 Results and Techniques

Throughout, we consider undirected graphs, and they are unweighted unless stated otherwise. In a weighted graph, i.e., when edges have nonnegative weights, the *weighted degree* of a vertex is the sum of weights on incident edges, and the graph is considered regular if all of its weighted degrees are equal. Typically, a sparsifying subgraph must be weighted even when the host graph is unweighted, in order to exhibit comparable parameters with far fewer edges.

Before we can state our results we first need to recall some basic definitions from spectral graph theory. Given a weighted graph G, let D be the diagonal $n \times n$ matrix of weighted degrees, and let A be the weighted adjacency matrix. The *Laplacian* of G is defined as $L := D - A$, and the *normalized Laplacian* is the matrix $\hat{L} := D^{-1/2} L D^{-1/2}$.

▶ **Definition 3** (Effective Resistance)**.** Let $G(V, E, w)$ be a weighted graph, and let P the Moore-Penrose pseudo-inverse of its Laplacian matrix. The *effective resistance* (also called *resistance distance*) between two nodes $u, v \in V$ is

$$R_G(u, v) := (e_u - e_v)^T P (e_u - e_v),$$

where e_u and e_v denote the standard basis vectors in $\mathbb{R}^V$ that correspond to u and v respectively.

When the graph G is clear from context we will omit it and write $R(u, v)$. We can now define the main objects that we study.

▶ **Definition 4** (Resistance Sparsifier)**.** Let $G(V, E, w)$ be a weighted graph, and let $\varepsilon \in (0, 1)$. A $(1+\varepsilon)$-*resistance sparsifier* for G is a subgraph $H(V, E', w')$ with reweighted edges such that $(1 - \varepsilon) R_H(u, v) \le R_G(u, v) \le (1 + \varepsilon) R_H(u, v)$, for all $u, v \in V$.

It will turn out that in order to understand resistance sparsifiers, we need to use expansion properties.

▶ **Definition 5** (Graph Expansion)**.** The *edge-expansion* (also known as the *Cheeger constant*) of a weighted graph $G(V, E, w)$ is

$$\phi(G) := \min \left\{ \frac{w(S, \bar{S})}{|S|} : S \subset V,\ 0 < |S| \le |V|/2 \right\},$$

where $w(S, \bar{S})$ denotes the total weight of edges with exactly one endpoint in $S \subset V$. The *spectral expansion* of G, denoted $\lambda_2(G)$, is the second-smallest eigenvalue of the graph's normalized Laplacian.

Our main result is the following. Throughout this paper, "efficiently" means in randomized polynomial time.

▶ **Theorem 6.** *Fix $\beta, \gamma > 0$, let n be sufficiently large, and $1/n^{0.99} < \varepsilon < 1$. Every D-regular graph G on n nodes with $D \ge \beta n$ and $\phi(G) \ge \gamma D$ contains (as a subgraph) a $(1+\varepsilon)$-resistance sparsifier with at most $\varepsilon^{-1} n (\log n)^{O(1/\beta\gamma^2)}$ edges, and it can be found efficiently.*

While dense regular expanders may seem like a simple case, even this special case requires significant technical work. The most obvious idea, of sparsifying through random sampling, does not work — selecting each edge of G uniformly at random with probability $\tilde{O}(1/(D\varepsilon))$ (the right probability for achieving a subgraph with $\tilde{O}(n/\varepsilon)$ edges) need not yield a $(1 + \varepsilon)$-resistance sparsifier. Intuitively, this is because the variance of independent random sampling is too large (see Theorem 26 for the precise effect), and the easiest setting to see this is the case of sparsifying the complete graph. If we sparsify through independent random sampling, then to get a $(1 + \varepsilon)$-resistance sparsifier requires picking each edge independently with probability at least $1/(\varepsilon^2 n)$, and we end up with n/ε^2 edges. To beat this, we need to use correlated sampling. More specifically, it turns out that a random $O(1/\varepsilon)$-regular graph is a $(1+\varepsilon)$-resistance sparsifier of the complete graph, despite not being a $(1+\varepsilon)$-spectral sparsifier. So instead of sampling edges independently (the natural approach, and in fact the approach used to construct spectral sparsifiers by Spielman and Srivastava [14]), we need to sample a random regular graph.

In order to prove Theorem 6, we actually need to generalize this approach beyond the complete graph. But what is the natural generalization of a random regular graph when the graph we start with is not the complete graph? It turns out that what we need is an expander, which is sparse but maintains regularity of its degrees. This motivates our main structural result, that every dense regular expander contains a sparse regular expander (as a subgraph). This can be seen as a type of sparsification result that retains regularity.

▶ **Theorem 7.** *Fix $\beta, \gamma > 0$ and let n be sufficiently large. Every D-regular graph G on n nodes with $D \geq \beta n$ and $\phi(G) \geq \gamma D$ contains a weighted d-regular subgraph H with $d = (\log n)^{O(1/\beta\gamma^2)}$ and $\phi(H) \geq \frac{1}{3}$. All edge weights in H are in $\{1, 2\}$, and H can be found efficiently.*

To prove this theorem, we analyze a modified version of the cut-matching game of Khandekar, Rao, and Vazirani [8]. This game has been used in the past to construct expander graphs, but in order to use it for Theorem 7 we need to generalize beyond matchings, and also show how to turn the graphs it creates (which are not necessarily subgraphs of G) into subgraphs of G.

The expansion requirement for G in Theorem 7 is equivalent to $\lambda_2(G) = \Omega(1)$, when β and γ are viewed as absolute constants. We note that H is a much weaker expander, satisfying only $\lambda_2(H) = \Omega(1/\text{polylog}(n))$, but this is nonetheless sufficient for Theorem 6. Also, H is regular in weighted degrees. For completeness we give a variant of Theorem 7 that achieves an unweighted H by requiring stronger expansion from G, but this is not necessary for our application to resistance sparsifiers, which anyway involves reweighting the edges.

▶ **Theorem 8.** *For every $\beta > 0$ there is $0 < \gamma < 1$ such the following holds for sufficiently large n. Every D-regular graph G on n nodes with $D \geq \beta n$ and $\phi(G) \geq \gamma D$ contains an (unweighted) d-regular subgraph H with $d = (\log n)^{O(1/\beta\gamma)}$ and $\phi(H) \geq \frac{1}{3}$, and it can be found efficiently.*

The algorithm underlying Theorems 6, 7 and 8 turns out to be quite straightforward: decompose the host graph into disjoint perfect matchings or Hamiltonian cycles (which are "atomic" regular components), and subsample a random subset of them of size d to form the target subgraph. However, since the decomposition leads to large dependencies between inclusion of different edges in the subgraph, it is unclear how to approach this algorithm with direct probabilistic analysis. Instead, our analysis uses the adaptive framework of [8] to quantify the effect of gradually adding random matching/cycles from the decomposition to the subgraph.

1.2 Related Work

The line of work most directly related to resistance sparsifiers is the construction of spectral sparsifiers. This was initiated by Spielman and Teng [13], and was later pushed to its limits by Spielman and Teng [15], Spielman and Srivastava [14], and Batson, Spielman, and Srivastava [2], who finally proved that every graph has a $(1+\varepsilon)$-spectral sparsifier with $O(n/\varepsilon^2)$ edges and that this bound is tight (see also [1]).

The approach by Spielman and Srivastava [14] is particularly closely related to our work. They construct almost-optimal spectral sparsifiers (a logarithmic factor worse than [2]) by sampling each edge independently with probability proportional to the effective resistance between the endpoints. This method naturally leads us to try the same thing for resistance sparsification, but as discussed, independent random sampling (even based on the effective resistances) cannot give improved resistance sparsifiers. Interestingly, in order to make their algorithm extremely efficient they needed a way to estimate effective resistances very quickly, so along the way they showed how to create a sketch of size $O(n \log n/\varepsilon^2)$ from which every resistance distance can be read off in $O(\log n)$ time (essentially through an ℓ_2-squared embedding and a Johnson-Lindenstrauss dimension reduction).

2 Sparse Regular Expanding Subgraphs

In this section we prove Theorem 7, building towards it in stages. Our starting point is the Cut-Matching game of Khandekar, Rao and Vazirani (KRV) [8], which is a framework to constructing sparse expanders by iteratively adding perfect matchings across adaptively chosen bisections of the vertex set. The resulting graph H is regular, as it is the union of perfect matchings, and if the matchings are contained in the input graph G then H is furthermore a subgraph of G, as desired. In Section 2.1, we employ this approach to prove Theorem 7 in the case $D/n = \frac{3}{4} + \Omega(1)$.

To handle smaller D, we observe that the perfect matchings in the KRV game can be replaced with a more general structure that we call a *weave*, defined as a set of edges where for every vertex at least one incident edge crosses the given bisection. To ensure that H is regular (all vertices have the same degree), we would like the weaves to be regular. We thus decompose the input graph to disjoint regular elements – either perfect matchings or Hamiltonian cycles – and use them as building blocks to construct regular weaves. Leveraging the fact that for some bisections, G contains no perfect matching but does contain a weave, we use this extension in Section 2.2 to handle the case $D/n = \frac{1}{2} + \Omega(1)$.

Finally, for the general case $D/n = \Omega(1)$, we need to handle a graph G that contains no weave on some bisections. The main portion of our proof constructs a weave that is not contained in G, but rather embeds in G with small (polylogarithmic) congestion. Repeating this step sufficiently many times as required by the KRV game, yields a subgraph H as desired.

Notation and terminology

For a regular graph G, we denote $\deg(G)$ the degree of each vertex. We say that a graph H is an *edge-expander* if $\phi(H) > \frac{1}{3}$. A *bisection* of a vertex set of size n is a partition $(S, \bar{S})$ with equal sizes $\frac{1}{2}n$ if n is even, or with sizes $\lfloor \frac{1}{2}n \rfloor$ and $\lceil \frac{1}{2}n \rceil$ if n is odd.

2.1 The Cut-Matching Game

Khandekar, Rao and Vazirani [8] described the following game between two players. Start with an empty graph (no edges) H on a vertex set of even size n. In each round, the *cut player* chooses a bisection, and the *matching player* answers with a perfect matching across the bisection. The game ends when H is an edge-expander. Informally, the goal of the cut player is to reach this as soon as possible, and that of the matching player is to delay the game's ending.

▶ **Theorem 9** ([8, 7]). *The cut player has an efficiently computable strategy that wins (i.e., is guaranteed to end the game) within $O(\log^2 n)$ rounds, and a non-efficient strategy that wins within $O(\log n)$ rounds.*

The following result illustrates the use of the KRV framework in our setting.

▶ **Theorem 10.** *Let $\delta > 0$ and let n be even and sufficiently large ($n \geq n_0(\delta)$). Then every n-vertex graph $G(V, E)$ with minimum degree $D \geq (\frac{3}{4} + \delta)n$ contains an edge-expander H that is d-regular for $d = O(\log n)$, and also an efficiently computable edge-expander H' that is a d'-regular for $d' = O(\log^2 n)$.*

Proof. Apply the Cut-Matching game on V with the following player strategies. For the cut player, execute the efficient strategy from Theorem 9 that wins within $O(\log^2 n)$ rounds. For the matching player, given a bisection $(S, \bar{S})$, consider the bipartite subgraph $G[S, \bar{S}]$ of G induced by $(S, \bar{S})$. Each vertex in S has in G at least $D \geq \frac{3}{4}n$ neighbors, but at most $\frac{1}{2}n - 1$ of them are in S, and the rest must be in $\bar{S}$, which implies that $G[S, \bar{S}]$ has minimum degree $\geq \frac{1}{4}n$. Hence, as a simple consequence of Hall's theorem (see Proposition 29), it contains a perfect matching that can be efficiently found. The matching player returns this matching as his answer. We then remove this matching from G before proceeding to the next round, to ensure that different iterations find disjoint matchings. The slackness parameter δ (and n being sufficiently large) ensure that the minimum degree of G does not fall below $\frac{3}{4}n$ during the $O(\log^2 n)$ iterations, so the above argument holds in all rounds.

The game ends with an edge-expander H' which is a disjoint union of $d' = O(\log^2 n)$ perfect matchings contained in G, and hence is a d'-regular subgraph of G, as required. To obtain the graph H, apply the same reasoning but using the non-efficient strategy from Theorem 9 that wins within $O(\log n)$ rounds. ◀

2.2 The Cut-Weave Game

For values of D below $\frac{3}{4}n$, we can no longer guarantee that every bisection in G admits a perfect matching. However, we observe that one can allow the matching player a wider range of strategies while retaining the ability of the cut player to win within a small number of rounds.

▶ **Definition 11** (weave)**.** Given a bisection $(S, \bar{S})$ of a vertex set V, a *weave* on $(S, \bar{S})$ is a subgraph in which every node has an incident edge crossing $(S, \bar{S})$.

▶ **Definition 12** (Cut-Weave Game)**.** The *Cut-Weave game* with parameter r is the following game of two players. Start with a graph H on a vertex set of size n and no edges. In each round, the *cut player* chooses a bisection of the vertex set, and the *weave player* answers with an r-regular weave on the bisection. The edges of the weave are added to H.

Note that the $r = 1$ case is the original Cut-Matching game (when n is even). The following theorem is an extension of Theorem 9. For clarity of presentation, its proof is deferred to Section 3.

▶ **Theorem 13.** *In the Cut-Weave game with parameter r, the cut player has an efficient strategy that wins within $O(r \log^2 n)$ rounds, and furthermore ensures $\phi(H) \geq \frac{1}{2}r$.*

In order to construct regular weaves, we employ a decomposition of G into disjoint Hamiltonian cycles. The following theorem was proven by Perkovic and Reed [12], and recently extended by Csaba, Kühn, Lo, Osthus and Treglown [4].

▶ **Theorem 14.** *Let $\delta > 0$. Every D-regular graph G on n nodes with $D \geq (\frac{1}{2} + \delta)n$, admits a decomposition of its edges into $\lfloor \frac{1}{2}D \rfloor$ Hamiltonian cycles and possibly one perfect matching (if D is odd). Furthermore, the decomposition can be found efficiently.*

Now we can use the Cut-Weave framework to make another step towards Theorem 7.

▶ **Theorem 15.** *Let $\delta > 0$ and let n be sufficiently large. Then every n-vertex graph $G(V, E)$ with minimum degree $D \geq (\frac{1}{2} + \delta)n$ contains a d-regular edge-expander H with $d = O(\log^3 n)$, which furthermore can be efficiently found.*

Proof. We simulate the Cut-Weave game with $r = 16\delta^{-1} \log n$. The proof is the same as Theorem 10, only instead of a perfect matching we need to construct an r-regular weave across a given bisection $(S, \bar{S})$. We apply Theorem 14 to obtain a Hamiltonian decomposition of G. For simplicity, if D is odd we discard the one perfect matching from Theorem 14. Let $\mathcal{C}$ be the collection of Hamiltonian cycles in the decomposition.

Suppose w.l.o.g. $|S| = \lceil \frac{1}{2} n \rceil$. Every $v \in S$ has at most $|S| - 1 \leq \frac{1}{2} n$ neighbors in S, and hence at least δn incident edges crossing to $\bar{S}$. We set up a Set-Cover instance of the cycles $\mathcal{C}$ against the nodes in S, where a node v is considered covered by a cycle C is v has an incident edge crossing to $\bar{S}$, that belongs to C. This is a dense instance: since each cycle visits v only twice, v can be covered by $\frac{1}{2}\delta n$ cycles. Therefore, $4\delta^{-1} \log n$ randomly chosen cycles form a cover with high probability (see Proposition 30 for details). We then repeat the same procedure to cover the nodes on side $\bar{S}$. The result is a collection of $8\delta^{-1} \log n = \frac{1}{2} r$ disjoint Hamiltonian cycles, whose union forms an r-regular weave on $(S, \bar{S})$, which we return as the answer of the weave player. Applying Theorem 13 with $r = O(\log n)$ concludes the proof of Theorem 15. ◀

Observe that in the proof of Theorem 15, the weave player is in fact oblivious to the queries of the cut player: all she does is sample random cycles from $\mathcal{C}$, and the output subgraph H is the union of those cycles. Therefore, in order to construct H, it is sufficient to decompose G into disjoint Hamiltonian cycles, and choose a random subset of size $O(\log^3 n)$ of them. There is no need to actually simulate the cut player, and in particular, the proof does not require her strategy (from Theorem 13) to be efficient.

2.3 Reduction to Double Cover

We now begin to address the full range of parameters stated in Theorem 7. In this range there is no Hamiltonian decomposition theorem (or a result of similar flavor) that we are aware of, so we replace it with a basic argument which incurs edge weights $w : V \times V \to \{0, 1, 2\}$ in the target subgraph H, as well as a loss in its degree.

Given the input graph $G(V, E)$, we construct its *double cover*, which is the bipartite graph $G''(V'', E'')$ defined by $V'' = V \times \{0, 1\}$ and $E'' = \{((v, 0)(u, 1)) : vu \in E\}$. It is easily seen that if G is D-regular then so is G'', and since $|V''| = 2|V|$ we have $D \geq \frac{1}{2}\beta|V''|$. It also well known that $\lambda_2(G) = \lambda_2(G'')$, and therefore by the discrete Cheeger inequalities,

$$\phi(G'') \geq \tfrac{1}{2}\lambda_2(G'')D = \tfrac{1}{2}\lambda_2(G)D \geq \tfrac{1}{2}\gamma^2(G)D.$$

G'' satisfies the requirements of Theorem 7 with $\beta'' = \frac{1}{2}\beta$ and $\gamma'' = \frac{1}{2}\gamma^2$. Suppose we find in G'' a d-regular edge-expander H'' with $d = (\log n)^{O(1/\beta''\gamma'')} = (\log n)^{O(1/\beta\gamma^2)}$. We carry it over to a subgraph H of G, by including each edge $uv \in E$ in H with weight $|\{(v, 0)(u, 1), (u, 0)(v, 1)\} \cap E(H'')|$, where $E(H'')$ denotes the edge set of H''. Each edge then appears in H with weight either 1 or 2 (or 0, which means it is not present in H). It can be easily checked that H is d-regular in weighted degrees, and $\phi(H) \geq \frac{1}{2}\phi(H'')$. Therefore H is a suitable target subgraph for Theorem 7.

The above reduction allows us to restrict our attention to regular bipartite graphs G, but on the other hand we are forced to look for a subgraph H which is unweighted and d-regular with $d = (\log n)^{O(1/\beta\gamma)}$ (which is tighter than stated in Theorem 7). We take this approach in the remainder of the proof. The gain is that such G admits a decomposition into disjoint perfect matchings, which can be efficiently found, as a direct consequence of Hall's theorem. We will use this fact where we have previously used Theorem 14.

2.4 Constructing an Embedded Weave

We now get to the main technical part of the proof. Given a bisection $(S, \bar{S})$ queried by the cut player, we need to construct an r-regular weave on the bisection, where this time we choose $r = (\log n)^{O(1/\beta\gamma)}$. Unlike the proof of Theorem 15, we cannot hope to find a weave which is a subgraph of G, since if $D < \frac{1}{2}n$, any bisection in which one side contains some vertex and all its neighbors would not admit a weave in G. Instead, we aim for a weave which embeds into G with polylogarithmic congestion.

We will use two types of graph operations: The *union* of two graphs on the same vertex set V is obtained by simply taking the set union of their edge sets, whereas the *sum* of the two graphs is given by keeping parallel edges if they appear in both graphs. We now construct the weave in 4 steps.

Step 1

Fix $\mu = \frac{\beta\gamma^2}{4}$. We partition the entire vertex set V into subsets $S_0, S_1, \ldots, S_t$ by the following process:

1. Set $S_0 \leftarrow \bar{S}$ and $T \leftarrow S$.
2. While $T \neq \emptyset$, take $S_i \subseteq T$ to be the subset of nodes with at least μD neighbors in S_{i-1}, and set $T \rightarrow T \setminus S_i$.

▶ **Lemma 16.** *The process terminates after* $t \leq \frac{2}{\beta\gamma}$ *iterations.*

Proof. Consider an iteration $i \leq \frac{2}{\beta\gamma}$ that ends with $T \neq \emptyset$. Denote $\bar{T} = V \setminus T = \cup_{j=0}^{i} S_j$. By the hypothesis $\phi(G) \geq \gamma D$ we have at least $\gamma D|T|$ edges crossing from T to $\bar{T}$, so by averaging over the nodes in T, there is $v \in T$ with γD neighbors in $\bar{T}$. For every $j < i$, v must have less than μD neighbors in S_j, or it would already belong to $S_{j+1} \subseteq \bar{T}$. Summing over $j = 0, \ldots, i-1$, we see that v has less than $i\mu D \leq \frac{1}{2}\gamma D$ neighbors in $\bar{T} \setminus S_i$, so at least $\frac{1}{2}\gamma D$ neighbors in S_i. This implies $|S_i| \geq \frac{1}{2}\gamma D$. We have shown that each of the first $\frac{2}{\beta\gamma}$ iterations either terminates the process or removes $\frac{1}{2}\gamma D \geq \frac{1}{2}\gamma\beta n$ nodes from T, so after $\frac{2}{\beta\gamma}$ iterations we must have $T = \emptyset$. ◀

Step 2

By Section 2.3 we have a decomposition of all the edges in G into a collection $\mathcal{M}$ of D disjoint perfect matchings. For every $i = 1, \ldots, t$, we now cover the nodes in S_i with perfect matchings, similar to the proof of Theorem 15. A node $v \in S_i$ is considered covered by a matching if v has an incident edge with the other endpoint in S_{i-1}, and that edge lies on the matching. Since v has μD incident edges crossing to S_{i-1}, and each matching touches v with at most one edge, we have μD matchings that can cover v. Therefore $k = \frac{1}{\mu}\log n$ randomly chosen matchings from $\mathcal{M}$ form a cover of S_i (see Proposition 30), which we denote as K_i. Thus, for each i we have a subgraph K_i which is k-regular, such that each node in S_i has an incident edge in K_i with the other endpoint in S_{i-1}. Denote henceforth

$$K = \cup_{i=1}^{t} K_i.$$

Note that K is a regular subgraph of G, since it is a union of disjoint perfect matchings from $\mathcal{M}$, and $\deg(K) \leq kt$.

Step 3

In this step we construct a graph K^* from the subgraph K. As discussed, K^* will not be a subgraph of G but will embed into it with reasonable congestion. Let us formally define the notion of graph embedding that we will be using.

▶ **Definition 17** (Graph embedding with congestion). Let $G(V, E)$ and $G'(V, E')$ be graphs on the same vertex set. Denote by $\mathcal{P}_G$ the set of simple paths in G. An *embedding* of G' into G is a map $f : E' \to \mathcal{P}_G$ such that every edge in G' is mapped to a path in G with the same endpoints.

The *congestion* of f on an edge $e \in E$ is $\operatorname{cng}_f(e) := |e' \in E' : e \in f(e')|$. The congestion of f is $\operatorname{cng}(f) := \max_{e\in E} \operatorname{cng}_f(e)$. We say that G' embeds into G with congestion c if there is an embedding f with $\operatorname{cng}(f) = c$.

The following claim is a simple observation and we omit its proof.

▶ **Claim 18.** *If G' embeds into G with congestion c, then $\phi(G) \geq \frac{1}{c}\phi(G')$.*

We generate K^* with the following inductive construction.

▶ **Lemma 19.** *Let $\rho_0 = c_0 = 0$. We can efficiently construct subgraphs $K_1^*, \dots, K_t^*$ (which may have parallel edges and self-loops), such that for every $i = 1, \dots, t$,*

1. *K_i^* is ρ_i-regular, where $\rho_i = k(1 + \rho_{i-1})$.*
2. *K_i^* embeds into K with congestion c_i, where $c_i = 1 + kc_{i-1}$.*
3. *Every $v \in S_i$ has an incident edge in K_i^* with the other endpoint in S_0.*

Proof. We go by induction on i. For the base case $i = 1$ we simply set $K_1^* = K_1$. The claim holds as we recall that

1. K_1 is k-regular.
2. K_1 is a subgraph of K, hence it embeds into K with congestion $1 = 1 + kc_0$.
3. By Step 2, every $v \in S_1$ has an incident edge in K_1 crossing to S_0.

We turn to the inductive step $i > 1$. Start with a graph K' which is a fresh copy of K_{i-1}^*, with each edge duplicated into k parallel edges. By induction, K' is $(k\rho_{i-1})$-regular. Now sum K_i into K'; recall this means keeping parallel edges instead of unifying them. Since K_i is k-regular, K' is ρ_i-regular.

Let $v \in S_i$. By Step 2, there is an edge $vw \in K_i$ such that $w \in S_{i-1}$. By induction, there is an edge $wu \in K_{i-1}^*$ such that $u \in S_0$. Note that both edges vw and wu are present in K'. Perform the following crossing operation on K': Remove the edges vw and wu, and add an edge vu and a self-loop on w.

Perform this on every $v \in S_i$. The resulting graph is K_i^*. We need to show that it is well defined in the following sense: we might be using the same edge wu for several v's, and we need to make sure each wu appears sufficiently many times, to be removed in all the crossing operations in which it is needed. Indeed, we recall that K_i is the union of k disjoint perfect matchings, and therefore each $w \in S_{i-1}$ has at most k edges in K_i incoming from S_i. Since K' contains k copies of each edge wu, we have enough copies to be removed in all necessary crossing operations.

Lastly we show that K_i^* satisfies all the required properties.

1. Since K' was ρ_i-regular, and the switching operations do not effect vertex degrees, we see that K_i^* is ρ_i-regular.
2. Each edge vu in K_i^* which is not original from K', corresponds to a path (of length 2) in K' that was removed upon adding that edge; hence K_i^* embeds into K' with congestion 1. K' is the sum of K_i, which is a subgraph of K, and k copies of K_{i-1}^*, which by induction embeds into K with congestion c_{i-1}. Hence K' embeds into K with congestion $1 + kc_{i-1} = c_i$. Therefore, K_i^* embeds into K with congestion c_i.
3. For every $v \in S_i$, we added to K_i^* an edge vu such that $u \in S_0$. ◀

We now take $K^* = \sum_{i=1}^t K_i^*$. By Claim 19, K^* is $(\sum_{i=1}^t \rho_i)$-regular, embeds into K with congestion $\sum_{i=1}^t c_i$, and every $v \in S$ has an incident edge $vu \in K^*$ such that $u \in \bar{S}$. (To see why the latter point holds, recall that we put $\bar{S} = S_0$.)

Step 4

In this final step we repeat Steps 1–3, only with the roles of S and $\bar{S}$ interchanged. This results in a subgraph $\bar{K}$ of G which is kt-regular, and a graph $\bar{K}^*$ which is $(\sum_{i=1}^t \rho_i)$-regular, embeds into $\bar{K}$ with congestion $\sum_{i=1}^t c_i$, and every $v \in \bar{S}$ has an incident edge $vu \in \bar{K}^*$ such that $u \in S$.

Our final weave is $K^* + \bar{K}^*$. By the above it is clearly a weave, and moreover it is r-regular and embeds into $K \cup \bar{K}$ (and hence into G, which contains $K \cup \bar{K}$) with congestion c, where $r = 2\sum_{i=1}^t \rho_i$ and $c = 2\sum_{i=1}^t c_i$. By inspecting the recurrence formulas from Claim 19, in which ρ_i and c_i were defined, we can bound $\rho_i, c_i \leq (2k)^i \leq (2k)^t$ for every i, and hence $r, c \leq 2t(2k)^t$. Recalling that $t \leq \frac{2}{\beta\gamma} + 1$ and $k = \frac{1}{\mu}\log n = O(\log n)$, we find $r, c \leq (\log n)^{O(1/\beta\gamma)}$.

2.5 Completing the Proof of Theorem 7

We play the Cut-Weave game for L rounds, where $L = O(r\log^2 n)$ is the number of rounds required by the efficient strategy in Theorem 13. For each round $\ell = 1, \ldots, L$, we constructed above an r-regular weave $W_\ell^* = K^* + \bar{K}^*$, that embeds into a subgraph $W_\ell = K \cup \bar{K}$ of G with congestion c. Let $H = \cup_{\ell=1}^L W_\ell$ and $H^* = \sum_{1=\ell}^L W_\ell^*$. Then H is a union of disjoint perfect matchings from $\mathcal{M}$, and hence regular. Moreover $\deg(H) \leq 2ktL$, since H is the union of L subgraphs $\{W_\ell\}_{\ell=1}^L$, where each W_ℓ is a union W_ℓ of two kt-regular graphs $K, \bar{K}$.

Now consider H^*. Since each W_ℓ^* embeds into W_ℓ with congestion c, we see that H^* embeds into H with congestion (at most) cL. By Theorem 13 we have $\phi(H^*) \geq \frac{1}{2}r$, and this now implies $\phi(H) \geq \frac{r}{2cL}$.

Recalling the parameters:

$$t = O(1) \;\; ; \;\; k = O(\log n) \;\; ; \;\; r, c = O(\log^{O(1/\beta\gamma)} n) \;\; ; \;\; L = O(r\log^2 n),$$

we see that H is a d-regular subgraph of $d = (\log n)^{O(1/\beta\gamma)}$ and $\phi(H) \geq 1/(\log n)^{O(1/\beta\gamma)}$. We can now repeat this Cut-Weave game $(\log n)^{O(1/\beta\gamma)}$ disjoint times, because if each time we remove the graph H we have found, we decrease the degree $D = \beta n$ of each node by only polylog (n). By repeating the game this many times and taking the union of the disjoint resulting subgraphs, we find a regular subgraph H of G with $\deg(H) = (\log n)^{O(1/\beta\gamma)}$ and $\phi(H) \geq 1$. Lastly recall that unfolding the reduction from Section 2.3 puts on H edge weight in $\{1, 2\}$, and weakens the degree bound to $\deg(H) = (\log n)^{O(1/\beta\gamma^2)}$. This completes the proof of Theorem 7.

Regarding the algorithm to construct H, the observation made after Theorem 15 applies here as well. The weave player's strategy is oblivious to the queries of the cut player, since she just samples random matchings from $\mathcal{M}$ to form H. The cut player strategy does not actually need to be simulated, nor the graphs K^* need to actually be constructed. The algorithm to construct H then amounts to the following: Construct the double cover graph G" of G; decompose G" into disjoint perfect matchings; choose a random subset of $(\log n)^{O(1/\beta\gamma^2)}$ of them to form a subgraph H" of G"; and unfold the double cover construction to obtain the final subgraph H from H".

2.6 Proof of Theorem 8

The theorem follows from replacing the reduction to the double cover in Section 2.3 by a Hamiltonian decomposition result that holds for this stronger expansion requirement, due to Kühn and Osthus [9, Theorem 1.11]. The trade-off between β and γ is inherited from their theorem (in which it is unspecified). Circumventing Section 2.3 also improves the dependence of d on γ. The proof of Theorem 8 is otherwise identical to the proof of Theorem 7.

3 Proof of the Cut-Weave Theorem

Recall the setting of the Cut-Weave game with parameter r: The game starts with a graph G_0 on n vertices and without edges. In each round $t = 1, 2, \dots$, the weave player queries a bisection of the vertex set, and the weave player answers with an r-regular weave H_t on that bisection. The weave is then unified into the graph, putting $G_t = G_{t-1} \cup H_t$.

We now prove Theorem 13 by an adaptation of the analysis from [8]. The main change is in Lemma 25.

For each step t, let M_t be the matrix describing one step of the natural lazy random walk on H_t: W.p. $\frac{1}{2}$ stay in the current vertex, and with probability $\frac{1}{2r}$ move to a neighbor. The cut player strategy is as follows:

- Choose a random unit vector $z \perp \mathbf{1}$ in $\mathbb{R}^n$.
- Compute $u = M_t M_{t-1} \dots M_1 z$.
- Output the bisection $(S, \dots S)$ where S is the $\lfloor n/2 \rfloor$ vertices with smallest values in u.

Let us analyze the game with this strategy. In the graph G_t (which equals $\cup_{t'=1}^{t} H_{t'}$), we consider the following t-steps random walk: Take one (lazy) step on H_1, then on H_2, and so on until H_t. In other words, the walk is given by applying sequentially M_1, then M_2, and so on.

Let $P_{ij}(t)$ denote the probability to go from node j to node i within t steps. Let P_i denote the vector $(P_{i1}, P_{i2}, \dots, P_{ji})$. We use the following potential function:

$$\Psi(t) = \sum_{i,j \in V} (P_{ij} - 1/n)^2 = \sum_{i=1}^{n} \|P_i - \mathbf{1}/n\|_2^2.$$

▶ **Lemma 20.** *For every t and every $i \in V$, we have $\sum_{j \in V} P_{ij}(t) = 1$.*

Proof. By induction on t: It holds initially, and in each step t, vertex i trades exactly half of its total present probability with its neighbors in H_t. (Note that this relies on the fact that H_t is regular.) ◀

▶ **Lemma 21.** *If $\Psi(t) < 1/4n^2$ then $G = G_t$ has edge-expansion at least $\frac{1}{2}r$.*

Proof. If $\Psi(t) < 1/4n^2$ then $P_{ji}(t) \geq \frac{1}{2n}$ for all $i, j \in V$. Hence the graph K_t on V, in which each edge ij has weight $P_{ji}(t) + P_{ij}(t)$, has edge-expansion $\frac{1}{2}$. We finish by showing that K_t embeds into G_t with congestion $1/r$. Proof by induction: Consider the transition from

G_{t-1} to G_t, which is unifying H_t into G_{t-1}. Let $i, j \in V$ be connected with an edge in H_t, and let k be any vertex. In the transition from K_{t-1} to K_t, we need to ship $\frac{1}{2r}$ of the type-k probability in i (namely $\frac{1}{2r}P_{ik}$) to j, and similarly, ship $\frac{1}{2r}P_{jk}$ probability from j to i. (The "type-k" probabiility is probability mass that was originally located in k.) In total, we need to ship $\frac{1}{2r}\sum_{k\in V} P_{ik} = \frac{1}{2r}$ from i to j and a similar amount from j to i. In total the edge ij in H_t needs to support $\frac{1}{r}$ flow (of probability) in the transition, so the claim follows. ◀

We turn to analyzing the change in potential in a single fixed round t. To simplify notation we let

$$P_{ji} = P_{ji}(t) \quad ; \quad Q_{ji} = P_{ji}(t+1).$$

Moreover recall we have a vector u generated by the cut player in the current round:

$$u = M_t M_{t-1} \dots M_1 z.$$

Denote its entries by $u_1, \dots, u_n$. We are now adding the graph H_{t+1} to G_t to produce G_{t+1}.

▶ **Lemma 22.** *For every i, u_i is the projection of P_i on r, i.e. $u_i = P_i^T z$.*

Proof. Fix i. Abbreviate $M = M_t M_{t-1} \dots M_1(\frac{1}{n}\mathbf{1})$. If ϕ is any distribution on the vertices then $P_i^T \phi$ is the probability that the random walk lands in vertex i after t steps, meaning

$$(M\phi)_i = P_i^T \phi. \tag{1}$$

Let $z' = \frac{1}{n\|z\|_\infty} z$. Applying (1) with $\phi = z' + \frac{1}{n}\mathbf{1}$ gives $(M(z' + \frac{1}{n}\mathbf{1}))_i = P_i^T(z' + \frac{1}{n}\mathbf{1})$. Applying (1) again with $\phi = \frac{1}{n}\mathbf{1}$ gives $(M\frac{1}{n}\mathbf{1})_i = P_i^T(\frac{1}{n}\mathbf{1})$ and together we get $(Mz')_i = P_i^T z'$, which implies $u_i = (Mz)_i = P_i^T z$. ◀

▶ **Lemma 23.** *With probability $1 - 1/n^{\Omega(1)}$ over the choice of z, for all pairs $i, j \in V$,*

$$\|P_i - P_j\|_2^2 \geq \frac{n-1}{C \log n} |u_i - u_j|^2.$$

Proof. Similar to [8, Lemma 3.4]. ◀

▶ **Lemma 24.** *Let $E(S, \bar{S})$ denote the set of edges in H_{t+1} that cross the bisection $(S, \bar{S})$ produced by the cut player (from the vector u). Then,*

$$(n-1)\mathbb{E}\left[\sum_{ij \in E(S,\bar{S})} |u_i - u_j|^2\right] \geq \Psi(t).$$

Proof. Denote by $\deg_{(S,\bar{S})}(i)$ the number of edges in $E(S, \bar{S})$ incident to vertex i. Note that $\deg_{(S,\bar{S})}(i) \geq 1$ for every $i \in V$, since H_{t+1} is a weave on $(S, \bar{S})$. Recall that S contains the vertices with smallest entries in u. Hence there is a number $\eta \in \mathbb{R}$ such that $i \leq \eta \leq j$ for each edge $ij \in E(S,\bar{S})$. Hence,

$$\begin{aligned}\sum_{ij\in E(S,\bar{S})} |u_i-u_j|^2 &\geq \sum_{ij\in E(S,\bar{S})} ((u_i-\eta)^2+(\eta-u_j)^2) \\ &= \sum_{i\in V} \deg_{(S,\bar{S})}(i)(u_i-\eta)^2 \\ &\geq \sum_{i\in V}(u_i-\eta)^2 \\ &= \sum_{i\in V} u_i^2 - 2\eta\sum_{i\in V} u_i + n\eta^2 \\ &\geq \sum_{i\in V} u_i^2,\end{aligned}$$

where the last equality is by noting that $z \perp \mathbf{1}$, hence $u \perp \mathbf{1}$, hence $\sum_i u_i = 0$.

Next, since $u_i = P_i^T z$ and $z \perp \mathbf{1}$ we have $u_i = (P_i - \mathbf{1}/n)^T z$. Hence u_i is the projection of $P_i - \mathbf{1}/n$ on z. By properties of random projections we have $\mathbb{E}[u_i^2] = \frac{1}{n-1}\|P_i - \mathbf{1}/n\|_2^2$ (see details in [8]), hence

$$\mathbb{E}\left[\sum_{i\in V} u_i^2\right] = \frac{1}{n-1}\sum_{i\in V}\|P_i-\mathbf{1}/n\|_2^2 = \frac{1}{n-1}\Psi(t),$$

and the lemma follows from combining this with the above. ◀

▶ **Lemma 25.** *Let E_{t+1} denote the edge set of H_{t+1}. The potential reduction is*

$$\Psi(t) - \Psi(t+1) = \frac{1}{r}\sum_{ij\in E_{t+1}} \|P_i - P_j\|_2^2.$$

Proof. We construct from G a graph G' by splitting each vertex i into r copies $i_1,\ldots,i_r$, assigning arbitrarily one edge from the r edges incident to i in E_{t+1} to the copies, and distributing the type-j probability in i, for each j, evenly among the copies. We denote by P_{ji_k} the amount of type-j probability on i_k before adding E_{t+1} to G', and by Q_{ji_k} the type-j probability in i after adding E_{t+1}. Note that we have defined $P_{ji_k} = \frac{1}{r}P_{ji}$ for all $i,j \in V$ and $k \in [r]$, but for the Q_{ji_k}'s all we know is that $\sum_{k=1}^r Q_{ji_k} = Q_{ji}$, so Q_{ji} may be distributed arbitrarily among the Q_{ji_k}'s. As usual P_{i_k} denotes the vector with entries P_{ji_k}, and Q_{i_k} is defined similarly.

Define the potential of G' as:

$$\Psi'(t) = \sum_{i\in V}\sum_{k=1}^r \|P_{i_k} - \mathbf{1}/nr\|_2^2.$$

We thus have

$$\Psi(t) = \sum_{i\in V}\|P_i-\mathbf{1}/n\|_2^2 = r\sum_{k=1}^r\sum_{i\in V}\|\frac{1}{r}P_i - \mathbf{1}/nr\|_2^2 = r\sum_{k=1}^r\sum_{i\in V}\|P_{i_k}-\mathbf{1}/nr\|_2^2 = r\Psi'(t).$$

To relate $\Psi(t+1)$ to $\Psi'(t+1)$, we use the general fact that for any constants c and X, the solution to $\min\|x - c\mathbf{1}\|$ s.t. $x \in \mathbb{R}^r$, $\sum_i x_i = X$ is attained on $x = \frac{X}{r}\mathbf{1}$. Since we have

$\sum_{k=1}^{r} Q_{ji_k} = Q_{ji}$ for all i, j, we infer

$$\begin{aligned}\Psi(t+1) &= \sum_{i\in V} \|Q_i - \mathbf{1}/n\|_2^2 \\ &= \sum_{i,j\in V} (Q_{ji} - 1/n)^2 \\ &= \sum_{i,j\in V} r \sum_{k=1}^{r} (\frac{1}{r}Q_{ji} - 1/nr)^2 \\ &\leq \sum_{i,j\in V} r \sum_{k=1}^{r} (Q_{ji_k} - 1/nr)^2 \\ &= r \sum_{i\in V} \sum_{k=1}^{r} \|Q_{i_k} - \mathbf{1}/nr\|_2^2 \\ &= r\Psi'(t+1).\end{aligned}$$

We have thus proven,

$$\Psi(t) - \Psi(t+1) \geq r(\Psi'(t) - \Psi'(t+1)).$$

Now observe that E_{t+1} is, by construction, a perfect matching on G'. Therefore by [8, Lemma 3.3] (which the current lemma generalizes),

$$\begin{aligned}\Psi'(t) - \Psi'(t+1) &\geq \sum_{i_k, j_{k'} \in E_{t+1}} \|P_{i_k} - P_{j_{k'}}\|_2^2 \\ &= \sum_{i_k, j_{k'} \in E_{t+1}} \|\frac{1}{r}P_i - \frac{1}{r}P_j\|_2^2 \\ &= \frac{1}{r^2} \sum_{i,j\in E_{t+1}} \|P_i - P_j\|_2^2,\end{aligned}$$

and the lemma follows. ◀

Proof of Theorem 13. The initial potential is $\Psi(0) = n - 1$, and by Lemma 21 we need to get it below $1/4n^2$. Putting Lemmas 23 to 25 together, we see that in each step we have in expectation $\Psi(t+1) \leq (1 - \frac{1}{Cr\log n})\Psi(t)$. Hence, in expectation, it is enough to play for $O(r\log^2 n)$ rounds. ◀

4 Resistance Sparsification

We prove Theorem 6 by combining Theorem 7 with the following known result.

▶ **Theorem 26** (von Luxburg, Radl and Hein [16]). *Let G be a non-bipartite weighted graph with maximum edge weight $w_{\max}$ and minimum weighted degree $d_{\min}$. Let u, v be nodes in G with weighted degrees d_u, d_v respectively. Then*

$$\left| R_G(u,v) - \left(\frac{1}{d_u} + \frac{1}{d_v}\right)\right| \leq 2\left(\frac{1}{\lambda_2(G)} + 2\right)\frac{w_{\max}}{d_{\min}^2}.$$

Qualitatively, the theorem asserts that in a sufficiently regular expander, the resistance distance is essentially determined by vertex degrees. Therefore an expanding subgraph H of G with the *same* weighted degrees can serve as a resistance sparsifier. In particular, in order to resistance-sparsify a regular expander, all we need is a regular expanding subgraph, as we have by Theorem 7. Since Theorem 26 does not apply to bipartite graphs, we will use the following variant that holds also for bipartite graphs as long as they are regular. Its proof appears in Section A.1.

▶ **Theorem 27.** *Let G be a weighted graph which is d-regular in weighted degrees, with maximum edge weight $w_{\max}$. Let u, v be nodes in G. Then*

$$\left| R_G(u,v) - \frac{2}{d} \right| \leq 12 \left(\frac{1}{\lambda_2(G)} + 2 \right) \frac{w_{\max}}{d^2}.$$

Proof of Theorem 6. Using Theorem 7 we obtain a d-regular subgraph H of G with $\phi(H) > \frac{1}{3}$. By removing the obtained subgraph H from G and iterating, we can apply the theorem $3d/\varepsilon$ times and obtain disjoint subgraphs H. Since $d = (\log n)^{O(1)}$ and $D = \Omega(n)$, the degree of G does not significantly change in the process, and the requirements of Theorem 7 continue to hold throughout the iterations (with a loss only in constants). Taking the union of the disjoint subgraphs produced in this process, we obtain a subgraph H of G which is $(3d^2/\varepsilon)$-regular with $\phi(H) \geq d/\varepsilon$. By the discrete Cheeger inequality,

$$\lambda_2(H) \geq \frac{1}{2} \left(\frac{\phi(H)}{\deg(\mathrm{H})} \right)^2 \geq \frac{1}{18d^2}.$$

Recall that H has edge weights in $\{1, 2\}$. We now multiply each weight by $\varepsilon D/(3d^2)$, rendering it D-regular in weighted degrees. This does not affect $\lambda_2(H)$ since it is an eigenvalue of the *normalized* Laplacian.

Let $u, v \in V$. Apply Theorem 27 on both G and H. As G is D-regular with $w_{\max} = 1$ and $\lambda_2(G) = \Omega(1)$, we know that $R_G(u,v) = \frac{2}{D} \pm O\left(\frac{1}{D^2}\right)$. And as H is D-regular with $w_{\max} = O(\frac{\varepsilon D}{d^2})$ and $\lambda_2(H) = \Omega(1/d^2)$, we know that $R_H(u,v) = \frac{2}{D} \pm O\left(\frac{\varepsilon}{D}\right)$. Putting these together, we get $\frac{R_H(u,v)}{R_G(u,v)} = 1 \pm O\left(\varepsilon + \frac{1}{D}\right) = 1 \pm O(\varepsilon)$, where the last equality holds for sufficiently large n since $D = \Omega(n)$. Scaling ε down by the constant hidden in the last $O(\varepsilon)$ notation yields the theorem. ◀

Acknowledgements. We thank Uriel Feige for useful comments on this work.

References

1 Alexandr Andoni, Robert Krauthgamer, and David P. Woodruff. The sketching complexity of graph cuts. *CoRR*, abs/1403.7058, 2014. arXiv:1403.7058.

2 Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM J. Comput.*, 41(6):1704–1721, 2012. doi:10.1137/090772873.

3 A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *28th Annual ACM Symposium on Theory of Computing*, pages 47–55. ACM, 1996. doi:10.1145/237814.237827.

4 Béla Csaba, Daniela Kühn, Allan Lo, Deryk Osthus, and Andrew Treglown. Proof of the 1-factorization and Hamilton decomposition conjectures. *ArXiv e-prints*, abs/1401.4159, 2014. arXiv:1401.4159.

5 M. M. Deza and M. Laurent. *Geometry of cuts and metrics.* Springer-Verlag, Berlin, 1997.

6 Michael Kapralov and Rina Panigrahy. Spectral sparsification via random spanners. In *3rd Innovations in Theoretical Computer Science Conference*, pages 393–398. ACM, 2012. doi:10.1145/2090236.2090267.

7 Rohit Khandekar, Subhash A. Khot, Lorenzo Orecchia, and Nisheeth K. Vishnoi. On a cut-matching game for the sparsest cut problem. Technical Report UCB/EECS-2007-177, EECS Department, University of California, Berkeley, 2007. URL: `http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-177.html`.

8 Rohit Khandekar, Satish Rao, and Umesh V. Vazirani. Graph partitioning using single commodity flows. *J. ACM*, 56(4), 2009. `doi:10.1145/1538902.1538903`.

9 Daniela Kühn and Deryk Osthus. Decompositions of complete uniform hypergraphs into hamilton berge cycles. *J. Comb. Theory, Ser. A*, 126:128–135, 2014. `doi:10.1016/j.jcta.2014.04.010`.

10 L. Lovász. Random walks on graphs: a survey. In *Combinatorics, Paul Erdős is eighty, Vol. 2 (Keszthely, 1993)*, volume 2 of *Bolyai Soc. Math. Stud.*, pages 353–397. János Bolyai Math. Soc., Budapest, 1996.

11 D. Peleg and A. A. Schäffer. Graph spanners. *J. Graph Theory*, 13(1):99–116, 1989. `doi:10.1002/jgt.3190130114`.

12 L. Perkovic and B. Reed. Edge coloring regular graphs of high degree. In *Discrete Math.,165/166*, pages 567–578, 1997.

13 D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *36th Annual ACM Symposium on Theory of Computing*, pages 81–90. ACM, 2004. `doi:10.1145/1007352.1007372`.

14 Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, December 2011. `doi:10.1137/080734029`.

15 Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, July 2011. `doi:10.1137/08074489X`.

16 Ulrike von Luxburg, Agnes Radl, and Matthias Hein. Hitting and commute times in large random neighborhood graphs. *Journal of Machine Learning Research*, 15(1):1751–1798, 2014. URL: `http://jmlr.org/papers/v15/vonluxburg14a.html`.

A Appendix: Omitted Proofs

A.1 Proof of Theorem 27

In the non-bipartite case, Theorem 27 follows from Theorem 26. We henceforth assume that $G = (V, E, w)$ is bipartite with bipartition $V = V_1 \cup V_2$. Note that since it is regular, we must have $|V_1| = |V_2| = \frac{1}{2}|V|$. Furthermore, as a weighted regular bipartite graph, G is a convex combination of perfect matchings and hence is regular also in unweighed degrees. Let d' denote the unweighted degree of each vertex in G. If $d' \leq 2$ then it is easy to verify that the theorem holds (due to poor expansion), so we henceforth assume $d' \geq 3$.

For brevity we denote the error term in Theorem 26 as

$$\mathrm{err} := 2\left(\frac{1}{\lambda_2(G)} + 2\right)\frac{w_{\max}}{d^2}.$$

We will use the notion of *hitting time*: For a pair of vertices u, v, the hitting time $H_G(u, v)$ is defined as the expected time it takes a random walk in G that starts at u, to hit v. Define the *normalized hitting time* $h_G(u, v) = \frac{1}{2W} H_G(u, v)$, where W is the sum of all edge weights in G. We then have,

$$R_G(u, v) = h_G(u, v) + h_G(v, u). \tag{2}$$

We will use the following bound on the normalized hitting time, which is given in the same theorem by von Luxburg, Radl and Hein [16].

▶ **Theorem 28.** *In the same setting of Theorem 26,*

$$\forall u \neq v \in V, \qquad h_G(u,v) = \frac{1}{d_v} \pm \mathrm{err}.$$

(Like Theorem 26, this theorem does not apply to bipartite graphs, and this is the obstacle we are now trying to circumvent.)

We begin by handling pairs of vertices contained within the same partition side, say V_1. We construct from G a weighted graph G_1 on the vertex set V_1, with weights w_1, by putting

$$\forall i \neq j \in V_1, \quad w_1(i,j) = \frac{1}{d} \sum_{k \in V_2} w(i,k) w(j,k).$$

We argue that $H_{G_1}(u,v) = \frac{1}{2} H_G(u,v)$. This follows by observing that we set the weights w_1 such that for any $i,j \in V_1$, the probability to walk in one step from i to j in G_1 equals the probability to walk in two steps from i to j in G via an intermediate node in V_2. Furthermore, we have normalized the weights w_1 such that G_1 is d-regular in weighted degrees. Recalling that $|V_1| = \frac{1}{2}|V|$, we have

$$h_{G_1}(u,v) = \frac{1}{d|V_1|} H_{G_1}(u,v) = \frac{2}{d|V|} \cdot \frac{1}{2} H_G(u,v) = h_G(u,v).$$

Recalling that the unweighted degree in G is $d' \geq 3$, we see that by construction, G_1 contains a triangle and hence is non-bipartite. Hence we can apply to it Theorem 28 and obtain $h_{G_1}(u,v) = \frac{1}{d} \pm \mathrm{err}_1$, where err_1 is the error term of G_1. Note that for every $i \neq j \in V_1$ we have $w_1(i,j) \leq \frac{w_{\max}}{d} \sum_{k \in V_2} w(i,k) = w_{\max}$, so the maximum edge weight in G_1 is bounded by $w_{\max}$, and $\lambda_2(G_1) \geq \lambda_2(G)$ (easy to verify by construction), so $\mathrm{err}_1 \leq \mathrm{err}$, and we have $h_{G_1}(u,v) = \frac{1}{d} \pm \mathrm{err}$. Hence,

$$h_G(u,v) = \frac{1}{d} \pm \mathrm{err}.$$

Recalling that $R_G(u,v) = h_G(u,v) + h_G(v,u)$, we have established that

$$R_G(u,v) = \frac{2}{d} \pm 2\mathrm{err}$$

for every pair $u,v \in V_1$. The same arguments hold for every pair $u,v \in V_2$ as well. We are left to handle the case $u \in V_1$, $v \in V_2$. Recalling the definition of hitting time, we have

$$\begin{aligned}
H_G(u,v) &= 1 + \frac{w(u,v)}{d} \cdot 0 + \sum_{x \in V_2 \setminus \{v\}} \frac{w(u,x)}{d} H_G(x,v) && \text{(factoring out the first step)} \\
&= 1 + \frac{w(u,v)}{d} \cdot 0 + \sum_{x \in V_2 \setminus \{v\}} \frac{w(u,x)}{d} \cdot 2W \cdot h_G(x,v) \\
&= 1 + 2W \sum_{x \in V_2 \setminus \{v\}} \frac{w(u,x)}{d} \left(\frac{1}{d} \pm \mathrm{err} \right) && \text{(since } v,x \in V_2\text{)} \\
&= 1 + 2W \left(1 - \frac{w(u,v)}{d} \right) \left(\frac{1}{d} \pm \mathrm{err} \right).
\end{aligned}$$

Therefore

$$h_G(u,v) = \frac{1}{2W} + \left(1 - \frac{w(u,v)}{d} \right) \left(\frac{1}{d} \pm \mathrm{err} \right),$$

which implies

$$h_G \leq \frac{1}{2W} + \frac{1}{d} \pm \mathrm{err}$$

and

$$h_G(u,v) \geq \frac{1}{2W} + \left(1 - \frac{w_{\max}}{d}\right)\left(\frac{1}{d} \pm \mathrm{err}\right) = \frac{1}{2W} + \frac{1}{d} \pm 2\mathrm{err}.$$

Together, $h_G(u,v) = \frac{1}{d} + \frac{1}{2W} \pm 2\mathrm{err}$. Now, since for an arbitrary vertex i we have

$$d = \deg(i) = \sum_{j \in V} w(i,j) \leq n w_{\max},$$

we see that $\frac{1}{2W} = \frac{1}{nd} \leq \frac{w_{\max}}{d^2} \leq \mathrm{err}$ and hence

$$h_G(u,v) = \frac{1}{d} \pm 3\mathrm{err}.$$

Plugging this into $R_G(u,v) = h_G(u,v) + h_G(v,u)$, we find

$$R_G(u,v) = \frac{2}{d} \pm 6\mathrm{err},$$

which completes the proof of Theorem 27. ◀

A.2 Further Omitted Proofs

▶ **Proposition 29.** *Let $G(V,U;E)$ be a bipartite graph on n nodes with $|V| = |U| = \frac{1}{2}n$, and minimum degree $\geq \frac{1}{4}n$. Then G contains a perfect matching.*

Proof. Let $S \subset V$ be non-empty, and denote $N(S) \subset U$ the set of nodes with a neighbor in S. If $|S| \leq \frac{1}{4}n$ then since any $v \in S$ has $\frac{1}{4}n$ neighbors in U, we have $|N(S)| \geq N(\{v\}) \geq \frac{1}{4}n \geq |S|$. If $|S| > \frac{1}{4}n$ then by the minimum degree condition on side U, every $u \in U$ must have a neighbor in S, and hence $|N(S)| = |U| = |V| \geq |S|$. The same arguments apply for $S \subset U$, so the condition of Hall's Marriage Theorem is verified, and it implies that G contains a perfect matching. ◀

▶ **Proposition 30.** *Consider an instance of Set Cover with a set S of n elements, and a family $\mathcal{M}$ of subsets of S. Suppose each $x \in S$ belongs to at least a μ-fraction of the subsets in $\mathcal{M}$. Then for sufficiently large n, we can efficiently find a cover $M \subset \mathcal{M}$ with $|M| \leq \frac{1.1}{\mu} \log n$.*

Proof. Pick q uniformly random sets (with replacement) from $\mathcal{M}$ to form M. The probability that a given element in S is not covered by M is upper-bounded by $(1-\mu)^q$. Taking a union bound over the element, we need to ensure that $n(1-\mu)^q < 1$ in order to ensure that with constant probability, M is a solution to the given Set Cover instance. This can be achieved by $q \leq \frac{1.1}{\mu} \log n$. ◀

Reconstruction/Non-reconstruction Thresholds for Colourings of General Galton-Watson Trees[*]

Charilaos Efthymiou

Georgia Institute of Technology, College of Computing
266 Ferst Drive, Atlanta, GA-30332, USA
`efthymiou@gmail.com`

Abstract

The broadcasting models on trees arise in many contexts such as discrete mathematics, biology, information theory, statistical physics and computer science. In this work, we consider the k-colouring model. A basic question here is whether the assignment at the root affects the distribution of the colourings at the vertices at distance h from the root. This is the so-called *reconstruction problem.* For the case where the underlying tree is d-ary it is well known that $d/\ln d$ is the *reconstruction threshold.* That is, for $k=(1+\epsilon)d/\ln d$ we have non-reconstruction while for $k=(1-\epsilon)d/\ln d$ we have reconstruction.

Here, we consider the largely unstudied case where the underlying tree is chosen according to a predefined distribution. In particular, we consider the well-known Galton-Watson trees. The corresponding model arises naturally in many contexts such as the theory of spin-glasses and its applications on random Constraint Satisfaction Problems (rCSP). The study on rCSP focuses on Galton-Watson trees with offspring distribution $\mathcal{B}(n,d/n)$, i.e. the binomial with parameters n and d/n, where d is fixed. Here we consider a *broader* version of the problem, as we assume *general offspring distribution* which includes $\mathcal{B}(n,d/n)$ as a special case.

Our approach relates the corresponding bounds for (non)reconstruction to certain *concentration properties* of the offspring distribution. This allows to derive reconstruction thresholds for a very wide family of offspring distributions, which includes $\mathcal{B}(n,d/n)$. A very interesting corollary is that for distributions with expected offspring d, we get reconstruction threshold $d/\ln d$ under *weaker concentration* conditions than what we have in $\mathcal{B}(n,d/n)$.

Furthermore, our reconstruction threshold for the random colorings of Galton-Watson with offspring $\mathcal{B}(n,d/n)$, implies the reconstruction threshold for the random colourings of $G(n,d/n)$.

1998 ACM Subject Classification G.2 Discrete Mathematics, G.3 Probability and Statistics, E.4 Coding and Information Theory

Keywords and phrases Random Colouring, Reconstruction Problem, Galton-Watson Tree

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.756

1 Introduction

The broadcasting models on trees and the closely related reconstruction problem are studied in statistical physics, biology, communication theory, e.g. see [8, 21, 14]. Our work is motivated from the study of *random Constraint Satisfaction Problems* (rCSP) such as random graph colouring, random k-SAT etc. This is mainly because the models on random trees capture some of the most fundamental properties of the corresponding models on random (hyper)graphs, e.g. [7, 15, 20].

[*] This work was supported by ARC – Georgia Institute of Technology, College of Computing.

The most fundamental problem in the study of broadcasting models is to determine the reconstruction/non-reconstruction threshold. I.e. whether the configuration of the root biases the distribution of the configuration of distant vertices. The transition from non-reconstruction to reconstruction can be achieved by adjusting appropriately the parameters of the model. Typically, this transition exhibits a *threshold behaviour*. So far, the main focus of the study was to determine the precise location of this threshold for various models when the underlying graph is a fixed tree, mostly regular. In these cases, typically the reconstruction threshold is expressed in terms of the maximum degree of the underlying tree, e.g. [3, 26, 2, 4].

In a lot of applications, e.g. phylogeny reconstruction, rCSP, usually the underlying tree is random. Motivated by such problems, in this work we study the reconstruction problem for the colouring model when the underlying tree is chosen according to some predefined probability distribution. In particular, we consider *Galton-Watson* trees (GW-trees) with some *general* offspring distribution.

In our setting, the main technical challenge is to deal with the so-called "effect of high degrees". That is, we expect to have vertices in the tree which are of degree much higher than the expected offspring. The deviation from the expected degree is so large that expressing the (non)reconstruction bounds in terms of maximum degree leads to highly suboptimal results. Similar challenges appear in related problems in random graphs $G(n, d/n)$ e.g. sampling colourings [11, 10, 13, 27].

It is a folklore conjecture that when the offspring distribution is "reasonably" concentrated about its expectation, then the reconstruction threshold can be expressed in terms of the expected offspring of the underlying tree. Somehow, the concentration makes the high degree vertices sufficiently rare, such that their effect on the phenomenon is negligible. Our aim is to make the intuitive base of this relation *rigorous* by adopting the most generic assumptions about the offspring distribution.

More specifically, our result summarizes as follows: We provide a concentration criterion for the distributions over the non-negative integers about the expectation. For a GW-tree with offspring distribution that satisfies this criterion, the transition from non-reconstruction to reconstruction exhibits a threshold behaviour at the critical point $d/\ln d$, where d is the expected offspring. Interestingly, the aforementioned concentration criterion is much weaker than the standard tail bounds we have for many natural distributions, e.g. $\mathcal{B}(n, d/n)$ with fixed d.

On the other hand, when the concentration of the offspring distribution is not sufficiently high to provide thresholds, we still get upper and lower bounds for reconstruction and non-reconstruction, respectively. These bounds are expressed in terms of the tails of the offspring distribution.

Concluding, let us remark that the reconstruction threshold we get for the random colourings of GW-tree with offspring $\mathcal{B}(n, d/n)$, allows to compute the corresponding threshold for the random colourings of $G(n, d/n)$ [7, 15, 20]. See Section 2.1 for more discussion.

2 Definitions and Results

For the sake of brevity, we define the colouring model and the reconstruction problem, first, in terms of a fixed complete Δ-ary T of height h, where $\Delta, h > 0$ are integers. Later we will extend these definitions w.r.t. GW trees.

The broadcasting models on a tree T are models where information is sent from the root over the edges to the leaves. For some finite set of spins (colours) $S = \{1, 2, \ldots, k\}$, a configuration on T is an element in $S^{V(T)}$, i.e. it is an assignment of spins to the vertices of T. The spin of the root r is chosen according to some initial distribution over S. The information propagates along the edges of the tree as follows: There is a $k \times k$ stochastic matrix M such that if the vertex v is assigned spin i, then its child u is assigned spin j with probability $M_{i,j}$. The *k-colouring* model we consider here corresponds to having M such that

$$M_{i,j} = \begin{cases} \frac{1}{k-1} & \text{for } i \neq j \\ 0 & \text{otherwise.} \end{cases}$$

We let μ be the *uniform distribution* over the k-colourings of T. We also refer to μ as the Gibbs distribution. The broadcasting process gives rise to Gibbs distributions on the underlying tree T. Fixing the spin (colour assignment) at the root of T, the configuration we get after the broadcasting process has finished is distributed as in μ conditional the spin of the root.

The reconstruction problem can be cast very naturally in terms of the corresponding Gibbs distribution. More specifically, let $r(T)$ (or r_T) denote the root of the tree T. Also, let $L_h(T)$ be the set of vertices at distance h from the root $r(T)$. Finally, we let μ^i be the distribution μ conditional that the spin at r_T is $i \in S$. Reconstructibility is defined as follows:

▶ **Definition 1.** For any $i, j \in S$ let $||\mu^i - \mu^j||_{L_h}$ denote the total variation distance of the projections of μ^i and μ^j on L_h. We say that a model is *reconstructible* on a tree T if there exists $i, j \in S$ for which

$$\lim_{h\to\infty} ||\mu^i - \mu^j||_{L_h(T)} > 0.$$

When the above limit is zero for every i, j, then we say that the model has *non-reconstruction*.

Non-reconstruction implies, also, that *typical* colourings of the vertices at level h of the tree have a vanishing effect on the distribution of the colouring of $r(T)$, as h grows.

For the colouring model on Δ-ary trees it is well-known that the reconstruction threshold is at the critical value $\Delta/\ln\Delta$, see [3, 22, 25, 26]. That is, for any given fixed $\epsilon > 0$ and sufficiently large Δ, we have non-reconstruction when $k \geq (1+\epsilon)\Delta/\ln\Delta$ while for $k \leq (1-\epsilon)\Delta/\ln\Delta$ we have reconstruction.

Rather than considering a fixed tree, here, we consider a Galton Watson tree (GW-trees) with some *general* offspring distribution. In particular, we let the following:

▶ **Definition 2.** Let ξ be a distribution over the non negative integers. We let $\mathcal{T}_\xi$ denote a Galton-Watson tree with offspring distribution ξ. Also, given some integer $h > 0$, we let $\mathcal{T}_\xi^h$ denote the restriction of $\mathcal{T}_\xi$ to its first h levels[1].

For the sake of brevity any distribution ξ on the non-negative integers is represented as a stochastic vector. That is, for Z distributed as in ξ and any integer $i \geq 0$, it holds that $\Pr[Z = i] = \xi(i)$ (or ξ_i).

For the case of a random tree, e.g. Galton-Watson tree, the notion of reconstructibility, extends as follows:

[1] In other words, $\mathcal{T}_\xi^h$ is the induced subtree of $\mathcal{T}_\xi$ which contains all the vertices within graph distance h from the root.

▶ **Definition 3.** We say that a model is *reconstructible* on $\mathcal{T}_\xi$ if there exists $i, j \in S$ for which

$$\lim_{h\to\infty} \mathbb{E}||\mu^i - \mu^j||_{L_h} > 0,$$

where the expectation is w.r.t. the instances of the tree. When the above limit is zero for every $i, j \in S$, then we say that the model has *non-reconstruction*.

So as to have a threshold behavior for reconstruction, it is natural to require a certain kind of parametrization for the offspring distribution ξ. This parametrization allows to adjust the expectation from low to high. In this work we assume that we are dealing with such distribution.

▶ **Definition 4.** Consider $\mathcal{T}_\xi$ for some offspring distribution ξ with expected offspring d_ξ. For the k-colouring model on $\mathcal{T}_\xi$ we have a *reconstruction threshold* θ for some function $\theta : \mathbb{R}^+ \to \mathbb{R}^+$, if the following holds: For any $\alpha > 0$ and $d_\xi > d_\xi(\alpha)$, we have non-reconstruction when $k \geq (1+\alpha)\theta(d_\xi)$, while we have reconstruction when $k \leq (1-\alpha)\theta(d_\xi)$.

One of the main results of this work is to show that we have a threshold behaviour for the reconstruction/non-reconstruction transition for the k-colourings of $\mathcal{T}_\xi$ when ξ is *well concentrated*. The notion of well concentration is defined as follows:

▶ **Definition 5.** A distribution ξ over the positive integers with expectation d_ξ is defined to be "well concentrated" if the following is true: There is an absolute constant $c > 0$ such that for any fixed $\gamma > 0$, sufficiently large d_ξ and any $x \geq (1+\gamma)d_\xi$ it holds that

$$\sum_{j\geq x} \xi_j \leq x^{-c} \qquad \textit{and} \qquad \sum_{j\leq(1-\gamma)d_\xi} \xi_j \leq (d_\xi)^{-c}. \tag{1}$$

The quantity c is independent of the distribution ξ. We do not compute the exact value of c but it is implicit from our analysis.

The following theorem is one of the main results in our work.

▶ **Theorem 6.** *Let ξ be a well concentrated distribution over the non-negative integers. Then, the colouring model on $\mathcal{T}_\xi$, with expected offspring d_ξ, has reconstruction threshold $d_\xi/\ln d_\xi$.*

The above theorem follows as a corollary of a more general and more technical result, Theorem 10. This theorem is more general as it covers non-threshold cases, too.

It is not hard to show that $\mathcal{B}(n, d/n)$ is well concentrated. This follows trivially by just using standard Chernoff bounds (e.g. [24]). Then, Theorem 6 implies the following corollary.

▶ **Corollary 7.** *Consider $\mathcal{T}_\xi$ where ξ is the distribution $\mathcal{B}(n, d/n)$, where d is fixed. Then, the colouring model on $\mathcal{T}_\xi$, has reconstruction threshold $d/\ln d$.*

As a matter of fact, it is elementary to verify that $\mathcal{B}(n, d/n)$ is, by no means, the less well concentrated offspring distribution we can have. That is, a distribution with less heavy tails than $\mathcal{B}(n, d/n)$ can be well concentrated.

2.1 From Galton-Watson trees to Random Graphs

The non-reconstruction phenomenon in rCSP seems to be central in the algorithmic problems. In particular, it has been related to the *efficiency* of local algorithms which search for satisfying solutions. That is, when we have non-reconstruction, usually there is an efficient (simple) local algorithm which finds satisfying assignments efficiently e.g. [6, 16]. On the other hand, in the reconstruction regime there is no efficient algorithm which finds solutions.

For this reason, the transition from non-reconstruction to reconstruction on rCSPs has been attributed the name "algorithmic barrier"[2], see [1].

The ingenious, however, mathematically non-rigorous *Cavity Method*, introduced by physicists [18, 17], makes very impressive predictions about the fundamental properties of rCSP. One of the most interesting parts of these predictions involves the Gibbs distribution and its spatial mixing properties like reconstructibility. The Cavity Method predicts that the spatial mixing properties of the Gibbs distribution over the colouring of $G(n, d/n)$ can be studied by means of the Gibbs distribution of the k-colourings over a Galton-Watson tree with offspring distribution $\mathcal{B}(n, d/n)$, where d is fixed independent of n. That is, choose some vertex v in $G(n, d/n)$ and some fixed radius neighborhood around v. The projection of Gibbs distribution on this neighborhood is, somehow, "similar" to the corresponding Gibbs distribution over the Galton-Watson tree. The above line of arguments, led to conjecture that the colouring model on a random graph $G(n, d/n)$ has the same reconstruction threshold as that of the GW tree with offspring $\mathcal{B}(n, d/n)$.

All the above considerations from the Cavity Method have been studied on a rigorous basis in [7, 15, 20]. We have a quite accurate picture of the relation between the local projection of Gibbs distribution on $G(n, d/n)$ and the Gibbs distribution on the $\mathcal{B}(n, d/n)$ Galton-Watson tree. In particular, we have mathematically rigorous arguments which imply that indeed the reconstruction thresholds for $G(n, d/n)$ and GW-tree coincide as far as the colouring model is concerned [3]. That is, Corollary 7 implies that, indeed, the reconstruction threshold for the colouring model on $G(n, d/n)$ is $d/\ln d$.

3 High Level Description

In this section, we give a high level overview of how do we derive upper and lower bounds for reconstruction and non-reconstruction, respectively. Consider an instance of $\mathcal{T}_\xi^h$ for some distribution ξ over the non-negative integers and some integer $h > 0$.

▶ Remark. For a set of vertices Λ in the tree, we use the term *random colouring of* Λ to indicate the following way of colouring Λ: Take a random colouring of the tree and keep only the colouring of the vertices in Λ. Also, when we refer to *a typical colouring of vertex set* Λ, we imply that this colouring is typical w.r.t. the aforementioned distribution.

Depending on the tails of ξ we choose appropriate quantities Δ_+ and Δ_- such that $\Delta_- \leq d_\xi \leq \Delta_+$. Given these two quantities we show that we have non-reconstruction for $k \geq (1+\alpha)\Delta_+/\ln \Delta_+$ and we have reconstruction for $k \leq (1-\alpha)\Delta_-/\ln \Delta_-$, for the colouring model on $\mathcal{T}_\xi^h$, where $\alpha > 0$ is fixed. We show (non)reconstruction by arguing about the *structure* of $\mathcal{T}_\xi^h$.

3.1 Non Reconstruction

First, we focus on non-reconstruction. Given Δ_+, we define a set of structural specifications such that if $\mathcal{T}_\xi^h$ satisfies them, then we have non-reconstruction for $k \geq (1+\alpha)\Delta_+/\ln \Delta_+$. We should consider Δ_+ to be a parameter for the specifications.

In particular, given Δ_+, we introduce the notion of *mixing* vertex. Roughly speaking, a vertex $v \in \mathcal{T}_\xi^h$ is mixing if the following is true: A typical k-colouring of the vertices at level h

[2] We should mention that this observation is empirical as there is no corresponding (rigorous) computational hardness result.
[3] For more details on the convergence between the distribution on the GW-tree and $G(n, d/n)$, see [7].

(e.g. above remark) does not bias the colouring of v by too much when $k \geq (1+\alpha)\Delta_+/\ln\Delta_+$. A vertex is biased if it is forced to choose from a relatively small set of colours. Perhaps a simple example of a vertex u *not* being mixing is when the subtree rooted at u has minimum degree much larger than Δ_+.

Whether some vertex in $\mathcal{T}_\xi^h$ is mixing or not depends on the subtree that hangs below it. An inductive definition of a mixing vertex, roughly, is as follows: A non leaf vertex v is mixing if the number of its children is at most Δ_+ while no more than $o(\Delta_+)$ of its children are non-mixing vertices. We consider the leaves of the tree to be mixing vertices, by default.

Furthermore, our specifications require that the mixing vertices are *sufficiently many* and *well spread* over the tree. To be more specific, we want the following: Every path from the root of $\mathcal{T}_\xi^h$ to a vertex at level h contains a sufficiently large number of vertices which are mixing. Additionally, we would like that the number of vertices at level h should not deviate significantly from their expectation.

Then, we argue that non-reconstruction holds for the colouring model on any, *arbitrary*, instance of $\mathcal{T}_\xi^h$ which satisfies the aforementioned specifications when $k \geq (1+\alpha)\Delta_+/\ln\Delta_+$. The choice of $\Delta_+ \geq d_\xi$ is the smallest possible that guarantees that $\mathcal{T}_\xi^h$ satisfies the structural specifications with probability that tends to 1 as $h \to \infty$.

For showing non-reconstruction, given a fixed tree of the desired structure, we use an idea introduced in [4]. The authors there show non-reconstruction by upper bounding appropriately the second moment of a quantity called "magnetization of the root". This approach has turned out to be quite popular for showing non-reconstruction bounds for various models on fixed trees e.g. [3, 26, 2, 4]. Additionally to [4], our approach builds on the very elegant combinatorial formalization from [3], which uses the notion of *unbiasing boundary* to deal with the magnetization of the root.

The approach in [3], for Δ-ary trees, shows non-reconstruction by arguing that the typical colourings of the vertices at level h do not bias the colouring of the vertices in the largest part of the underlying (regular) tree. The additional challenge here is that the trees we consider are highly non-regular. So as to get an effect similar to that of an unbiasing boundary from the colorings at level h, we need to argue about the subtree structure of each vertex in the tree. At this point we use the specification requirement. In other words, the setting we develop here with the mixing vertices somehow allows, to a certain extent, to apply the idea of unbiasing boundaries to control the magnetization of the root of the non-regular trees we deal with.

3.2 Reconstruction

As opposed to non-reconstruction, the reconstruction bound is well known in the special case where the offspring distribution is $\mathcal{B}(n, d/n)$, e.g. [19, 25]. Our approach deviates from both [19, 25] in that it applies to GW-trees with a general offspring distributions, while it focuses on the structural properties of the underlying tree, i.e. as we do for the non-reconstruction case.

We are based on the following observation. Consider some fixed tree T of height h and some positive integer k. Take a random k-colouring of the vertices at level h of that tree. Consider, now, the probability that this colouring "freezes" the colouring of the root of T. The assignment at the root gets frozen when the colouring of the vertices at level h specifies *uniquely* the colouring at the root. A sufficient condition for reconstruction is that the probability that the colouring of the root gets frozen is bounded away from zero for any $h > 0$. The reconstruction bound for a Δ-ary tree follows exactly from this argument. That is, for $k \leq (1-\alpha)\Delta/\ln\Delta$, a random colouring of $L_h(T)$ freezes the colour assignment of the root with probability bounded away from zero for any $h > 0$, see [25, 22].

The above argument extends naturally to the case of a non-regular tree T' of height h. More specifically, if T' has a h-level, Δ-ary subtree, rooted at $r(T')$, then the colouring model on T' has reconstruction for $k \leq (1-\alpha)\Delta/\ln\Delta$.

As far as the reconstruction for $\mathcal{T}_\xi$ is regarded, we work as follows: We consider some parameter Δ_- which depends on the offspring distribution ξ. We show that $\mathcal{T}_\xi^h$ has a Δ_--ary subtree with h levels rooted at $r(\mathcal{T}_\xi^h)$, with probability bounded away from zero for any $h > 0$. The considerations in the previous paragraphs and Definition 3, imply that the colouring model in $\mathcal{T}_\xi$ has reconstruction for $k \leq (1-\alpha)\Delta_-/\ln\Delta_-$. Our choice of Δ_- is the largest possible that guarantees exactly the subtree specification for $\mathcal{T}_\xi^h$.

4 Upper and Lower Bounds

We start our analysis by focusing on the upper and the lower bounds for reconstruction and non-reconstruction, respectively. Consider $\mathcal{T}_\xi^h$ and the k-colouring model on this tree. We define appropriate quantities Δ_- and Δ_+ which depend on the statistics of the offspring distribution ξ. As far as Δ_+ is concerned, we have the following:

▶ **Definition 8.** Consider a distribution ξ over the non negative integers with expectation d_ξ. Given some fixed $\delta \in (0, 1/10)$, we let $\Delta_+ = \Delta_+(\delta) \geq d_\xi$ be the minimum integer such that the following holds: There are $q \in [0, 3/4)$ and $\beta \geq 4$, independent of d_ξ, such that

$$q \geq \sum_{i>\Delta_+} \xi_i + \Pr\left[\mathcal{B}(\Delta_+, q) \geq (\Delta_+)^\delta\right] \tag{2}$$

and

$$\sum_{t>\Delta_+} t \cdot \xi_t \leq \exp\left(-2\beta \ln d_\xi\right), \qquad \Pr\left[\mathcal{B}(\Delta_+, q) > (\Delta_+)^\delta\right] \leq \exp\left(-2\beta \ln d_\xi\right). \tag{3}$$

We discuss how do we choose δ in the range $(0, 1/10)$ a bit later. Given ξ and δ we choose the minimum Δ_+ that satisfy the above requirements. Then we use Δ_+ as a parameter to specify a set of structural specifications for trees (roughly described in Section 3). For any instance of $\mathcal{T}_\xi$ which satisfies these specification we have non-reconstruction for any $k \geq (1+\alpha)\Delta_+/\ln\Delta_+$, where α is fixed.

▶ Remark. It turns out that there is a relation between the quantities α and δ. This means that given α we should choose appropriately δ, or the other way around.

The notion of mixing vertex is related to Definition 8 in the following way: A vertex v in $\mathcal{T}_\xi^h$ is mixing if the number of its children is at most Δ_+, while at most $(\Delta_+)^\delta$ of them are non-mixing.

To get further intuition, perhaps, it is useful to consider the condition in (2) and its implication in terms of the mixing vertices. Since we need the tree to have sufficiently many and well-spread mixing vertices, it is natural to require that the probability of a vertex in $\mathcal{T}_\xi^h$ to be mixing is sufficiently large regardless of its level in the tree. We satisfy exactly this requirement from (2).

Let q be an upper bound for the probability of each child of some $v \in \mathcal{T}_\xi^h$ to be non-mixing[4]. It is straightforward to show that the r.h.s. of (2) is an upper bound for v to be non-mixing. Moreover, if (2) holds, then clearly q is an upper bound for v to be non-mixing,

[4] The probability of a vertex being non-mixing depends only on the subtree rooted at this vertex.

too. That is, if some vertex at some level l of the tree is non-mixing with probability at most q, then (2) guarantees that for any vertex at level $l-1$ the probability of it being non-mixing has the same upper bound q. This implies that regardless of its level at the tree, each vertex v is mixing with probability at least $1-q$. For further details see in Section 8.

As far as Δ_{-} is concerned, we have the following.

▶ **Definition 9.** Let ξ be a distribution over the non negative integers. Given some $\delta \in (0, 1/10)$, we let $\Delta_{-} = \Delta_{-}(\delta) \le d_{\xi}$ be the maximum integer such that the following holds: There is $g \in [0, 3/4)$ such that

$$g \ge \sum_{i<\Delta_{-}} \xi_i + \sum_{i \ge \Delta_{-}} \xi_i \cdot \Pr\left[\mathcal{B}(i, 1-g) < (\Delta_{-}) - (\Delta_{-})^{\delta}\right]. \tag{4}$$

The arguments for reconstruction are based on showing that with probability bounded away from zero for any h, the following holds for $\mathcal{T}_{\xi}^{h}$: The root of $\mathcal{T}_{\xi}^{h}$ has a subtree of height h such that each non leaf vertex has at least $\Delta_{-} - (\Delta_{-})^{\delta}$ many children. In the full version of this extended abstract, in [12], we show that the condition in (4) guarantees that $\mathcal{T}_{\xi}^{h}$ has exactly this property.

The following theorem is the main technical result of this work. The trees considered in Theorem 10 do not necessarily have well concentrated offspring distribution ξ.

▶ **Theorem 10.** *Let some fixed $\alpha > 0$. Consider an instance of $\mathcal{T}_{\xi}^{h}$ such that the expected offspring d_{ξ} is sufficiently large. Set $\delta = \min\{\alpha/2, 1/10\}$, i.e. the variable that specifies both Δ_{+} and Δ_{-}.*

For μ, the Gibbs distribution over the k-colourings of $\mathcal{T}_{\xi}^{h}$, the following is true:
non-reconstruction: *For $k = (1+\alpha)\Delta_{+}/\ln\Delta_{+}$ and any $i, j \in [k]$ it holds that*
$\mathbb{E}||\mu^i - \mu^j||_{L_h} \le 8k^2(2\Delta_{+})^{-0.45\delta h}$.
reconstruction: *For $k = (1-\alpha)\Delta_{-}/\ln\Delta_{-}$ there are $i, j \in [k]$ such that*
$$\mathbb{E}||\mu^i - \mu^j||_{L_h} \ge \frac{1}{4}\left(1 - \frac{2}{\log k}\right).$$
Both of the expectations above are taken w.r.t. the tree instances.

The whole proof of Theorem 10 appears in the full version of this work in [12]. In this extended abstract we provide a sketch for the proof of the most interesting part of the theorem, the non-reconstruction part. See in Section 5.

Given Theorem 10, it is elementary to show that Theorem 6 holds. I.e. given that the offspring distribution is well concentrated (Definition 5), we to show that Δ_{-} and Δ_{+} are sufficiently close to each other. The derivations are simple they can be found in the full version of this paper in [12].

Notation. For any tree T we let $r(T)$ or r_T denote its root. Let $L_h(T)$ denote the set of vertices at graph distance h from $r(T)$. For every vertex $v \in T$, we define $\tilde{T}_v$ the subtree of T as follows: Delete the edge between v and its parent in T. Then $\tilde{T}_v$ is the connected component that contains v. We use the convention that $r(\tilde{T}_v) = v$.

We use capital letters of the Latin alphabet to indicate random variables which are colourings of the tree T, e.g. X, Y, etc. We use small letters of the greek alphabet to indicate fixed colourings, e.g. σ, τ, etc. We use the notation σ_{Λ} or $X(\Lambda)$ to indicate that the vertices in Λ have a colour assignment specified by the colourings σ, X, respectively.

Given a tree T, we let μ denote the Gibbs distribution for its k-colourings. Usually we consider μ under certain boundary conditions. That is, given some $\Lambda \subset T$, and σ, a k-colouring of T, we consider the Gibbs distribution conditional that the vertices in Λ have fixed colouring σ_Λ. In this case we denote the corresponding Gibbs distribution as μ^{σ_Λ}. For $\Xi \subseteq T$ we let μ_Ξ denote the *marginal* of the Gibbs distribution for the vertices in Ξ. We denote marginals over the vertex set Ξ of a Gibbs distribution with boundary σ_Λ in the natural way, i.e. $\mu^{\sigma_\Lambda}_\Xi$.

5 Proof of Theorem 10 – Non Reconstruction

First, consider a fixed tree T of height h and we let $L = L_h(T)$. From [23] we have that

$$||\mu^i - \mu||_{r_T} \leq k \cdot \sum_{\sigma(L)\in[k]^L} \mu_L(\sigma_L) \cdot ||\mu^{\sigma(L)} - \mu||_{r_T}. \tag{5}$$

Furthermore, from the definition of the total variation distance we have that

$$\begin{aligned}\sum_{\sigma(L)\in[k]^L} \mu_L(\sigma_L) \cdot ||\mu^{\sigma(L)} - \mu||_{r_T} &= \frac{1}{2} \sum_{\sigma(L)\in[k]^L} \mu_L(\sigma_L) \cdot \sum_{c\in[k]} \left|\mu^{\sigma(L)}_{r_T}(c) - 1/k\right| \\ &= \frac{1}{2} \sum_{c\in[k]} \sum_{\sigma(L)\in[k]^L} \mu_L(\sigma_L) \cdot \left|\mu^{\sigma(L)}_{r_T}(c) - 1/k\right|. \end{aligned}\tag{6}$$

The quantity $\left|\mu^{\sigma(L)}_{r(T)}(c) - 1/k\right|$, is usually called *magnetization of the root* $r(T)$, e.g. see [5]. The inner sum is the average magnetization at the root, w.r.t. boundary conditions at the set L. We bound this average magnetization by using the following standard result.

▶ **Proposition 11.** *Consider a fixed tree T of height h and some integer $k > 0$. For every $c \in [k]$ the following is true: Let X be a random k-colouring of T conditional that $X(r_T) = c$. It holds that*

$$\sum_{\sigma(L)\in[k]^L} \mu_L(\sigma(L)) \cdot \left|\mu^{\sigma(L)}_{r_T}(c) - 1/k\right| \leq \sqrt{\frac{1}{k} \cdot \left|\left|\mu^{X_L}(\cdot) - \mu^{Z^q_L}(\cdot)\right|\right|_{\{r_T\}}}, \tag{7}$$

where Z^q is random colouring of T conditional that $Z^q(r_T) = q$, where q maximizes the r.h.s. of (7)*.*

The proof of Proposition 11, which is very similar to the proof of Lemma 1 in [4], appears also in the full version of this work in [12].

The quantity on the r.h.s. of (7) is a deterministic one, i.e. it depends only the tree T, c and k. We let

$$\mathbb{G}_{c,k}(T) = \left|\left|\mu^{X_L}(\cdot) - \mu^{Z^q_L}(\cdot)\right|\right|_{\{r_T\}}.$$

Consider $\mathcal{T}^h_\xi$ as in the statement of Theorem 10. The quantity $\mathbb{G}_{c,k}(\mathcal{T}^h_\xi)$ is a random variable. In the light of (6), (5) and Proposition 11, it suffices to show that $\mathbb{E}\left[\mathbb{G}_{c,k}(\mathcal{T}^h_\xi)\right]$ tends to zero with h sufficiently fast, for any $c \in [k]$.

▶ **Definition 12** (Mixing Root). Let Δ_+ and δ be as in the statement of Theorem 10. For a tree T of height h, its root is called mixing if the following holds: When $h = 0$, then $r(T)$ is mixing, by default. When $h > 0$, $r(T)$ is mixing if and only if $\texttt{deg}(r_T) \leq \Delta_+$ and there are at most $(\Delta_+)^\delta$ many vertices v children of $r(T)$ such that $\tilde{T}_v$ does not have a mixing root.

▶ **Definition 13.** Given $\zeta \in [0, 1]$ and some integer $t > 0$, we let $\mathcal{A}_{t,\zeta}$ denote the set of trees T of height at most t such that the following holds: Every path $\mathcal{P}$ of length t from $r(T)$ to $L_t(T)$ contains at least $(1 - \zeta)t$ vertices v such that $\tilde{T}_v$ has a mixing root.

Before presenting our next result, we need to do the following remark. In Definition 8, given ξ and δ, among others the following inequality should hold for Δ_+,

$$\sum_{t \geq \Delta_+} t \cdot \xi_t < \exp\left(-2\beta \ln d_\xi\right),$$

where $\beta \geq 4$. Given Δ_+ and ξ the exact value of the parameter β is already specified. That is, when we define Δ_+ and ξ, the value of β is implicit.

▶ **Proposition 14.** *Assume that the distribution ξ, δ, Δ_+ are as defined in the statement of Theorem 10. Let $\mathcal{C} = \beta \ln d_\xi$. Also, let $\zeta \in (0,1)$ and $\theta = \theta(\zeta) > 1$ be such that $(1-\zeta)\theta < 1$ and $\beta(1-\theta) < -1$. Then, for every $h \geq 1$ it holds that*

$$\Pr[\mathcal{T}_\xi^h \in \mathcal{A}_{h,\zeta}] \geq 1 - \exp\left[-(1-\theta(1-\zeta))\mathcal{C} \cdot h\right].$$

The proof of Proposition 14 appears in Section 8.

▶ **Theorem 15.** *Let ξ, δ, Δ_+ and α be as in the statement of Theorem 10. Also, let $\zeta \in (0,1)$ and let the integer $h \geq 1$. For $k = (1+\alpha)\Delta_+/\ln \Delta_+$, it holds that*

$$\mathbb{E}\left[\mathbb{G}\left(\mathcal{T}_\xi^h\right) \middle| \mathcal{T}_\xi^h \in \mathcal{A}_{h,\zeta}\right] \leq \frac{4(2\Delta_+)^{-0.9(3/4-\zeta)\delta h}}{\Pr[\mathcal{T}_\xi^h \in \mathcal{A}_{h,\zeta}]}.$$

The proof of Theorem 15 appears in Section 6.

Set $\zeta = 1/4$, and $\theta = 1.3$, applying Proposition 14 we get that

$$\Pr[\mathcal{T}_\xi^h \notin \mathcal{A}_{h,\zeta}] \leq d_\xi^{-0.1h}. \tag{8}$$

For the same values of ζ, θ as above, (8) with Theorem 15 gives that

$$\mathbb{E}\left[\mathbb{G}(\mathcal{T}_\xi^h) \middle| \mathcal{T}_\xi^h \in \mathcal{A}_{h,\zeta}\right] \leq 8(2\Delta_+)^{-0.45\delta h}. \tag{9}$$

Since we always have $0 \leq \mathbb{G}(T) \leq 1$, for ζ and θ as above, we get that

$$\mathbb{E}\left[\mathbb{G}(\mathcal{T}_\xi^h)\right] \leq \mathbb{E}\left[\mathbb{G}(\mathcal{T}_\xi^h) \middle| \mathcal{T}_\xi^h \in \mathcal{A}_{h,1/4}\right] + \Pr\left[\mathcal{T}_\xi^h \notin \mathcal{A}_{h,1/4}\right] \leq 16(2\Delta_+)^{-0.45\delta h},$$

where the last inequality follows from (8) and (9). The theorem follows.

6 Proof of Theorem 15

Consider first the quantity $\mathbb{G}_{c,k}(T)$, for some fixed tree T. Then, it holds that

$$\mathbb{G}_{c,k}(T) = \left|\left|\mu^{X_L}(\cdot) - \mu^{Z_L^q}(\cdot)\right|\right|_{r_T}. \tag{10}$$

An important remark from Proposition 11 is that it allows to use any kind of correlation between the X, Z^q. For this reason we assume that (X, Z^q) is distributed as in $\nu_{c,q}^T$. We are going to specify this distribution later. First we get the following result.

▶ **Proposition 16.** *Let ξ, δ, Δ_+ and α be as in the statement of Theorem 15. Also let $0 \leq \gamma \leq \delta$. Then for $k = (1+\alpha)\Delta_+/\ln \Delta_+$, it holds that*

$$\begin{aligned}\mathbb{E}\left[\mathbb{G}_{c,k}\left(\mathcal{T}_\xi^h\right) \middle| \mathcal{T}_\xi^h \in \mathcal{A}_{h,\zeta}\right] &\\ \leq \frac{1}{\Pr\left[\mathcal{T}_\xi^h \in \mathcal{A}_{h,\zeta}\right]} &\left(2\exp\left(-\frac{1}{8}(\Delta_+)^{\frac{h/4-1}{2}\delta + \frac{7}{8}\frac{\alpha}{1+\alpha}}\right) \cdot \mathbb{E}\left[\left|L_h\left(\mathcal{T}_\xi^h\right)\right|\right]\right.\\ &\left. + 2(2(\Delta_+)^{-\gamma})^{(3/4-\zeta)h} \cdot \mathbb{E}[H(X_L, Z_L^q)]\right).\end{aligned} \tag{11}$$

For the above proposition we remark the following: On the r.h.s. of (11) the rightmost expectation term is w.r.t. both the joint distribution of X, Z^q and the distribution over the tree $\mathcal{T}^h_\xi$. The rest expectations are w.r.t. the distributions over trees only, i.e. $\mathcal{T}^h_\xi$. The proof of Proposition 16 appears in Section 7.

For showing the theorem we bound appropriately the two expectations on the r.h.s. of (11). It is elementary that

$$\mathbb{E}\left[\left|L_h\left(\mathcal{T}^h_\xi\right)\right|\right] = \left(d_\xi\right)^h. \tag{12}$$

For bounding $\mathbb{E}\left[H(X_L, Z^q_L)\right]$ we need to specify a coupling between the random variables X and Z^q which minimizes their expected Hamming distance. Observe that the expected hamming distance is both w.r.t. the coupling and the randomness of the trees.

The coupling of X and Z^q we use, can be defined inductively as follows: We colour the vertices from the root down to the leaves. For a vertex v whose father w is such that $X(w) = Z^q(w)$ we couple $X(v)$ and $Z^q(v)$ identically, i.e. $X(v) = Z^q(v)$. On the other hand, when $X(w) \neq Z^q(w)$ we set $X(v) = Z^q(v)$ unless $X(v) = Z^q(w)$, then we set $Z^q(v) = X(w)$.

Let w be a vertex in the tree and let u be a child of w. Then, for the coupling above, it holds that

$$\Pr\left[X(u) \neq Z^q(u) | X(w) \neq Z^q(w)\right] = k^{-1}.$$

In $\mathcal{T}^h_\xi$, the expected number of children per (non-leaf) vertex is d_ξ. Then, it is elementary to show that for a disagreeing vertex, the expected number of disagreeing children is $d_\xi/k \leq \frac{\ln \Delta_+}{1+\alpha}$, since $\Delta_+ > d_\xi$. Furthermore, it holds that

$$\mathbb{E}[H(X_L, Y_L)] \leq \left(\frac{\ln \Delta_+}{(1+\alpha)}\right)^h. \tag{13}$$

Observe that the above expectation is w.r.t. *both* tree instances and the joint distribution of the two random colourings.

The theorem follows by combining (13), (12) and Proposition 16.

7 Proof of Proposition 16

The previous setting allows to use ideas based on the notion of biasing-unbiasing boundary (introduced in [3]) to prove Proposition 16. To be more precise, the definition of biasing non-biasing boundaries we use here is slightly different than that [3], but the approach is similar.

▶ **Definition 17** (Non-Biasing Boundary). For $\alpha, \gamma, \delta, \Delta_+$ as in the statement of Proposition 16, we let $k = (1+\alpha)\Delta_+/\ln \Delta_+$, and let some integer $t \geq 1$. Consider a tree H of height t such that $r(H)$ is mixing. For σ, a k-colouring of H, we say that σ_L does not bias the root if the following holds:

- if $t = 1$, then $\sigma(L_t(G))$ uses all but at least $(\Delta_+)^\gamma$ many colours.
- if $t > 1$, then the following holds: We let $v_1, \ldots, v_s$ be the children of the root of H, where $s \leq \Delta_+$. Also, let $\mathbb{S} \subseteq \{\tilde{H}_{v_1}, \tilde{H}_{v_2}, \ldots, \tilde{H}_{v_s}\}$ contain only the subtrees whose roots are mixing. Then, there are at most Δ_+^δ many subtrees $\tilde{H}_{v_i} \in \mathbb{S}$ such that $\sigma(L_{t-1}(\tilde{H}_{v_i}))$ biases the root $r(\tilde{H}_{v_i})$.

Also, we let $\mathcal{U}(T)$ denote the set of all boundary conditions on L which are not biasing.

Note the notion of non-biasing boundary condition makes sense only for trees with mixing root.

▶ **Lemma 18.** *Let γ, α, Δ_+ be as in the statement of Proposition 16. Let $k = (1+\alpha)\frac{\Delta_+}{\ln \Delta_+}$, also let some integer $t \geq 1$. Consider a fixed tree T of height t and let $L = L_t(T)$. For σ, a k-colouring of T, such that σ_L is biasing for the root of T the following is true: There is at least one $c \in [k]$ such that for X, a random k-colouring of T, it holds that*

$$\Pr[X_{r(T)} = c | X_L = \sigma_L] \geq (\Delta_+)^{-\gamma}.$$

For a proof of Lemma 18 see in the full version of this work in [12].

▶ **Definition 19.** Let $\alpha, \gamma, \delta, \Delta_+, h$ be as in the statement of Proposition 16. Consider a tree T of height h and let $L = L_h(T)$. For every vertex $w \in L$ we define the set of boundaries $\mathcal{U}_w \subseteq [k]^L$ as follows: Let $\mathcal{P}$ denote the path that connects r_T and w and we let

$$\mathcal{M} = \left\{ v \in \mathcal{P} : dist(r_T, v) \leq (3/4)h,\ \tilde{T}_v \text{ has mixing root} \right\}.$$

Then $\mathcal{U}_w$ contains the boundary conditions on L which do not bias the root of any of the subtrees $\tilde{T}_v$ where $v \in \mathcal{M}$.

▶ **Proposition 20.** *Let $\alpha, \gamma, \delta, \Delta_+, h, \zeta$ be as in the statement of Proposition 16. Let some fixed tree $T \in \mathcal{A}_{h,\zeta}$ and let $L = L_h(T)$. Consider σ, τ to be two k-colourings of T such that $H(\sigma_L, \tau_L) = 1$. Furthermore, assume that $\sigma(w) \neq \tau(w)$ for some $w \in L$, while both $\sigma_L, \tau_L \in \mathcal{U}_w$. Then it holds that*

$$||\mu^{\sigma_L} - \mu^{\tau_L}||_{r(T)} \leq \Delta^*_{\zeta,h} = (2\Delta_+^{-\gamma})^{(3/4-\zeta)h}.$$

Proof. For showing the proposition we use disagreement percolation coupling construction. This approach is somehow standard and it has been used in different contexts, e.g. [9, 11]. For the full proof of the propositions see in the full version of this work in [12]. ◀

▶ **Proposition 21.** *Let $\alpha, \gamma, \delta, \Delta_+, h, \zeta$ be as in the statement of Proposition 16. Consider a fixed tree $T \in \mathcal{A}_{h,\zeta}$. Let X be a random k-colouring of T. For $k = (1+\alpha)\Delta_+/\ln \Delta_+$ and any $w \in L_h(T)$ it holds that*

$$\Pr\left[X_L \notin \mathcal{U}_w\right] \leq 2\exp\left(-\frac{1}{8}(\Delta_+)^{\frac{h/4-1}{2}\delta + \frac{7}{8}\frac{\alpha}{1+\alpha}}\right).$$

For the proof of Proposition 21 see in the full version of this work in [12].

Proof of Proposition 16. First, consider some fixed tree $T \in \mathcal{A}_{h,\zeta}$ and we let $L = L_h(T)$. Usually we fix a colouring of L and we call it (the colouring) boundary condition. We also use the term "free" boundary to indicate the absence of any boundary condition on L or some of its vertices.

Consider two colourings of the leaves $\sigma(L)$ and $\tau(L)$. We let m be the Hamming distance between $\sigma(L)$ and $\tau(L)$, i.e. $m = H(\sigma_L, \tau_L)$. Let $v_1, \ldots, v_m$ be the vertices in L for which σ_L and τ_L disagree. Consider the sequence of boundary conditions $Z_0, \ldots, Z_{2m} \in [k]^L$ such that $\sigma_L = Z_1$, $\tau_L = Z_{2m}$ while the rest of the members are as follows: For $i \leq m$, we get Z_i from Z_{i-1} be substituting the assignment of v_i from $\sigma(v_i)$ to "free". Also, for $i \geq m$ we get Z_{i+1} from Z_i by substituting $Z(v_{i-m})$ from "free" to $\tau(v_{i-m})$. It is direct that $H(Z_i, Z_{i+1}) = 1$.

From triangle inequality, it holds that

$$||\mu^{\sigma_L} - \mu^{\tau_L}||_{r(T)} \leq \sum_{i=0}^{2m-1} ||\mu^{Z_i} - \mu^{Z_{i+1}}||_{r(T)}. \qquad (14)$$

Also, it is not hard to see that for every $w \in L$ the following is true: if $\sigma_L \in \mathcal{U}_w$, then $Z_i \in \mathcal{U}_w$ for every $i = 1, \dots, m$. Similarly, if $\tau_L \in \mathcal{U}_w$, then $Z_i \in \mathcal{U}_w$ for every $i = m, \dots, 2m$.

Let the event $\mathbb{U}_{v_i}^{\sigma,\tau} =$ "$\sigma_L, \notin \mathcal{U}_{v_i} \bigcup \tau_L \notin \mathcal{U}_{v_i}$". Then it holds that

$$||\mu^{Z_i} - \mu^{Z_{i+1}}||_{r(T)} \leq \mathbb{I}_{\{\mathbb{U}_{v_i}\}} + \left(1 - \mathbb{I}_{\{\mathbb{U}_{v_i}\}}\right) \Delta^*_{\zeta,h}, \tag{15}$$

where $\Delta^*_{\zeta,h}$ is defined in the statement of Proposition 20. In words, the above inequality states the following: if at least one of the σ_L, τ_L are not in $\mathcal{U}_{v_i}$, then the l.h.s. of (15) is at most 1. On the other hand, if both $\sigma_L, \tau_L \in \mathcal{U}_{v_i}$ then the total variation distance on the l.h.s. can be upper bounded by using Proposition 20.

Plugging (15) into (14) we have that

$$||\mu^{\sigma_L} - \mu^{\tau_L}||_{r(T)} \leq 2 \cdot \sum_{v \in L_h(T)} \mathbb{I}_{\{\sigma_v \neq \tau_v\}} \cdot \left[\mathbb{I}_{\{\mathbb{U}_v\}} + \left(1 - \mathbb{I}_{\{\mathbb{U}_v\}}\right) \cdot \Delta^*_{\zeta,h}\right]. \tag{16}$$

Now, we consider the quantity $\mathbb{G}_{c,k}(T)$, i.e. $\mathbb{G}_{c,k}(T) = ||\mu^{X_L} - \mu^{Z^q_L}||_{r(T)}$. For bounding $\mathbb{G}_{c,k}(T)$ we are going to use (16). That is

$$\begin{aligned}
\mathbb{G}_{c,k}(T) &= ||\mu^{X_L} - \mu^{Z^q_L}||_{r(T)} \leq \sum_{\sigma_L, \tau_L \in [k]^L} \Pr\left[X_L = \sigma_L, Z^q_L = \tau_L\right] \cdot ||\mu^{\sigma_L} - \mu^{\tau_L}||_{r(T)} \\
&\leq 2 \cdot \sum_{\sigma_L, \tau_L \in [k]^L} \Pr\left[X_L = \sigma_L, Z^q_L = \tau_L\right] \\
&\qquad \cdot \sum_{v \in L_h(T)} \mathbb{I}_{\{\sigma_v \neq \tau_v\}} \cdot \left(\mathbb{I}_{\{\mathbb{U}_v^{\sigma,\tau}\}} + \left(1 - \mathbb{I}_{\{\mathbb{U}_v^{\sigma,\tau}\}}\right) \Delta^*_{\zeta,h}\right) \quad \text{[from (16)]} \\
&\leq 2 \cdot \sum_{v \in L_h(T)} \left(\Pr\left[X(v) \neq Z^q(v), \mathbb{U}_v^{X_L, Z^q_L}\right] + \Pr\left[X(v) \neq Z^q(v)\right] \cdot \Delta^*_{\zeta,h}\right) \\
&\leq 2 \cdot \sum_{v \in L_h(T)} \Pr\left[\mathbb{U}_v^{X_L, Z^q_L}\right] + 2 \cdot \sum_{v \in L_h(T)} \Pr\left[X(v) \neq Z^q(v)\right] \cdot \Delta^*_{\zeta,h}.
\end{aligned}$$

Due to symmetry it holds that $\Pr\left[X(L) \notin \mathcal{U}_v\right] = \Pr\left[Z^q(L) \notin \mathcal{U}_v\right]$. Using this observation and a union bound, the above inequality implies that

$$\begin{aligned}
\mathbb{G}_{c,k}(T) &\leq 4 \sum_{v \in L} \Pr\left[X(L) \notin \mathcal{U}_{\mathcal{P}_v}\right] + \Delta^*_{\zeta,h} \sum_{v \in L} \Pr\left[X(v) \neq Z^q(v)\right] \\
&\leq 2 \exp\left(-\frac{1}{8} (\Delta_+)^{\frac{h/4-1}{2}\delta + \frac{7}{8}\frac{\alpha}{1+\alpha}}\right) \cdot |L_h(T)| + 2\Delta^*_{\zeta,h} \cdot \mathbb{E}_{\nu_{c,q}}[H(X_L, Z^q_L)],
\end{aligned}$$

where in the last inequality we used Proposition 21 to bound $\Pr\left[X(L) \notin \mathcal{U}_{\mathcal{P}_v}\right]$. $\mathbb{E}_{\nu_{c,q}}[H(X(L), Z^q(L))]$ is the expected Hamming distance between X_L and Z^q_L and depends only on the joint distribution of X, Z^q, which is denoted as $\nu_{c,q}$.

The proposition follows by averaging over $\mathcal{T}^h_\xi$, conditional that we have a tree in $\mathcal{A}_{h,\zeta}$. That is

$$\begin{aligned}
&\mathbb{E}\left[\mathbb{G}_{c,k}\left(\mathcal{T}^h_\xi\right) \mid \mathcal{T}^h_\xi \in \mathcal{A}_{h,\zeta}\right] \\
&\qquad \leq \frac{1}{\Pr\left[\mathcal{T}^h_\xi \in \mathcal{A}_{h,\zeta}\right]} \Bigg(2 \exp\left(-\frac{1}{8}(\Delta_+)^{\frac{h/4-1}{2}\delta + \frac{7}{8}\frac{\alpha}{1+\alpha}}\right) \cdot \mathbb{E}\left[\left|L_h\left(\mathcal{T}^h_\xi\right)\right|\right] \\
&\qquad\qquad + 2(2\Delta_+^{-\gamma})^{(3/4-\zeta)h} \cdot \mathbb{E}[H(X_L, Z^q_L)]\Bigg).
\end{aligned}$$

The rightmost expectation term is w.r.t. both $\nu_{c,q}$ and the distribution of random trees $\mathcal{T}_\xi^h$. In the above derivations we used the following, easy to derive, inequality

$$\mathbb{E}\left[f\left(\mathcal{T}_\xi^h\right)|\mathcal{T}_\xi^h \in \mathcal{A}_{h,\zeta}\right] \leq \mathbb{E}\left[f\left(\mathcal{T}_\xi^h\right)\right]/\Pr\left[\mathcal{T}_\xi^h \in \mathcal{A}_{h,\zeta}\right],$$

where f is any non-negative functions on the support of the distribution $\mathcal{T}_\xi^h$. The proposition follows. ◀

8 Proof of Proposition 14

For $i = (1-\zeta)h$ we let $Q_{h,i} = \Pr\left[\mathcal{T}_\xi^h \notin \mathcal{A}_{h,\zeta}\right]$. Also, we let $Q_{h,i}^t = \Pr\left[\mathcal{T}_\xi^h \notin \mathcal{A}_{h,\zeta} \middle| \texttt{deg}(r(T_\xi^h)) = t\right]$ Using a simple union bound we get the following: For $t \leq (\Delta_+)^\delta$ it holds that

$$Q_{h,i}^t \leq t \cdot Q_{h-1,i-1}. \tag{17}$$

Intuitively, the above is implied by the following: If $\texttt{deg}(r(T_\xi^h)) \leq (\Delta_+)^\delta$, then, regardless of its children, the root $r(T_\xi^h)$ is mixing. Conditional that $\texttt{deg}(r(T_\xi^h)) \leq (\Delta_+)^\delta$ holds, so as to have $\mathcal{T}_\xi^h \notin \mathcal{A}_{h,\zeta}$, there should be a vertex v, child of $r(T_\xi^h)$ such that the following is true: The subtree $\tilde{T}_v$ has a path from its root to its vertices of at level $h-1$ which contain less than $i-1$ mixing vertices.

Using similar arguments, for $(\Delta_+)^\delta \leq t \leq \Delta_+$, we get the following lemma, whose proof appear in Section 8.1.

▶ **Lemma 22.** *For $(\Delta_+)^\delta < t \leq \Delta_+$, it holds that*

$$Q_{h,i}^t \leq 2t\left(Q_{h-1,i-1} + Q_{h-1,i} \cdot \Pr\left[\mathcal{B}(\Delta_+, q) \geq (\Delta_+)^\delta\right]\right).$$

Finally, using a simple union bound we get that for $t > \Delta_+$ it holds that

$$Q_{h,i}^t \leq t \cdot Q_{h-1,i}. \tag{18}$$

The above follows by a line of arguments similar to those we used for (17) and by noting that if $\texttt{deg}(r(T_\xi^h)) \geq \Delta_+$, then the root of T_ξ^h is non-mixing.

We are bounding $Q_{h,i}$ by using (17), (18) and Lemma 22. We have that

$$\begin{aligned}
Q_{h,i} &= \sum_{t=0}^{n} Q_{h,i}^t \xi_t \\
&= Q_{h-1,i-1} \cdot \sum_{t=0}^{(\Delta_+)^\delta} t \cdot \xi_t + 2Q_{h-1,i-1} \cdot \sum_{t=(\Delta_+)^\delta+1}^{\Delta_+} t \cdot \xi_t + \\
&\quad +2Q_{h-1,i} \cdot \Pr\left[\mathcal{B}(\Delta_+, q) \geq (\Delta_+)^\delta\right] \cdot \sum_{t=(\Delta_+)^\delta+1}^{\Delta_+} t \cdot \xi_t + Q_{h-1,i} \cdot \sum_{t\geq(\Delta_+)+1} t \cdot \xi_t \\
&\leq 2Q_{h-1,i-1} \sum_{t=0}^{\Delta_+} t \cdot \xi_t + Q_{h-1,i} \left(2\Pr\left[\mathcal{B}(\Delta_+, q) \geq (\Delta_+)^\delta\right] \sum_{t=(\Delta_+)^\delta}^{\Delta_+} t \cdot \xi_t \right. \\
&\qquad \left. + \sum_{t\geq(\Delta_+)+1} t \cdot \xi_t \right) \\
&\leq 2d_\xi \cdot Q_{h-1,i-1} + Q_{h-1,i} \left(2d_\xi \cdot \Pr\left[\mathcal{B}(\Delta_+, q) \geq (\Delta_+)^\delta\right] + \sum_{t\geq(\Delta_+)+1} t \cdot \xi_t \right).
\end{aligned} \tag{19}$$

The following lemma uses (19) to derive an upper bound on $Q_{h,i}$.

► **Lemma 23.** *Let $h, \beta, \mathcal{C}$ be as in the statement of Proposition 14. Also, let $\lambda \in (0,1)$ and $\theta' > 1$ be a fixed numbers such that $\beta(1-\theta') < -1$ and $\lambda\theta' < 1$. Then for $i = \lambda h$ and $Q_{h,i}$ that satisfies the inequality in (19), it holds that*

$$Q_{h,i} \leq \exp\left[-(1-\lambda\theta')\cdot\mathcal{C}\cdot h\right]. \tag{20}$$

The proof of Lemma 23 appears in Section 8.2

The proposition follows by using the above lemma and setting $\lambda = (1-\zeta)$ and $\theta' = \theta$, where ζ and θ are defined in the statement of Proposition 14.

8.1 Proof of Lemma 22

Let q_{h-1} be the probability for each child of $r(\mathcal{T}^h_\xi)$ to be non-mixing. Conditional that $r(\mathcal{T}^h_\xi)$ has degree t, the number of non-mixing children of $r(\mathcal{T}^h_\xi)$ is binomially distributed with parameters, t, q_{h-1}, i.e. $\mathcal{B}(t, q_{h-1})$. Letting $Q^M_{h,i} = Pr\left[\mathcal{T}^h_\xi \notin \mathcal{A}_{h,\zeta} \,\middle|\, r\left(T^h_\xi\right) \text{ is mixing}\right]$ and $Q^N_{h,i} = Pr\left[\mathcal{T}^h_\xi \notin \mathcal{A}_{h,\zeta} \,\middle|\, r\left(T^h_\xi\right) \text{ is not mixing}\right]$, it holds that

$$\begin{aligned} Q^t_{h,i} &\leq \sum_{j=0}^{(\Delta_+)^\delta} \binom{t}{j} q^j_{h-1}(1-q_{h-1})^{t-j}\left[(t-j)Q^M_{h-1,i-1} + jQ^N_{h-1,i-1}\right] + \\ &\quad + \sum_{j=(\Delta_+)^\delta+1}^{t} \binom{t}{j} q^j_{h-1}(1-q_{h-1})^{t-j}\left[(t-j)Q^M_{h-1,i} + jQ^N_{h-1,i}\right]. \end{aligned}$$

Using the standard equality that $(t-j)\binom{t}{j} = t\binom{t-1}{j}$, we get that

$$\begin{aligned} Q^t_{h,i} &\leq t(1-q_{h-1})Q^M_{h-1,i-1}\sum_{j=0}^{(\Delta_+)^\delta}\binom{t-1}{j}q^j_{h-1}(1-q_{h-1})^{t-1-j} \\ &\quad + tq_{h-1}Q^N_{h-1,i-1}\sum_{j=1}^{(\Delta_+)^\delta}\binom{t-1}{j-1}q^{j-1}_{h-1}(1-q_{h-1})^{t-j} \\ &\quad + t(1-q_{h-1})Q^M_{h-1,i}\sum_{j=(\Delta_+)^\delta+1}^{t-1}\binom{t-1}{j}q^j_{h-1}(1-q_{h-1})^{t-1-j} \\ &\quad + tq_{h-1}Q^N_{h-1,i}\sum_{j=(\Delta_+)^\delta+1}^{t}\binom{t-1}{j-1}q^{j-1}_{h-1}(1-q_{h-1})^{t-j}. \end{aligned}$$

It is not hard to see that for any h, i it holds that $q_h Q^N_{h,i} \leq Q_{h,i}$ and $(1-q_h)Q^M_{h,i} \leq Q_{h,i}$. Using these two inequalities we get that

$$\begin{aligned} Q^t_{h,i} &\leq tQ_{h-1,i-1}\left(\Pr\left[\mathcal{B}(t-1,q_{h-1}) \leq (\Delta_+)^\delta\right] + \Pr\left[\mathcal{B}(t-1,q_{h-1}) \leq (\Delta_+)^\delta - 1\right]\right) \\ &\quad + tQ_{h-1,i}\left(\Pr\left[\mathcal{B}(t-1,q_{h-1}) \geq (\Delta_+)^\delta + 1\right] + \Pr\left[\mathcal{B}(t-1,q_{h-1}) \geq (\Delta_+)^\delta\right]\right) \\ &\leq 2tQ_{h-1,i-1} + 2tQ_{h-1,i}\Pr\left[\mathcal{B}(t-1,q_{h-1}) \geq (\Delta_+)^\delta\right]. \end{aligned} \tag{21}$$

Note that that $\Pr\left[\mathcal{B}(t-1,q_{h-1}) \geq (\Delta_+)^\delta\right]$ is increasing with t. That is, for $t \leq \Delta_+$ it holds that

$$\Pr\left[\mathcal{B}(t-1,q_{h-1}) \geq (\Delta_+)^\delta\right] \quad \leq \quad \Pr\left[\mathcal{B}(\Delta_+,q_{h-1}) \geq (\Delta_+)^\delta\right]. \tag{22}$$

At this point we observe that the quantity q, defined in Definition 8, is an upper bound for q_h, for every h. This follows by an inductive argument, i.e. induction on h the number of levels of $\mathcal{T}^h_\xi$.

Clearly, for $h = 0$, the assertion is true. The tree with zero levels consists of only one vertex, which is a leaf. By default the leaves are mixing vertices, i.e. the probability of a leaf to be non-mixing is zero. Since $q \in [0, 3/4)$, q is an upper bound for the vertex to be non-mixing.

Given $h > 0$, assume that the assertion is true for $\mathcal{T}^{h'}_\xi$, for any $h' \leq h$. We are going to show that this is true for T^h_ξ. Let $\mathbf{N}$ be the number of non-mixing children of the root of T^h_ξ. It holds that

$$\Pr[r(\mathcal{T}^h_\xi) \text{ is non-mixing}] \leq \Pr[\mathtt{deg}(r(\mathcal{T}^h_\xi)) > \Delta_+] + \Pr[\mathbf{N} > (\Delta_+)^\delta | \mathtt{deg}(r(\mathcal{T}^h_\xi)) \leq \Delta_+].$$

Given that $\mathtt{deg}(r(\mathcal{T}^h_\xi)) = D$, for some integer $D \geq 0$, $\mathbf{N}$ is a binomial variable with parameters D, q_{h-1}. Due to our induction hypothesis it holds that $q_{h-1} < q$. Since we have conditioned that $D < \Delta_+$, it is clear that $\mathbf{N}$ is dominated by a binomial variable with parameters Δ_+, q, that is

$$\begin{aligned}\Pr[r(\mathcal{T}^h_\xi) \text{ is non-mixing}] &\leq \Pr[\mathtt{deg}(r(\mathcal{T}^h_\xi)) > \Delta_+] + \Pr[\mathcal{B}(\Delta_+, q) > (\Delta_+)^\delta] \\ &\leq \sum_{i \geq \Delta_+} \xi_i + \Pr[\mathcal{B}(\Delta_+, q) > (\Delta_+)^\delta] \leq q,\end{aligned}$$

where the last inequality follows from the definition of q, i.e. in Definition 8. The above inequality with (22) imply that

$$\Pr\left[\mathcal{B}(\Delta_+, q_{h-1}) \geq (\Delta_+)^\delta\right] \leq \Pr\left[\mathcal{B}(\Delta_+, q) \geq (\Delta_+)^\delta\right],$$

as $\mathcal{B}(\Delta_+, q_{h-1})$ is stochastically dominated by $\mathcal{B}(\Delta_+, q)$, since, $q_{h-1} \leq q$, for any h.

The lemma follows by plugging the above inequality into (21).

8.2 Proof of Lemma 23

We are going to use induction to prove the lemma. First we are going to show that if (20) is true for some $h > 1$ then it is also true for $h + 1$. Let $\lambda = \frac{i}{h}$, $\lambda^- = \frac{i-1}{h-1}$ and $\lambda^+ = \frac{i}{h-1}$. We rewrite (19) in terms of λ, λ^+ and λ^- as follows:

$$\begin{aligned}&Q_{\{h,\lambda h\}} \\ &\leq 2d \cdot Q_{\{h-1,\lambda^-(h-1)\}} + Q_{\{h-1,\lambda^+(h-1)\}} \left(2d \Pr\left[\mathcal{B}(\Delta_+, q) \geq (\Delta_+)^\delta\right] + \sum_{t \geq (\Delta_+)+1} t \cdot \xi_t\right).\end{aligned} \tag{23}$$

Using the induction hypothesis and noting that $\lambda^- = \lambda - \frac{1-\lambda}{h-1}$ we have that

$$\begin{aligned}Q_{\{h-1,\lambda^-(h-1)\}} &\leq \exp\left[-(1-\theta\lambda^-)(h-1)\mathcal{C}\right] \\ &\leq \exp\left[-\left(1-\theta'\left(\lambda - \frac{1-\lambda}{h-1}\right)\right)(h-1)\mathcal{C}\right] \\ &\leq \exp\left[-(1-\theta'\lambda)(h-1)\mathcal{C}\right] \cdot \exp\left[-\theta'(1-\lambda)\mathcal{C}\right] \\ &\leq \exp\left[-(1-\theta'\lambda)\, h\, \mathcal{C}\right] \cdot \exp\left[(1-\theta')\mathcal{C}\right].\end{aligned}$$

As far as $Q_{\{h-1,i\}}$ is regarded, we use the fact that $\lambda^+ = \lambda + \frac{\lambda}{h-1}$ and we get that

$$\begin{aligned}
Q_{\{h-1,\lambda^+\cdot(h-1)\}} &\leq \exp\left[-(1-\theta'\lambda^+)(h-1)\mathcal{C}\right] \\
&\leq \exp\left[-\left(1-\theta'\lambda-\frac{\theta'\lambda}{h-1}\right)(h-1)\mathcal{C}\right] \\
&\leq \exp\left[-\left(1-\theta'\lambda\right)(h-1)\mathcal{C}\right]\cdot\exp\left[\theta'\lambda\mathcal{C}\right] \\
&\leq \exp\left[-\left(1-\theta'\lambda\right)h\mathcal{C}\right]\exp\left[\mathcal{C}\right] . \qquad (24)
\end{aligned}$$

Substituting the bounds for $Q_{\{h-1,i-1\}}, Q_{\{h-1,i\}}$ above into (23) we get that

$$\begin{aligned}
&Q_{\{h,\lambda h\}} \\
&\leq \exp\left[-\left(1-\theta'\lambda\right)h\mathcal{C}\right] \\
&\quad\times\left(2d\cdot\exp\left[\left(1-\theta'\right)\mathcal{C}\right]+\exp\left(\mathcal{C}\right)\left(2d\Pr\left[\mathcal{B}(\Delta_+,q)\geq(\Delta_+)^{\delta}\right]+\sum_{t\geq(\Delta_+)+1}t\cdot\xi_t\right)\right).
\end{aligned}$$

From to our assumption that $\beta(1-\theta') < -1$ it is direct that

$$2d\cdot\exp\left[\left(1-\theta'\right)\mathcal{C}\right] = 2d^{1+\beta(1-\theta')} \leq 1/5.$$

Also due to our assumptions about Δ_+, δ we get that

$$\exp\left(\mathcal{C}\right)\left(2d\Pr\left[\mathcal{B}(\Delta_+,q)\geq(\Delta_+)^{\delta}\right]+\sum_{t\geq\Delta_++1}t\cdot\xi_t\right) \quad\leq\quad \frac{2}{5}.$$

Using the two bounds above (**??**) writes as follows:

$$Q_{\{h,\lambda h\}} \quad\leq\quad \exp\left[-\left(1-\theta'\cdot\lambda\right)h\mathcal{C}\right].$$

It remains to show the base of the induction, i.e the case $h = 1$. Since the leaves of the trees are, by default, mixing, for any fixed $\lambda \in (0,1)$ and $h = 1$ it holds that

$$Q_{\{h,\lambda\cdot h\}} \leq \Pr[deg(r(T)) \geq \Delta_+] = \sum_{t\geq\Delta_+}\xi_t \leq \exp\left[-2\mathcal{C}\right] \leq \exp\left[-\left(1-\theta'\cdot\lambda\right)\mathcal{C}\right],$$

as $\lambda, \theta > 0$ while $\lambda\cdot\theta' < 1$. The lemma follows.

Acknowledgement. The author of this work would like to thank Guilhem Semerjian for our communication and the discussion on the problem. Also, the author would like to thank Amin Coja-Oghlan for the discussions, his comments and the suggestions for improving the content of this work.

References

1 D. Achlioptas and A. Coja-Oghlan. Algorithmic barriers from phase transitions. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 793–802, Oct 2008.

2 Nayantara Bhatnagar, Allan Sly, and Prasad Tetali. Reconstruction threshold for the hardcore model. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, pages 434–447, 2010.

3 Nayantara Bhatnagar, Juan Carlos Vera, Eric Vigoda, and Dror Weitz. Reconstruction for colorings on trees. *SIAM J. Discrete Math.*, 25(2):809–826, 2011.

4 Christian Borgs, Jennifer T. Chayes, Elchanan Mossel, and Sébastien Roch. The kesten-stigum reconstruction bound is tight for roughly symmetric binary channels. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 518–530, 2006.

5 J. T. Chayes, L. Chayes, James P. Sethna, and D. J. Thouless. A mean field spin glass with short-range interactions. *Comm. Math. Phys.*, 106(1):41–89, 1986.

6 Amin Coja-Oghlan. A better algorithm for random k-sat. *SIAM J. Comput.*, 39(7):2823–2864, 2010.

7 Amin Coja-Oghlan, Charilaos Efthymiou, and Nor Jaafari. Local convergence of random graph colorings. *CoRR*, abs/1501.06301, 2015.

8 Constantinos Daskalakis, Elchanan Mossel, and Sébastien Roch. Optimal phylogenetic reconstruction. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 159–168, 2006.

9 Martin E. Dyer, Abraham D. Flaxman, Alan M. Frieze, and Eric Vigoda. Randomly coloring sparse random graphs with fewer colors than the maximum degree. *Random Struct. Algorithms*, 29(4):450–465, 2006.

10 Charilaos Efthymiou. A simple algorithm for random colouring $G(n, d/n)$ using $(2 + \epsilon)d$ colours. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 272–280, 2012.

11 Charilaos Efthymiou. MCMC sampling colourings and independent sets of $G(n, d/n)$ near uniqueness threshold. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 305–316, 2014.

12 Charilaos Efthymiou. Reconstruction/non-reconstruction thresholds for colourings of general galton-watson trees. *CoRR*, abs/1406.3617, 2014.

13 Charilaos Efthymiou. Switching colouring of g(n, d/n) for sampling up to gibbs uniqueness threshold. In *Algorithms – ESA 2014 – 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 371–381, 2014.

14 William Evans, Claire Kenyon, Yuval Peres, and Leonard J. Schulman. Broadcasting on trees and the ising model. *Ann. Appl. Probab.*, 10(2):410–433, 05 2000.

15 Antoine Gerschenfeld and Andrea Montanari. Reconstruction for models on random graphs. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 194–204, 2007.

16 Geoffrey R Grimmett and Colin JH McDiarmid. On colouring random graphs. *Mathematical Proceedings of the Cambridge Philosophical Society Mathematical Proceedings of the Cambridge Philosophical Society*, 77(02):311–324, 1975.

17 Florent Krzakała, Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *Proceedings of the National Academy of Sciences*, 104(25):10318–10323, 2007.

18 Marc Mézard, Giorgio Parisi, and Riccardo Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002.

19 Michael Molloy. The freezing threshold for k-colourings of a random graph. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19–22, 2012*, pages 921–930, 2012.

20 Andrea Montanari, Ricardo Restrepo, and Prasad Tetali. Reconstruction and clustering in random constraint satisfaction problems. *SIAM J. Discrete Math.*, 25(2):771–808, 2011.

21 E. Mossel. Phase transitions in phylogeny. *Trans. Amer. Math. Soc.*, 356(6):2379–2404 (electronic), 2004.

22 E. Mossel and Y. Peres. Information flow on trees. *Ann. Appl. Probab.*, 13(3):817–844, 2003.

23 Elchanan Mossel. Reconstruction on trees: Beating the second eigenvalue. *Ann. Appl. Probab.*, 11(1):285–300, 02 2001.

24 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press„ 1995.

25 Guilhem Semerjian. On the freezing of variables in random constraint satisfaction problems. *Journal of Statistical Physics*, 130(2):251– 293, January 2008.

26 Allan Sly. Reconstruction of random colourings. *Communications in Mathematical Physics*, 288(3):943–961, 2009.

27 Yitong Yin and Chihao Zhang. Sampling colorings almost uniformly in sparse random graphs. *CoRR*, abs/1503.03351, 2015.

A Randomized Online Quantile Summary in $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ Words

David Felber and Rafail Ostrovsky

UCLA Computer Science Department
Los Angeles, CA, USA
{dvfelber,rafail}@cs.ucla.edu

Abstract

A quantile summary is a data structure that approximates to ε-relative error the order statistics of a much larger underlying dataset.

In this paper we develop a randomized online quantile summary for the cash register data input model and comparison data domain model that uses $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words of memory. This improves upon the previous best upper bound of $O(\frac{1}{\varepsilon}\log^{3/2}\frac{1}{\varepsilon})$ by Agarwal et al. [1]. Further, by a lower bound of Hung and Ting [4] no deterministic summary for the comparison model can outperform our randomized summary in terms of space complexity. Lastly, our summary has the nice property that $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words suffice to ensure that the success probability is $1-e^{-\text{poly}(1/\varepsilon)}$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.3 Probability and Statistics

Keywords and phrases order statistics, data stream, streaming algorithm

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.775

1 Introduction

A quantile summary S is a fundamental data structure that summarizes an underlying dataset X of size n, in space much less than n. Given a query ϕ, S returns a sample y of X such that the rank of y in X is (probably) approximately ϕn. Quantile summaries are used in sensor networks to aggregate data in an energy-efficient manner and in database query optimizers to generate query execution plans.

Quantile summaries have been developed for a variety of different models and metrics. The data input model we consider is the standard online cash register streaming model, in which a new item is added to the dataset at each new timestep, and the total number of items is not known until the end. The data domain model we consider is the comparison model, in which stream items come from an arbitrary ordered domain (and specifically, not necessarily from the integers).

Formally, our quantile summary problem is defined over a totally ordered domain $\mathcal{D}$ and by an error parameter $\varepsilon \leq 1/2$. There is a dataset X that is initially empty. Time occurs in discrete steps. In timestep t, stream item x_t arrives and is then processed, and then any quantile queries ϕ in that step are received and processed. To be definite, we pick the first timestep to be 1. We write X_t or $X(t)$ for the t-item prefix stream $x_1 \dots x_t$ of X. The goal is to maintain at all times t a summary S_t of the dataset X_t that, given any query ϕ in $(0,1]$, can return a sample $y = y(\phi)$ so that $|R(y, X_t) - \phi t| \leq \varepsilon t$, where $R(a, Z)$ is the *rank of item a in set Z*, defined as $|\{z \in Z : z \leq a\}|$. For randomized summaries, we only require that $\forall t \forall \phi$, $P(|R(y, X_t) - \phi t| \leq \varepsilon t) \geq 2/3$; that is, y's rank is only probably close to ϕt, not definitely close. In fact, it will be easier to deal with the rank directly, so we define $\rho = \phi t$ and use that in what follows.

1.1 Previous work

The two most directly relevant pieces of prior work ([1, 2] and [6]) are randomized online quantile summaries for the cash register/comparison model. Aside from oblivious sampling algorithms (which require storing $\Omega(1/\varepsilon^2)$ samples) the only other such work of which we are aware is an approach by Wang, Luo, Yi, and Cormode [11] that combines the methods of [1, 2] and [6] into a hybrid with the same space bound as [1, 2].

The newer of the two is that of Agarwal, Cormode, Huang, Phillips, Wei, and Yi [1, 2]. Among other results, Agarwal et al. develop a randomized online quantile summary for the cash register/comparison model that uses $O(\frac{1}{\varepsilon}\log^{3/2}\frac{1}{\varepsilon})$ words of memory. This summary has the nice property that any two such summaries can be combined to form a summary of the combined underlying dataset without loss of accuracy or increase in size.

The earlier such summary is that of Manku, Rajagopalan, and Lindsay [6], which uses $O(\frac{1}{\varepsilon}\log^2\frac{1}{\varepsilon})$ space. At a high level, their algorithm downsamples the input stream in a non-uniform way and feeds the downsampled stream into a deterministic summary, while periodically adjusting the downsampling rate.

We note here for those familiar with the result of Manku et al. that, while our algorithm at a high level may appear similar, there are important differences. We defer a discussion of similarities and differences to Section 4 after the presentation of our algorithm in Section 3.

For the comparison model, the best deterministic online summary to date is the (GK) summary of Greenwald and Khanna [3], which uses $O(\frac{1}{\varepsilon}\log\varepsilon n)$ space. This improved upon a deterministic (MRL) summary of Manku, Rajagopalan, and Lindsay [5] and a summary implied by Munro and Paterson [7], which use $O(\frac{1}{\varepsilon}\log^2\varepsilon n)$ space.

A more restrictive domain model than the comparison model is the bounded universe model, in which elements are drawn from the integers $\{1,\ldots,u\}$. For this model there is a deterministic online summary by Shrivastava, Buragohain, Agrawal, and Suri [9] that uses $O(\frac{\log u}{\varepsilon})$ space.

Not much exists in the way of lower bounds for this problem. There is a simple lower bound of $\Omega(1/\varepsilon)$ which intuitively comes from the fact that no one sample can satisfy more than $2\varepsilon n$ different rank queries. For the comparison model, Hung and Ting [4] developed a deterministic $\Omega(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ lower bound. Whether this bound can be extended to hold for our weaker probabilistic guarantee, and whether our algorithm can be modified to satisfy the stronger deterministic guarantee, are both open questions.

1.2 Our contributions

In the next section we describe a simple $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ streaming summary that is online except that it requires n to be given up front and that it is unable to process queries until it has seen a constant fraction of the input stream. This simple summary is not new (it is mentioned in Wang et al. [11], for example) but the discussion provides exposition for Section‘3, in which we develop this summary into a fully online summary with the same asymptotic space complexity that can answer queries at any point in time. We close in Section 4 by examining the similarities and differences between our summary and previous work and discuss a design approach for similar streaming problems.

2 A simple streaming summary

Before we describe the algorithm we must first describe its two main components in a bit more detail than was used in the introduction. The two components are Bernoulli sampling and the GK summary [3].

2.1 Bernoulli sampling

Bernoulli sampling downsamples a stream X of size n to a sample stream S by choosing to include each next item into S with independent probability m/n. (As stated this requires knowing the size of X in advance.) At the end of processing X, the expected size of S is m, and the expected rank of any sample y in S is $E(R(y,S)) = \frac{m}{n}R(y,X)$. In fact, for any times $t \leq n$ and partial streams X_t and S_t, where S_t is the sample stream of X_t, we have $E(|S_t|) = mt/n$ and $E(R(y,S_t)) = \frac{m}{n}R(y,X_t)$. To generate an estimate for $R(y,X_t)$ from S_t we use $\hat{R}(y,X_t) = \frac{n}{m}R(y,S_t)$. The following theorem bounds the probability that S is very large or that $\hat{R}(y,X_t)$ is very far from $R(y,X_t)$. A generalization of this theorem is due Vapnik and Chervonenkis [10]; the proof of this special case is a simple known application of Chernoff bounds.

▶ **Theorem 1.** *For all times $t \geq n/64$,*

$$P(|S_t| > 2tm/n) \; < \; \exp(-m/192)$$

Further, for all times $t \geq n/64$ and items y,

$$P(|\hat{R}(y,X_t) - R(y,X_t)| > \varepsilon t/8) \; < \; 2\exp(-\varepsilon^2 m/12288)$$

Proof. For the first part,

$$P(|S_t| > 2tm/n) \; < \; \exp(-tm/3n) \; < \; \exp(-m/192)$$

since $t \geq n/64$. For the second part,

$$P(|\hat{R}(y,X_t) - R(y,X_t)| > \varepsilon t/8) \; = \; P(|R(y,S_t) - E(R(y,S_t))| > \varepsilon tm/8n)$$

The Chernoff bound is

$$P(|R(y,S_t) - E(R(y,S_t))| > \delta E(R(y,S_t))) \; < \; 2\exp(-\min\{\delta,\delta^2\}E(R(y,S_t))/3)$$

Here, $\delta = \varepsilon t/8R(y,S_t)$, so

$$P \; < \; 2\exp(-\varepsilon^2 t^2 m/192nE(R(y,S_t))) \; \leq \; 2\exp(-\varepsilon^2 m/12288)$$

finishing the proof. ◀

This means that, given any $1 \leq \rho \leq t$, if we choose to return the sample $y \in S_t$ with $R(y,S_t) = \rho m/n$, then $R(y,X_t)$ is likely to be close to ρ, as long as m is $\Omega(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon})$.

2.2 GK summary

The GK summary is a deterministic summary that can answer queries to relative error over any portion of the received stream. Let G_t be the summary after inserting the first t items X_t from stream X into G. Greenwald and Khanna guarantee in [3] that with only $O(\frac{1}{\varepsilon}\log(\varepsilon t))$ words, given any $1 \leq \rho \leq t$, G_t can return a sample $y \in X_t$ so that $|R(y,X_t) - \rho| \leq \varepsilon t/8$. We call this the *GK guarantee.*

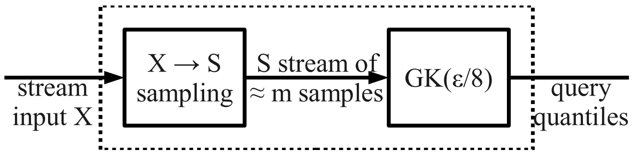


Figure 1 The big picture.

2.3 A simple streaming summary

We combine Bernoulli sampling with the GK summary by downsampling the input data stream X to a sample stream S and then feeding S into a GK summary G. It looks like in Figure 1.

The key reason this gives us a small summary is that we never need to store S; each time we sample an item into S we immediately feed it into G. Therefore, we only use as much space as $G(S(X_t))$ uses. In particular, for $m = O(\text{poly}(1/\varepsilon))$, we use only $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ words. To answer a query ρ for X_t, we scale ρ by m/n, ask $G(S(X_t))$ for that, and return the resulting sample y.

We formalize this intuition in the following lemma, which combines the ideas in the proof of Theorem 1 with the GK guarantee to yield approximation and correctness guarantees.

▶ **Lemma 2.** *Fix some time $t \geq n/64$ and some rank $\rho \leq t$, and consider querying $G(S(X_t))$ with $q = \min\{\rho m/n, |S|\}$, obtaining y as the result.*

Say that $S = S(X_t)$ is good *if $||S| - mt/n| \leq \varepsilon mt/8n$ and if none of the first $\leq mt/n$ samples z in S has $|R(z, S) - \frac{m}{n} R(z, X_t)| > \varepsilon mt/8n$.*

If S is good then $|R(y, X_t) - \rho| \leq \varepsilon t/2$.

Further, if $m \geq \frac{400000 \ln 1/\varepsilon}{\varepsilon^3}$ then $P(S$ is not good$) \leq \varepsilon^3 e^{-1/\varepsilon}/8$.

Proof. First, by the GK guarantee, $G(S)$ returns some item y with $|R(y, S) - q| \leq \varepsilon t/8$. If S is good, then $|q - \rho m/n| \leq \varepsilon mt/8n$, and also $|R(y, S) - \frac{m}{n} R(y, X_t)| \leq \varepsilon mt/8n$. By the triangle inequality, $|\frac{m}{n} R(y, X_t) - \rho m/n| \leq 3\varepsilon mt/8n$. Equivalently, $|R(y, X_t) - \rho| \leq 3\varepsilon t/8$.

Now, following the proof of Theorem 1, we have that

$$P(||S_t| - mt/n| > \varepsilon mt/8n) \ < \ 2 \exp(-\varepsilon^2 m/12288)$$

and also for each of the first $\leq m$ samples z that

$$P(|R(z, S) - \tfrac{m}{n} R(z, X_t)| > \varepsilon mt/8n) \ < \ 2 \exp(-\varepsilon^2 m/12288)$$

By the union bound, $P(S$ is not good$) \leq 4m \exp(-\varepsilon^2 m/12288)$. Choosing $m \geq \frac{400000 \ln 1/\varepsilon}{\varepsilon^3}$ suffices to bound this quantity by $\varepsilon^3 e^{-1/\varepsilon}/8$. ◀

2.4 Caveats

There are two serious issues with this summary. The first is that it requires us to know the value of n in advance to perform the sampling. Also, as a byproduct of the sampling, we can only obtain approximation guarantees after we have seen at least $1/64$ (or at least some constant fraction) of the items. This means that while the algorithm is sufficient for approximating order statistics over streams stored on disk, more is needed to get it to work for online streaming applications, in which (1) the stream size n is not known in advance, and (2) queries can be answered approximately at all times $t \leq n$ and not just when $t \geq n/64$.

Adapting this basic streaming summary idea to work online constitutes the next section and the bulk of our contribution. We start with a high-level overview of our online summary algorithm. In Section 3.1 we formally define an initial version of our algorithm whose expected size at any given time is $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words. In Section 3.2 we show that our algorithm guarantees that $\forall n \forall \rho,\ P(|R(y, X_n) - \rho| \leq \varepsilon n) \geq 1 - \exp(-1/\varepsilon)$. In Section 3.3 we discuss the slight modifications necessary to get a deterministic $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ space complexity, and also perform a time complexity analysis.

3 An online summary

Our algorithm works in *rows*, which are illustrated in figure 2. Row r is a summary of the first $2^r 32m$ stream items. Since we don't know how many items will actually be in the stream, we can't start all of these rows running at the outset. Therefore, we start each row $r \geq 1$ once we have seen $1/64$ of its total items. However, since we can't save these items for every row we start, we need to construct an approximation of this fraction of the stream, which we do by using the summary of the previous row, and join this approximating stream with the new items that arrive while the row is live. We then wait until the row has seen a full half of its items before we permit it to start answering queries; this dilutes the influence of approximating the $1/64$ of its input that we couldn't store.

Operation within a row is very much like the operation of our fixed-n streaming summary. We feed the joint approximate prefix + new item stream through a Bernoulli sampler to get a sample stream, which is then fed into a GK summary (which is stored). After row r has seen half of its items, its GK summary becomes the one used to answer quantile queries. When row $r + 1$ has seen $1/64$ of *its* total items, row r generates an approximation of those items from its GK summary and feeds them as a stream into row $r + 1$.

Row 0 is slightly different in order to bootstrap the algorithm. There is no join step since there is no previous row to join. Also, row 0 is active from the start. Lastly, we get rid of the sampling step so that we can answer queries over timesteps $1 \ldots m/2$.

After the first $32m$ items, row 0 is no longer needed, so we can clean up the space used by its GK summary. Similarly, after the first $2^r 32m$ items, row r is no longer needed. The upshot of this is that we never need storage for more than six rows at a time. Since each GK summary uses $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words, the six live GK summaries also only use $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ words.

Our error analysis, on the other hand, will require us to look back as many as $\Theta(\log 1/\varepsilon)$ rows to ensure our approximation guarantee. We stress that we will not need to actually *store* these $\Theta(\log 1/\varepsilon)$ rows for our guarantee to hold; we will only need that they didn't have any bad events (as will be defined) when they *were* alive.

3.1 Algorithm description

Our algorithm works in rows. Each row r has its own copy G_r of the GK algorithm that approximates its input to $\varepsilon/8$ relative error. For each row r we define several streams: A_r is the prefix stream of row r, B_r is its suffix stream, R_r is its prefix stream replacement (generated by the previous row), J_r is the joint stream R_r followed by B_r, S_r is its sample stream, and Q_r is a one-time stream generated from G_r by querying it with ranks $\rho_1 \ldots \rho_{8/\varepsilon}$, where $\rho_q = q(\varepsilon/8)(m/32)$ for $r \geq 1$ and $\rho_q = q\varepsilon m/8$ for $r = 0$.

The prefix stream $A_r = X(2^{r-1}m)$ for row $r \geq 1$, importantly, is not directly received by row r. Instead, at the end of timestep $2^{r-1}m$, row $r-1$ generates Q_{r-1} and duplicates each of those $8/\varepsilon$ items $2^{r-1}\varepsilon m/8$ times to get the replacement prefix R_r, which is then immediately fed into row r before timestep $2^{r-1}m+1$ begins.

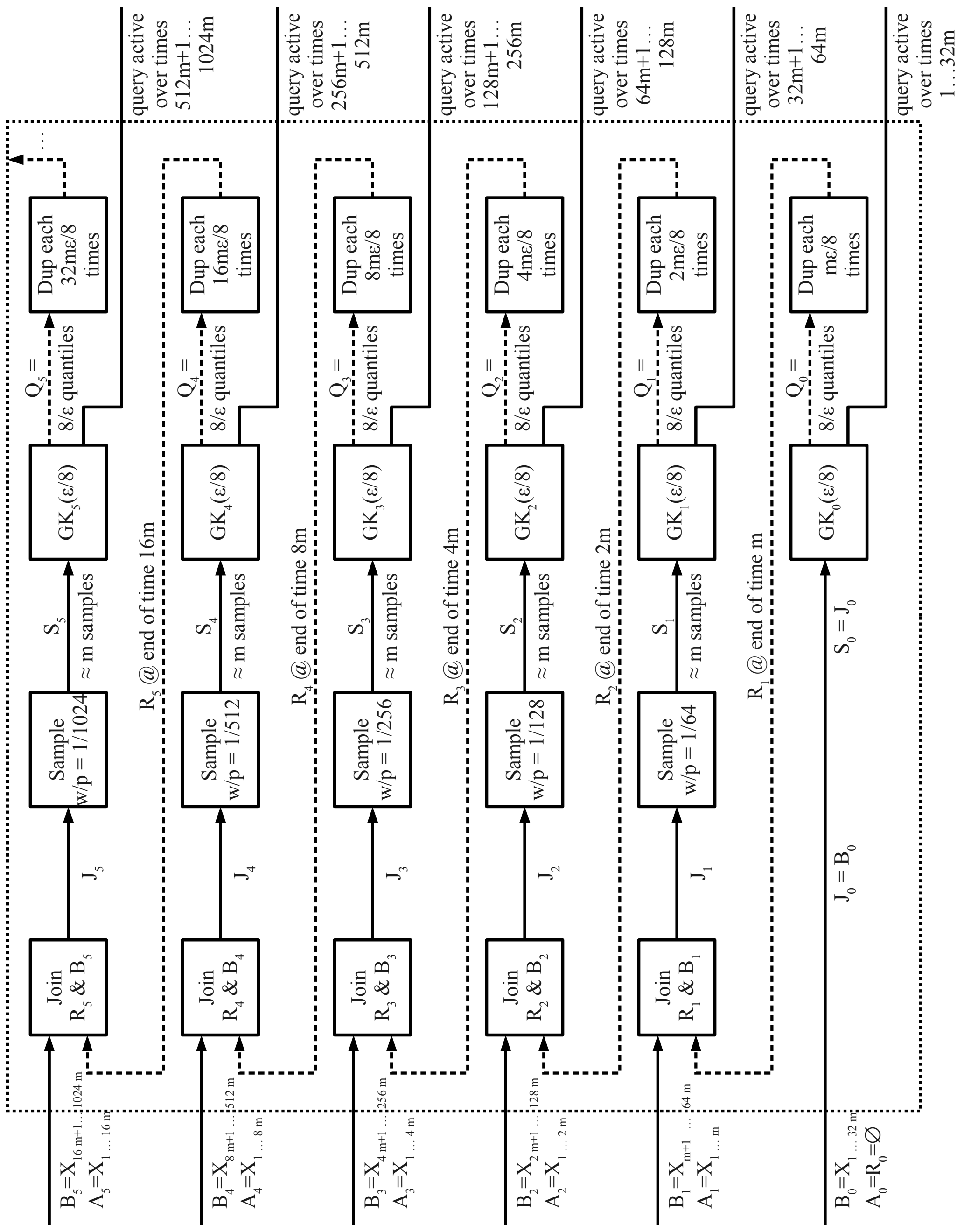


Figure 2 Each row r has its own copy G_r of the GK algorithm that approximates its input to $\varepsilon/8$ relative error. A_r is the prefix stream of row r, B_r is its suffix stream, R_r is its prefix stream replacement (generated by the previous row), J_r is the joint stream R_r followed by B_r, S_r is its sample stream, and Q_r is a one-time stream generated from G_r at time $2^r m$ to get the replacement prefix R_{r+1}.

Each row can be *live* or not and *active* or not. Row 0 is live in timesteps $1 \dots 32m$ and row $r \geq 1$ is live in timesteps $2^{r-1}m{+}1 \dots 2^r 32m$. Live rows require space; once a row is no longer live we can free up the space it used. Row 0 is active in timesteps $1 \dots 32m$ and row $r \geq 1$ is active in timesteps $2^r 16m{+}1 \dots 2^r 32m$. This definition means that exactly one row $r(t)$ is active in any given timestep t. Any queries that are asked in timestep t are answered by $G_{r(t)}$. Given query ρ, we ask $G_{r(t)}$ for $\rho/2^{r(t)}32$ (if $r \geq 1$) or for ρ (if $r = 0$) and return the result.

At each timestep t, when item x_t arrives, it is fed as the next item in the suffix stream B_r for each live row r. B_r joined with R_r defines the joined input stream J_r. For $r \geq 1$, J_r is downsampled to the sample stream S_r by sampling each item independently with probability $1/2^r 32$. For row 0, no downsampling is performed, so $S_0 = J_0$. Lastly, S_r is fed into G_r.

Figure 2 shows the operation of and the communication between the first six rows. Solid arrows indicate continuous streams and dashed arrows indicate one-time messages. Algorithm 1 is a pseudocode listing of the algorithm.

```
Initially, allocate space for $G_0$. Mark row 0 as live and active.
**for** $t = 1, 2, \dots$ **do**
    **foreach** *live row* $r \geq 0$ **do**
        **with probability** $1/2^r 32$ **do**
            Insert $x_t$ into $G_r$.
    **if** $t = 2^{r-1}m$ *for some* $r \geq 1$ **then**
        Allocate space for $G_r$. Mark row $r$ as live.
        Query $G_{r-1}$ with $\rho_1 \dots \rho_{8/\varepsilon}$ to get $y_1 \dots y_{8/\varepsilon}$.
        **for** $q = 1 \dots 8/\varepsilon$ **do**
            **for** $1 \dots 2^{r-1}\varepsilon m/8$ **do**
                **with probability** $1/2^r 32$ **do**
                    Insert $y_q$ into $G_r$.
    **if** $t = 2^r 16m$ *for some* $r \geq 1$ **then**
        Mark row $r$ as active. Unmark row $r{-}1$ as active.
    **if** $t = 2^r 32m$ *for some* $r \geq 0$ **then**
        Unmark row $r$ as live. Free space for $G_r$.
**on** *query* $\rho$ **do**
    Let $r = r(t)$ be the active row.
    Query $G_r$ for rank $\rho/2^r 32$ (if $r \geq 1$) or for rank $\rho$ (if $r = 0$).
    Return the result.
```

Algorithm 1. Procedural listing of the algorithm in Section 3.1.

3.2 Error analysis

Define $C_r = x(2^r 32m{+}1), x(2^r 32m{+}2), \dots$ and Y_r to be R_r followed by B_r and then C_r. That is, Y_r is just the continuation of J_r for the entire length of the input stream.

Fix some time t. All of our claims will be relative to time t; that is, if we write S_r we mean $S_r(t)$. Our error analysis proceeds as follows. We start by proving that $R(y, Y_r)$ is a good approximation of $R(y, Y_{r-1})$ when certain conditions hold for S_{r-1}. By induction, this means that $R(y, Y_r)$ is a good approximation of $R(y, X{=}Y_0)$ when the conditions hold for all of $S_0 \dots S_{r-1}$, and actually it's enough for the conditions to hold for just $S_{r-\log 1/\varepsilon} \dots S_{r-1}$ to get a good approximation. Having proven this claim, we then prove that the result $y = y(\rho)$

of a query to our summary has $R(y,X)$ close to ρ. Lastly, we show that $m = O(\text{poly}(1/\varepsilon))$ suffices to ensure that the conditions hold for $S_{r-\log 1/\varepsilon} \ldots S_{r-1}$ with very high probability $(1-e^{-1/\varepsilon})$.

▶ **Lemma 3.** *Let α_r be the event that $|S_r| > 2m$ and let β_r be the event that any of the first $\leq 2m$ samples z in S_r has $|2^r 32 R(z,S_r) - R(z,Y_r)| > \varepsilon t/8$. Say that S_r is* good *if neither α_r nor β_r occur (or if $r = 0$).*

For all rows $r \geq 1$ such that $t \geq t_r = 2^{r-1}m$, and all for all items y, if S_{r-1} is good then we have that $|R(y,Y_r) - R(y,Y_{r-1})| \leq 2^r \varepsilon m$.

Proof. At the end of time t_r we have $Y_r(t_r) = R_r(t_r)$, which is each item $y(\rho_q)$ in Q_{r-1} duplicated $\varepsilon t_r/8$ times. If $S_{r-1}(t_r)$ is good then $|R(y(\rho_q), Y_{r-1}(t_r)) - 2^{r-1}32\rho_q| \leq \varepsilon t_r/2$ following Lemma 2.

Fix q so that $y(\rho_q) \leq y < y(\rho_{q+1})$, where $y(\rho_0)$ and $y(\rho_{1+8/\varepsilon})$ are defined to be $\inf \mathcal{D}$ and $\sup \mathcal{D}$ for completeness. Fixing q this way implies that $R(y, Y_r(t_r)) = 2^{r-1}32\rho_q$. By the above bound on $R(y(\rho_q), Y_{r-1}(t_r))$ we also have that

$$2^{r-1}32\rho_q - \varepsilon t_r/2 \;\leq\; R(y, Y_{r-1}(t_r)) \;<\; 2^{r-1}32\rho_{q+1} + \varepsilon t_r/2$$

Recalling that $\rho_q = q\varepsilon m/256$, these bounds imply that

$$|R(y, Y_r(t_r)) - R(y, Y_{r-1}(t_r))| \;\leq\; 2^r \varepsilon m$$

For each time t after t_r, the new item x_t changes the rank of y in both streams Y_r and Y_{r-1} by the same additive offset, so

$$|R(y, Y_r) - R(y, Y_{r-1})| \;=\; |R(y, Y_r(t_r)) - R(y, Y_{r-1}(t_r))| \;\leq\; 2^r \varepsilon m$$

yielding the lemma. ◀

By applying this lemma inductively we can bound the difference between Y_r and $X = Y_0$:

▶ **Corollary 4.** *For all $r \geq 1$ such that $t \geq t_r = 2^{r-1}m$, if all of $S_0(t_1), S_1(t_2), \ldots, S_{r-1}(t_r)$ are good, then $|R(y,Y_r) - R(y,X)| \leq 2 \cdot 2^r \varepsilon m$.*

To ensure that all of these S_i are good would require m to grow with n, which would be bad. Happily, it is enough to require only the last $\log_2 1/\varepsilon$ sample summaries to be good, since the other items we disregard constitute only a small fraction of the total stream.

▶ **Corollary 5.** *Let $d = \log_2 1/\varepsilon$. For all $r \geq 1$ such that $t \geq t_r = 2^{r-1}m$, if all of $S_{r-1}(t_r), \ldots, S_{r-d}(t_{r-d+1})$ are good, then $|R(y,Y_r) - R(y,X)| \leq 2^{r+2}\varepsilon m$.*

Proof. By Lemma 3 we have $|R(y,Y_r) - R(y,Y_{r-d})| \leq 2^{r+1}\varepsilon m$. At time $t \geq t_{r-d}$, Y_{r-d} and X share all except possibly the first $2^{(r-d)-1}m = 2^{r-1}m/2^d = 2^{r-1}\varepsilon m$ items. Thus

$$|R(y,Y_r) - R(y,X)| \leq |R(y,Y_r) - R(y,Y_{r-d})| + |R(y,Y_{r-d}) - R(y,X)| \leq 2^{r+1}\varepsilon m + 2^r \varepsilon m$$

proving the corollary. ◀

We now prove that if the last several sample streams were good then querying our summary will give us a good result.

▶ **Lemma 6.** *Let $d = \log_2 \frac{1}{\varepsilon}$ and $r = r(t)$. If all $S_r(t), S_{r-1}(t_r), \ldots, S_{r-d}(t_{r-d+1})$ are good, then querying our summary with rank ρ (= querying the active GK summary G_r with $\rho/2^r 32$ if $r \geq 1$, or with ρ if $r = 0$) returns $y = y(\rho)$ such that $|R(y,X) - \rho| \leq \varepsilon t$.*

Proof. For $r \geq 1$ we have by Corollary 5 that $|R(y, Y_r) - R(y, X)| \leq 2^{r+2}\varepsilon m \leq \varepsilon t/2$. We apply Lemma 2 once more at row r, which tells us that $|R(y, Y_r) - \rho| \leq \varepsilon t/2$, and combine these bounds with the triangle inequality.

For $r = 0$, the GK guarantee alone proves the lemma. ◀

Lastly, we prove that $m = O(\text{poly}(1/\varepsilon))$ suffices to ensure that all of $S_r(t)$, $S_{r-1}(t_r), \ldots,$ $S_{r-d}(t_{r-d+1})$ are good with probability at least $1 - e^{-1/\varepsilon}$.

▶ **Lemma 7.** *Let $d = \log_2 1/\varepsilon$ and $r = r(t)$. If $m \geq \frac{400000 \ln 1/\varepsilon}{\varepsilon^3}$ then all $S_r(t), S_{r-1}(t_r), \ldots,$ $S_{r-d}(t_{r-d+1})$ are good with probability at least $1 - e^{-1/\varepsilon}$.*

Proof. There are at most $1+\log_2 1/\varepsilon \leq 4/\varepsilon$ of these summary streams total. Lemma 2 and the union bound give us

$$P(\text{some } S_r \text{ is bad}) \leq \frac{4}{\varepsilon}\frac{\varepsilon^3}{8}e^{-1/\varepsilon} \leq e^{-1/\varepsilon}$$

which implies our claim. ◀

3.3 Space and time complexity

A minor issue with the algorithm is that, as written in section 3.1, we do not actually have a bound on the worst-case space complexity of the algorithm; we only have a bound on the space needed at any given point in time. This issue is due to the fact that there are low probability events in which $|S_r|$ can get arbitrarily large and the fact that over n items there are a total of $\Theta(\log n)$ sample streams. The space complexity of the algorithm is $O(\max |S_r|)$, and to bound this value with constant probability using the Chernoff bound appears to require that $\max |S_r| = \Omega(\log \log n)$, which is too big.

Fortunately, fixing this problem is simple. Instead of feeding every sample of S_r into the GK summary G_r, we only feed each next sample if G_r has seen $< 2m$ samples so far. That is, we deterministically restrict G_r to receiving only $2m$ samples. Lemmas 3 through 6 condition on the goodness of the sample streams S_r, which ensures that the G_r receive at most $2m$ samples each, and the claim of Lemma 7 is independent of the operation of G_r. Therefore, by restricting each G_r to receive at most $2m$ inputs we can ensure that the space complexity is deterministically $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ without breaking our error guarantees.

From a practical perspective, the assumption in the streaming setting is that new items arrive over the input stream X at a high rate, so both the worst-case per-item processing time as well as the amortized time to process n items are important. For our per-item time complexity, the limiting factor is the duplication step that occurs at the end of each time $t_r = 2^{r-1}m$, which makes the worst-case per-item processing time as large as $\Theta(n)$. Instead, at time t_r we could generate Q_{r-1} and store it in $O(1/\varepsilon)$ words, and then on each arrival $t = 2^{r-1}m+1 \ldots 2^r m$ we could insert both x_t and also the next item in R_r. By the time $t_{r+1} = 2t_r$ that we generate Q_r, all items in R_r will have been inserted into J_r. Thus the worst-case per-item time complexity is $O(\frac{1}{\varepsilon}T_{\text{GK}}^{\max})$, where $T_{\text{GK}}^{\max}$ is the worst-case per-item time to query or insert into one of our GK summaries. Over $2^r 32m$ items there are at most $2m$ insertions into any one GK summary, so the amortized time over n items in either case is $O(\frac{m \log(n/m)}{n} T_{\text{GK}})$, where T_{GK} is the amortized per-item time to query or insert into one of our GK summaries. Algorithm 2 includes the changes of this section.

4 Discussion

Our starting point is a very natural idea used in Manku et al. [6]: downsample the input stream and feed the resulting sample stream into a deterministic summary data structure

```
Initially, allocate space for $G_0$. Mark row 0 as live and active.
for $t = 1, 2, \ldots$ do
    foreach live row $r \geq 0$ do
        with probability $1/2^r 32$ do
            Insert $x_t$ into $G_r$ if $G_r$ has seen $< 2m$ insertions.
        if $r \geq 1$ and $2^{r-1}m < t \leq 2^r m$ and $G_r$ has seen $< 2m$ insertions then
            with probability $1/2^r 32$ do
                Also insert item $t-2^{r-1}m$ of $R_r$ into $G_r$.
    if $t = 2^{r-1}m$ for some $r \geq 1$ then
        Allocate space for $G_r$. Mark row $r$ as live.
        Query $G_{r-1}$ with $\rho_1 \ldots \rho_{8/\varepsilon}$ to get $Q_{r-1} = y_1 \ldots y_{8/\varepsilon}$.
        Store $Q_{r-1}$, to implicitly define $R_r$.
    if $t = 2^r 16m$ for some $r \geq 1$ then
        Mark row $r$ as active. Unmark row $r-1$ as active.
    if $t = 2^r 32m$ for some $r \geq 0$ then
        Unmark row $r$ as live. Free space for $G_r$.
on query $\rho$ do
    Let $r = r(t)$ be the active row.
    Query $G_r$ for rank $\rho/2^r 32$ (if $r \geq 1$) or for rank $\rho$ (if $r = 0$).
    Return the result.
```

Algorithm 2. Procedural listing of the algorithm in Section 3.3. The changes between Sections 3.1 and 3.3 are that G_r never has more than $2m$ insertions and that stream R_r is paired with items in B_r.

(compare our Figure 1 with figure 1 on page 254 of [6]). At a very high level, we are simply replacing their deterministic $O(\frac{1}{\varepsilon}\log^2 \varepsilon n)$ MRL summary [5] with the deterministic $O(\frac{1}{\varepsilon}\log \varepsilon n)$ GK summary [3].

However, our implementation of this idea differs conceptually from the implementation of Manku et al. in two important ways. First, we use the GK algorithm strictly as a black box, whereas Manku et al. peek into the internals of their MRL algorithm, using its algorithm-specific interface (New, Collapse, Output) rather than the more generic interface (Insert, Query). At an equivalent level, dealing with the GK algorithm is already unpleasant—the space complexity analysis in [3] is quite involved, and in fact a simpler analysis of the GK algorithm is an open problem [8]. Using the generic interface, our implementation could just as easily replace the GK boxes in the diagram in Figure 2 with MRL boxes; or, for the bounded universe model, with boxes running the q-digest summary of Shrivastava et al. [9].

The second way in which our algorithm differs critically from that of Manku et. al. is that we operate on *streams* rather than on stream *items*. We use this approach in our proof strategy too; the key step in our error analysis, Lemma 3, is a statement about (what to us are) static objects, so we can trade out the complexity of dealing with time-varying data structures for a simple induction. We believe that developing streaming algorithms with analyses that hinge on analyzing streams rather than just stream items is likely to be a useful design approach for many problems.

Acknowledgements. We thank the anonymous reviewers especially for suggesting that we clarify our usage of Theorem 1 in Section 3 and for providing historical context and further insight into previous work, among the many useful comments and suggestions they provided.

Research supported in part by NSF grants CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

1 Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. In *Proceedings of the 31st Symposium on Principles of Database Systems*, PODS'12, pages 23–34, New York, NY, USA, 2012. ACM.

2 Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff M Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Transactions on Database Systems (TODS)*, 38(4):26, 2013.

3 Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. *ACM SIGMOD Record*, 30(2):58–66, 2001.

4 Regant YS Hung and Hingfung F Ting. An\ omega (\ frac {1}{\ varepsilon}\ log\ frac {1}{\ varepsilon}) space lower bound for finding ε-approximate quantiles in a data stream. In *Frontiers in Algorithmics*, pages 89–100. Springer, 2010.

5 Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G Lindsay. Approximate medians and other quantiles in one pass and with limited memory. *ACM SIGMOD Record*, 27(2):426–435, 1998.

6 Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. *ACM SIGMOD Record*, 28(2):251–262, 1999.

7 J. I. Munro and M. S. Paterson. Selection and sorting with limited storage. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, SFCS'78, pages 253–258, Washington, DC, USA, 1978. IEEE Computer Society.

8 Problem 2: Quantiles – open problems in sublinear algorithms (suggested by graham cormode at kanpur 2006), 2006. `http://sublinear.info/2`.

9 Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: new aggregation techniques for sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 239–249. ACM, 2004.

10 Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

11 Lu Wang, Ge Luo, Ke Yi, and Graham Cormode. Quantiles over data streams: an experimental study. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 737–748. ACM, 2013.

On Constant-Size Graphs That Preserve the Local Structure of High-Girth Graphs[*]

Hendrik Fichtenberger[1], Pan Peng[1,2], and Christian Sohler[1]

1 **Department of Computer Science, TU Dortmund, Germany**
`{firstname.lastname}@tu-dortmund.de`

2 **State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China**

Abstract

Let $G = (V, E)$ be an undirected graph with maximum degree d. The k-disc of a vertex $v \in V$ is defined as the rooted subgraph that is induced by all vertices whose distance to v is at most k. The k-disc frequency vector of G, $\mathrm{freq}_k(G)$, is a vector indexed by all isomorphism types of k-discs. For each such isomorphism type Γ, the k-disc frequency vector counts the fraction of vertices that have k-disc isomorphic to Γ. Thus, the frequency vector $\mathrm{freq}_k(G)$ of G captures the *local structure* of G. A natural question is whether one can construct a much smaller graph H such that H has a similar local structure. N. Alon proved that for any $\epsilon > 0$ there always exists a graph H whose size is independent of $|V|$ and whose frequency vector satisfies $\|\mathrm{freq}_k(G) - \mathrm{freq}_k(H)\|_1 \le \epsilon$. However, his proof is only existential and neither gives an explicit bound on the size of H nor an efficient algorithm. He gave the open problem to find such explicit bounds [9]. In this paper, we solve this problem for the special case of high girth graphs. We show how to efficiently compute a graph H with the above properties when G has girth at least $2k+2$ and we give explicit bounds on the size of H.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases local graph structure, k-disc frequency vector, graph property testing

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.786

1 Introduction

Given a graph $G = (V, E)$, the problem to find a smaller graph H that approximates G with respect to some of its properties is a basic problem in the area of graph algorithms. For example, spanner graphs [4] approximate G with respect to the shortest path structure, combinatorial sparsifiers [2] approximate G with respect to the cut structure, spectral sparsifiers [14] approximate G with respect to the spectral structure, and for a dense graph G the regularity lemma [15] may be thought of as providing a constant size weighted graph that captures an important part of the combinatorial structure of G.

In this paper we consider a different type of approximation. We study the problem of constructing a small graph H that has approximately the same *local structure* as G, where G is assumed to be undirected and to have a maximum degree bounded by d. The motivation to consider such an approximation is that any algorithm that only uses local information will behave similarly on inputs G and H. This is, for example, interesting in the context of property testing in the bounded degree graph model introduced by Goldreich and Ron [7],

* The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 307696.

where we are given oracle access to the adjacency lists of a graph G with maximum degree d and the goal is to distinguish graphs with a given property Π from graphs that are ϵ-far from Π, that is, graphs that have to be changed in more than $\epsilon d|V|$ edges to obtain a graph with property Π. It is known that many constant time property testers in this model depend only on the local structure of the input graph. For example, all minor-closed properties can be tested in this way [3, 8]. If one allows the property testing algorithm to be (non-uniformly) depending on n (and ϵ and d) then every hyperfinite property is testable [13], that is, every graph property that contains only graphs that can be partitioned into connected components of constant size by removing an ϵ-fraction of the edges.

We now continue to make the problem more precise. Given a vertex $v \in V$, the k-disc of v is defined as the rooted subgraph that is induced by all vertices whose distance to v is at most k. The k-disc frequency vector of G, $\mathrm{freq}_k(G)$, is an L-dimensional vector indexed by all isomorphism types of k-discs, where L is the number of such isomorphism types. For each isomorphism type Γ, the k-disc frequency vector counts the fraction of vertices that have k-disc Γ. In other words, $\mathrm{freq}_k(G)$ is the frequency distribution of local neighborhoods over the vertices of G. Given G with maximum degree d and a parameter ϵ our problem is to compute a smaller graph H such that $\|\mathrm{freq}_k(G) - \mathrm{freq}_k(H)\|_1 \leq \epsilon$, where $\mathrm{freq}_k(G)\,, \mathrm{freq}_k(H)$ denote the frequency vectors of G and H, respectively.

1.1 Previous Work

There is a surprisingly simple proof by Alon showing that for every $\epsilon > 0$ and constants d and k, there is an $M(\epsilon)$ such that for every d-bounded degree graph G, there is a graph H of size $M(\epsilon)$ such that the ℓ_1-norm distance of $\mathrm{freq}_k(G)$ and $\mathrm{freq}_k(H)$ is bounded by ϵ (see [11, Proposition 19.10] for the proof). In other words, for every d-bounded graph G of arbitrary size, there exists a small graph H of constant size that approximates the local neighborhood distribution of G. This result may be viewed as an analogue to a weak version of Szemerédi's regularity lemma for dense graphs [15] (see [10, Section 5.5] for more details).

The proof by Alon is based on a compactness argument and does not give explicit bound on $M(\epsilon)$. Obtaining such a bound was suggested by Alon as an open problem [9].

The problem is also related to the theory of graph limits and may be viewed as a finite version of the Aldous-Lyons conjecture [1]. A special case of this conjecture was solved by Elek [5]: He proved that every involution-invariant probability measure on the space of d-bounded trees arises as the local limit of some (infinite) sequence of d-bounded graphs.

1.2 Our Results

In this paper, we give a bound on $M(\epsilon)$ for the special case when the input graph has high girth, where the girth of a graph G is defined as the length of the shortest cycle in G. In other words, we focus on the class of graphs where all k-discs are trees. This class contains some very interesting graphs already. For example, it is known that a random regular graph with high girth is an expander graph with high probability (cf. [6, 12]).

We develop an algorithm that, given oracle access to a graph G with maximum degree d, computes in constant time and with a constant number of queries a small graph H such that $\|\mathrm{freq}_k(G) - \mathrm{freq}_k(H)\|_1 \leq \epsilon$. Here, a query asks for the adjacency list of a vertex $v \in V(G)$.

▶ **Theorem 1.** *Let $d \geq 2$, $k \geq 1$, $\epsilon, \delta \in (0,1)$ and define $\varphi := \frac{300 d^{3k+2} L^3}{\epsilon^2 \delta}$. Let $G = (V,E)$ be a d-bounded degree graph of size $|G| \geq 2\varphi^2/\delta$ with $\mathrm{girth}(G) \geq 2k+2$. Then, there is an* *algorithm that outputs, with probability* $1-\delta$*, a graph* H *such that*

$$\|\mathrm{freq}_k(G)-\mathrm{freq}_k(H)\|_1 \le \epsilon \textit{ and } |V(H)| \le \varphi\,.$$

The algorithm has time and query complexity $\mathcal{O}(1)$ *for constant* d*,* k*,* ϵ *and* δ*.*

If we allow the algorithm to be less efficient (but deterministic), the size of H can be reduced by a factor of L^2/ϵ.

▶ **Theorem 2.** *Let* $d \ge 2$*,* $k \ge 1$*,* $\epsilon \in (0,1)$ *and let* $G=(V,E)$ *be a* d*-bounded graph with* $\mathrm{girth}(G) \ge 2k+2$*. Then, there is a deterministic algorithm that outputs a graph* H *such that*

$$\|\mathrm{freq}_k(G)-\mathrm{freq}_k(H)\|_1 \le \epsilon \textit{ and } |V(H)| \le 36\frac{d^{3k+2}L}{\epsilon}\,.$$

The algorithm has time complexity $\mathcal{O}(|V(G)|)$*.*

We remark that our results can be directly generalized to graphs that are close to having high girth. For any $\varepsilon > 0$ and integer k, two d-bounded graphs G and G' are called to be *ε-close* to each other if one can obtain G' by inserting/deleting at most εdn edges to/from G. By noting that the ℓ_1-norm distance of the frequency vectors of two graphs G and G' is small if they are close to each other, we have the following corollary.

▶ **Corollary 3.** *Let* $d \ge 2$*,* $k \ge 1$*,* $\epsilon \in (0,1)$ *and let* $G=(V,E)$ *be a* d*-bounded graph that is* $\frac{\varepsilon}{6d^{k+1}}$*-close to some graph* G' *with* $\mathrm{girth}(G') \ge 2k+2$*. Then, there exists a graph* H *of size at most* $72\frac{d^{3k+2}L}{\epsilon}$ *such that* $\|\mathrm{freq}_k(G)-\mathrm{freq}_k(H)\|_1 \le \epsilon$*.*

1.3 Proof Overview and Techniques

Our result is based on the following transformation of a graph G that fully preserves the local structure of G: Let $(u_1,v_2),(u_2,v_1) \in E$ be two edges with the properties that (a) the distance from u_1 to v_1 and the distance from v_2 to u_2 in G are large and (b) the local neighborhoods of u_1 and u_2 are isomorphic and (c) the local neighborhoods of v_2 and v_1 are isomorphic. Then one can replace the edges $(u_1,v_2),(u_2,v_1)$ by $(u_1,v_1),(u_2,v_2)$ without changing the local structure of the graph. We believe that this local transformation might be also interesting in the context of lower bounds in property testing, since if we consider sufficiently large local neighborhoods, the behavior of any constant-query property testing algorithm does not change under this transformation.

Our algorithm now works as follows. We use random sampling to identify a subset $U \subseteq V$ of constant size that has approximately the same distribution of neighborhoods (with respect to G) as V. Then we use our transformation to turn G into a graph G' where U has a small cut (relative to the size of U) to $V \setminus U$ and the neighborhood distribution of G is preserved. Then the graph $G'[U]$ has constant size and a similar distribution of neighborhoods as G.

2 Preliminaries

Let $G=(V,E)$ be an undirected graph. We will assume G to be a *d-bounded degree* graph, that is, the maximum degree of a vertex in G is upper bounded by d. Throughout the paper, d is assumed to be a constant. Given two vertices $u,v \in V$, let $\mathrm{dt}_G(u,v)$ be the length of the shortest path between u and v. The girth of G, $\mathrm{girth}(G)$, is defined as the length of the shortest cycle in G. The cut of $V_1,V_2 \subseteq V$ where $V_1 \cap V_2 = \emptyset$ is defined as $E \cap \{(u,v) \mid u \in V_1 \wedge v \in V_2\}$.

For any $v \in V$, the *k-disc* of v, denoted by $\mathrm{disc}_k(G,v)$, is defined as the subgraph that is induced by the vertices that are at distance at most k to v and is rooted at v. Two k-discs are isomorphic if and only if there exists a *root-preserving graph isomorphism*, that is, a graph isomorphism that identifies the roots. For any two k-discs Γ' and Γ'', we write $\Gamma' \simeq \Gamma''$ if Γ' is isomorphic to Γ'', and write $\Gamma' \not\simeq \Gamma''$ otherwise. We denote the number of all non-isomorphic d-bounded degree rooted graphs with radius at most k (that is, k-discs) by $L := L(d,k)$. The set of all such graphs is denoted by $\mathcal{T}_k = (\Gamma_1, \ldots, \Gamma_L)$. Since G has d-bounded degree, the size of each of its k-discs $\Gamma_i \in \mathcal{T}_k$ is bounded by $1 + d + \ldots + d^k \le 3d^k/2$.

▶ **Fact 4.** *The size of a k-disc $\Gamma \in \mathcal{T}_k$ is at most $3d^k/2$.*

The *k-disc count vector* $\mathrm{cnt}_k(G)$ of a graph G is an L-dimensional vector where the i-th entry counts the number of k-discs in G that are isomorphic to $\Gamma_i \in \mathcal{T}_k$. By Fact 4, L is finite. Note that the total number of k-discs in G is exacly $|V(G)|$. Given a k-disc isomorphism type Γ, $\mathrm{cnt}_k(G)_\Gamma$ is defined as the entry in $\mathrm{cnt}_k(G)$ that corresponds to Γ. Given a subset of vertices $S \subseteq V$, let $\mathrm{cnt}_k(S \mid G)$ be the k-disc count vector such that the ith entry counts the number of k-discs of G with root vertex in S that are isomorphic to Γ_i.

The *k-disc frequency vector* of G, denoted by $\mathrm{freq}_k(G)$, is the vector where the i-th entry counts the fraction of k-discs in G that are isomorphic to $\Gamma_i \in \mathcal{T}_k$, or equivalently, $\mathrm{freq}_k(G) := \mathrm{cnt}_k(G)\,/|V(G)|$. We define $\mathrm{freq}_k(S \mid G) := \mathrm{cnt}_k(S \mid G)\,/|S|$ and $\mathrm{freq}_k(G)_\Gamma := \mathrm{freq}_k(G)_\Gamma\,/|V(G)|$ similarly.

In the following, we consider both k-discs and $(k-1)$-discs. We use Γ to denote k-discs and Δ to denote $(k-1)$-discs.

For any integer k and $\Gamma', \Gamma'' \in \mathcal{T}_k$, we call an edge $(u,v) \in E$ a *(Γ', Γ'')-edge* if $\mathrm{disc}_k(G,u) \simeq \Gamma'$ and $\mathrm{disc}_k(G,v) \simeq \Gamma''$. For any two subsets $V_1, V_2 \subseteq V$ and any two k-disc types Γ', Γ'', we let $\mathrm{e}(\Gamma', \Gamma'' \,|\, V_1, V_2)$ denote the number of (Γ', Γ'')-edges from V_1 to V_2, that is, the number of edges (u,v) such that $u \in V_1$, $v \in V_2$, $\mathrm{disc}_k(G,u) \simeq \Gamma'$ and $\mathrm{disc}_k(G,v) \simeq \Gamma''$.

For any k-disc $\Gamma \in \mathcal{T}_k$ and $(k-1)$-disc $\Delta \in \mathcal{T}_{k-1}$, Γ is called *Δ-extensive* if the $(k-1)$-disc of the root of Γ is isomorphic to Δ. We denote the set of all Δ-extensive k-discs Γ by $\mathrm{ext}(\Delta)$. Given a k-disc $\Gamma \in \mathcal{T}_k$ with root r and a $(k-1)$-disc $\Delta \in \mathcal{T}_{k-1}$, let $\mathrm{neigh}(\Gamma, \Delta)$ be the number of neighbors of r whose $(k-1)$-disc is isomorphic to Δ, that is,

$$\mathrm{neigh}(\Gamma, \Delta) := \left|\{v \mid (r,v) \in E(\Gamma) \wedge \mathrm{disc}_{k-1}(\Gamma, v) \simeq \Delta\}\right|.$$

For any $\Delta_1, \Delta_2 \in \mathcal{T}_{k-1}$, let $\mathrm{neigh}_\Sigma(\Delta', \Delta'')$ be the total number of (Δ', Δ'')-edges starting at the root of any k-disc $\Gamma \in \mathrm{ext}(\Delta')$, that is, $\mathrm{neigh}_\Sigma(\Delta', \Delta'') := \sum_{\Gamma \in \mathrm{ext}(\Delta')} \mathrm{neigh}(\Gamma, \Delta'')$. Note that $\mathrm{neigh}_\Sigma(\cdot,\cdot)$ is not necessarily symmetric. Since $|\mathrm{ext}(\Delta')| \le |\mathcal{T}_k| \le L$ and for every $\Gamma \in \mathrm{ext}(\Delta')$, its root's degree is at most d, we get the following bound.

▶ **Fact 5.** *For every pair of $(k-1)$-discs Δ', $\Delta'' \in \mathcal{T}_{k-1}$, we have* $\mathrm{neigh}_\Sigma(\Delta', \Delta'') \le Ld$.

3 Rewiring Edges

In this section, we show that for every partitioning $V_1 \,\dot\cup\, V_2 = V$ of a graph $G = (V,E)$ with girth at least $2k+2$ and $\mathrm{freq}_k(V_1 \mid G)\,, \mathrm{freq}_k(V_2 \mid G) \approx \mathrm{freq}_k(G)$, one can reduce the size of the cut of V_1 and V_2 to some constant by rewiring edges without any effect on $\mathrm{freq}_k(V_i \mid G)$, $i \in \{1,2\}$, and $\mathrm{freq}_k(G)$. Removing the remaining edges in the cut changes the k-disc frequency vectors only slightly. Thus, two smaller graphs with approximately the same k-disc frequency vector as G are obtained.

To this end, our first lemma shows that the fraction of (Δ', Δ'')-edges that start in an arbitrary subset $V_1 \subseteq V$ is approximately the same as for another arbitrary subset $V_2 \subseteq V$ if the frequency distributions of the k-discs in V_1 and V_2 are close.

▶ **Lemma 6.** *Let $G = (V, E)$ be a d-bounded degree graph, $k \in \mathbb{N}$, $\lambda \in [0, 1]$ and let $V_1, V_2 \subseteq V$ be such that $|\mathrm{freq}_k(V_1 \mid G)_\Gamma - \mathrm{freq}_k(V_2 \mid G)_\Gamma| \leq \lambda$ for all k-discs $\Gamma \in \mathcal{T}_k$. Then, for all $(k-1)$-discs $\Delta', \Delta'' \in \mathcal{T}_{k-1}$ such that $\Delta' \not\simeq \Delta''$, it holds that*

$$\left|\frac{\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V)}{|V_1|} - \frac{\mathrm{e}(\Delta', \Delta'' \,|\, V_2, V)}{|V_2|}\right| \leq \lambda \cdot \mathrm{neigh}_\Sigma(\Delta', \Delta'') \,.$$

Proof. Consider any Δ'-extensive k-disc $\Gamma \in \mathrm{ext}(\Delta')$. Then the number of (Δ', Δ'')-edges in G such that the root of Δ' belongs to V_1 equals

$$\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V) = \sum_{\Gamma \in \mathrm{ext}(\Delta')} \mathrm{cnt}_k(V_1 \mid G)_\Gamma \cdot \mathrm{neigh}(\Gamma, \Delta'') \,.$$

An analogous equation holds for $\mathrm{e}(\Delta', \Delta'' \,|\, V_2, V)$. Note that since $\Delta' \not\simeq \Delta''$, even edges that start and end in V_1 are counted only once in the right-hand side of the equation because $\mathrm{ext}(\Delta') \cap \mathrm{ext}(\Delta'') = \emptyset$. Therefore,

$$\begin{aligned}
&\left|\frac{\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V)}{|V_1|} - \frac{\mathrm{e}(\Delta', \Delta'' \,|\, V_2, V)}{|V_2|}\right| \\
&= \left|\frac{\sum_{\Gamma \in \mathrm{ext}(\Delta')} \mathrm{cnt}_k(V_1 \mid G)_\Gamma \cdot \mathrm{neigh}(\Gamma, \Delta'')}{|V_1|} - \frac{\sum_{\Gamma \in \mathrm{ext}(\Delta')} \mathrm{cnt}_k(V_2 \mid G)_\Gamma \cdot \mathrm{neigh}(\Gamma, \Delta'')}{|V_2|}\right| \\
&\leq \sum_{\Gamma \in \mathrm{ext}(\Delta')} |\mathrm{freq}_k(V_1 \mid G)_\Gamma - \mathrm{freq}_k(V_2 \mid G)_\Gamma| \cdot \mathrm{neigh}(\Gamma, \Delta'') \\
&\leq \lambda \cdot \sum_{\Gamma \in \mathrm{ext}(\Delta')} \mathrm{neigh}(\Gamma, \Delta'') \\
&= \lambda \cdot \mathrm{neigh}_\Sigma(\Delta', \Delta'') \,.
\end{aligned}$$

◀

If V_1, V_2 is a partitioning of V, the former result can be improved. In particular, we show that if $\mathrm{freq}_k(V_1 \mid G) \approx \mathrm{freq}_k(V_2 \mid G)$, then for almost every (Δ', Δ'')-edge from V_1 to V_2 there is a counterpart, that is, a (Δ', Δ'')-edge from V_2 to V_1. We will later use this result to reduce the size of the cut without altering the k-disc frequency vector by swapping the endpoints of edges in the cut so that the new edges lie completely in V_1 and V_2, respectively.

▶ **Lemma 7.** *Let $G = (V, E)$ be a d-bounded degree graph, $k \in \mathbb{N}$, $\lambda \in [0, 1]$ and let $V_1 \dot\cup V_2 = V$ be a partitioning of V such that $|\mathrm{freq}_k(V_1 \mid G)_\Gamma - \mathrm{freq}_k(V_2 \mid G)_\Gamma| \leq \lambda$ for all k-discs $\Gamma \in \mathcal{T}_k$. Then, for all $(k-1)$-discs $\Delta', \Delta'' \in \mathcal{T}_{k-1}$, it holds that*

$$|\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2) - \mathrm{e}(\Delta', \Delta'' \,|\, V_2, V_1)| \leq \frac{|V_1||V_2|}{|V|} \cdot \lambda \cdot \Big[\mathrm{neigh}_\Sigma(\Delta', \Delta'') + \mathrm{neigh}_\Sigma(\Delta'', \Delta')\Big] \,.$$

Proof. If $\Delta' \simeq \Delta''$, the bound holds trivially because $\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2) = \mathrm{e}(\Delta', \Delta'' \,|\, V_2, V_1)$. Therefore, assume that $\Delta' \not\simeq \Delta''$ now. Note that by symmetry it holds that

$$\mathrm{e}(\Delta', \Delta'' \,|\, V_i, V_j) = \mathrm{e}(\Delta'', \Delta' \,|\, V_j, V_i) \qquad i, j \in \{1, 2\} \,. \tag{1}$$

Furthermore, since $V_1 \,\dot\cup\, V_2$ is a partitioning of V, we have

$$\mathrm{e}(\Delta', \Delta'' \,|\, V_i, V) = \mathrm{e}(\Delta', \Delta'' \,|\, V_i, V_1) + \mathrm{e}(\Delta', \Delta'' \,|\, V_i, V_2) \qquad i \in \{1, 2\} \tag{2}$$

and an analogous equation for $\mathrm{e}(\Delta'', \Delta' \,|\, V_i, V)$. Now, we have

$$\begin{aligned}
&|\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2) - \mathrm{e}(\Delta', \Delta'' \,|\, V_2, V_1)| \\
&= \frac{|V_1||V_2|}{|V|} \cdot \frac{(|V_1| + |V_2|)}{|V_1||V_2|} \cdot \Big|\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2) - \mathrm{e}(\Delta', \Delta'' \,|\, V_2, V_1)\Big| \\
&= \frac{|V_1||V_2|}{|V|} \cdot \left|\left(\frac{1}{|V_1|} + \frac{1}{|V_2|}\right) \cdot \Big(\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2) - \mathrm{e}(\Delta'', \Delta' \,|\, V_1, V_2)\Big) + 0 - 0\right|
\end{aligned}$$

$$
\begin{aligned}
&= \frac{|V_1||V_2|}{|V|} \cdot \Bigg| \left(\frac{1}{|V_1|} + \frac{1}{|V_2|} \right) \cdot \Big(\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2) - \mathrm{e}(\Delta'', \Delta' \,|\, V_1, V_2) \Big) \\
&\quad + \frac{\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_1) - \mathrm{e}(\Delta'', \Delta' \,|\, V_1, V_1)}{|V_1|} - \frac{\mathrm{e}(\Delta', \Delta'' \,|\, V_2, V_2) - \mathrm{e}(\Delta'', \Delta' \,|\, V_2, V_2)}{|V_2|} \Bigg| \\
&= \frac{|V_1||V_2|}{|V|} \cdot \Bigg| \frac{\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_1) + \mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2)}{|V_1|} - \frac{\mathrm{e}(\Delta', \Delta'' \,|\, V_2, V_2) + \mathrm{e}(\Delta', \Delta'' \,|\, V_2, V_1)}{|V_2|} \\
&\quad - \frac{\mathrm{e}(\Delta'', \Delta' \,|\, V_1, V_1) + \mathrm{e}(\Delta'', \Delta' \,|\, V_1, V_2)}{|V_1|} + \frac{\mathrm{e}(\Delta'', \Delta' \,|\, V_2, V_2) + \mathrm{e}(\Delta'', \Delta' \,|\, V_2, V_1)}{|V_2|} \Bigg| \\
&= \frac{|V_1||V_2|}{|V|} \cdot \Bigg| \frac{\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V)}{|V_1|} - \frac{\mathrm{e}(\Delta', \Delta'' \,|\, V_2, V)}{|V_2|} - \frac{\mathrm{e}(\Delta'', \Delta' \,|\, V_1, V)}{|V_1|} + \frac{\mathrm{e}(\Delta'', \Delta' \,|\, V_2, V)}{|V_2|} \Bigg| \\
&\le \frac{|V_1||V_2|}{|V|} \cdot \lambda \Big[\mathrm{neigh}_\Sigma(\Delta', \Delta'') + \mathrm{neigh}_\Sigma(\Delta'', \Delta') \Big],
\end{aligned}
$$

where the fourth equation follows from Eq. (1), the fifth equation follows from Eq. (2) and the inequality follows from applying Lemma 6. ◀

The former result enables us to analyze our main technical tool, that is, the rewiring of edges. First, we will prove that under some condition we can rewire two (Δ', Δ'')-edges without altering the k-disc frequency distribution of the graph or the partitions. This part of the proof shows that there exists, for every vertex $v \in V$, an isomorphism function that maps the k-disc of v in the original graph to the k-disc of v in the rewired graph. We then show that if we cannot find such (Δ', Δ'')-edges, the cut of V_1 and V_2 is small. This implies that the removal of the remaining edges changes the k-disc frequency vector of the graph only slightly.

▶ **Lemma 8.** *Let $G = (V, E)$ be a d-bounded graph with $\mathrm{girth}(G) \ge 2k + 2$, $k \in \mathbb{N}$, $\lambda \in [0, 1]$ and let $V_1 \,\dot\cup\, V_2 = V$ be a partitioning of V such that $|\mathrm{freq}_k(V_1 \mid G)_\Gamma - \mathrm{freq}_k(V_2 \mid G)_\Gamma| \le \lambda$ for all k-discs $\Gamma \in \mathcal{T}_k$. Then either there exists a graph $H = (V, F)$ such that*

$$\mathrm{girth}(H) \ge 2k + 2 \tag{3}$$

$$|F \cap (V_1 \times V_2)| \le |E \cap (V_1 \times V_2)| - 2 \tag{4}$$

$$\mathrm{disc}_k(H, w) \simeq \mathrm{disc}_k(G, w) \qquad \forall w \in V \tag{5}$$

or the cut between V_1 and V_2 is small:

$$\mathrm{e}(V_1, V_2) \le 6 d^{2k+2} L + 2\lambda L d \cdot \min(|V_1|, |V_2|) \,. \tag{6}$$

Proof. Consider the following condition (see Fig. 1):

($\star$) There exist $(u_1, v_2) \in (V_1 \times V_2) \cap E$ and $(u_2, v_1) \in (V_2 \times V_1) \cap E$ such that $\mathrm{dt}_G(u_1, v_1)$, $\mathrm{dt}_G(v_2, u_2) \ge 2k{+}1$, $\mathrm{disc}_{k-1}(G, u_1) \simeq \mathrm{disc}_{k-1}(G, u_2)$ and $\mathrm{disc}_{k-1}(G, v_2) \simeq \mathrm{disc}_{k-1}(G, v_1)$.

Informally, it states that, for a suitable choice of $(k-1)$-discs Δ' and Δ'', there exists a (Δ', Δ'')-edge from V_1 to V_2 and a (Δ', Δ'')-edge from V_2 to V_1 such that two endpoints of different edges are not too close. We prove in the following that if condition ($\star$) is satisfied, then there exists a graph H with the desired properties, and that Eq. (6) holds otherwise.

▶ **Claim 9.** *If condition ($\star$) is satisfied, there exists a graph $H = (V, F)$ such that Ineq. (3) and (4) and Expr. (5) are satisfied.*

Proof. Suppose that condition ($\star$) is satisfied. We define an intermediate graph $G' := (V, E')$ that is obtained by deleting the edges (u_1, v_2) and (u_2, v_1) from G, that is, $E' := E \setminus$

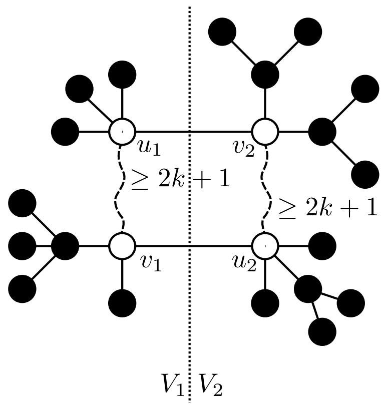


Figure 1 If condition ($\star$) is satisfied (as here for $k = 2$), it is possible to replace the edges (u_1, v_2) and (u_2, v_1) by (u_1, v_1) and (u_2, v_2), respectively, without changing the k-disc vector of G. Otherwise, the cut of V_1 and V_2 is small and removing these edges affects only few k-discs in V_1.

$\{(u_1, v_2), (u_2, v_1)\}$. We further define $H := (V, F)$ to be the graph that is obtained by adding the edges (u_1, v_1) and (u_2, v_2) to G', that is, $F := E' \cup \{(u_1, v_1), (u_2, v_2)\}$.

Observe that since $\mathrm{dt}_G(u_1, v_1), \mathrm{dt}_G(u_2, v_2) \geq 2k + 1$, Ineq. (3) holds, and by definition of H, Ineq. (4) holds. Thus, it remains to prove that Expr. (5) also holds, that is, for any vertex w, the k-discs of w in G and H are isomorphic. In what follows, we carefully construct a root-preserving bijection $f : V(\mathrm{disc}_k(G, w)) \to V(\mathrm{disc}_k(H, w))$ such that for all $x, y \in V(\mathrm{disc}_k(G, w))$, $(x, y) \in E(\mathrm{disc}_k(G, w))$ if and only if $(f(x), f(y)) \in E(\mathrm{disc}_k(H, w))$ to formally prove this somewhat intuitive observation.

Let $w \in V$. We distinguish between the cases that neither (u_1, v_2) nor (u_2, v_1), either one of them, or both are contained in $\mathrm{disc}_k(G, w)$. First, we will specify two isomorphism functions $g_u : V(\mathrm{disc}_k(G, u_1)) \to V(\mathrm{disc}_k(G, u_2))$ and $g_v : V(\mathrm{disc}_k(G, v_2)) \to V(\mathrm{disc}_k(G, v_1))$ for $\mathrm{disc}_k(G, u_1) \simeq \mathrm{disc}_k(G, u_2)$ and $\mathrm{disc}_k(G, v_2) \simeq \mathrm{disc}_k(G, v_1)$, respectively. If there is more than one candidate for g_u and g_v respectively, we make an arbitrary choice unless stated otherwise. We will then define f using these two functions g_u and g_v and prove that f is an isomorphism between $\mathrm{disc}_k(G, w)$ and $\mathrm{disc}_k(H, w)$.

Case 1: $(u_1, v_2) \notin \mathrm{disc}_k(G, w)$, $(u_2, v_1) \notin \mathrm{disc}_k(G, w)$. In this case, we define $f(x) := x$ for all $x \in V(\mathrm{disc}_k(G, w))$. We claim that neither (u_1, v_1) nor (v_2, u_2) belongs to $E(\mathrm{disc}_k(H, w))$. Without loss of generality assume that $(u_1, v_1) \in E(\mathrm{disc}_k(H, w))$. Then $\mathrm{dt}_G(w, u_1), \mathrm{dt}_G(w, v_1) \leq k$, which implies that $\mathrm{dt}_G(u_1, v_1) \leq \mathrm{dt}_G(u_1, w) + \mathrm{dt}_G(w, v_1) \leq 2k$. This is a contradiction to the assumption that $\mathrm{dt}_G(u_1, v_1) \geq 2k + 1$. The same argument shows that $(u_2, v_2) \notin E(\mathrm{disc}_k(G, w))$. Therefore, the k-discs $\mathrm{disc}_k(G, w)$ and $\mathrm{disc}_k(H, w)$ do not contain any of the edges (u_1, v_2), (u_2, v_1), (u_1, v_1), (u_2, v_2) and thus $\mathrm{disc}_k(G, w) \simeq \mathrm{disc}_k(H, w)$ by our definition of H.

Case 2: $(u_1, v_2) \in \mathrm{disc}_k(G, w)$, $(u_2, v_1) \notin \mathrm{disc}_k(G, w)$. In this case it holds that $u_2, v_1 \notin \mathrm{disc}_k(G, w)$, since otherwise, either $\mathrm{dt}_G(u_1, v_1) \leq 2k$ or $\mathrm{dt}_G(v_2, u_2) \leq 2k$, which contradicts condition ($\star$).

Now we observe that since $\mathrm{girth}(G) \geq 2k + 2$, the k-disc $\mathrm{disc}_k(G, w)$ is a tree. This implies that the deletion of the edge (u_1, v_2) will partition $\mathrm{disc}_k(G, w)$ into two connected components, say P_{u_1} and P_{v_2}, which represent the set of vertices in $\mathrm{disc}_k(G, w)$ that are connected to u_1 after deleting (u_1, v_2) and the set of remaining vertices that are connected

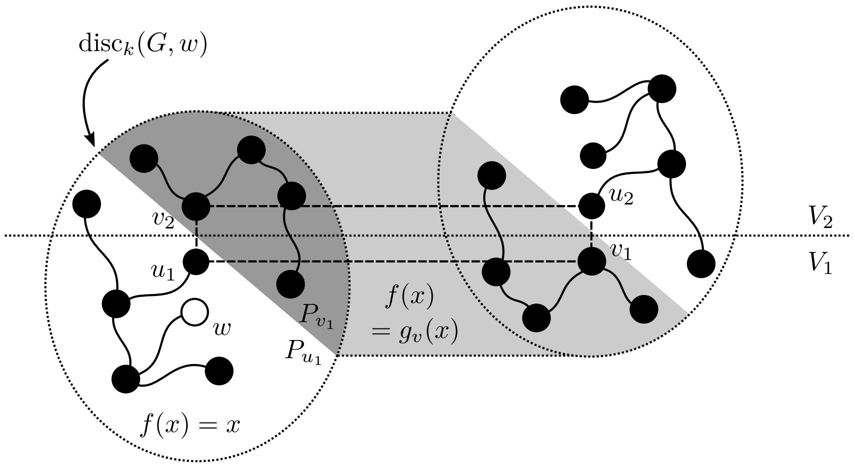


Figure 2 The k-disc of w in G, $\mathrm{disc}_k(G, w)$, is partitioned into two parts P_{u_1} (white background) and P_{v_2} (dark gray background) by the edge (u_1, v_2). The dashed edges are only present in either G or H but not in G'. If $w \in P_{u_1}$ as in the figure, then $f(x) := x$ for any $x \in P_{u_1}$, and $f(x) := g_v(x)$ for any $x \in P_{v_2}$, where g_v is chosen such that the image of P_{v_2} under g_v is a subset of $\mathrm{disc}_k(G', v_1)$ (light gray background).

to v_2, respectively. Without loss of generality assume that $w \in P_{u_1}$. The case that $w \in P_{v_2}$ can be analyzed similarly.

Let $f(x) = x$ if $x \in P_{u_1}$ and $f(x) = g_v(x)$ if $x \in P_{v_2}$. If there is more than one candidate for g_v, we make an arbitrary choice among all isomorphism functions that map P_{v_2} to (a subset of) $\mathrm{disc}_{k-1}(G', v_1)$, that is, the $(k-1)$-disc of v_1 after deleting (u_2, v_1) (see Fig. 2). Since $\mathrm{disc}_{k-1}(G, v_2) \simeq \mathrm{disc}_{k-1}(G, v_1)$ by $(\star)$, there is always an isomorphism function that satisfies this condition. Moreover, f is a bijection because g_v is a bijection, and the image of $V(\mathrm{disc}_k(G, w))$ under f is $V(\mathrm{disc}_k(H, w))$ by the construction of H. We now prove that f is an isomorphism function between $\mathrm{disc}_k(G, w)$ and $\mathrm{disc}_k(H, w)$.

First note that $f(w) = w$, $f(u_1) = u_1$ and $f(v_2) = g_v(v_2) = v_1$. Now consider any $x, y \in V(\mathrm{disc}_k(G, w))$. If $x, y \in P_{u_1}$ or $x, y \in P_{v_2}$, then $f(x) = x$, $f(y) = y$ or $f(x) = g_v(x)$, $f(y) = g_v(y)$, respectively. Therefore, $(x, y) \in E(G)$ if and only if $(f(x), f(y)) \in E(H)$.

Now consider the case that $x \in P_{u_1}$ and $y \in P_{v_2}$. If $x = u_1$ and $y = v_2$, then we know that $(x, y) \in E(G)$ and also that $(f(x), f(y)) = (u_1, v_1) \in E(H)$ by the definition of H. Otherwise, either $x \neq u_1$ or $y \neq v_2$. In this case, there is no edge (x, y) in G since $\mathrm{disc}_k(G, w)$ is a tree and x, y lie on different sides of the edge (u_1, v_2). Recall that $f(u_1) = u_1$ and $f(v_2) = g_v(v_2) = v_1$. Since f is a bijection, either $f(x) \neq u_1$ or $f(y) \neq v_1$. Observe that $\mathrm{disc}_k(H, w)$ is a tree by Ineq. (3). Hence there is no edge between $f(x)$ and $f(y)$ in H as they lie on different sides of the edge (u_1, v_1). The case that $x \in P_{v_2}$ and $y \in P_{u_1}$ is symmetric.

Therefore, the function f is a root-preserving isomorphism function between $\mathrm{disc}_k(G, w)$ and $\mathrm{disc}_k(H, w)$.

Case 3: $(u_1, v_2) \notin \mathrm{disc}_k(G, w)$, $(u_2, v_1) \in \mathrm{disc}_k(G, w)$. This case can be analyzed similarly to the foregoing case.

Case 4: $(u_1, v_2) \in \mathrm{disc}_k(G, w), (u_2, v_1) \in \mathrm{disc}_k(G, w)$. Note that this case cannot happen because otherwise we would have $\mathrm{dt}_G(u_1, v_1), \mathrm{dt}_G(u_2, v_2) \leq 2k$, which contradicts the assumption that $\mathrm{dt}_G(u_1, v_1)$, $\mathrm{dt}_G(u_2, v_2) \geq 2k + 1$. This completes the case analysis and the proof of Claim 9. ◀

▶ **Claim 10.** *If condition* $(\star)$ *is not satisfied, then Eq.* (6) *holds.*

Proof. Suppose that condition $(\star)$ is not satisfied. First, note that we have

$$\mathrm{e}(V_1, V_2) = \sum_{\Delta' \in \mathcal{T}_{k-1}} \sum_{\Delta'' \in \mathcal{T}_{k-1}} \mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2)\,.$$

Now, let $\Delta', \Delta'' \in \mathcal{T}_{k-1}$ be any two $(k-1)$-disc isomorphism types. The key observation is that if (u_1, v_2) is a (Δ', Δ'')-edge from V_1 to V_2, then for every (Δ', Δ'')-edge (u_2, v_1) from V_2 to V_1 the distance between u_2 and v_2 or the distance between v_1 and u_1 must be smaller than $2k+1$ (otherwise, the edges could be rewired and $(\star)$ would be satisfied). Since the graph is degree-bounded, this implies an upper bound on the number of possible endpoints u_2, v_1 and thus implies an upper bound on $\mathrm{e}(\Delta', \Delta'' \,|\, V_2, V_1)$. It follows that $\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2)$ is also bounded by Lemma 7. In case there is no (Δ', Δ'')-edge from V_1 to V_2, the number of (Δ', Δ'')-edges from V_2 to V_1 can be bounded directly by Lemma 7.

We proceed to make this precise. For every choice of $\Delta', \Delta'' \in \mathcal{T}_{k-1}$, we distinguish two cases as mentioned before: whether we can find a (Δ', Δ'')-edge from V_1 to V_2 or not.

Case 1: There exist $u_1 \in V_1$ and $v_2 \in V_2$ such that $(u_1, v_2) \in E$, $\mathrm{disc}_{k-1}(G, u_1) \simeq \Delta'$ and $\mathrm{disc}_{k-1}(G, v_2) \simeq \Delta''$, that is, $\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2) > 0$. Since condition $(\star)$ is not satisfied, at least one endpoint of every (Δ', Δ'')-edge (u_2, v_1) from V_2 to V_1 must have distance less than $2k+1$ to u_1 or v_2. Without loss of generality, fix such an edge with $\mathrm{dt}_G(u_2, v_2) < 2k+1$. The case $\mathrm{dt}_G(u_1, v_1) < 2k+1$ can be analyzed similarly. There are at most $3d^{2k}/2$ vertices with distance less than $2k+1$ to v_2 by Fact 4. Each of these near vertices can be adjacent to at most d vertices in V_1 whose $(k-1)$-discs are isomorphic to Δ''. Taking the symmetric case $\mathrm{dt}_G(u_1, v_2) < 2k+1$ into account, we have $\mathrm{e}(\Delta', \Delta'' \,|\, V_2, V_1) \leq 2 \cdot 3d^{2k}/2 \cdot d \leq 3d^{2k+1}$. Now by Lemma 7, it holds that

$$\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2) + \mathrm{e}(\Delta', \Delta'' \,|\, V_2, V_1) \leq 6d^{2k+1} + \frac{|V_1||V_2|}{|V|} \cdot \lambda \Big[\mathrm{neigh}_\Sigma(\Delta', \Delta'') + \mathrm{neigh}_\Sigma(\Delta'', \Delta')\Big]\,.$$

Case 2: There do not exist $u_1 \in V_1$ and $v_2 \in V_2$ such that $(u_1, v_2) \in E$, $\mathrm{disc}_{k-1}(G, u_1) \simeq \Delta'$ and $\mathrm{disc}_{k-1}(G, v_2) \simeq \Delta''$, that is, $\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2) = 0$. By Lemma 7, we have

$$\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2) + \mathrm{e}(\Delta', \Delta'' \,|\, V_2, V_1) \leq 0 + \frac{|V_1||V_2|}{|V|} \cdot \lambda \Big[\mathrm{neigh}_\Sigma(\Delta', \Delta'') + \mathrm{neigh}_\Sigma(\Delta'', \Delta')\Big]\,.$$

This completes the case analysis. Note that each k-disc $\Gamma \in \mathcal{T}_k$ determines the $(k-1)$-discs of its root and of its at most d neighbors. Moreover, the number of different k-disc isomorphism types in G is at most L. Therefore, the number of pairs Δ', Δ'' such that there exists an edge between a vertex with $(k-1)$-disc Δ' and a vertex with $(k-1)$-disc Δ'' is at most Ld, that is, $\mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2) \neq 0$ for at most Ld pairs Δ', Δ'', and we have

$$\begin{aligned}
&\mathrm{e}(V_1, V_2)\\
&= \sum_{\Delta' \in \mathcal{T}_{k-1}} \sum_{\Delta'' \in \mathcal{T}_{k-1}} \mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2)\\
&\leq Ld \cdot \max_{\Delta', \Delta'' \in \mathcal{T}_{k-1}} \mathrm{e}(\Delta', \Delta'' \,|\, V_1, V_2)\\
&\leq Ld \cdot \left(6d^{2k+1} + \frac{\lambda |V_1||V_2|}{|V|} \cdot \max_{\Delta', \Delta'' \in \mathcal{T}_{k-1}} \Big[\mathrm{neigh}_\Sigma(\Delta', \Delta'') + \mathrm{neigh}_\Sigma(\Delta'', \Delta')\Big]\right)\\
&\leq 6d^{2k+2}L + \lambda \cdot \min(|V_1|, |V_2|) \cdot 2Ld,
\end{aligned}$$

where the last step follows from Fact 5. This completes the proof of Claim 10 and Lemma 8. ◀

4 Proof of the Main Theorems

We first prove Theorem 1 by arguing along the execution of Algorithm 1.

Algorithm 1

```
1:  function PartitionAndRewire($G = (V, E)$, $\varphi$)
2:      $V_1 \leftarrow$ sample $\varphi$ vertices from $V$ uniformly at random
3:      $V_2 \leftarrow V \setminus V_1$
4:      $E' \leftarrow E$, $G' \leftarrow (V, E')$
5:      for all $(u_1, v_2) \in (V_1 \times V_2) \cap E'$ and $(u_2, v_1) \in (V_2 \times V_1) \cap E'$ do
6:          if $\mathrm{disc}_{k-1}(G', u_1) \simeq \mathrm{disc}_{k-1}(G', u_2) \wedge \mathrm{disc}_{k-1}(G', v_2) \simeq \mathrm{disc}_{k-1}(G', v_1)$
                $\wedge\, \mathrm{dt}_{G'}(u_1, v_1) \geq 2k+1 \wedge \mathrm{dt}_{G'}(u_2, v_2) \geq 2k+1$ then
7:              $E' \leftarrow E' \setminus \{(u_1, v_2), (u_2, v_1)\} \cup \{(u_1, v_1), (u_2, v_2)\}$
8:              $G' \leftarrow (V, E')$
9:              Goto line 5
10:         end if
11:     end for
12:     return $H := G'[V_1]$
13: end function
```

Proof of Theorem 1. We prove that the output of Algorithm 1 is a graph with the desired properties. First, we sample φ vertices $v_1, \ldots, v_\varphi$ from G uniformly at random (cf. line 2). Let E_1 denote the event that all the sampled vertices are different. Note that for any i, j such that $1 \leq i < j \leq \varphi$, the probability that $v_i = v_j$ is at most $1/|V|$, which implies that $\Pr[E_1] \geq 1 - \frac{\varphi^2}{|V|} \geq 1 - \frac{\delta}{2}$ because $|V| \geq 2\varphi^2/\delta$.

Let $V_2 := V \setminus V_1$. For each $i \leq \varphi$, let $\vec{\boldsymbol{f}}_i \in \{0,1\}^L$ denote the random vector that equals the indicator vector $\vec{\mathbf{1}}_\Gamma$ if the k-disc of v_i is isomorphic to Γ. Note that $\mathrm{freq}_k(V_1 \mid G) = \sum_i \vec{\boldsymbol{f}}_i/\varphi$ and that $\Pr[\vec{\boldsymbol{f}}_i = \vec{\mathbf{1}}_\Gamma] = \mathrm{freq}_k(G)_\Gamma$, and thus $\mathrm{E}[\mathrm{freq}_k(V_1 \mid G)] = \mathrm{E}[\vec{\boldsymbol{f}}_i] = \mathrm{freq}_k(G)$. Let $X := \|\mathrm{freq}_k(G) - \mathrm{freq}_k(V_1 \mid G)\|_2^2$. We bound the deviation between $\mathrm{freq}_k(G)$ and $\sum_i \vec{\boldsymbol{f}}_i/\varphi$. It holds that

$$\begin{aligned}
\mathrm{E}[X] = \mathrm{E}\left[\left\|\mathrm{freq}_k(G) - \frac{\sum_{i=1}^{\varphi} \vec{\boldsymbol{f}}_i}{\varphi}\right\|_2^2\right] &= \mathrm{E}\left[\frac{1}{\varphi^2}\left\|\sum_{i=1}^{\varphi}(\mathrm{freq}_k(G) - \vec{\boldsymbol{f}}_i)\right\|_2^2\right] \\
&= \frac{1}{\varphi^2}\mathrm{E}\left[\sum_{i=1}^{\varphi}\|\mathrm{freq}_k(G) - \vec{\boldsymbol{f}}_i\|_2^2\right] \\
&= \frac{1}{\varphi^2}\sum_{i=1}^{\varphi}\mathrm{E}\left[\|\mathrm{freq}_k(G) - \vec{\boldsymbol{f}}_i\|_2^2\right] \\
&= \frac{1}{\varphi}\mathrm{E}\left[\|\mathrm{freq}_k(G) - \vec{\boldsymbol{f}}_1\|_2^2\right] \\
&\leq \frac{1}{\varphi}\mathrm{E}\left[\|\mathrm{freq}_k(G) - \vec{\boldsymbol{f}}_1\|_1\right] \\
&\leq \frac{2}{\varphi},
\end{aligned}$$

where the third equation follows from the fact that all $\vec{\boldsymbol{f}}_i$ are independent of each other; the penultimate inequality follows from the fact that the absolute values of all entries of

$\mathrm{freq}_k(G') - \vec{\boldsymbol{f}}_1$ are at most 1. Now by Markov's inequality,

$$\Pr\left[\left\|\mathrm{freq}_k(G) - \frac{\sum_i \vec{\boldsymbol{f}}_i}{\varphi}\right\|_2^2 \geq \frac{2}{\delta}\cdot\frac{2}{\varphi}\right] \leq \Pr\left[X \geq \frac{2}{\delta}\cdot \mathrm{E}[X]\right] \leq \frac{\delta}{2}\,.$$

Therefore, if we let $\lambda = \frac{\varepsilon}{6Ld^{k+1}}$, then with probability at least $1-\delta/2$,

$$\left\|\mathrm{freq}_k(G) - \frac{\sum_i \vec{\boldsymbol{f}}_i}{\varphi}\right\|_1 \leq \sqrt{L}\cdot\left\|\mathrm{freq}_k(G) - \frac{\sum_i \vec{\boldsymbol{f}}_i}{\varphi}\right\|_2 \leq \sqrt{L}\cdot\sqrt{\frac{4}{\delta\varphi}} \leq \lambda/2,$$

where the last inequality follows from our choice of $\varphi = \frac{300d^{3k+2}L^3}{\varepsilon^2\delta} = \frac{300d^kL}{36\lambda^2\delta} \geq \frac{16L}{\lambda^2\delta}$. This further implies that (with probability at least $1-\delta/2$)

$$\|\mathrm{freq}_k(G) - \mathrm{freq}_k(V_1 \mid G)\|_1 = \left\|\mathrm{freq}_k(G) - \frac{\sum_i \vec{\boldsymbol{f}}_i}{\varphi}\right\|_1 \leq \frac{\lambda}{2}\,. \tag{7}$$

Let E_2 denote the event that $\|\mathrm{freq}_k(G) - \mathrm{freq}_k(V_1 \mid G)\|_1 \leq \frac{\lambda}{2}$. Thus $\Pr[E_2] \geq 1-\delta/2$. If E_2 occurs, then $\|\mathrm{freq}_k(G) - \mathrm{freq}_k(V_2 \mid G)\|_1 \leq \frac{\lambda}{2}$ because $|V_2| \geq |V_1|$, and therefore $\|\mathrm{freq}_k(V_1 \mid G) - \mathrm{freq}_k(V_2 \mid G)\|_1 \leq \lambda$.

Conditioning on both events E_1 and E_2, which occur with probability $\Pr[E_1 \cap E_2] \geq 1-2\cdot\frac{\delta}{2} = 1-\delta$, we apply Lemma 8 with G, λ and partition V_1, V_2 as follows: Let $G' = (V, E')$ be a copy of G. As long as condition ($\star$) is satisfied, we replace G' by the *rewired* graph that satisfies Ineq. (3) and (4) and Expr. (5) (cf. lines 5 to 11). After rewiring, there remain at most $6d^{2k+2}L + 2\lambda L d\varphi$ edges in the cut of V_1 and V_2, which are (virtually) deleted by returning the graph $H := G'[V_1]$.

The k-disc of a vertex is altered if and only if an edge is inserted to the k-disc or removed from it. The maximum size of a k-disc is at most $3d^k/2$ by Fact 4. Therefore, removing a single edge alters at most $3d^k/2$ k-discs. By Lemma 8 it holds that

$$\|\mathrm{freq}_k(V_1 \mid G) - \mathrm{freq}_k(H)\|_1 \leq \frac{3d^k/2\cdot\left(6d^{2k+2}L + 2\lambda L d\varphi\right)}{\varphi} \leq \frac{9d^{3k+2}L}{\varphi} + 3Ld^{k+1}\lambda \leq \frac{3\varepsilon}{4}, \tag{8}$$

where the last inequality follows from our choice of $\varphi = \frac{300d^{3k+2}L^3}{\varepsilon^2\delta}$ and $\lambda = \frac{\varepsilon}{6Ld^{k+1}}$. It follows from Eqs. (7) and (8) and the triangle inequality that

$$\|\mathrm{freq}_k(G) - \mathrm{freq}_k(H)\|_1 \leq \frac{\lambda}{2} + \frac{3\varepsilon}{4} \leq \varepsilon\,.$$

Now we analyze the query (and time) complexity of the above algorithm. Note that the algorithm only needs to sample φ vertices and query all the $(k+2)$-discs of vertices in V_1. In particular, the rewiring step (cf. line 6) can be performed as follows: we consider all the vertices that are endpoints of some edges leaving V_1 by exploring the neighbors of all vertices in V_1. We want to find $u_1, v_1 \in V_1$ with $\mathrm{dt}_G(u_1, v_1) \geq 2k+1$ such that $(u_1, v_2) \in (V_1 \times V_2) \cap E'$ and $(u_2, v_1) \in (V_2 \times V_1) \cap E'$. To test if we should rewire the corresponding edges or not, we only need to consider the $(k+1)$-discs of $v_2, u_2 \in V_2$ to determine if $\mathrm{dt}_G(v_2, u_2) \geq 2k+1$. This implies that we only need to query the $(k+2)$-discs of all vertices in V_1. It follows that the algorithm makes at most $\varphi \cdot \frac{3d^{k+2}}{2} = \mathcal{O}(1)$ queries for constants d, ϵ and k to the oracle of G. Also note that since $|V_1| \in \mathcal{O}(1)$, the number of rewiring steps as well as the number of eges with at least one end in V_1 is at most $|V_1|d \in \mathcal{O}(1)$. Therefore, the algorithm has constant time complexity. ◀

Algorithm 2

```
1: function REWIREANDSPLIT($G = (V, E)$, $\varphi$)
2:     Partition $V$ into $V_1, V_2$ such that
           $|V_1| = \varphi$ and $|\text{freq}_k(V_1 \mid G)_\Gamma - \text{freq}_k(V_2 \mid G)_\Gamma| \leq 1/\varphi$ for all $\Gamma$
3:     $E' \leftarrow E$, $G' \leftarrow (V, E')$
4:     for all $(u_1, v_2) \in (V_1 \times V_2) \cap E'$ and $(u_2, v_1) \in (V_2 \times V_1) \cap E'$ do
5:         if $\text{disc}_{k-1}(G', u_1) \simeq \text{disc}_{k-1}(G', u_2) \wedge \text{disc}_{k-1}(G', v_2) \simeq \text{disc}_{k-1}(G', v_1)$
               $\wedge\, \text{dt}_{G'}(u_1, v_1) \geq 2k + 1 \wedge \text{dt}_{G'}(u_2, v_2) \geq 2k + 1$ then
6:             $E' \leftarrow E' \cap \{(u_1, v_2), (u_2, v_1)\} \cup \{(u_1, v_1), (u_2, v_2)\}$
7:             $G' \leftarrow (V, E')$
8:             Goto line 4
9:         end if
10:    end for
11:    return $H := G'[V_1]$
12: end function
```

Now, we prove Theorem 2 by arguing along the execution of Algorithm 2.

Proof of Theorem 2. We prove that the output of Algorithm 2 is a graph with the desired properties. Let $\varphi := 12Ld^{3k+2}/\epsilon$. Without loss of generality assume that $\varphi \leq |V(G)|/3$ (otherwise just output $H := G$ directly). First, we partition V into two parts V_1 and V_2 such that $\varphi \leq |V_1| \leq 2\varphi$ and for any k-disc Γ, $|\text{freq}_k(G)_\Gamma - \text{freq}_k(V_i \mid G)_\Gamma| \leq 1/\varphi$ (cf. line 2). Such a partition can be constructed as follows: For each k-disc $\Gamma \in \mathcal{T}_k$, we put $\lceil \varphi \cdot \text{freq}_k(G)_\Gamma \rceil$ vertices v with $\text{disc}_k(G, v) \simeq \Gamma$ into V_1 and the remaining ones into V_2. Thus, $\varphi \leq |V_1| \leq \varphi + |\mathcal{T}_k| \leq 2\varphi$ and we have

$$|\text{freq}_k(G)_\Gamma - \text{freq}_k(V_1 \mid G)_\Gamma| \leq \left| \frac{\varphi \cdot \text{freq}_k(G)_\Gamma - \lceil \varphi \cdot \text{freq}_k(G)_\Gamma \rceil}{\varphi} \right| \leq 1/\varphi\,.$$

Since $|V_2| = n - 2\varphi \geq \varphi$, we also have $|\text{freq}_k(G)_\Gamma - \text{freq}_k(V_2 \mid G)_\Gamma| \leq 1/\varphi$. By the triangle inequality, the partitions V_1 and V_2 satisfy the prerequisite of Lemma 8 with $\lambda = 2/\varphi$. Additionally, it follows that

$$\|\text{freq}_k(G) - \text{freq}_k(V_1 \mid G)\|_1 \leq \frac{L}{\varphi}\,. \tag{9}$$

Let $G' = (V, E')$ be a copy of G. As long as G' and the partition V_1, V_2 satisfy the prerequisite of Lemma 8 and condition ($\star$), we *rewire* the edges of G' according to Lemma 8 so that G' will satisfy the properties given by Ineq. (3) and (4) and Expr. (5), (cf. lines 4 to 10). When G' does not satisfy condition ($\star$) anymore, we let $H := G'[V_1]$ and we are done. Note that at the end of the process, G' satisfies Eq. (6), which implies that the number of edges between V_1 and V_2 in G', that is, the *boundary* of H, is at most

$$6d^{2k+2}L + 2\lambda dL \cdot \min(|V_1|, |V_2|) \leq 6d^{2k+2}L + \frac{4dL}{\varphi} \cdot \varphi \leq 7d^{2k+2}L.$$

Now note that for any vertex $v \in H$, the k-disc of v in H differs from the k-disc of v in G' only if v is within distance at most k to the boundary of H, which in turn has size at most $7d^{2k+2}L$. By Fact 4, we have that the total number of vertices in H with different k-discs in H and G' is at most $3d^k/2 \cdot 7d^{2k+2}L \leq 11d^{3k+2}L$, which implies that

$$\|\text{freq}_k(V_1 \mid G) - \text{freq}_k(H)\|_1 \leq \frac{11d^{3k+2}L}{\varphi}\,. \tag{10}$$

It follows from Eqs. (9) and (10) and the triangle inequality that

$$\|\mathrm{freq}_k(G) - \mathrm{freq}_k(H)\|_1 \leq \frac{L + 11d^{3k+2}L}{\varphi} \leq \frac{12d^{3k+2}L}{\varphi} \leq \epsilon,$$

where the last inequality follows from our choice of φ.

Finally, we note that the graph H can be constructed by the following deterministic algorithm. We first compute the frequency vector $\mathrm{freq}_k(G)$ of G, which takes time $|V(G)|$. Then we consider all d-bounded graphs of size at most φ and output the graph H with frequency vector that is closest to $\mathrm{freq}_k(G)$ in ℓ_1-norm distance, which can be done in constant time. In total, the running time of the algorithm is $O(|V(G)|)$. ◀

Finally, we give a short proof of Corollary 3.

Proof of Corollary 3. We first note that the number of vertices whose k-discs may be altered by inserting / deleting a single edge $e = (u, v)$ is upper bounded by the number of k-discs that contain this edge. The number of such k-discs is exactly the number of vertices w such that there exists a path of length at most k from u to w and a path of length at most k from v to w, and is thus upper bounded by $1 + d + d(d-1) + \cdots + d(d-1)^{k-1} \leq 3d^k/2$.

Let $\delta = \frac{\varepsilon}{6d^{k+1}}$. Since G' is δ-close to G, G' can be obtained from G by inserting / deleting at most $\delta d|V|$ edges, and thus the total number of vertices that may have different k-discs in G and G' is at most $\delta d|V| \cdot 3d^k/2$. Finally, since a vertex that has different k-discs in G and G' may contribute at most $2/|V|$ to the $s\ell_1$-norm distance of $\mathrm{freq}_k(G)$ and $\mathrm{freq}_k(G')$, we have

$$\|\mathrm{freq}_k(G) - \mathrm{freq}_k(G')\|_1 \leq \frac{2}{|V|} \cdot \left(\delta d|V| \cdot \frac{3d^k}{2}\right) = 3\delta d^{k+1} \leq \frac{\varepsilon}{2}\,.$$

Now since G' satisfies that $\mathrm{girth}(G') \geq 2k + 2$, by Theorem 2, we know that there exists a graph H with size at most $72\frac{d^{3k+2}L}{\varepsilon}$ such that $\|\mathrm{freq}_k(G') - \mathrm{freq}_k(H)\|_1 \leq \varepsilon/2$. Therefore, $\|\mathrm{freq}_k(G) - \mathrm{freq}_k(H)\|_1 \leq \varepsilon$. ◀

Acknowledgements. We thank anonymous referees for their valuable comments concerning the presentation of the paper.

References

1 D. Aldous and R. Lyons. Processes on Unimodular Random Networks. *Electronic Journal of Probability*, 12(54):1454–1508, 2007.

2 A. Benczúr and D. Karger. Approximating s-t Minimum Cuts In $\tilde{O}(n^2)$ Time. In *Proceedings of the 28th annual ACM Symposium on Theory of Computing*, pages 47–55. ACM, 1996.

3 I. Benjamini, O. Schramm, and A. Shapira. Every Minor-Closed Property of Sparse Graphs Is Testable. *Advances in Mathematics*, 223(6):2200–2218, 2010.

4 P. Chew. There Are Planar Graphs Almost as Good as the Complete Graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.

5 G. Elek. On the Limit of Large Girth Graph Sequences. *Combinatorica*, 30(5):553–563, 2010.

6 J. Friedman. *A proof of Alon's second eigenvalue conjecture and related problems.* American Mathematical Society, 2008.

7 O. Goldreich and D. Ron. Property Testing in Bounded Degree Graphs. *Algorithmica*, 32:302–343, 2002.

8 A. Hassidim, J. Kelner, H. Nguyen, and K. Onak. Local Graph Partitions for Approximation and Testing. In *Proceedings of the 50th annual IEEE Symposium on Foundations of Computer Science*, pages 22–31. IEEE, 2009.

9 P. Indyk, A. McGregor, I. Newman, and K. Onak. Bertinoro workshop on sublinear algorithms 2011. `http://sublinear.info/42`. In *Open Problems in Data Streams, Property Testing, and Related Topics*, 2011.

10 L. Lovász. Very Large Graphs. *arXiv preprint arXiv:0902.0132*, 2009.

11 L. Lovász. *Large Networks and Graph Limits*. American Mathematical Society, 2012.

12 B. McKay, N. Wormald, and B. Wysocka. Short Cycles in Random Regular Graphs. *Electronic Journal of Combinatorics*, 11(1):66, 2004.

13 I. Newman and C. Sohler. Every Property of Hyperfinite Graphs Is Testable. *SIAM Journal on Computing*, 42(3):1095–1112, 2013.

14 D. Spielman and S. Teng. Spectral Sparsification of Graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

15 E. Szemerédi. Regular Partitions of Graphs. Technical report, DTIC Document, 1975.

Dimension Expanders via Rank Condensers[*]

Michael A. Forbes[†1] and Venkatesan Guruswami[‡2]

1 **School of Mathematics, Institute for Advanced Study**
Princeton, NJ, USA
`miforbes@csail.mit.edu`

2 **Computer Science Department, Carnegie Mellon University**
Pittsburgh, PA, USA
`guruswami@cmu.edu`

Abstract

An emerging theory of "linear algebraic pseudorandomness" aims to understand the linear algebraic analogs of fundamental Boolean pseudorandom objects where the rank of subspaces plays the role of the size of subsets. In this work, we study and highlight the interrelationships between several such algebraic objects such as subspace designs, dimension expanders, *seeded rank condensers*, *two-source rank condensers*, and rank-metric codes. In particular, with the recent construction of near-optimal subspace designs by Guruswami and Kopparty [12] as a starting point, we construct good (seeded) rank condensers (both *lossless* and *lossy* versions), which are a small collection of linear maps $\mathbb{F}^n \to \mathbb{F}^t$ for $t \ll n$ such that for every subset of $\mathbb{F}^n$ of small rank, its rank is preserved (up to a constant factor in the lossy case) by at least one of the maps.

We then compose a tensoring operation with our lossy rank condenser to construct constant-degree dimension expanders over polynomially large fields. That is, we give $O(1)$ explicit linear maps $A_i : \mathbb{F}^n \to \mathbb{F}^n$ such that for any subspace $V \subseteq \mathbb{F}^n$ of dimension at most $n/2$, $\dim\bigl(\sum_i A_i(V)\bigr) \geqslant (1+\Omega(1))\dim(V)$. Previous constructions of such constant-degree dimension expanders were based on Kazhdan's property T (for the case when $\mathbb{F}$ has characteristic zero) or monotone expanders (for every field $\mathbb{F}$); in either case the construction was *harder* than that of usual vertex expanders. Our construction, on the other hand, is *simpler*.

For two-source rank condensers, we observe that the lossless variant (where the output rank is the product of the ranks of the two sources) is equivalent to the notion of a linear rank-metric code. For the lossy case, using our seeded rank condensers, we give a reduction of the general problem to the case when the sources have high ($n^{\Omega(1)}$) rank. When the sources have $O(1)$ rank, combining this with an "inner condenser" found by brute-force leads to a two-source rank condenser with output length nearly matching the probabilistic constructions.

1998 ACM Subject Classification F.2.1 Numerical Algorithms and Problems

Keywords and phrases dimension expanders, rank condensers, rank-metric codes, subspace designs, Wronskians

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.800

[*] A full version of this work can be found at `http://arxiv.org/abs/1411.7455`.
[†] This work was performed when the author was a graduate student at MIT CSAIL (which was supported by Scott Aaronson's Waterman award, NSF CCF-1249349), and when the author was a Google Research Fellow at the Simons Institute for the Theory of Computing.
[‡] Some of this work was done when the author was a visiting researcher at Microsoft Research New England, Cambridge, MA. Research supported in part by NSF grant CCF-0963975.

1 Introduction

The broad area of pseudorandomness deals with efficiently generating objects that exhibit the desirable properties of "random-like" objects despite being constructed either explicitly or with limited randomness. Pseudorandomness is a central and influential theme in many areas such as complexity theory, derandomization, coding theory, cryptography, high-dimensional geometry, graph theory, and additive combinatorics. The topic has witnessed much progress over the years and continues to be intensively studied. We now have non-trivial constructions of various pseudorandom objects such as expander graphs, randomness extractors and condensers, Ramsey graphs, list-decodable codes, compressed sensing matrices, Euclidean sections, and pseudorandom generators for various concrete models. Despite the seemingly different definitions and contexts of these objects, insights in pseudorandomness have uncovered intimate connections between them, and this has led to a rich theory of "Boolean pseudorandomness" drawing a common pool of broadly useful techniques (see for instance the recent survey by Vadhan [26].)

Recently, there is an emerging theory of "linear-algebraic pseudorandomness" aimed at understanding the linear-algebraic analogs of fundamental Boolean pseudorandom objects where the dimension of subspaces plays the role analogous to min-entropy. Examples of such algebraic objects include dimension expanders, subspace-evasive sets, subspace designs, rank-preserving condensers, etc. In addition to their intrinsic interest, these notions also have surprising applications; for instance, subspace-evasive sets to the construction of Ramsey graphs [22] and list-decodable codes [13, 15], subspace designs to list decoding both in the Hamming metric and the rank metric [16, 14], and rank-preserving condensers to affine extractors [10] [1] and polynomial identity testing [18, 9].

In this work, we study several interesting pseudorandom objects in the linear-algebraic world, such as subspace evasive sets, subspace designs, dimension expanders, *seeded rank condensers*, and *two-source rank condensers*. The last two notions are also introduced in this work, though closely related concepts were studied earlier in the literature. We briefly and informally define these notions now, with more precise statements appearing in later sections. A subspace evasive set is a (large) subset of $\mathbb{F}^n$ that has small intersection with every low-dimensional subspace of $\mathbb{F}^n$. Subspace designs are a (large) collection of subspaces such that every low-dimensional subspace intersects few of them. Dimension expanders are a (small) collection of linear maps $A_i : \mathbb{F}^n \to \mathbb{F}^n$ such that for every subspace $V \subseteq \mathbb{F}^n$ of bounded dimension, the dimension of $\sum_i A_i(V)$ is at least $\alpha \cdot \dim(V)$ for a constant $\alpha > 1$ (so that the dimension grows, or *expands*). Rank condensers are a (small) collection of linear maps $\mathbb{F}^n \to \mathbb{F}^t$ (for $t \ll n$) such that for every subspace of dimension r, its image under at least one of the maps has large dimension. That is, the ambient dimension n is *condensed* to t while roughly the preserving the rank (to r in the *lossless* case (so that no rank is lost), and to $\Omega(r)$ in the *lossy* case). A two-source rank condenser is a map $E : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}^t$ such that for every pair $A, B \subseteq \mathbb{F}^n$ with rank r each, $f(A \times B)$ has rank $\Omega(r^2)$ (or even r^2 in the lossless case) – the tensor product construction is lossless but requires $t = n^2$, so the challenge here is to "derandomize" the tensor product and achieve $t \ll n^2$ (and even $t \ll n$

[1] Despite the usage of rank condensers in the Gabizon-Raz [10] construction of affine extractors, affine extractors seem to not quite fit the restricted notion of a "linear-algebraic pseudorandom object" in the sense of this paper. That is, the objects we consider focus on functions and their interactions with the rank of certain sets of vectors. In contrast, affine extractors (maps which convert uniform distributions over large-enough subspaces of the input into uniform distributions over full-dimensional subspaces) require further statistical properties. The weaker notion of an affine disperser (a map which is almost surjective on its range when applied to a large-enough subspace of the input) similarly requires one-sided statistical guarantees.

for the lossy case for $r \ll \sqrt{n}$).

We remark that there are two perspectives on the above objects. The first is that of *subspaces*, so that we only consider subspaces and their dimension. The second is that of *sets of vectors*, where we consider arbitrary sets of vectors measured by their rank (the dimension of their span). When the underlying functions are linear (or multilinear) these viewpoints are equivalent. For example, one can equally discuss dimension expanders as expanding the dimension of subspaces or as increasing the rank of matrices through matrix multiplication. In this work, we take both views, using "dimension" to refer to subspaces and "rank" to refer to the dimension of the span of a set of vectors.

Conceptually, our work highlights close interconnections between the above pseudorandomness notions. In particular, we show that subspace designs (which were introduced in the context of list decoding variants of algebraic-geometric codes of Guruswami and Xing [16]) are the *same* concept as lossless rank condensers while emphasizing a different regime of parameters. This connection also highlights that a strong variant of subspace designs yields lossy rank condensers. The near-optimal explicit construction of (strong) subspace designs of Guruswami-Kopparty [12] then yields lossless and lossy rank condensers with parameters close to the existential constructions. Our main technical application is an explicit construction of constant-degree dimension expanders over polynomially large fields, that expands all subspaces of $\mathbb{F}^n$ of dimension $n/2$ (say) by a factor $\alpha > 1$. We achieve this construction by first increasing the rank in a trivial way by increasing the dimension of the ambient space, and then using a lossy rank condenser to reduce the ambient space back to $\mathbb{F}^n$ while preserving the rank up to a constant factor. While previous constructions of dimension expanders were at least as complicated as constructions of standard expander graphs (or more so), our construction and analysis is rather elementary. Unfortunately, unlike previous work, our techniques are currently best suited to large fields due to connections with Reed-Solomon codes. However, we do obtain dimension expanders over small fields by paying various logarithmic penalties.

Turning to two-source rank condensers, our original motivation to propose them was a possible route to iteratively construct subspace-evasive sets that might offer some way around the exponential dependence on intersection size that seems inherent to constructions based on algebraic varieties. While there appears to be serious obstacles to such an approach, the notion seems a fundamental one to study regardless. In this work, we focus on two-source rank condensers $f : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}^t$ where the map f is bilinear as this seems like a natural class of constructions to study. We observe that the lossless variant is *equivalent* to the notion of a linear rank-metric code. Known optimal constructions of rank-metric codes such as the Gabidulin codes thereby yield lossless two-source condensers with optimal output length (equal to $\Theta(nr)$ for rank-r subsets of $\mathbb{F}^n$). For lossy two-source rank condensers, we can enumerate over the seeds of our seeded lossy condenser, applying it to both sources separately and condensing the sources to $r^{\Theta(1)}$ dimensions (from the original n). For small r (e.g., constant), we can "concatenate" this construction with a near-optimal lossy two-source condenser found by brute-force to obtain output length $\Theta(n/r)$, matching the non-constructive bound. In general, our method reduces the problem to the case of relatively high "rate" (when $r \approx n^{1/3}$), which is typically easier to tackle.

Organization: In the next three sections, we state (informal versions of) our results, all of ideas behind them, and brief discussions of prior work for seeded rank condensers (section 2), dimension expanders (section 3), and two-source rank condensers (section 4). An expanded treatment with formal statements and proofs can be found in the full version of this work (arXiv:1411.7455).

2 Subspace Designs and Rank Condensers

We begin by discussing the notion of a *subspace design*, as recently defined by Guruswami and Xing [16], and contrast this with the notion of a *seeded (single source) rank condenser* to which we add the qualifier of *lossless*, as defined by Forbes, Saptharishi and Shpilka [8]. We will describe how these objects are essentially the same notion, where the rank condenser can be considered the "primal" object and the subspace design the "dual" object. We then introduce *lossy rank condensers*, a new notion that is key to our construction of dimension expanders see section 3) and describe how the construction of subspace designs of Guruswami and Kopparty [12] implies nearly optimal lossy rank condensers.

2.1 Subspace Designs

We begin with the definition of a subspace design, which is a collection of subspaces $\{H_i\}_i$ such that small-dimensional subspaces V intersect few of the H_i.

▶ **Definition 1** (Guruswami-Xing [16] and Guruswami-Kopparty [12]). Let $\mathbb{F}$ be a field. A collection $\mathcal{H} = \{H_i\}_i$ of subspaces $H_i \subseteq \mathbb{F}^n$ is a **weak** (r, L)**-subspace design** if for every subspace $V \subseteq \mathbb{F}^n$ with $\dim V = r$,

$$|\{i \mid \dim(H_i \cap V) > 0\}| \leqslant L \; .$$

The collection $\mathcal{H}$ is a **strong** (r, L)**-subspace design** if for every subspace $V \subseteq \mathbb{F}^n$ with $\dim V = r$,

$$\sum_i \dim(H_i \cap V) \leqslant L \; .$$

The collection $\mathcal{H}$ is **explicit** if given an index $i \in [|\mathcal{H}|]$ a basis for the i-th subspace in $\mathcal{H}$ can be constructed in $\mathsf{poly}(n, \log |\mathcal{H}|)$ operations in $\mathbb{F}$.

We note here that the above subspaces H_i are not constrained to be of equal dimension. Allowing the dimension of the H_i to vary could conceivably allow for improved constructions, but no construction so far uses this freedom. As such, we will primarily concern ourselves with the case when the dimensions are equal.

Guruswami-Xing [16] defined subspace designs as a way to prune list-decodable codes to ensure a small list-size while maintaining high rate. As such, one wishes for the size $|\mathcal{H}|$ of the design to be large while maintaining L of moderate size. In particular, they showed that large designs exist non-constructively.

▶ **Proposition** (Guruswami-Xing [16]). *Let $\mathbb{F}_q$ be a finite field. Let $\epsilon > 0$, $n \geqslant 8/\epsilon$ and $s \leqslant \epsilon n/2$. Then there is a strong $(s, 8s/\epsilon)$-subspace design $\mathcal{H}$ of $(1-\epsilon)n$-dimensional subspaces in $\mathbb{F}_q^n$ with $|\mathcal{H}| = q^{\epsilon n/8}$.*

Note that the *co*-dimension of the subspaces in $\mathcal{H}$ is ϵn, which is twice that of the maximum dimension $s \approx \epsilon n/2$. We now further remark on the variations of this definition. The following relation between the weak and strong versions is immediate.

▶ **Lemma 2** (Guruswami-Kopparty [12]). *Let $\mathbb{F}$ be a field, and let $\mathcal{H}$ be a collection of subspaces in $\mathbb{F}^n$. Then if $\mathcal{H}$ is a strong (r, L)-subspace design, then $\mathcal{H}$ is a weak (r, L)-subspace design. If $\mathcal{H}$ is a weak (r, L)-subspace design, then $\mathcal{H}$ is a strong (r, rL)-subspace design.*

We also observe that as every dimension $\leqslant r$ subspace can be padded to a dimension r subspace, we immediately can see that subspace designs apply to smaller subspaces as well.

▶ **Lemma 3.** *Let $\mathbb{F}$ be a field, and let $\mathcal{H}$ be a weak/strong (r, L)-subspace design in $\mathbb{F}^n$. Then $\mathcal{H}$ is a (s, L)-subspace design over $\mathbb{F}^n$ for every $1 \leqslant s \leqslant r$.*

While the above seems to allow one to focus on dimension r as opposed to dimension $\leqslant r$, this is not strictly true as one can achieve a better list size L for dimension $s \ll r$. Similarly, the above lemma relating strong and weak designs seems to suggest that qualitatively (up to polynomial factors) these notions are the same. However, as described in the full version, obtaining the appropriate (strong) list size simultaneously for all $s \leqslant r$ will be crucial for our application to constant-degree dimension expanders.

2.2 Seeded Lossless Rank Condensers

Strong subspace designs ask that for any small subspace V there is some $H_i \in \mathcal{H}$ so that $H_i \cap V$ is *small* (that is, by averaging, $\dim H_i \cap V \leqslant L/|\mathcal{H}|$). Equivalently, the amount of dimension in V that is outside H_i is *large* so that in some sense the dimension of V is preserved. This perspective is more naturally phrased in the language of *(seeded) rank condensers*, as defined by Forbes, Saptharishi and Shpilka [8]. The definition we use here is tuned to the equivalence with subspace designs, and we recover their definition as the lossless version of what we term here a *lossy seeded rank condenser* (see Theorem 6). We will discuss prior work and motivation for rank condensers that is less immediately relevant in the full version.

We begin with the definition of rank condensers, which are a collection of linear maps $\varphi : \mathbb{F}^n \to \mathbb{F}^t$ (given explicitly as matrices $E \in \mathbb{F}^{t\times n}$) such that for any small-dimensional subspace V, most of the maps have $\dim \varphi(V) = \dim V$.

▶ **Definition 4.** Let $\mathbb{F}$ be a field and $n \geqslant r \geqslant 1$. A collection of matrices $\mathcal{E} \subseteq \mathbb{F}^{t\times n}$ is a **weak (seeded)** (r, L)**-lossless rank condenser** if for all matrices $M \in \mathbb{F}^{n\times r}$ with $\operatorname{rank} M = r$,

$$|\{E \mid E \in \mathcal{E}, \operatorname{rank} EM < \operatorname{rank} M\}| \leqslant L .$$

The collection $\mathcal{E}$ is a **strong (seeded)** (r, L)**-lossless rank condenser** if for all matrices $M \in \mathbb{F}^{n\times r}$ with $\operatorname{rank} M = r$,

$$\sum_{E\in\mathcal{E}} (\operatorname{rank} M - \operatorname{rank} EM) \leqslant L .$$

The collection $\mathcal{E}$ is **explicit** if given an index $i \in [|\mathcal{E}|]$ the i-th matrix of $\mathcal{E}$ can be constructed in $\mathsf{poly}(t, n, \log |\mathcal{E}|)$ operations in $\mathbb{F}$.

As we have many types of condensers in this paper (weak, strong, lossless, lossy, two-source, etc.) we will often just refer to them as "condensers" (perhaps with some relevant parameters such as "(r, ϵ)") when the relevant adjectives are clear from context.

As it can only increase the quality of the condenser, one naturally considers the case when $\operatorname{rank} E = t$ for all $E \in \mathcal{E}$. However, we do not impose this restriction just as we do not impose the condition that subspaces in subspace designs all have the same dimension. In fact, by the equivalence of subspace designs and lossless rank condensers one can see that these two restrictions are equivalent.

We briefly remark that as all of the pseudorandom objects we consider in this work are linear (or in the case of two-source condensers, bilinear) we will often freely pass between subspaces $V \subseteq \mathbb{F}^n$ of dimension r and matrices $M \in \mathbb{F}^{n\times r}$ of rank r, using that we can choose a basis for V so that col-span $M = V$. As such, we will often treat a matrix $M \in \mathbb{F}^{n\times r}$ as a list of r vectors in $\mathbb{F}^n$.

We now note that subspace designs are equivalent to lossless rank condensers.

▶ **Proposition 5.** *Let $\mathbb{F}$ be a field and $n \geqslant r \geqslant 1$. Let $\mathcal{H} = \{H_i\}_{i\in[N]}$ be a collection of subspaces $H_i \subseteq \mathbb{F}^n$ and let $\mathcal{E} = \{E_i\}_{i\in[N]} \subseteq \mathbb{F}^{t\times n}$ be a collection of matrices, where we have that* row-span $E_i = (H_i)^\perp$ *for $i \in [N]$. Then $\mathcal{H}$ is a weak/strong (r, L)-subspace design iff $\mathcal{E}$ is a weak/strong (r, L)-lossless rank condenser.*

While the above proposition is quite simple, it offers a unifying perspective of these different objects which was key to obtaining further results.

2.3 Seeded Lossy Rank Condensers

While the above seeded lossless rank condensers already have applications to list-decodable codes, rank condensers were defined in Forbes, Saptharishi and Shpilka [8] for quite different reasons. We now give a definition closer to their motivation.

▶ **Definition 6.** Let $\mathbb{F}$ be a field and $n \geqslant r \geqslant 1$ and $\epsilon \geqslant 0$. A collection of matrices $\mathcal{E} \subseteq \mathbb{F}^{t\times n}$ is a **(seeded)** (r, ϵ)**-lossy rank condenser** if for all matrices $M \in \mathbb{F}^{n\times r}$ with $\operatorname{rank} M = r$,

$$\operatorname{rank} EM \geqslant (1 - \epsilon) \operatorname{rank} M \ ,$$

for some $E \in \mathcal{E}$. The collection $\mathcal{E}$ is a **(seeded)** $(\leqslant r, \epsilon)$**-lossy rank condenser** if it a (s, ϵ)-lossy condenser for all $1 \leqslant s \leqslant r$.

The collection $\mathcal{E}$ is **explicit** if given an index $i \in [|\mathcal{E}|]$ the i-th matrix of $\mathcal{E}$ can be constructed in $\mathsf{poly}(t, n, \log |\mathcal{E}|)$ operations in $\mathbb{F}$.

This notion is a natural linear-algebraic analogue of condensers for *min-entropy* from the realm of Boolean pseudorandomness. One contrast is that we do not require that *most* $E \in \mathcal{E}$ have the desired condensing property as this does not seem important for our applications, although we note that our constructions can meet this stronger requirement with the appropriate modifications[2].

In is worthwhile to contrast this object with subspace designs or lossless rank condensers. The goal of subspace designs was (due to connections with list-decodable codes) to construct a *large* design while less focus was on the exact list-size bound. Here, we have the somewhat different goal of obtaining a *small* collection of matrices, which is akin to obtaining a very small list size in a subspace design. The focus on the collection being small is from the use of such condensers in derandomization, as we will need to enumerate over each matrix in the collection.

In particular, the notion of a $(r, 0)$-lossy rank condenser is of interest because it is *lossless*, which is important for many applications. In particular, this notion was previously defined as a "*rank condenser (hitting set)*" in the work of Forbes, Saptharishi and Shpilka [8], but the construction and usage of these objects predates them[3]. In particular, Gabizon and Raz [10] constructed a $(r, 0)$-condenser with size nr^2, and they used this to construct affine extractors over large fields. Karnin and Shpilka [18] named the construction of Gabizon and Raz [10] to be "rank preserving subspaces" and used this construction to make a *polynomial*

[2] More precisely, this stronger definition requiring most E to condense rank is closer to the definition of a min-entropy condenser. Only requiring *some* E to condense rank is more akin to the notion of a *somewhere condenser* as defined by Barak-Kindler-Shaltiel-Sudakov-Wigderson [3].

[3] We note that the works we highlight are not necessarily the first or last in their respective lines of research, and rather we only highlight those that (to the best of our knowledge) had results concerning lossless rank condensers.

identity testing[4] algorithm of Dvir and Shpilka [7] work in the *black box* model. Forbes and Shpilka [9] later gave an improved construction of a rank condenser with only nr size, and showed how they can be used to make another polynomial identity testing algorithm of Raz and Shpilka [24] work in the black-box model. Forbes, Saptharishi and Shpilka [8], building on the work of Agrawal, Saha, and Saxena [1], analyzed "multivariate" lossless rank condensers as they arose naturally in a polynomial identity testing algorithm.

Beyond applications to polynomial identity testing, Lokshtanov, Misra, Panolan and Saurabh [19] used these condensers to derandomize a fixed-parameter-tractable algorithm of Marx [21] for ℓ-matroid intersection. Cheung, Kwok and Lau [6] rediscovered the rank condenser of Gabizon and Raz [10] and (among other things) used this to give faster randomized algorithms for exact linear algebra. Forbes, Saptharishi and Shpilka [8] showed a generic recipe to construct such rank condensers from *any* error-correcting code (over large fields). Given these applications and connections present in (r, ϵ)-lossy rank condensers for $\epsilon = 0$, we expect the $\epsilon > 0$ version will similarly have many applications.

We now quote the parameters given by the probabilistic method.

▶ **Proposition 7.** *Let $\mathbb{F}_q$ be a finite field. Let $n \geqslant r \geqslant 1$, $\epsilon \geqslant 0$ and $t > (1-\epsilon)r$. Then there is a collection $\mathcal{E}$ of k matrices $\mathcal{E} \subseteq \mathbb{F}_q^{t\times n}$ that is a (r, ϵ)-lossy rank condenser whenever*

$$k \geqslant \frac{rn + o_q(1)}{(t-(1-\epsilon)r)(\lfloor \epsilon r\rfloor + 1) - o_q(1)} \ . \tag{2.1}$$

For $\epsilon > 0$, there is a collection $\mathcal{E}$ of size k that is a $(\leqslant r, \epsilon)$-lossy rank condenser whenever

$$k \geqslant \frac{n + o_q(1)}{\epsilon(t-(1-\epsilon)r) - o_q(1)} \ .$$

Thus we can make the output size t of the condenser to be almost equal to the guaranteed dimension bound of $(1-\epsilon)r$. Further, we see that there is essentially no penalty in (existentially) insisting for a $(\leqslant r, \epsilon)$-condenser over a (r, ϵ)-condenser. However, we show in the full version that the notion of $(\leqslant r, \epsilon)$-condenser is provably stronger.

2.4 Our Results

We now turn to our constructions of condensers. We begin with the following construction, which is the rank condenser of Forbes and Shpilka [9] and was named the *folded Wronskian* by Guruswami-Kopparty [12].

▶ **Construction 8** (Folded Wronskian). *Let $\mathbb{F}$ be a field. Let $\omega \in \mathbb{F}$ be an element of multiplicative order $\geqslant n$. Define the matrix $\mathrm{W}_{t,\omega}(x) \in \mathbb{F}[x]^{[\![t]\!]\times[\![n]\!]}$ by $(\mathrm{W}_{t,\omega}(x))_{i,j} := (\omega^i x)^j$.*

Identifying $\mathbb{F}^{[\![n]\!]}$ with the vector space of degree $< n$ polynomials $\mathbb{F}[x]^{<n}$, the matrix $\mathrm{W}_{t,\omega}(x)$ defines the linear map $\mathrm{W}_{t,\omega}(x) : \mathbb{F}[x]^{<n} \to \mathbb{F}[x]^t$ given by

$$f(x) \mapsto (f(x), f(\omega x), \ldots, f(\omega^{t-1}x)) \ .$$

That is, we define $[\![n]\!] := \{0, \ldots, n-1\}$ so that in the above the indices i and j are indexed from zero. When the value of ω is clear from context we will just write "W_t". Note

[4] The *polynomial identity testing problem* is when given a algebraic circuit C (perhaps from a restricted class of circuits) to *deterministically* decide whether the circuit C computes the identically zero polynomial. The *black box* version is where we only allow access to C by evaluating the polynomial it computes. See Shpilka and Yehudayoff [25] for more on this problem.

that the fact that ω has large multiplicative order means that we require a large field, in particular that $|\mathbb{F}| > n$.

The key result that forms the starting point for our constructions is the following analysis of the folded Wronskian by Guruswami and Kopparty [12]. While their analysis was originally in the context of subspace designs, we state their result here in the language of lossless rank condensers as it is more natural in our context.

▶ **Theorem 9** (Guruswami-Kopparty [12]). *Assume the setup of Theorem 8 where we take $t \geqslant r \geqslant 1$. Let $S \subseteq \{(\omega^\ell)^j \mid j \geqslant 0\}$ where $\ell \geqslant t - r + 1$. Then $\{\mathrm{W}_t(\alpha) \mid \alpha \in S\} \subseteq \mathbb{F}^{t\times n}$ is a strong $(r, \frac{r(n-r)}{t-r+1})$-lossless rank condenser.*

We note here that the above parameters are slightly stronger than what Guruswami and Kopparty [12] obtain, as they only obtain a list bound of $\frac{r(n-1)}{t-r+1}$. This improved bound follows by using some of the analysis from Forbes, Saptharishi and Shpilka [8] as explained in the full version. Note that this construction essentially matches the non-constructive bound (2.1) when $\epsilon = 0$.

The above analysis indicates that for a matrix $M \in \mathbb{F}^{n\times r}$ of rank r that the total rank loss over all maps in $\mathcal{E}$ is at most $\frac{r(n-r)}{t-r+1}$. Thus, by an averaging argument, at most $1/k \cdot \frac{r(n-r)}{t-r+1}$ such maps can have a rank loss of $\geqslant k$. This observation thus shows that the above construction is not just a *lossless* rank condenser but also a *lossy* condenser (with different parameters).

▶ **Corollary 10.** *Let $\mathbb{F}$ be a field. Let $n, t \geqslant r \geqslant 1$ and $\epsilon > 0$, where $\omega \in \mathbb{F}$ is an element of multiplicative order $\geqslant \mathsf{poly}(n)$. Define $\mathcal{E} := \{\mathrm{W}_{t,\omega}((\omega^t)^j) \mid 0 \leqslant j < \frac{n}{\epsilon(t-r+1)}\}$, that is, the folded Wronskian evaluated at $\frac{n}{\epsilon(t-r+1)}$ distinct powers of ω^t. Then $\mathcal{E}$ is an explicit $(\leqslant r, \epsilon)$-lossy rank condenser.*

To motivate our below application to dimension expanders, suppose that $r = n/3$, $t = n/2$ and $\epsilon > 0$. This says then that we construct a rank condenser that maps $\mathbb{F}^n$ to $\mathbb{F}^{n/2}$ that maps rank $n/3$ subspaces to rank $(1-\epsilon)n/3$ subspaces. Further, this condenser is a collection of at most

$$\frac{n}{\epsilon(n/2 - n/3)} = 6/\epsilon$$

maps such that one map from the collection always preserves the desired rank. To obtain these parameters, it is key to the analysis that we have a *strong* lossless condenser and that it obtains the (near-optimal) bound given by Guruswami and Kopparty [12]. Note that these condensing parameters are similar to the min-entropy condensers of Raz [23] and Barak-Kindler-Shaltiel-Sudakov-Wigderson [3], which use a constant number of random bits to condense a source with a constant-rate of min-entropy.

3 Dimension Expanders

We now turn to our main object of interest, *dimension expanders*. Dimension expanders were defined by Barak, Impagliazzo, Shpilka and Wigderson [2] in an attempt to translate challenges in the explicit construction of objects in Boolean pseudorandomness into the regime of linear algebra. Indeed, in combinatorics there is a well-established analogy between subsets of $[n]$ and subspaces of vector spaces over finite fields. In the context of pseudorandomness, we can then translate questions that manipulate the *size* of subsets $S \subseteq \{0,1\}^n$ (or more generally, the min-entropy of distributions over $\{0,1\}^n$) into questions about manipulating the *dimension* of subspaces $V \subseteq \mathbb{F}^n$. While these regimes seem different, it is conceivable that such linear algebraic constructions could yield new constructions in Boolean pseudorandomness (such as how the inner-product function is a two-source extractor). Indeed, as in the work of Guruswami and Wang [13], this idea has borne fruit (if in a perhaps unexpected way) by showing how linear-algebraic pseudorandom objects can improve list-decodable codes. We now define dimension expanders.

▶ **Definition 11.** Let $\mathbb{F}$ be a field, $n \geqslant 1$, $\epsilon > 0$ and $\alpha \in \mathbb{R}$ with $\alpha \geqslant 1$. A collection of matrices $\mathcal{A} = \{A_1, \ldots, A_d\} \subseteq \mathbb{F}^{n \times n}$ is a (ϵ, α)**-dimension expander of degree** d if for all subspaces $V \subseteq \mathbb{F}^n$ of dimension $\leqslant \epsilon n$ that

$$\dim \sum_{i=1}^{d} A_i(V) = \dim \operatorname{span}\{A_i(V)\}_{i=1}^{d} \geqslant \alpha \cdot \dim V \ .$$

The collection $\mathcal{A}$ is **explicit** if given an index $i \in [|\mathcal{A}|]$ the i-th matrix in $\mathcal{A}$ can be constructed in $\mathsf{poly}(n, \log |\mathcal{A}|)$ operations in $\mathbb{F}$.

We remark that in the above definition one can generally assume that all of the maps A_i are of full-rank, as that can only increase $\dim \sum_{i=1}^{d} A_i(V)$. Similarly, one can assume that A_1 equals the identity matrix I_n as we can use the transform $A_i \mapsto A_1^{-1} A_i$ as again this does not affect the size of the outputted dimension. While these assumptions are thus without loss of generality, we will not impose them.

In general we will be most interested in $(\Omega(1), 1 + \Omega(1))$-dimension expanders of constant degree, which we shall thus call "dimension expanders" without any quantification. This parameter regime is of interest because it matches that of the probabilistic method, which we quote the results of below.

▶ **Proposition 12.** *Let $\mathbb{F}_q$ be a finite field, $n \geqslant 1$, $\epsilon > 0$ and $\alpha \in \mathbb{R}$ with $\alpha \geqslant 1$. Then there exist a collection matrices $\mathcal{A} = \{A_1, \ldots, A_d\} \subseteq \mathbb{F}^{n \times n}$ which is a (ϵ, α)-dimension expander of degree d whenever*

$$d \geqslant \alpha + \frac{1}{1 - \alpha\epsilon} + o_q(1) \ .$$

Put into more concrete terms, we see that one can existentially obtain $(1/2d, d - O(1))$-dimension expansion with degree d. That we have an expansion of $(1 - \epsilon)d$ in a degree d expander is akin to *lossless (vertex) expanders* which have a similar degree/expansion relation, and these expanders have applications beyond those of normal expanders (see Capalbo, Reingold, Vadhan and Wigderson [5] and references therein). While previous work focused on obtaining constant-degree dimension expanders, our work raises the questions of obtaining *lossless* dimension expanders so that we match the above bound. Our work, as discussed below, lends itself to being particularly quantitative with regards to the size and parameters of the construction. However, we do not obtain lossless dimension expanders, and to the best of our knowledge, neither do the other previous constructions of dimension expanders discussed below.

While we discuss prior work in depth in the full version, we briefly summarize the state of art in dimension expanders in the following theorems.

▶ **Theorem** (Lubotzky and Zelmanov [20] and Harrow [17])**.** *Let $\mathbb{F}$ be a field of characteristic zero and $n \geqslant 1$. There exists an explicit $O(1)$-sized collection $\mathcal{A} \subseteq \mathbb{F}^{n \times n}$ such that $\mathcal{A}$ is a $(1/2, 1 + \Omega(1))$-dimension expander over $\mathbb{F}^n$.*

This construction requires characteristic zero as it uses a notion of distance that lacks a good definition in finite characteristic.

▶ **Theorem** (Bourgain and Yehudayoff [4]). *Let $n \geqslant 1$. There exists an explicit $O(1)$-sized collection $\mathcal{A} \subseteq \{0,1\}^{n\times n}$ such that $\mathcal{A}$ is a $(1/2, 1+\Omega(1))$-dimension expander over $\mathbb{F}^n$, over every field $\mathbb{F}$.*

Note that the above construction is only a function of n, and not of the field, so that this *single* construction is a dimension expander over *all* fields.

As explained in the full version of this work, both of the above constructions in some way attempt to extend existing ideas about expander graphs into the world of dimension expanders. The first replicates the representation theory approach to constructing expanding Cayley graphs, and the second shows how bipartite expanders (with the strong requirement of *monotonicity*) extend to also be dimension expanders.

Our Work: In our work we take a different approach to constructing dimension expanders that treats such expanders as part of an emerging theme of *linear-algebraic* pseudorandomness as seen by recent linear-algebraic approaches to list-decoding [11, 15, 13, 16, 14] and linear-algebraic derandomization of subclasses of polynomial identity testing [18, 9]. The first consequence of this perspective is that we work in fields that are at least polynomially large as this is the setting of Reed-Solomon codes. To obtain dimension expanders over smaller fields, a natural solution within this theory is to use "code concatenation" ideas from coding theory. Unfortunately the idea of code concatenation is somewhat subtle in our setting and so only supplies a concatenation (based on converting Reed-Solomon codes to BCH codes) that incurs a logarithmic loss in the parameters. The second consequence is that we build our dimension expanders out of the existing linear-algebraic pseudorandom objects that have emerged from prior work. That is, just how in Boolean pseudorandomness the notions of expanders, extractors and list-decodable codes are all related (see for example Vadhan [26]), we leverage such connections to construct our expanders from the above mentioned rank condensers.

We now explain our construction, which while ultimately was motivated by the connections between two-source rank condensers and dimension expanders, can be explained in a self-contained manner. The first observation is that one can easily obtain "$(1, d)$-expanders" of degree $d \in \mathbb{N}$ if one is willing to allow the ambient space to grow. That is, consider the tensor product $\mathbb{F}^n \otimes \mathbb{F}^d = \mathbb{F}^{nd}$. By properties of the tensor product, for $V \subseteq \mathbb{F}^n$ of rank $r \leqslant n$ we know that $V \otimes \mathbb{F}^d$ is of rank rd in $\mathbb{F}^{nd}$. Further, $V \otimes \mathbb{F}^d$ can be seen as the image of d maps $T_i : \mathbb{F}^n \to \mathbb{F}^{nd}$ where the i-th map places the space $\mathbb{F}^n$ into the "i-th block" of $(\mathbb{F}^n)^d = \mathbb{F}^{nd}$. In analogy to bipartite expander graphs, this is akin to giving each left vertex its own "private neighborhood" of right vertices into which it expands.

While trivial, the above step now allows us to convert a question of *expansion* to a question of *condensing*. That is, tensoring achieves expansion only because the output of the maps are larger than the input, while the non-trivial aspect of dimension expanders is to expand while keeping the output size the *same*. However, tensoring *has* expanded dimension and thus we can now focus on reducing the output size. Specifically, suppose that we consider $V \subseteq \mathbb{F}^n$ of rank $r = n/2d$. Then its image under the above tensoring is $W := \sum_i T_i(V)$ of dimension $n/2$. This subspace W lies in an nd-dimensional space and we wish return it to an n-dimensional space while not losing too much in the dimension. However, this last problem is exactly the question of *lossy rank condensing*. For constant d, the above discussion shows that we can condense such constant-rate dimension in a lossy way using a *constant* number of maps. In this example, we can condense W to $\mathbb{F}^n$ using $\frac{dn}{\epsilon(n-n/2)} = \frac{2d}{\epsilon}$ maps, at least one of which produces a $(1-\epsilon)n/2$ dimensional space. Thus, this expands $V \subseteq \mathbb{F}^n$ of rank $\frac{n}{2d}$ to be of dimension $(1-\epsilon)n/2$ within $\mathbb{F}^n$, all while using $d \cdot \frac{2d}{\epsilon} = \frac{2d^2}{\epsilon}$ maps (we multiply the

$$\begin{tikzcd}
\mathbb{F}^n \arrow[r, "\text{tensoring}", "\text{degree } d"'] & \mathbb{F}^{nd} \arrow[r, "\begin{array}{c}(\leqslant \epsilon dn, \delta)\text{-lossy}\\ \text{condenser}\end{array}", "\text{degree } \frac{dn}{\delta(n-\epsilon dn)}"'] & \mathbb{F}^n \\
\dim \epsilon n \arrow[r] & \dim \epsilon dn \arrow[r] & \dim (1-\delta)\epsilon dn
\end{tikzcd}$$

Figure 1 Constructing dimension expanders from tensoring and lossy rank condensers.

number of maps due to the composition). We summarize this composition in Figure 1.

We note that the above discussion has only discussed constant-rate rank, that is, subspaces of $\mathbb{F}^n$ with rank $\Omega(n)$. Dimension expanders however are required to expand *all* small subspaces. Our construction also handles this case as the lossy rank condensers we use will preserve a $(1-\delta)$ fraction of the input rank, as long as that rank is small enough. In the above sketch there is also the technicality that we must tensor with $\mathbb{F}^d$ with d being *integral*, which restricts $d \geqslant 2$ as $d = 1$ does not yield expansion. With this construction alone one would only obtain expansion in $\mathbb{F}^n$ for rank $< n/d \leqslant n/2$, but we manage to sidestep this restriction by a simple truncation argument. Putting the above pieces together we obtain the following theorem.

▶ **Theorem 13** (Main Theorem). *Let $n, d \geqslant 1$ and let $0 < \epsilon \leqslant \eta < 1$ be constants. Let $\mathbb{F}$ be a field with $|\mathbb{F}| \geqslant \mathsf{poly}(n)$. There is an explicit $(\epsilon, \eta/\epsilon)$-dimension expander in $\mathbb{F}^n$ of degree $\Theta\left(\frac{1}{\epsilon^2(1-\eta)^2}\right)$. If $\epsilon < 1/d$ then for any $\delta > 0$ there is an explicit $(\epsilon, (1-\delta)d)$-dimension expander in $\mathbb{F}^n$ with degree $\frac{d^2}{\delta(1-\epsilon d)}$.*

These expanders yield an expansion of α with degree $\approx \alpha^2$, and thus are not lossless. In particular, existential methods show that there are $(\epsilon, \eta/\epsilon)$-dimension expanders with degree $\approx 1/\epsilon + \frac{1}{1-\eta}$. In remains an interesting challenge to obtain such lossless dimension expanders. In particular, we note that we get "all of the dimension" from the tensoring step using only *one* map from the condenser. This occurs despite the fact that *most* maps in the condenser preserve all of the dimension (assuming we double the seed length). It seems natural to hope that an integrated analysis of the tensoring and condensing stages would show that the construction has a better expansion than what we obtain.

Over small fields our results are comparatively weaker as we simulate a larger field within the small field (as how one transforms Reed-Solomon codes to BCH codes), so that we pay various logarithmic penalties.

▶ **Corollary 14.** *Let $\mathbb{F}_q$ be finite and $n, d \geqslant 1$. Then there are explicit $\left(\Theta\left(\frac{1}{d \log_q dn}\right), \Theta(d)\right)$-dimension expanders in $\mathbb{F}_q^n$ of degree $\Theta(d^2 \log_q dn)$.*

4 Two-Source Rank Condensers

In the context of Boolean pseudorandomness, it is well known (see for example Vadhan [26]) that strong min-entropy seeded extractors (extractors that output the entropy of the source *plus* the entropy of the seed) are equivalent to a form of vertex expansion. Such extractors are a special case of (seedless) two-source min-entropy extractors where one of the sources is very small and of full entropy. Thus, as a generalization of the dimension expanders we have already defined, we can thus define the notion of a *(seedless) two-source rank condenser.*

While it is often most natural to consider the two sources to be of equal dimension, to highlight the connection to dimension expanders we consider sources with unbalanced dimension.

▶ **Definition 15.** Let $\mathbb{F}$ be a field and $n \geqslant r \geqslant 1$ and $m \geqslant s \geqslant 1$. A function $f : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^t$ is a **(seedless)** (r, s, ϵ)**-two-source rank condenser** if for all sets $A \subseteq \mathbb{F}^n$ and $B \subseteq \mathbb{F}^m$ with rank $A = r$ and rank $B = s$,

$$\operatorname{rank} f(A \times B) = \operatorname{rank}\{f(\overline{v}, \overline{w})\}_{\overline{v}\in A, \overline{w}\in B} \geqslant (1-\epsilon) \operatorname{rank} A \cdot \operatorname{rank} B\ .$$

The function f is a $(\leqslant r, s, \epsilon)$-condenser if it is a (r', s, ϵ)-condenser for all $1 \leqslant r' \leqslant r$, and $(\leqslant r, \leqslant s, \epsilon)$-condensers are defined similarly. If $\epsilon = 0$ we say the rank condenser is **lossless** and it is otherwise **lossy**. The function f is **bilinear** if $f(\overline{v}, \overline{w}) = (\overline{v}^{\text{tr}} E_i \overline{w})_{i=1}^t$ for $E_i \in \mathbb{F}^{n\times m}$. The function f is **explicit** if it can be evaluated in $\mathsf{poly}(n, m, t)$ steps.

While this definition is naturally motivated as a generalization of dimension expanders, we originally were motivated to study these objects due to potential applications for constructing *subspace evasive sets*, as we describe in the full version.

Note that in general we allow the function f to be arbitrary, but in this work we will restrict ourselves to bilinear functions f as they are the most natural. In this case, as discussed after Theorem 4, we see that the function f acts on *subspaces* so that we ask that for subspaces $V \subseteq \mathbb{F}^n$ and $W \subseteq \mathbb{F}^m$ that $\dim f(V, W) \geqslant (1-\epsilon) \dim V \cdot \dim W$. In this way, f can be thought of as a *derandomized tensor product.*

We now quote the parameters as given by the probabilistic method.

▶ **Proposition 16.** *Let $\mathbb{F}_q$ be a finite field. Let $n \geqslant r \geqslant 1$ and $m \geqslant s \geqslant 1$ and $\epsilon \geqslant 0$. Then there exists a function $f : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^t$ which is a bilinear (r, s, ϵ)-two-source rank condenser, assuming that*

$$t \geqslant \frac{n}{\epsilon s} + \frac{m}{\epsilon r} + (1-\epsilon) rs + o_q(1)\ .$$

for $\epsilon > 0$. Further, there exists an f which is a $(\leqslant r, s, \epsilon)$-condenser assuming that

$$t \geqslant \frac{n}{\epsilon s} + \frac{m}{\epsilon} + (1-\epsilon) rs + o_q(1)\ .$$

If $\epsilon = 0$, then there exists an f which is a $(r, s, 0)$-condenser assuming that

$$t \geqslant rn + sm + rs + o_q(1)\ .$$

In particular, in the balanced case of $n = m$ and $r = s$ this shows that any $t \geqslant \frac{2n}{\epsilon r} + (1-\epsilon)r^2 + o_q(1)$ suffices. Note that unlike the single-source setting, there is a large penalty for condensing all small enough sources. Thus, the above gives $(r, r, 1/2)$-condensers with output $\approx \frac{n}{r} + r^2$ but to obtain a $(\leqslant r, r, 1/2)$-condenser the resulting output size is $\approx n + r^2$ (and the full version shows that a linear dependence on n is needed in this case).

Note that in our definitions of seeded rank condensers there was no analogue of *strong* min-entropy extractors, which are extractors that also recover the entropy of the seed in addition to the entropy of the source. That is, in our setting, there is no "rank of the seed" to recover as the seed is simply an index into the collection $\mathcal{E}$. The notion of a two-source rank condenser in some sense allows the second source to be a "seed" in that we can associate elements of $\mathcal{E}$ with elements in a basis for $\mathbb{F}^m$. However, we do not pursue this analogy further as two-source rank condensers meeting the probabilistic method do not seem to yield good lossy rank condensers in all regimes as two-source condensers can require an output size which is linear in the input size.

However, the connection between two-source extractors and expanders does hold tightly for the notion of rank, as we show. Note that for this connection it suffices to have condensers that work when one of the two sources has full rank.

▶ **Proposition 17.** *Let $\mathbb{F}_q$ be a finite field. For large n and all other parameters constant, constructions of bilinear $(\leqslant \delta n, m, \epsilon)$-two-source rank condensers $f : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^t$ that meet the probabilistic method bound yield constructions of (δ, α)-dimension expanders in $\mathbb{F}^n$ meeting the probabilistic method bound.*

We also give constructions of two-source condensers using seeded rank condensers. That is, for two sources we use a seeded rank condenser to condense each source and use a union bound to show that the seed-length only doubles. We then enumerate over seeds and for each seed we then tensor the two condensed sources together. While this approach seems wasteful, we show that it yields *optimal* lossless two-source rank condensers by appropriate pruning. In particular, we observe that this is the same construction as given by Forbes and Shpilka [9] for an object known as a *rank-metric code*. We push this observation further to see that bilinear lossless two-source rank condensers are *equivalent* to rank-metric codes. Using this connection, we obtain optimal such condensers over *any* field using known constructions of rank-metric codes.

▶ **Theorem 18.** *Let $\mathbb{F}$ be a field and $n \geqslant r \geqslant 1$ and $m \geqslant s \geqslant 1$. Then there is an explicit bilinear $f : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^t$ which is a $(r, s, 0)$-two-source rank condenser with $t \leqslant O(\min\{r, s\} \cdot (n + m))$.*

We then turn to constructions of *lossy* two-source condensers, where our results are considerably weaker. However, we are able to give near-optimal results for *constant* r by using a brute force "inner condenser" and using our condense-then-tensor results as an "outer condenser".

▶ **Proposition 19.** *Let $\mathbb{F}$ be a field and $n \geqslant r \geqslant 1$, where $|\mathbb{F}| \geqslant \mathsf{poly}(n)$ and $r \leqslant O(1)$. Then there is an explicit bilinear $(r, r, 1 - (1 - \epsilon)^3)$-two source rank condenser $f : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}^t$ with $t \leqslant O(n/\epsilon^2 r)$.*

5 Open Questions

This work leaves several directions for future work.

1. Can one obtain (r, ϵ)-lossy seeded rank *extractors*, where the output is $\approx (1 - \epsilon)r$? Our methods require the output to be $\geqslant r$.
2. Can one develop of theory of "code concatenation" to improve our results for small fields?
3. Can one obtain lossy two-source rank condensers with output size $o(nr)$ for $r = \omega(1)$?
4. Can one obtain *lossless* dimension expanders, where the degree/expansion relationship matches the probabilistic method?
5. What is the complexity of computing dimension expansion? That is, given matrices $A_1, \ldots, A_d \in \mathbb{F}^{n\times n}$, compute the largest α so that $\mathcal{A} := \{A_i\}_{i=1}^d$ is a $(1/2, \alpha)$-dimension expander.

Acknowledgments. We would like to thank Swastik Kopparty, Prasad Raghavendra, Amir Shpilka, Amir Yehudayoff, Avi Wigderson, and anonymous reviewers for helpful comments.

References

1 Manindra Agrawal, Chandan Saha, and Nitin Saxena. Quasi-polynomial hitting-set for set-depth-Δ formulas. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC 2013)*, pages 321–330, 2013. Full version at arXiv:1209.2333.

2 Boaz Barak, Russell Impagliazzo, Amir Shpilka, and Avi Wigderson. Personal Communication to Dvir-Shpilka [?], 2004.

3 Boaz Barak, Guy Kindler, Ronen Shaltiel, Benny Sudakov, and Avi Wigderson. Simulating independence: New constructions of condensers, Ramsey graphs, dispersers, and extractors. *J. ACM*, 57(4), 2010. Preliminary version in the *37th Annual ACM Symposium on Theory of Computing (STOC 2005)*.

4 Jean Bourgain and Amir Yehudayoff. Expansion in $\mathrm{SL}_2(\mathbb{R})$ and monotone expanders. *Geometric and Functional Analysis*, 23(1):1–41, 2013. Preliminary version in the *44th Annual ACM Symposium on Theory of Computing (STOC 2012)*. This work is the full version of [?].

5 Michael R. Capalbo, Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002)*, pages 659–668, 2002.

6 Ho Yee Cheung, Tsz Chiu Kwok, and Lap Chi Lau. Fast matrix rank algorithms and applications. *J. ACM*, 60(5):31, 2013. Preliminary version in the *44th Annual ACM Symposium on Theory of Computing (STOC 2012)*.

7 Zeev Dvir and Amir Shpilka. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM J. Comput.*, 36(5):1404–1434, 2007. Preliminary version in the *37th Annual ACM Symposium on Theory of Computing (STOC 2005)*.

8 Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 867–875, 2014. Full version at arXiv:1309.5668.

9 Michael A. Forbes and Amir Shpilka. On identity testing of tensors, low-rank recovery and compressed sensing. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC 2012)*, pages 163–172, 2012. Full version at arXiv:1111.0663.

10 Ariel Gabizon and Ran Raz. Deterministic extractors for affine sources over large fields. *Combinatorica*, 28(4):415–440, 2008. Preliminary version in the *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005)*.

11 Venkatesan Guruswami. Linear-algebraic list decoding of folded reed-solomon codes. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity (CCC 2011)*, pages 77–85, 2011. The full version of this paper is merged into Guruswami-Wang [13].

12 Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. *Combinatorica*, pages 1–25, 2014. Preliminary version in the *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*.

13 Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of reed-solomon codes. *IEEE Transactions on Information Theory*, 59(6):3257–3268, 2013. Preliminary versions appeared in Proceedings of the 26th Annual IEEE Conference on Computational Complexity (CCC 2011) and Proceedings of the 15th International Workshop on Randomization and Computation (RANDOM 2011).

14 Venkatesan Guruswami and Carol Wang. Evading subspaces over large fields and explicit list-decodable rank-metric codes. In *Proceedings of the 18th International Workshop on Randomization and Computation (RANDOM 2014)*, pages 748–761, 2014. Full version at arXiv:1311.7084.

15 Venkatesan Guruswami and Chaoping Xing. Folded codes from function field towers and improved optimal rate list decoding. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC 2012)*, pages 339–350, 2012. Full version at arXiv:1204.4209.

16 Venkatesan Guruswami and Chaoping Xing. List decoding Reed-Solomon, algebraic-geometric, and Gabidulin subcodes up to the Singleton bound. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC 2013)*, pages 843–852, 2013. Full version in the Electronic Colloquium on Computational Complexity (ECCC), Technical Report TR12-146.

17 Aram W. Harrow. Quantum expanders from any classical cayley graph expander. *Quantum Information & Computation*, 8(8–9):715–721, 2008.

18 Zohar S. Karnin and Amir Shpilka. Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. *Combinatorica*, 31(3):333–364, 2011. Preliminary version in the *23rd Annual IEEE Conference on Computational Complexity (CCC 2008)*.

19 Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. *arXiv*, 1404.4506, 2014.

20 Alexander Lubotzky and Efim Zelmanov. Dimension expanders. *J. Algebra*, 319(2):730–738, 2008.

21 Dániel Marx. A parameterized view on matroid optimization problems. *Theor. Comput. Sci.*, 410(44):4471–4479, 2009. Preliminary version in the *33rd International Colloquium on Automata, Languages and Programming (ICALP 2006)*.

22 Pavel Pudlák and Vojtěch Rödl. Pseudorandom sets and explicit constructions of Ramsey graphs. In *Complexity of computations and proofs*, volume 13 of *Quad. Mat.*, pages 327–346. Dept. Math., Seconda Univ. Napoli, Caserta, 2004.

23 Ran Raz. Extractors with weak random seeds. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC 2005)*, pages 11–20, 2005. Full version in the Electronic Colloquium on Computational Complexity (ECCC), Technical Report TR04-099.

24 Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Comput. Complex.*, 14(1):1–19, April 2005. Preliminary version in the *19th Annual IEEE Conference on Computational Complexity (CCC 2004)*.

25 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):2070–388, 2010.

26 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.

Swendsen-Wang Algorithm on the Mean-Field Potts Model[*]

Andreas Galanis[†1], Daniel Štefankovič[‡2], and Eric Vigoda[§3]

1 **University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK**
`andreas.galanis@cs.ox.ac.uk`

2 **University of Rochester, Rochester, NY, 14627, USA**
`stefanko@cs.rochester.edu`

3 **Georgia Institute of Technology, Atlanta, GA, 30332, USA**
`vigoda@cc.gatech.edu`

Abstract

We study the q-state ferromagnetic Potts model on the n-vertex complete graph known as the mean-field (Curie-Weiss) model. We analyze the Swendsen-Wang algorithm which is a Markov chain that utilizes the random cluster representation for the ferromagnetic Potts model to recolor large sets of vertices in one step and potentially overcomes obstacles that inhibit single-site Glauber dynamics. The case $q = 2$ (the Swendsen-Wang algorithm for the ferromagnetic Ising model) undergoes a slow-down at the uniqueness/non-uniqueness critical temperature for the infinite Δ-regular tree ([16]) but yet still has polynomial mixing time at all (inverse) temperatures $\beta > 0$ ([7]). In contrast for $q \geq 3$ there are two critical temperatures $0 < \beta_{\mathrm{u}} < \beta_{\mathrm{rc}}$ that are relevant, these two critical points relate to phase transitions in the infinite tree. We prove that the mixing time of the Swendsen-Wang algorithm for the ferromagnetic Potts model on the n-vertex complete graph satisfies: (i) $O(\log n)$ for $\beta < \beta_{\mathrm{u}}$, (ii) $O(n^{1/3})$ for $\beta = \beta_{\mathrm{u}}$, (iii) $\exp(n^{\Omega(1)})$ for $\beta_{\mathrm{u}} < \beta < \beta_{\mathrm{rc}}$, and (iv) $O(\log n)$ for $\beta \geq \beta_{\mathrm{rc}}$. These results complement refined results of Cuff et al. [10] on the mixing time of the Glauber dynamics for the ferromagnetic Potts model. The most interesting aspect of our analysis is at the critical temperature $\beta = \beta_{\mathrm{u}}$, which requires a delicate choice of a potential function to balance the conflating factors for the slow drift away from a fixed point (which is repulsive but not Jacobian repulsive): close to the fixed point the variance from the percolation step dominates and sufficiently far from the fixed point the dynamics of the size of the dominant color class takes over.

1998 ACM Subject Classification G.3 Probability and Statistics

Keywords and phrases Ferromagnetic Potts model, Swendsen-Wang dynamics, mixing time, mean-field analysis, phase transition

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.815

1 Introduction

The mixing time of Markov chains is of critical importance for simulations of statistical physics models. It is especially interesting to understand how phase transitions in these

[*] Full version available from the arXiv at http://arxiv.org/abs/1502.06593.
[†] The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 334828. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.
[‡] Research supported in part by NSF grant CCF-1318374.
[§] Research supported in part by NSF grant CCF-1217458.

models manifest in the behavior of the mixing time; these connections are the topic of this paper.

We study the q-state ferromagnetic Potts model. In the following definition the case $q = 2$ corresponds to the Ising model and $q \geq 3$ is the Potts model. For a graph $G = (V, E)$ the configurations of the model are assignments $\sigma : V \to [q]$ of spins to vertices, and let Ω denote the set of all configurations. The model is parameterized by $\beta > 0$, known as the (inverse) temperature. For a configuration $\sigma \in \Omega$ let $m(\sigma)$ be the number of edges in E that are monochromatic under σ and let its weight be $w(\sigma) = \exp(\beta m(\sigma))$. Then the Gibbs distribution μ is defined as follows, for $\sigma \in \Omega$, $\mu(\sigma) = w(\sigma)/Z(\beta)$, where $Z(\beta) = \sum_{\sigma\in\Omega} w(\sigma)$ is the normalizing constant, known as the partition function.

A useful feature for studying the ferromagnetic Potts model is its alternative formulation known as the random-cluster model. Here configurations are subsets of edges and the weight of such a configuration $S \subseteq E$ is

$$w(S) = p^{|S|}(1-p)^{|E\setminus S|}q^{k(S)},$$

where $p = 1 - \exp(-\beta)$ and $k(S)$ is the number of connected components in the graph $G' = (V, S)$ (isolated vertices do count). The corresponding partition function $Z_{\mathrm{rc}} = \sum_{S\subseteq E} w(S)$ satisfies $Z_{\mathrm{rc}} = (1-p)^{|E|}Z$.

The focus of this paper is the random-cluster (Curie-Weiss) model which in computer science terminology is the n-vertex complete graph $G = (V, E)$. The interest in this model is that it allows more detailed results and these results are believed to extend to other graphs of particular interest such as random regular graphs. For convenience we parameterize the model in terms of a constant $B > 0$ such that the Gibbs distribution is as follows:

$$\mu(\sigma) = \frac{1}{Z(\beta)}(1 - B/n)^{-m(\sigma)}. \tag{1}$$

(Note that $\beta = -\ln(1 - B/n) \sim B/n$ for large n.) The following critical points $\mathfrak{B}_u < \mathfrak{B}_o < \mathfrak{B}_{rc}$ for the parameter B are well-studied [1] and relevant to our study of the Potts model on the complete graph:

$$\mathfrak{B}_u = \sup\left\{B \geq 0 \,\middle|\, \frac{B - z}{B + (q-1)z} \neq e^{-z} \text{ for all } z > 0\right\} = \min_{z\geq 0}\left\{z + \frac{qz}{e^z - 1}\right\}, \tag{2}$$

$$\mathfrak{B}_o = \frac{2(q-1)\ln(q-1)}{q-2}, \qquad \mathfrak{B}_{rc} = q. \tag{3}$$

These thresholds correspond to the critical points for the infinite Δ-regular tree $\mathbb{T}_\Delta$ and random Δ-regular graphs by taking appropriate limits as $\Delta \to \infty$. (More specifically, if $B(\Delta)$ is a threshold on $\mathbb{T}_\Delta$ or the random Δ-regular graph then $\lim_{\Delta\to\infty} \Delta(B(\Delta) - 1)$ is the corresponding threshold in the Curie-Weiss model.) In this perspective, $\mathfrak{B}_u$ corresponds to the uniqueness/non-uniqueness threshold on $\mathbb{T}_\Delta$; $\mathfrak{B}_o$ corresponds to the ordered/disordered phase transition; and $\mathfrak{B}_{rc}$ was conjectured by Häggström to correspond to a second uniqueness/non-uniqueness threshold for the random-cluster model on $\mathbb{T}_\Delta$ with periodic boundaries (in particular, he conjectured that non-uniqueness holds iff $B \in (\mathfrak{B}_u, \mathfrak{B}_{rc})$). For a detailed exposition of these critical points we refer the reader to [10] (see also [11] for their relevance for random regular graphs).

[1] $\mathfrak{B}_o$ is β_c in [9, Equation (3.1)] and $\mathfrak{B}_u$ is equivalent to β_s in [10, Equation (1.1)] under the parametrization $z = B(qx - 1)/(q - 1)$. We follow the convention of counting monochromatic edges [9] as opposed to counting monochromatic pairs of vertices [10]; hence our thresholds are larger than those in [10] by a factor of 2.

The Glauber dynamics is a classical tool for studying the Gibbs distribution. These are the class of Markov chains whose transitions update the configuration at a randomly chosen vertex and are designed so that its stationary distribution is the Gibbs distribution. The limitation of local Markov chains, such as the Glauber dynamics, is that they are typically slow to converge at low temperatures (large B). The Swendsen-Wang algorithm is a more sophisticated Markov chain that utilizes the random cluster representation of the Potts model to potentially overcome bottlenecks that obstruct the simpler Glauber dynamics. It is formally defined as follows.

The Swendsen-Wang algorithm is a Markov chain (X_t) whose transitions $X_t \to X_{t+1}$ are as follows. From a configuration $X_t \in \Omega$:

- Let M be the set of monochromatic edges in X_t.
- For each edge $e \in M$, delete it with probability $1 - B/n$. Let M' denote the set of monochromatic edges that were not deleted.
- In the graph (V, M'), independently for each connected component choose a color uniformly at random from $[q]$ and assign all vertices in that component the chosen color. Let X_{t+1} denote the resulting spin configuration.

Recall, the mixing time T_{mix} of an ergodic Markov chain is defined as the number of steps from the worst initial state to get within total variation distance $\leq 1/4$ of its unique stationary distribution. For the Swendsen-Wang algorithm for the ferromagnetic Ising model on the complete graph, Cooper et al. [7] showed that $T_{\text{mix}} = O(\sqrt{n})$ for all temperatures. Long et al. [16] showed more refined results establishing that the mixing time is $\Theta(1)$ for $\beta < \beta_c$, $\Theta(n^{1/4})$ for $\beta = \beta_c$, and $\Theta(\log n)$ for $\beta > \beta_c$ where β_c is the uniqueness/non-uniqueness threshold.

For the Swendsen-Wang algorithm for the ferromagnetic Potts model, it was shown that the mixing time is exponentially large in $n = |V|$ at the critical point $B = \mathfrak{B}_o$ by Gore and Jerrum [13] for the complete graph, Cooper and Frieze [8] for $G(n,p)$ for $p = \Omega(n^{-1/3})$, Galanis et al. [11] for random regular graphs, and Borgs et al. [4, 5] for the d-dimensional integer lattice for $q \geq 25$ at the analogous critical point. For the Glauber dynamics for the ferromagnetic Potts model on the complete graph, Cuff et al. [10] showed that the mixing time satisfies (their results are significantly more precise than what we state here for convenience): $\Theta(n \log n)$ for $B < \mathfrak{B}_u$, exponentially slow mixing for $B > \mathfrak{B}_u$, and $\Theta(n^{4/3})$ mixing time for $B = \mathfrak{B}_u$ (and a scaling window of $O(n^{-2/3})$ around $\mathfrak{B}_u$).

We can now state our main result which is a complete classification of the mixing time of the Swendsen-Wang dynamics when the parameter B is a constant independent of n.

▶ **Theorem 1.** *For all $q \geq 3$, the mixing time T_{mix} of the Swendsen-Wang algorithm on the n-vertex complete graph satisfies:*

1. *For all $B < \mathfrak{B}_u$, $T_{\text{mix}} = O(\log n)$.*
2. *For $B = \mathfrak{B}_u$, $T_{\text{mix}} = O(n^{1/3})$.*
3. *For all $\mathfrak{B}_u < B < \mathfrak{B}_{rc}$, $T_{\text{mix}} = \exp(n^{\Omega(1)})$.*
4. *For all $B \geq \mathfrak{B}_{rc}$, $T_{\text{mix}} = O(\log n)$.*

In an independent work, Blanca and Sinclair [2] analyze a closely related chain to the Swendsen-Wang dynamics which is also suitable for sampling random cluster configurations. They provide an analogue of Theorem 1, though their analysis excludes the critical points $B = \mathfrak{B}_u$ and $B = \mathfrak{B}_{rc}$.

In the following section, we discuss the critical points $\mathfrak{B}_u, \mathfrak{B}_o, \mathfrak{B}_{rc}$, present a function F which captures a simplified view of the Swendsen-Wang dynamics, and then we present a lemma connecting the behavior of F with the critical points. We also present in Section 2 a high-level sketch of the proof of Theorem 1. In Section 3 we prove the slow mixing result (Part 3 of Theorem 1). We then prove the rapid mixing results for $B > \mathfrak{B}_{rc}$ in Section 4 and for $B = \mathfrak{B}_u$ in Section 5. The cases $B = \mathfrak{B}_{rc}$ and $B < \mathfrak{B}_u$ are given in Sections C and D, respectively, of the full version [12].

2 Proof Approach

2.1 Critical Points for Phase Transitions

We review the thresholds $\mathfrak{B}_u, \mathfrak{B}_o, \mathfrak{B}_{rc}$ for the mean-field Potts model, the reader is referred to [3] for further details which also apply to the random-cluster model. The thresholds $\mathfrak{B}_u, \mathfrak{B}_o, \mathfrak{B}_{rc}$ are related to the critical points of the following function of the partition function. We first need to introduce some notation. For a configuration $\sigma : V \to [q]$ and a color $i \in [q]$, let $\alpha_i(\sigma)$ be the fraction of vertices with color i in σ, i.e., $\alpha_i(\sigma) = |\{v \in V : \sigma(v) = i\}|/n$. We also denote by $\boldsymbol{\alpha}(\sigma)$ the vector $(\alpha_1(\sigma), \dots, \alpha_q(\sigma))$, and refer to it as the *phase* of σ.

For a q-dimensional probability vector $\boldsymbol{\alpha}$, let $\Omega^{\boldsymbol{\alpha}}$ be the set of configurations σ whose phase is $\boldsymbol{\alpha}$. Let

$$Z^{\boldsymbol{\alpha}} = \sum_{\sigma \in \Omega^{\boldsymbol{\alpha}}} w(\sigma) \text{ and } \Psi(\boldsymbol{\alpha}) := \lim_{n \to \infty} \frac{1}{n} \ln Z^{\boldsymbol{\alpha}}.$$

There are two relevant phases: the uniform phase $\mathbf{u} := (1/q, \dots, 1/q)$ and the majority phase $\mathbf{m} := (a, b, \dots, b)$ and its q permutations. For the majority phase, a, b are such that $a + (q-1)b = 1$ and $a > 1/q$ is a local maximum of

$$\Psi_1(a) := \Psi\Big(a, b, \dots, b\Big) = -a \ln a - (1-a) \ln \frac{1-a}{q-1} + \frac{B}{2}\left(a^2 + \frac{(1-a)^2}{q-1}\right) \tag{4}$$

and hence satisfies

$$\ln \frac{(q-1)a}{1-a} = B(a - (1-a)/(q-1)). \tag{5}$$

The thresholds $\mathfrak{B}_u, \mathfrak{B}_o, \mathfrak{B}_{rc}$ relate to the critical points of Ψ, see Figure 1 for an illustration of the following. For $B \leq \mathfrak{B}_u$ the uniform phase is the unique local maximum of Ψ. For $\mathfrak{B}_u < B < \mathfrak{B}_{rc}$ there are $q+1$ local maxima: the uniform phase and the q majority phases, and at $B = \mathfrak{B}_o$ they are all global maxima. Finally, for $B \geq \mathfrak{B}_{rc}$ the q majority phases are the only local maxima.

2.2 Connections to Simplified Swendsen-Wang

The following function from $[1/q, 1]$ to $[0, 1]$ will capture the behavior of the Swendsen-Wang algorithm.

$$F(z) := \frac{1}{q} + \left(1 - \frac{1}{q}\right) zx, \tag{6}$$

where $x = 0$ for $z \leq 1/B$ and for $z > 1/B$, $x \in (0, 1]$ is the unique solution of

$$x + \exp(-zBx) = 1. \tag{7}$$

The function F captures the size of the largest color class when there is a single heavy color where heavy means that the color class is supercritical in the percolation step of the

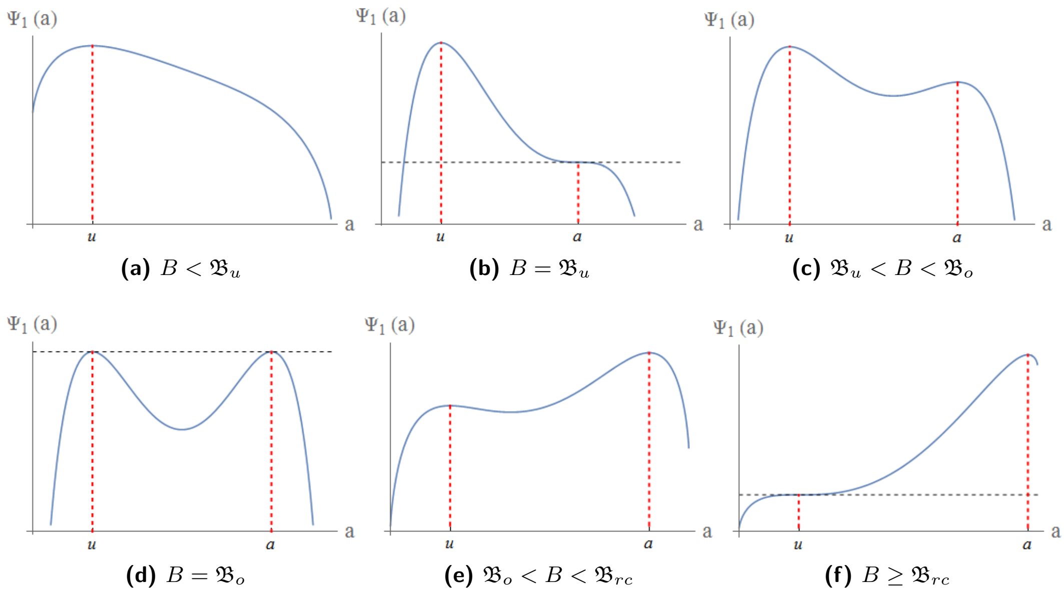


Figure 1 The function Ψ_1 (free energy) plotted in different regimes of B (defined in (4)). The critical points $\mathfrak{B}_u, \mathfrak{B}_o, \mathfrak{B}_{rc}$ are given by (2) and (3). In the regime $B < \mathfrak{B}_u$ (figure 1a), the function Ψ_1 has a unique local maximum at the disordered phase. At $B = \mathfrak{B}_u$ (figure 1b), the function Ψ_1 has a saddle point at the ordered phase. In the regime $\mathfrak{B}_u < B < \mathfrak{B}_{rc}$ (figures 1c, 1d and 1e) the function Ψ_1 has two local maxima; these are both global maxima iff $B = \mathfrak{B}_o$. In the regime $B \geq \mathfrak{B}_{rc}$ (figure 1f), the function Ψ_1 has a unique local maximum at the ordered phase and a saddle point at the disordered phase.

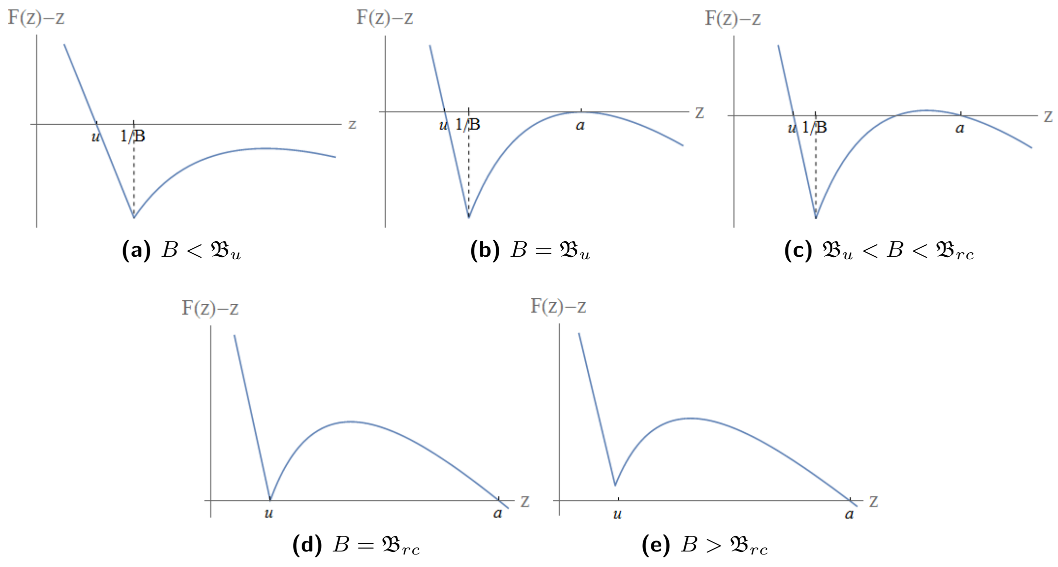
Swendsen-Wang process. Hence after the percolation step this heavy color will have a giant component and the other color classes will all be broken into small components. So say initially the one heavy color has size zn for $1/B < z < 1$ and let's consider its size after one step of the Swendsen-Wang dynamics. After the percolation step, this heavy color will have a giant component of size roughly xzn (where x is as in (7)) and all other components will be of size $O(\log n)$. Then a $1/q$ fraction of the small components will be recolored the same as the giant component, and hence the size of the largest color class will be (roughly) $nF(z)$ after this one step of the Swendsen-Wang dynamics.

Our next goal is to tie together the functions F and Ψ_1 so that we can relate the behavior of the Swendsen-Wang dynamics with the underlying phase transitions of the model. We first need some terminology. A critical point a of a function $f : \mathbb{R} \to \mathbb{R}$ is a *hessian maximum* if the second derivative of f at a is negative (this is a sufficient condition for a to be a local maximum). A fixpoint a of a function $F : \mathbb{R} \to \mathbb{R}$ is a *jacobian attractive fixpoint* if $|F'(a)| < 1$ (this is a sufficient condition for a to be an attractive fixpoint).

▶ **Lemma 2.** *The critical points of Ψ_1 correspond to fixpoints of F. The hessian maxima of Ψ_1 correspond to jacobian attractive fixpoints of F.*

Lemma 2 is proved in Section E of the full version [12].

The behavior of F is the basic tool for proving Theorem 1. Recall the earlier discussion of the uniform vector $\mathbf{u} := (1/q, \ldots, 1/q)$ and the q permutations of the majority phase $\mathbf{m} := (a, b, \ldots, b)$. The following lemma (proved in Section F of the full version [12]) provides some basic intuition about the proof of Theorem 1, see Figure 2 for a depiction of the various regimes.



(a) $B < \mathfrak{B}_u$ **(b)** $B = \mathfrak{B}_u$ **(c)** $\mathfrak{B}_u < B < \mathfrak{B}_{rc}$ **(d)** $B = \mathfrak{B}_{rc}$ **(e)** $B > \mathfrak{B}_{rc}$

Figure 2 The drift function $F(z) - z$, where F is defined by (6), (7). The critical points $\mathfrak{B}_u, \mathfrak{B}_o, \mathfrak{B}_{rc}$ are given by (2) and (3). In the regime $B < \mathfrak{B}_u$ (figure 2a), the function F has a unique attractive fixpoint at the disordered phase. At $B = \mathfrak{B}_u$ (figure 2b), F also has a (non-jacobian) *repulsive* fixpoint at the ordered phase. In the regime $\mathfrak{B}_u < B < \mathfrak{B}_{rc}$ (figures 2c), F has attractive fixpoints at the ordered and disordered phases. At $B = \mathfrak{B}_{rc}$ (figure 2d), the disordered phase is no longer attactive; it is jacobian repulsive. Finally, in the regime $B > \mathfrak{B}_{rc}$ (figure 2e), the function F has a unique attractive fixpoint at the ordered phase.

▶ **Lemma 3.** *For the function F,*

1. *For $B < \mathfrak{B}_u$, $u = 1/q$ is the unique fixpoint and it is jacobian attractive.*
2. *For $B = \mathfrak{B}_u$, there are 2 fixpoints: u and a where a is defined as in the majority phase* $\mathbf{m}$. *Of these, only u is (jacobian) attractive. The fixpoint a is repulsive but not jacobian repulsive.*
3. *For $\mathfrak{B}_u < B < \mathfrak{B}_{rc}$ there are 2 attractive fixpoints: u and a where a is defined as in the majority phase* $\mathbf{m}$. *Both of these are jacobian attractive.*
4. *For $B = \mathfrak{B}_{rc}$, both a and u are fixpoints. The fixpoint u is (jacobian) repulsive, while the fixpoint a is jacobian attractive.*
5. *For $B > \mathfrak{B}_{rc}$, a is the only fixpoint and it is jacobian attractive.*

The reason that u abruptly changes from a jacobian attractive fixpoint ($B < \mathfrak{B}_{rc}$) to a jacobian repulsive fixpoint ($B = \mathfrak{B}_{rc}$) stems from the fact that in the regime $B < \mathfrak{B}_{rc}$, F is constant in a small neighborhood around $1/q$ (precisely, in the interval $[1/q, 1/B]$), which is no longer the case for $B = \mathfrak{B}_{rc}$.

2.3 Proof Sketches

We explain the high-level proof approach for the various parts of Theorem 1 before presenting the detailed proofs in subsequent sections.

Slow mixing: For Part 3 of Theorem 1, the main idea is that the function F has 2 attractive fixpoints (see Lemma 3). At least one of the corresponding phases, $\mathbf{u}$ or $\mathbf{m}$, is a global maximum for Ψ. Consider the other phase, say it is $\mathbf{u}$ for concreteness. Consider the local ball around $\mathbf{u}$, these are configurations that are close in ℓ_∞ distance from $\mathbf{u}$. The key is that since $\mathbf{u}$ is an attractive fixpoint for F, if the initial state is in this local ball then with very high probability after one step of the Swendsen-Wang dynamics it will still be in the local ball (see Lemma 4, and Lemma 5 for the analogous lemma for $\mathbf{m}$). The result then follows since one needs to sample from the local ball around the phase which corresponds to the global maximum of Ψ to get close to the stationary distribution.

Fast mixing for $B > \mathfrak{B}_{rc}$: For a configuration σ and spin i, say the color class is heavy if the number of vertices with spin i is $> n/B$ and light if it is $< n/B$. If a color class is heavy then it is super-critical for the percolation step of Swendsen-Wang and hence there will be a giant component. The key is that for any initial state X_0, then with constant probability the largest components from all of the colors will choose the same new color and consequently there will be only one heavy color class and the other $q-1$ colors will be light. Hence we can assume there is one heavy color class and $q-1$ light color classes, and then the function F suitably describes the size of the largest color class during the evolution of the Swendsen-Wang dynamics. Since the only local maximum for F corresponds to the majority phase $\mathbf{m}$, after $O(\log n)$ steps we'll be close to $\mathbf{m}$ – the difference will be due to the stochastic nature of the process. Then it is straightforward to define a coupling for two chains (X_t, Y_t) whose initial states X_0, Y_0 are close to $\mathbf{m}$ so that after $T = O(\log n)$ steps we have that $X_T = Y_T$.

Fast mixing for $B = \mathfrak{B}_{rc}$: The basic outline is similar to the $B > \mathfrak{B}_{rc}$ case except here the argument is more intricate when the heaviest color lies in the scaling window (for the onset of a giant component). We need a more involved argument that we get away from initial configurations that are close to the uniform phase; informally, the uniform fixpoint is jacobian repulsive, so an initial displacement increases geometrically by a constant factor.

Fast mixing for $B < \mathfrak{B}_u$: Here the argument is similar to the $B > \mathfrak{B}_{rc}$ case, in fact it is easier. The critical point for a giant component in the percolation step is density $1/B$. In this case we have that $B < \mathfrak{B}_u$ and since $\mathfrak{B}_u < \mathfrak{B}_{rc} = q$ we have that in the uniform phase (which is the only local maxima) the color classes are all subcritical. Hence once we are close to the uniform phase all of the components after the percolation step will be of size $O(\log n)$. So the basic argument is similar to the $B > \mathfrak{B}_{rc}$ case in how we approach the local maxima, which is the uniform phase in this case. Then once we reach density $< 1/B$ then in the next step the configuration will be close to the uniform phase in the next step and then it is straightforward to couple two such configurations.

Fast mixing for $B = \mathfrak{B}_u$: This is the most difficult part. As in the $B > \mathfrak{B}_{rc}$ case with constant probability there will be at most one heavy color class after one step. We then track the evolution of the size of the heavy color class. The difficulty arises because the size of the component does not decrease in expectation at the majority fixpoint. However variance moves the size of the component into a region where the size of the component decreases in expectation. The formal argument uses a carefully engineered potential function that decreases because of the variance (the function is concave around the fixpoint) and expectation (the function is increasing) of the size of the largest color class, see Section 5.

3 Slow Mixing for $\mathfrak{B}_u < B < \mathfrak{B}_{rc}$

Let $\mathcal{B}(\mathbf{v},\delta)$ be the ℓ_∞-ball of configuration vectors of the q-state Potts model in K_n around $\mathbf{v}$ of radius δ, that is,

$$\mathcal{B}(\mathbf{v},\delta) = \{\mathbf{w}\in\mathbb{Z}^q \mid \|\mathbf{w}/n-\mathbf{v}\|_\infty \le \delta\}.$$

We will show that for $B < \mathfrak{B}_{rc}$ the Swendsen-Wang algorithm is exponentially unlikely to leave the vicinity of the uniform configuration.

▶ **Lemma 4.** *Assume $B < \mathfrak{B}_{rc}$. There exists $\varepsilon_0 > 0$ such that for all $\varepsilon \in (0,\varepsilon_0)$ for $S = \mathcal{B}(\mathbf{u},\varepsilon)$*

$$P_{\mathrm{SW}}(S,S) \ge 1-\exp(-\Theta(n^{1/2})).$$

The reason for Lemma 4 failing for $B > \mathfrak{B}_{rc}$ is that the first step of the Swendsen-Wang algorithm on a cluster of size n/q yields linear sized connected components, and these allow the algorithm to escape the neighborhood of $\mathbf{u}$.

We also analyze the behavior of the algorithm around the majority configuration (for the configuration to exist we need $B \ge \mathfrak{B}_u$).

▶ **Lemma 5.** *Assume $B > \mathfrak{B}_u$ and let $\mathbf{m} = (a,b,\dots,b)$ where $a > 1/q$ is the attractive fixpoint of F of Lemma 3. There exists $\varepsilon_0 > 0$ such that for all $\varepsilon \in (0,\varepsilon_0)$ for $S = \mathcal{B}(\mathbf{m},\varepsilon)$ we have*

$$P_{\mathrm{SW}}(S,S) \ge 1-\exp(-\Theta(n^{1/3})).$$

Combining Lemmas 4 and 5 we obtain Part 3 of Theorem 1.

▶ **Corollary 6.** *For $B \in (\mathfrak{B}_u,\mathfrak{B}_{rc})$, the mixing time of the Swendsen-Wang algorithm on the complete graph on n vertices is $\exp(\Omega(n^{1/3}))$.*

We prove here Lemma 5, the (very similar) proof of Lemma 4 is given in Section G of the full version [12].

We will need several known results on the $G(n,p)$ model in the supercritical regime ($p = c/n$, where $c > 1$). The size of the giant component is asymptotically normal [19]. We will use the following moderate deviation inequalities for the sizes of the largest and second largest components of G.

▶ **Lemma 7.** *Let $G \sim G(n,c/n)$ where $c > 1$. Let $\beta \in (0,1)$ be the solution of $x+\exp(-cx) = 1$. Let X,Y be the sizes of the largest and second largest components of G respectively. Then*

$$P(|X-\beta n| \ge n^{2/3}) \le \exp(-\Theta(n^{1/3})), \tag{8}$$
$$P(Y \ge n^{1/3}) \le \exp(-\Theta(n^{1/3})). \tag{9}$$

Equation (8) is proved in [1, Theorem 3.1]. Equation (9) of Lemma 7 is proved in Section A.1 of the full version [12].

▶ **Lemma 8** (see, e.g., [15], p.109)**.** *Let $t \in (0,1]$ be a constant. Let $G \sim G(n,c/n)$ where $c < 1$. Let X be the size of the largest component of G.*

$$P(X \ge n^t) \le \exp(-\Theta(n^t)).$$

Proof of Lemma 5. Let $X_0 \in S$ and let $\gamma := F'(a)$ (recall that $|\gamma| < 1$, since a is Jacobian attractive fixpoint by Lemma 3). The first step of the Swendsen-Wang algorithm chooses, for each color class, a random graph from $G(m,p)$, where $p = B/n$ and m is the number of vertices of that color. Let m_1 be the number of vertices of the dominant color. Since $X_0 \in S$ we have $m_1/n = a + \tau =: a'$ where $|\tau| < \varepsilon$. We can write

$$p = (m_1 B/n)/m_1 = (a'B)/m_1,$$

where $a'B > 1$ for sufficiently small $\varepsilon_0 > 0$ (using $aB > 1$ from Lemma 38 in the full version [12]). This means that the $G(m,p)$ process in this component is supercritical. Let $\beta \in (0,1]$ be the root of $x + \exp(-a'Bx) = 1$. By Lemma 7 the random graph will have, with probability $\geq 1 - \exp(-\Theta(n^{1/3}))$, one component of size $a'\beta n \pm n^{2/3}$ and all the other components will have size at most $n^{1/3}$.

Let m_2 be the number of vertices in one of the non-dominant colors. Since $X_0 \in S$ we have $m_2/n =: b'$ where

$$b - \varepsilon_0 \leq b - \varepsilon \leq b' \leq b + \varepsilon \leq b + \varepsilon_0. \tag{10}$$

We can write

$$p = (m_2 B/n)/m_2 = (b'B)/m_2,$$

where $b'B < 1$ for sufficiently small $\varepsilon_0 > 0$ (using $bB < 1$, again from Lemma 38 in the full version [12]). This means that the $G(m,p)$ process in this component is subcritical. By Lemma 8 (with $t = 1/3$), with probability $\geq 1 - \exp(-\Theta(n^{1/3}))$ the random graph will have all components of size at most $n^{1/3}$.

To summarize: starting from a configuration in S after the first step of the Swendsen-Wang algorithm we have, with probability $\geq 1 - q\exp(-\Theta(n^{1/3}))$ one large component of size $a'\beta n \pm n^{2/3}$ and the remaining components are of size $\leq n^{1/3}$ (small components). In the second step of the algorithm the components get colored by a random color. By symmetry, in expectation each color obtains $(n - a'\beta n \mp n^{2/3})/q$ vertices from the small components and by Azuma's inequality this number is $(n - a'\beta n \mp n^{2/3})/q \pm n^{5/6}$ with probability $\geq 1 - \exp(-\Theta(n^{1/3}))$. Combining the analysis of the first and the second step we obtain that at the end with probability $\geq 1 - 2q\exp(-\Theta(n^{1/3}))$ we have one color with $F(a')n \pm 2n^{5/6}$ vertices and the rest of the colors have $\frac{1-F(a')}{q-1}n \pm 2n^{5/6}$ vertices each.

For sufficiently small $\varepsilon_0 > 0$ there exists $\gamma' \in (\gamma, 1)$ such that for all $|\tau| < \varepsilon_0$ we have $|F(a+\tau) - a| < \gamma'\tau$. Hence for sufficiently small $\varepsilon_0 > 0$ and sufficiently large n we have $|F(a')n \pm 2n^{5/6} - an| \leq \varepsilon n$ and $|\frac{1-F(a')}{q-1}n \pm 2n^{5/6} - bn| \leq \varepsilon n$. This finishes the proof of the lemma. ◀

4 Fast mixing for $B > \mathfrak{B}_{rc}$

The lemmas stated in this section are proved in Section H of the full version [12].

Once the phases align then it is straightforward to couple the chains so that the configurations agree. The following lemma is essentially identical to [7, Lemma 4], which is also used in [16, Lemma 4.1].

▶ **Lemma 9** ([7], Lemma 4)**.** *For any constant $B > 0$, for all $q \geq 2$, all $\varepsilon > 0$, for $T = O(\log n)$ there is a coupling where* $\Pr[X_T \neq Y_T \mid \boldsymbol{\alpha}(X_0) = \boldsymbol{\alpha}(Y_0)] \leq \varepsilon$.

It is enough to get the phases within $O(\sqrt{n})$ distance from $\mathbf{m}$ and then there is a coupling so that with constant probability the phases will be identical after one additional step. More precisely, we have the following.

▶ **Lemma 10** ([16], Theorem 6.5)**.** *Let $B > \mathfrak{B}_u$. Let X_0, Y_0 be a pair of configurations where $\|\boldsymbol{\alpha}(X_0) - \mathbf{m}\|_\infty \leq Ln^{-1/2}$, $\|\boldsymbol{\alpha}(Y_0) - \mathbf{m}\|_\infty \leq Ln^{-1/2}$, for a constant $L > 0$. There exists a coupling such that with prob.* $\Theta(1)$, $\boldsymbol{\alpha}(X_1) = \boldsymbol{\alpha}(Y_1)$.

Let $\varepsilon > 0$. We say a color i is ε-heavy if $\alpha_i \geq (1+\varepsilon)/B$. We say that a color is ε-light if $\alpha_i \leq (1-\varepsilon)/B$. For a state X_t, we denote by S_t the size of the largest color class in X_t. We will show that the SW-algorithm has a reasonable chance of moving into a state where one color is ε-heavy and the remaining $q-1$ colors are ε-light.

▶ **Lemma 11.** *Assume $B > \mathfrak{B}_{rc}$ is a constant. There exists $\varepsilon > 0$ such that the following hold. For any n and any initial state X_0 with probability $\Theta(1)$ the next state X_1 has one ε-heavy color and the remaining $q-1$ colors are ε-light. Further, if X_0 has one ε-heavy color and the remaining $q-1$ colors are ε-light, then the same is true for X_1 with probability $1-o(1)$.*

Afterwards the behavior of the algorithm will be controlled by the function F (S_{t+1} will be close to $nF(S_t/n)$) and then with constant probability after $O(1)$ steps the state will be close to the majority phase $\mathbf{m}$.

▶ **Lemma 12.** *Assume $B > \mathfrak{B}_{rc}$ is a constant. For any constant $\delta > 0$ and any starting state X_0 after $T = O(1)$ steps with probability $\Theta(1)$ the SW-algorithm moves to state X_T such that $\|\boldsymbol{\alpha}(X_T) - \mathbf{m}\|_\infty \leq \delta$.*

Then we show that once we are within constant distance from $\mathbf{m}$ then in $O(\log n)$ steps the distance to $\mathbf{m}$ further decreases to $O(n^{-1/2})$.

▶ **Lemma 13.** *For $B > \mathfrak{B}_u$, there exist $\delta, L > 0$ such that the following is true. Suppose that we start at a state X_0 such that $\|\boldsymbol{\alpha}(X_0) - \mathbf{m}\|_\infty \leq \delta$. Then in $T = O(\log n)$ steps with probability $\Theta(1)$ the SW algorithm ends up in a state X_t such that*

$$\|\boldsymbol{\alpha}(X_T) - \mathbf{m}\|_\infty \leq Ln^{-1/2}. \tag{11}$$

From Lemmas 9, 10, 12 and 13 we conclude the following.

▶ **Corollary 14.** *Let $B > \mathfrak{B}_{rc}$ be a constant. The mixing time of the Swendsen-Wang algorithm on the complete graph on n vertices is $O(\log n)$.*

Proof. Consider two copies $(X_t), (Y_t)$ of the SW-chain. We will show that for $T = O(\log n)$, there exists a coupling of (X_t) and (Y_t) such that $\Pr(X_T = Y_T) = \Omega(1)$. It will then follow by elementary arguments that the mixing time is $O(\log n)$.

Let δ, L be as in Lemma 13. By Lemma 12, for $T_1 = O(1)$ with probability $\Theta(1)$ we have that

$$\|\boldsymbol{\alpha}(X_{T_1}) - \mathbf{m}\|_\infty \leq \delta \text{ and } \|\boldsymbol{\alpha}(Y_{T_1}) - \mathbf{m}\|_\infty \leq \delta.$$

By Lemma 13, for $T_2 = O(\log n)$ with probability $\Theta(1)$, we have that

$$\|\boldsymbol{\alpha}(X_{T_1+T_2}) - \mathbf{m}\|_\infty \leq Ln^{-1/2} \text{ and } \|\boldsymbol{\alpha}(Y_{T_1+T_2}) - \mathbf{m}\|_\infty \leq Ln^{-1/2}. \tag{12}$$

Let $T_3 = T_1 + T_2 + 1$. Conditioning on (12), by Lemma 10 there exists a coupling so that $\boldsymbol{\alpha}(X_{T_3}) = \boldsymbol{\alpha}(Y_{T_3})$ with probability $\Omega(1)$. Once the phases agree we can apply Lemma 9 to get the two chains to agree. More precisely, by Lemma 9, there exists $T_4 = O(\log n)$ and a coupling such that $\Pr(X_{T_3+T_4} = Y_{T_3+T_4} \mid \boldsymbol{\alpha}(X_{T_3}) = \boldsymbol{\alpha}(Y_{T_3})) = \Omega(1)$. Let $T = T_3 + T_4$. We have shown that for all X_0, Y_0 there is a coupling so that $\Pr(X_T = Y_T) = \Omega(1)$. For all $\eta > 0$, by repeating this coupling $O(\log(1/\eta))$ times we obtain a coupling so that for $T' = O(T \log(1/\eta))$ we have that $\Pr(X_{T'} \neq Y_{T'}) \leq \eta$, which completes the proof by setting $\eta = 1/4$. ◀

5 Fast Mixing at $B = \mathfrak{B}_u$

We will track the size of the largest color class. Roughly, our goal is to show that the chain reaches the uniform phase in $O(n^{1/3})$ steps.

As a starting point, we have the following analogue of Lemma 11.

▶ **Lemma 15.** *For sufficiently small (constant) $\varepsilon > 0$, for any starting state X_0 of the SW-chain, with probability $\Theta(1)$, there are at least $q-1$ colors in state X_1 which are ε-light. Further, if state X_0 has $q-1$ ε-light colors, then with probability $1-\exp(-n^{\Omega(1)})$, the same is true for X_1.*

Let S_t be the size of the largest color class in state X_t of the SW-chain. The key part of our arguments is to track the evolution of S_t when there are $(q-1)$ ε-light colors. The following lemma gives some statistics of S_t/n throughout the range $(1/B, 1]$, i.e., when the largest color class is supercritical in the percolation step of the SW-dynamics. Recall the function F defined in (6),(7).

▶ **Lemma 16.** *Let $\varepsilon > 0$ be a constant and condition on the event that X_t has $q-1$ colors which are ε-light.*

Assume that ζ satisfies $(1+\varepsilon)/B \leq \zeta/n \leq 1$. Let $Z = E[S_{t+1} \,|\, S_t = \zeta]$. Then, for all sufficiently large n, it holds that

$$nF(\zeta/n) - n^{1/10} \leq Z \leq nF(\zeta/n) + n^{1/10}. \tag{13}$$

Also, there exist absolute constants Q_1, Q_2 (depending only on ε) such that

$$nQ_1 \leq Var[S_{t+1} \,|\, S_t = \zeta] \leq nQ_2, \tag{14}$$

Finally, for every integer $k \geq 3$ and constant $\varepsilon' > 0$, there exists a constant $c > 0$ such that

$$E\Big[\big|S_{t+1} - Z\big|^k \,|\, S_t = \zeta\Big] \leq cn^{k/2+\varepsilon'}. \tag{15}$$

The trickiest part of our arguments is to argue that the SW-chain escapes the vicinity of the majority phase, i.e., when the largest color class S_t is roughly na (recall that a is the marginal of the majority phase and satisfies $F(a) = a$). In particular, note that when $S_t/n = a$, from (13) the expected value of S_{t+1}/n is a as well. More generally, the drift of the process in the window $|S_t - na| \leq \varepsilon n^{2/3}$ for some small $\varepsilon > 0$ is very weak. An expansion of F around the point a yields that in this region $nF(S_t/n) \approx S_t - c(S_t - an)^2/n$ for some constant $c > 0$, so the change (in expectation) of S_{t+1} relative to S_t is roughly $\varepsilon^2 n^{1/3}$. In particular how does the process escape this window?

The rough intuition is that inside the window the variance of the process aggregates the right way, that is, after $n^{1/3}$ steps, the process is displaced by the square root of the "aggregate variance", i.e., roughly $\sqrt{n^{1/3}n} = n^{2/3}$. In the meantime, it holds that $F(z) \leq z$ so S_t is bound to escape the window from its lower end. From that point on, the drift coming from the expectation of S_t (or else the function F) is sufficiently strong to take over and drive the process to the uniform phase.

The easiest way to capture the progress of the chain towards the uniform phase is by a potential function argument. Namely, we use Lemma 16 to show the following.

▶ **Lemma 17.** *There exist constants $M, \tau > 0$ such that for all constant $\varepsilon > 0$, for all sufficiently large n the following holds. There exists a three-times differentiable potential function $G : [1/q, 1] \to [0, Mn^{1/3}]$ with $G(1/q) = 0$ such that for any $\zeta \geq (1+\varepsilon)n/B$, if X_t has $(q-1)$ colors which are ε-light it holds that*

$$E[G(S_{t+1}/n) \,|\, S_t = \zeta] \leq G(\zeta/n) - \tau. \tag{16}$$

To motivate briefly our choice of G, by taking expectations in the second order expansion of $G(S_{t+1}/n)$ around $E[S_{t+1} \,|\, S_t = \zeta]$ we obtain

$$E[G(S_{t+1}/n) \,|\, S_t = \zeta] \approx G(F(\zeta/n)) + \frac{1}{2} Var[S_{t+1}/n \,|\, S_t = \zeta]\, G''(F(\zeta/n)). \tag{17}$$

(The precise conditions on the derivatives of G such that the approximation in (17) is sufficiently accurate are given in Lemma 26 of the full version [12].) From (17), in order to satisfy (16), the function G has to be carefully chosen to control the interplay between $G(F(x)) - G(x)$ and $G''(F(x))$. The first derivative of G should correspond to the drift $F(x) - x$ of the process coming from its expectation while the second derivative of G to the variance of the process. More precisely, when x is outside the critical window, the choice of the potential function is such that $G(F(x)) - G(x)$ is bounded above by a negative constant (i.e., its derivative is $1/(x - F(x))$); by our earlier remarks this should be sufficient to establish progress outside the critical window. Indeed, with this choice it turns out that $|G''(x)|/n$ is bounded above by a small constant outside the critical window, so that (16) is satisfied. Inside the critical window, where $x \approx F(x)$ and hence $G(F(x)) - G(x) \approx 0$, we choose G so that $G''(x)$ is negative. More precisely, to satisfy (16), since $Var[S_{t+1}/n \,|\, S_t = \zeta] = \Theta(1/n)$ from Lemma 16, we set $G''(x) = -Cn$ for some constant $C > 0$. The remaining part is then to interpolate between these two regimes keeping $G'(x)/G''(x)$ sufficiently large (so that (16) is satisfied) and $G(x)$ small (i.e., $O(n^{1/3})$); this is possible due to the quadratic behaviour of $F(z) - z$ around $z = a$. (See Lemma 27 in the full version [12] and its proof for the explicit specification of G.)

Lemmas 16 and 17 capture the SW-dynamics when the largest color class is supercritical. In the complementary regime, we have the following.

▶ **Lemma 18.** *Let $\varepsilon > 0$ be a sufficiently small constant. Suppose that X_0 is such that $q-1$ colors are ε-light and that $S_0 < (1+\varepsilon)n/B$. Then with probability $1 - \exp(-n^{\Omega(1)})$ it holds that $S_1 < (1+3q\varepsilon)n/q$.*

We next combine Lemmas 15, 17 and 18 to show the following.

▶ **Lemma 19.** *For $B = \mathfrak{B}_u$, there exists $L > 0$ such that the following is true. For any starting state X_0, in $T = O(n^{1/3})$ steps, with probability $\Theta(1)$ the SW algorithm ends up in a state X_T such that $\|\boldsymbol{\alpha}(X_T) - \mathbf{u}\|_\infty \le Ln^{-1/2}$.*

Proof. Let $\varepsilon > 0$ be a sufficiently small constant, to be picked later. We will assume that the state X_1 has $q-1$ ε-light colors since (by the first part of Lemma 15) this event happens with probability $\Theta(1)$. Henceforth, we will condition on this event.

Recall that S_t is the size of the largest color component at time t. We first prove that with probability $\Theta(1)$ for some $T = O(n^{1/3})$ it holds that $S_T < (1+\varepsilon)n/B$. Assuming this for the moment, then in the next step, i.e., at time $T+1$, by Lemma 18 all color classes have size at most $(1+3q\varepsilon)n/q$ and (for all sufficiently small ε) are thus subcritical in the percolation step of the SW-dynamics. It follows that the components sizes after the percolation step satisfy, by Lemma 22 in the full version [12], $E\Big[\sum_i |C_i|^2\Big] = O(n)$. Hence, after the coloring step, using Azuma's inequality with constant probability we have color classes of size $(n + O(n^{1/2}))/q$ (see the derivation of Equations (91) and (92) in the full version [12] for details).

It remains to argue that $T = O(n^{1/3})$. We will show in fact that $T = 3Mn^{1/3}$, where M is the constant in Lemma 17. Let P_t be the probability that at time t it holds that $S_t < (1+\varepsilon)n/B$. Our goal is to show that $P_T = \Theta(1)$. We will use Lemma 17 and the potential function G therein to bound P_T. In particular, we will show that for all n sufficiently large, for all $t = 1, \dots, T-1$, it holds that

$$E[G(S_{t+1}/n)] \le E[G(S_t/n)] - \tau(1-P_t) + \tau/2, \tag{18}$$

where τ is the constant in Lemma 17. Prior to that, let us conclude the argument assuming (18). Note that if $S_t < (1+\varepsilon)n/B$ then $S_{t+1} < (1+\varepsilon)n/B$ with probability at least $1-\exp(-n^{\Omega(1)})$ (by Lemma 18), so $P_t \le P_{t+1} + O(1/n)$. It thus follows from (18) that

$$E[G(S_T/n)] \le E[G(S_1/n)] - \tau T(1/2 - P_T) + o(1),$$

which gives $P_T \ge 1/2 - Mn^{1/3}/T + o(1)$ where M is the constant in Lemma 17. For $T = 3Mn^{1/3}$ we thus have $P_T \ge 1/6$ as wanted.

Finally, we prove (18) for $t = 1, \dots, T-1$. Note that Lemmas 17 and 18 apply whenever X_t has $q-1$ ε-light colors, so we will need to account for the (small-probability) event that this fails. Namely, let $\mathcal{E}_t$ denote the event that X_t has $q-1$ ε-light colors. Since we condition on the event that $\mathcal{E}_1$ holds, we have that $\bigcap_{t=2}^{T} \mathcal{E}_t$ holds with probability at least $1-\exp(-n^{\Omega(1)})$ (by the second part of Lemma 15).

Let $\mathcal{F}_t$ be the event that $S_t < (1+\varepsilon)n/B$ and note that $P_t = \Pr(\mathcal{F}_t)$. By taking expectations in inequality (16) of Lemma 17, we have

$$E\big[G(S_{t+1}/n) \mid \mathcal{E}_t, \neg\mathcal{F}_t\big] \le E[G(S_t/n) \mid \mathcal{E}_t, \neg\mathcal{F}_t] - \tau. \tag{19}$$

Note that if $S_t < (1+\varepsilon)n/B$, then by Lemma 18, with probability $1-\exp(-n^{\Omega(1)})$ we have $S_{t+1} < (1+3q\varepsilon)n/q$ and thus (by choosing ε sufficiently small) the continuity of G and $G(1/q) = 0$ yield $G(S_{t+1}/n) \le \tau/3$. It follows that

$$E\big[G(S_{t+1}/n) \mid \mathcal{E}_t, \mathcal{F}_t\big] \le \tau/3. \tag{20}$$

Let P'_t be the probability that at time t it holds that $S_t < (1+\varepsilon)n/B$ conditioned on the event $\mathcal{E}_t$, i.e., $P'_t := \Pr(\mathcal{F}_t \mid \mathcal{E}_t)$. Note that $P_t \ge P'_t(1-\exp(-n^{\Omega(1)})) \ge P'_t - \exp(-n^{\Omega(1)})$. Combining (19) and (20), we obtain

$$E[G(S_{t+1}/n) \mid \mathcal{E}_t] \le E[G(S_t/n) \mid \mathcal{E}_t] - \tau(1-P'_t) + \tau/3. \tag{21}$$

Since G is bounded by a polynomial and since the probability of the event $\neg\mathcal{E}_t$ is exponentially small, removing the conditioning in (21) only affects the inequality by an additive $o(1)$. Similarly, replacing P'_t with P_t in (21) only affects the inequality by an additive $o(1)$. This proves that (18) holds for all sufficiently large n, thus concluding the proof of Lemma 19. ◀

Using Lemma 19, it is not hard to obtain the following corollary.

▶ **Corollary 20.** *Let $B = \mathfrak{B}_u$. The mixing time of the Swendsen-Wang algorithm on the complete graph on n vertices is $O(n^{1/3})$.*

Proof. Consider two copies $(X_t), (Y_t)$ of the SW-chain. As in the proof of Corollary 14, it suffices to show that for $T = O(n^{1/3})$, there exists a coupling of (X_t) and (Y_t) such that $\Pr(X_T = Y_T) = \Omega(1)$.

By Lemma 19, for $T_1 = O(n^{1/3})$, it holds that with probability $\Theta(1)$

$$\|\boldsymbol{\alpha}(X_{T_1}) - \mathbf{u}\|_\infty \le Ln^{-1/2} \text{ and } \|\boldsymbol{\alpha}(Y_{T_1}) - \mathbf{u}\|_\infty \le Ln^{-1/2}. \tag{22}$$

Conditioning on (22), by an analogue of Lemma 10 (see Lemma 36 in the full version [12]), there exists a coupling such that with probability $\Theta(1)$ for $T_2 = T_1 + 1$, it holds that $\boldsymbol{\alpha}(X_{T_2}) = \boldsymbol{\alpha}(Y_{T_2})$. By Lemma 9, there exists $T_3 = O(\log n)$ and a coupling such that $\Pr(X_{T_2+T_3} = Y_{T_2+T_3} \mid \boldsymbol{\alpha}(X_{T_2}) = \boldsymbol{\alpha}(Y_{T_2})) = \Omega(1)$. It is now immediate to combine the couplings to obtain a coupling such that $\Pr(X_T = Y_T) = \Omega(1)$ with $T = T_2 + T_3 = O(n^{1/3})$, as desired. ◀

References

1 J. Ameskamp and M. Löwe. Moderate deviations for the size of the largest component in a super-critical Erdös-Rényi graph. *Markov Process. Related Fields*, 17(3):369–390, 2011.

2 A. Blanca and A. Sinclair. Dynamics for the mean-field random-cluster model. Available from the arXiv at: http://arxiv.org/abs/1412.6180

3 B. Bollobás, G. Grimmett, and S. Janson. The random-cluster model on the complete graph. *Probability Theory and Related Fields.* 104(3):283–317, 1996.

4 C. Borgs, J. T. Chayes, A. Frieze, J. H. Kim, P. Tetali, E. Vigoda, and V. H. Vu. Torpid mixing of some Monte Carlo Markov chain algorithms in statistical physics. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 218-229, 1999.

5 C. Borgs, J. T. Chayes, and P. Tetali. Tight Bounds for Mixing of the Swendsen-Wang Algorithm at the Potts Transition Point. *Probability Theory and Related Fields*, 152(3-4):509–557, 2012.

6 A. Coja-Oghlan, C. Moore, and V. Sanwalani. Counting connected graphs and hypergraphs via the probabilistic method. *Random Structures & Algorithms*, 31(3):288–329, 2007.

7 C. Cooper, M. E. Dyer, A. M. Frieze, and R. Rue. Mixing properties of the Swendsen-Wang process on the complete graph and narrow grids. *J. Math. Phys.* 41(3):1499–1527, 2000.

8 C. Cooper and A. M. Frieze. Mixing Properties of the Swendsen-Wang Process on Classes of Graphs. *Random Structures & Algorithms*, 15(3–4):242–261, 1999.

9 M. Costeniuc, R. S. Ellis, and H. Touchette. Complete analysis of phase transitions and ensemble equivalence for the Curie-Weiss-Potts model. *J. Math. Phys.*, 46(6):063301, 25 pages, 2005.

10 P. Cuff, J. Ding, O. Louidor, E. Lubetzky, Y. Peres, and A. Sly. Glauber Dynamics for the mean-field Potts Model. *Journal of Statistical Physics*, 149(3):432–477, 2012.

11 A. Galanis, D. Štefankovič, E. Vigoda, and L. Yang. Ferromagnetic Potts Model: Refined #BIS-hardness and Related Results. In *Proceedings of the 18th International Workshop on Randomization and Computation* (RANDOM), pages 677–691, 2014.

12 A. Galanis, D. Štefankovič, and E. Vigoda, Swendsen-Wang Algorithm on the Mean-Field Potts Model. Full version available from the arXiv at: http://arxiv.org/abs/1502.06593v2

13 V. K. Gore and M. R. Jerrum. The Swendsen-Wang process does not always mix rapidly. *Journal of Statistical Physics*, 97(1–2):67–86, 1999.

14 O. Häggström. The random-cluster model on a homogeneous tree. *Probability Theory and Related Fields*, 104(2):231–253, 1996.

15 S. Janson, T. Łuczak, and A. Rucinski. *Random graphs.* Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience, New York, 2000.

16 Y. Long, A. Nachmias, W. Ning, and Y. Peres. A Power Law of Order 1/4 for Critical Mean Field Swendsen-Wang Dynamics. *Memoirs of the AMS*, 232:number 1092, 2014.

17 A. Nachmias, and Y. Peres. The critical random graph, with martingales. *Israel Journal of Mathematics*, 176(1):29-41, 2010.

18 B. Pittel. On tree census and the giant component in sparse random graphs. *Random Structures & Algorithms*, 1(3):311–342, 1990.

19 V. E. Stepanov. The probability of the connectedness of a random graph $\mathcal{G}_m(t)$. *Teor. Verojatnost. i Primenen*, 15:58–68, 1970.

20 M. Ullrich. Rapid mixing of Swendsen-Wang dynamics in two dimensions. Ph.D. Thesis, Universität Jena, Germany, 2012. Available from the arXiv at: http://arxiv.org/abs/1212.4908

Decomposing Overcomplete 3rd Order Tensors using Sum-of-Squares Algorithms

Rong Ge[1] **and Tengyu Ma**[2]

1 **Microsoft Research**
1 Memorial Dr, Cambridge MA, USA
rongge@microsoft.com
2 **Computer Science Department**
Princeton University, USA
tengyu@cs.princeton.edu

Abstract

Tensor rank and low-rank tensor decompositions have many applications in learning and complexity theory. Most known algorithms use unfoldings of tensors and can only handle rank up to $n^{\lfloor p/2 \rfloor}$ for a p-th order tensor in $\mathbb{R}^{n^p}$. Previously no efficient algorithm can decompose 3rd order tensors when the rank is super-linear in the dimension. Using ideas from sum-of-squares hierarchy, we give the first quasi-polynomial time algorithm that can decompose a random 3rd order tensor decomposition when the rank is as large as $n^{3/2}/\operatorname{poly}\log n$.

We also give a polynomial time algorithm for certifying the injective norm of random low rank tensors. Our tensor decomposition algorithm exploits the relationship between injective norm and the tensor components. The proof relies on interesting tools for decoupling random variables to prove better matrix concentration bounds.

1998 ACM Subject Classification I.2.6 Learning

Keywords and phrases sum of squares, overcomplete tensor decomposition

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.829

1 Introduction

Tensors, as natural generalization of matrices, are often used to represent multi-linear relationships or data that involves higher order correlation. A p-th order tensor $T \in \mathbb{R}^{n^p}$ is a p-dimensional array indexed by $[n]^p$. A tensor T is rank-1 if it can be written as the outer-product of p vectors $T = a_1 \otimes \cdots \otimes a_p$, where $a_i \in \mathbb{R}^n$ (for $i = 1, \dots, p$). Equivalently, $T_{i_1,\dots,i_p} = \prod_{j=1}^{p} a_j(i_j)$ where $a_j(i_j)$ denotes the i_j-th entry of vector a_j.

Low rank tensors – similar to low rank matrices – are widely used in many applications. The rank of tensor T is defined as the minimum number m such that T can be written as the sum of m rank-1 tensors. This agrees with the definition of matrix rank. However, most of the corresponding tensor problems are much harder: for $p \geq 3$ computing the rank of the tensor (as well as many related problems) is NP-hard [22, 23]. Tensor rank is also not as well-behaved as matrix rank (see for example the survey [15]).

Unlike matrices, low rank tensor decompositions are often unique [24], which is important in many applications. In special cases (especially when rank m is less than dimension n) tensor decomposition can be efficiently computed. Such specialized tensor decompositions have been the key algorithmic ideas in many recent algorithms for learning latent variable models, including mixture of Gaussians, Independent Component Analysis, Hidden Markov Model and Latent Dirichlet Allocation (see [4]). In many cases tensor decomposition can be viewed as reinterpreting previous spectral learning results [14, 26, 2, 5]. This new interpretation has also inspired many new works (e.g. [3, 13, 19]).

A common limitation in early tensor decomposition algorithms is that they only work for the undercomplete case when rank m is at most the dimension n. Although there are some attempts to decompose tensors in the overcomplete case ($m > n$) [16, 13, 7, 18, 17], these works either require at least 4-th order tensors, or is polynomial time only when in mildly overcomplete case (when m is a constant factor larger than n). In many machine learning applications, the number of samples required to accurately estimate a 4-th order tensor is too large. In practice algorithms based on 3rd order tensor are much more preferable. Therefore we are interested in the key question: are there any efficient algorithms for overcomplete 3rd order tensor decomposition?

In the worst case setting, overcomplete 3rd order tensors are not well-understood. Kruskal [24] showed the tensor decomposition is unique when the rank $m \leq 1.5n - 1$ and the components are in general position, but there is no efficient algorithm known for finding this decomposition. Constructing an explicit 3rd order tensor with rank $\Omega(n^{1+\epsilon})$ will give nontrivial circuit complexity lowerbounds [29], while the best known rank bound for an explicit 3rd order matrix is only $3n - O(\log n)$ [1].

For many of the learning applications, it is natural to consider the average case problem where the components of the tensor are chosen according to a random distribution. In this case [6] give a polynomial time algorithm that can find the true components when $m = Cn$ for any constant $C > 0$ (however the runtime depends exponentially on C).

This paper also considers this average case setting and gives a quasi-polynomial algorithm for decomposing the tensor when m can be as large as $n^{3/2}$. The main idea of the algorithm is based on sum-of-squares (SoS) SDP hierarchy ([27, 25], see Section 2 and the recent survey [12]). The main difficulty in handling overcomplete 3rd order tensors is that there is no natural *unfolding* (i.e. mapping to a matrix) that can certify the rank of the tensor. We can unfold a 4-th order tensor T into a matrix M of size $n^2 \times n^2$ where $M_{(i_1,i_2),(i_3,i_4)} = T_{i_1,i_2,i_3,i_4}$. However, unfolding 3rd order tensor will result in a very unbalanced matrix of dimension $n \times n^2$ that cannot have rank more than n. Intuitively, the power of SoS-based algorithm is that it can provide higher-order "pseudo-moments" that will allow us to use nontrivial unfoldings.

In particular, the key component of the proof is a way of certifying *injective norm* (see Section 2) of random tensors, which is closely related to the problem of certifying the 2-to-4 norm of random matrices[8]. Recently, there has been an increasing number of applications of SoS hierarchy to learning problems. [9] give algorithms for finding the sparsest vectors in a subspace, which is closely related to many learning problems. [10] give a new algorithm for dictionary learning that can handle nearly linear sparsity, and also an algorithm for robust tensor decomposition.However their result requires a tensor of high order. [11] studies a related problem of tensor prediction, also using ideas of SoS hierarchies.

1.1 Our Results

In this paper we give a quasi-polynomial time algorithm for decomposing third-order tensors when the rank m is almost as large as $n^{3/2}$ and the components of the tensor is chosen randomly. More concretely, we define $\mathcal{D}_{m,n}$ to be a distribution of third order tensors of the following form:

$$T = \sum_{i=1}^{m} a_i^{\otimes 3},$$

where the vectors $a_i \in \mathbb{R}^n$ are uniformly random vectors in $\{\pm\frac{1}{\sqrt{n}}\}^n$ and $a_i^{\otimes 3}$ is short for $a_i \otimes a_i \otimes a_i$. Our goal is to recover these components a_i's. Since any permutation of a_i's is still a valid solution, we say two decompositions are ϵ-close if they are close after an arbitrary permutation:

▶ **Definition 1** (ϵ-close)**.** Two sets of vectors $\{a_i\}_{i\in[m]}$ and $\{\hat{a}_i\}_{i\in[m]}$ in $\mathbb{R}^n$ are ϵ-close if there exists a permutation $\pi : [m] \to [m]$ such that $\|\hat{a}_{\pi(i)} - a_i\| \le \epsilon$. Two decompositions of the tensor T are ϵ-close if their components are ϵ-close.

For tensors in distribution $\mathcal{D}_{m,n}$ our algorithm can recover the decomposition as long as $m \ll n^{3/2}$.

▶ **Theorem 2.** *Given a tensor $T = \sum_{i=1}^m a_i^{\otimes 3}$ sampled from distribution $\mathcal{D}_{m,n}$, when $m \ll n^{3/2}$ there is an algorithm that runs in time $n^{O(\log n)}$ and with high probability returns a decomposition $T \approx \sum_{i=1}^m \hat{a}_i^{\otimes 3}$ that is 0.1-close to the true decomposition.*

Our result easily generalizes to many other distributions for a_i (including a uniform random vector in unit sphere or a spherical Gaussian).

The algorithm does not output a very accurate solution (the accuracy can be improved to ϵ with an exponential dependency on $1/\epsilon$). However it is known that alternating minimization algorithms can refine the decomposition once we have a nice initial point[6]:

▶ **Theorem 3** ([6])**.** *Given a tensor T from distribution $\mathcal{D}_{m,n}$ ($m \ll n^{3/2}$), and an initial solution that is 0.1-close to the true decomposition, then for any $\epsilon > 0$ (that may depend on n) there is an algorithm that runs in time $\mathrm{poly}(n, \log 1/\epsilon)$ that with high probability finds a refined decomposition that is ϵ-close to the true decomposition.*

Combining the two results we have an algorithm that runs in time $n^{O(\log n)} \operatorname{poly}\log(1/\epsilon)$ that recovers a decomposition that is component-wise ϵ-close to the true decomposition.

▶ **Corollary 4.** *Given a tensor $T = \sum_{i=1}^m a_i^{\otimes 3}$ sampled from distribution $\mathcal{D}_{m,n}$, when $m \ll n^{3/2}$ for any $\epsilon > 0$ there is an algorithm that runs in time $n^{O((\log n))} \operatorname{poly}\log(1/\epsilon)$ and with high probability returns a decomposition $T \approx \sum_{i=1}^m \hat{a}_i^{\otimes 3}$ that is ϵ-close to the true decomposition.*

The main idea in proving Theorem 2 is the observation that when the tensor is generated randomly from $\mathcal{D}_{m,n}$, the true components are close to the maximizers of the multilinear form $T(x,x,x) = \sum_{i,j,k\in[n]} T_{i,j,k}x_ix_jx_k = \sum_{i=1}^m \langle a_i, x\rangle^3$. The maximum value of $T(x,x,x)$ on unit vectors $\|x\| = 1$ is known as the injective norm of the tensor. Computing or even approximating the injective norm is known to be hard [20, 21]. A key component of our approach is a sum-of-square algorithm (see Section 2 for preliminaries about sum-of-square algorithms) that certifies that the injective norm of a random tensor from $\mathcal{D}_{m,n}$ is small.

▶ **Theorem 5.** *For a tensor T in distribution $\mathcal{D}_{m,n}$, when $m \ll n^{3/2}$ with high probability the injective norm of T is bounded by $1 + o(1)$. Further, this can be certified in polynomial time.*

Our results (Theorem 2 and 5) still hold when we are given a tensor $\tilde{T}$ that is $1/\operatorname{poly}(n)$-close to T in the sense that the spectral norm of an unfolding of $\tilde{T} - T$ is $O(1/\operatorname{poly}\log(n))$. Theorem 3 (and hence Corollary 4) requires a tensor $\tilde{T}$ such that the unfolding of $\tilde{T} - T$ has spectral norm bounded by $\epsilon/\operatorname{poly}(n)$.

Organization

The rest of this paper is organized as follows: In Section 2 we introduce tensor notations and SoS hierarchies. Then we describe the main idea of the proof which relates tensor decomposition to the injective norm of tensor (Section 3). In Section 4 we give a polynomial time algorithm for certifying the injective norm of a random 3rd order tensor. Using this as a key tool in Section 5 we present the quasi-polynomial time algorithm that can decompose randomly generated tensors when $m \ll n^{3/2}$.

2 Preliminaries

2.1 Notations

In this paper we use $\|\cdot\|$ to denote the ℓ_2 norm of vectors and the spectral norm of matrices. That is, $\|v\| = \sqrt{\sum_i v_i^2}$ and $\|A\| = \sup_{\|u\|=1} \|Au\|$. Note that we will be using the sum-norm instead of expectation norm $\|v\|_{exp} = \sqrt{\mathbb{E}_i[v_i^2]}$ because the scaling of sum-norm is more natural for the tensor decomposition setting. We use $\langle u, v\rangle$ to denote the inner product of u and v. When A and B are two matrices, we use standard notation $A \preceq B$ to denote the fact that $B - A$ is a positive semidefinite. For a $m \times n$ matrix U and a $p \times q$ matrix V, we define the Kronecker product $U \otimes V$ as the $mp \times nq$ block matrix

$$U \otimes V = \begin{bmatrix} U_{1,1}V & \cdots & U_{1,n}V \\ \vdots & \ddots & \vdots \\ U_{m,1}V & \cdots & U_{m,n}V \end{bmatrix}$$

We use $\tilde{O}$ notations to hide dependencies on polylog factors in n and m. When we write $f \ll g$ we mean $f \le g/O(\operatorname{poly}\log n)$. Throughout the paper high probability means the probability is at least $1 - n^{-\omega(1)}$.

2.2 Tensors

Tensors are multi-dimensional arrays. In this paper for simplicity we only consider 3rd order symmetric tensors and their symmetric decompositions. For a third order symmetric tensor T, the value of $T_{i,j,k}$ only depends on the multi-set $\{i, j, k\}$, so $T_{i,j,k} = T_{j,i,k} = T_{k,i,j}$ (and more generally all the 6 permutations are equal). For a vector $v \in \mathbb{R}^n$, we use $v^{\otimes 3} \in \mathbb{R}^{n^3}$ to denote the symmetric third order tensor such that $v^{\otimes 3}_{i,j,k} = v_i v_j v_k$. Our goal is to decompose a tensor T as $T = \sum_{i=1}^m a_i^{\otimes 3}$.

There is a bijection between 3rd order symmetric tensors and homogeneous degree 3 polynomials. In particular, for a tensor T we define its corresponding polynomial $T(x, x, x) = \sum_{i,j,k=1}^n T_{i,j,k} x_i x_j x_k$. It is easy to verify that if $T = \sum_{i=1}^m a_i^{\otimes 3}$ then $T(x, x, x) = \sum_{i=1}^m \langle a_i, x\rangle^3$.

The injective norm $\|T\|_{inj}$ is defined to be the maximum value of the corresponding polynomial on the unit sphere, that is:

$$\|T\|_{inj} := \sup_{\|x\|=1} T(x, x, x).$$

It is not hard to prove when $m \ll n^{3/2}$, and the tensor T is chosen from the distribution $\mathcal{D}_{m,n}$, with high probability $1 - o(1) \le \|T\|_{inj} \le 1 + o(1)$, and in fact the value $T(x, x, x)$ is only close to 1 if x is close to one of the components a_i. We will give a (SoS) proof of this fact in Section 5

2.3 Sum-of-Square Algorithms and Proofs

Here we will only briefly introduce the notations and key concepts that are used in this paper, for more detailed discussions and references about SoS proofs we refer readers to [12] (especially Section 2).

Sum-of-squares proof system is a proof system for polynomial equalities and inequalities. Given a set of constraints $\{r_i(x) = 0\}$, and a degree bound d, we say there is a degree d SoS proof for $p(x) \geq q(x)$ if $p(x) - q(x)$ can be written as a sum of squares of polynomials modulo $r_i(x) = 0$, as defined formally below.

▶ **Definition 6** (SoS proof of degree d)**.** For a set of constraints $R = \{r_1(x) = 0, \ldots, r_t(x) = 0\}$, and an integer d, we write

$$p(x) \succeq_{R,d} q(x)$$

if there exists polynomials $h_i(x)$ for $i = 0, 1, \ldots, \ell$ and $g_j(x)$ for $j = 1, \ldots, t$ such that $\deg(h_0^2(p(x) - q(x))) \leq d$, $\deg(h_i) \leq d/2$ (for $i > 0$) and $\deg(g_j r_j) \leq d$ that satisfy

$$h_0(x)^2(p(x) - q(x)) = \sum_{i=1}^{\ell} h_i(x)^2 + \sum_{j=1}^{t} r_j(x) g_j(x),$$

We will drop the subscript d when it is clear form the context.

Note that the constraints set can be easily generalized to a set of inequalities by adding auxiliary variables. For example, constraint $r(x) \geq 0$ can be implemented as $r(x) = z^2$ where z is an auxiliary variable.

Many well-known inequalities can be proved using a low degree SoS proof, among them the most useful and important one is Cauchy-Schwarz inequality, which can be proved via degree-2 sum of squares. Another one is that $x^T Ax \preceq \|A\| \|x\|^2$. This is pretty useful when A is a random matrix where we can use random matrix theory to bound the spectral norm of A.

In order to turn an SoS arguments into an algorithm, we often consider the *pseudo-expectation*. Just as we have expectations for real distributions, we think of pseudo-expectation as expectations for pseudo-distributions that cannot be distinguished from true expectations using low degree polynomials. Pseudo-expectation can be viewed as a dual of SoS refutations.

▶ **Definition 7** (pseudo-expectation)**.** A degree d pseudo-expectation $\widetilde{\mathbb{E}}$ is a linear operator that maps degree d polynomials to reals. The operator satisfies $\widetilde{\mathbb{E}}[1] = 1$ and $\widetilde{\mathbb{E}}[p^2(x)] \geq 0$ for all polynomials $p(x)$ of degree at most $d/2$. We say a degree-d pseudo-expectation $\widetilde{\mathbb{E}}$ satisfies a set of equations $\{r_i(x) : i = 1 \ldots, \ell\}$ if for any i and any $q(x)$ such that $\deg(r_i q) \leq d$,

$$\widetilde{\mathbb{E}}\left[r_i(x) q(x)\right] = 0$$

By definition, if $p(x) \preceq_{R,d} q(x)$, and degree-d pseudo-expectation satisfies R, then we can take pseudo-expectation on both sides and obtain $\widetilde{\mathbb{E}}\left[p(x)\right] \leq \widetilde{\mathbb{E}}\left[q(x)\right]$. We will use this property of pseudo-expectation many times in the proofs.

The relationship between pseudo-expectations and SoS refutations can be summarized in the following informal lemma:

▶ **Lemma 8** ([27, 25], c.f. [12], informal stated)**.** *For a set of constraints R, either there is an SoS refutation of degree d that refutes R, or there is a degree d pseudo-expectation that satisfies R. Such a refutation/pseudo-expectation can be found in* $\mathrm{poly}(tn^d)$ *time.*

3 Relating Tensor Decompositions and Injective Norm

In this section we introduce the main idea of our proof. Given a tensor $T = \sum_{i=1}^m a_i^{\otimes 3}$ from distribution $\mathcal{D}_{m,n}$, we first make some observations about its corresponding polynomial $T(x,x,x) = \sum_{i=1}^m \langle a_i, x\rangle^3$.

When $x = a_1$, we know $T(a_1,a_1,a_1) = 1 + \sum_{i=2}^m \langle a_i, a_1\rangle^3$. Here conditioned on a_1, the second term is a sum of independent random variables ($\langle a_i, a_1\rangle^3$). By the distribution $\mathcal{D}_{m,n}$ we know these variables have mean 0 and absolute value around $1/n^{3/2}$. Standard concentration bounds show when $m \ll n^{3/2}$ with high probability $T(a_1,a_1,a_1) = 1 \pm o(1)$.

On the other hand, suppose x is a random vector in the unit sphere, then $T(x,x,x) = \sum_{i=1}^m \langle a_i, x\rangle^3$ is again a sum of random variables. By concentration bounds we know for any particular x, when $m \ll n^{3/2}$ with high probability $T(x,x,x) = o(1)$. This can actually be generalized to all vectors x that do not have large correlation with a_i's using ϵ-net arguments.

▶ **Observation.** *For a random tensor $T \sim \mathcal{D}_{m,n}$, when $m = n^{3/2}$ with high probability $T(x,x,x) \le 1 + o(1)$ for $\|x\| = 1$. Further when $T(x,x,x)$ is close to 1 the vector x is close to one of the components a_i's.*

Later we will give a SoS proof for this observation. Based on this observation, if we want to find a component, then it suffices to find a vector x such that $T(x,x,x)$ is close to 1. Using the idea of pseudo-expectations, we can do this in two steps:

1. Find a pseudo-expectation $\tilde{\mathbb{E}}[x]$ that satisfies the constraint $\|x\|^2 - 1 = 0$ and maximizes $\tilde{\mathbb{E}}[T(x,x,x)]$.
2. "Sample" from this pseudo-distribution with psuedo-expectations $\tilde{\mathbb{E}}$ to get a vector x such that $T(x,x,x) \approx 1$, in particular x will be close to one of the components a_i's.

In Section 4 we will prove the first part of the observation. In particular we show even though we are maximizing over pseudo-expectation $\tilde{\mathbb{E}}[x]$ (instead of real distributions over x), we can still guarantee the maximum value $\tilde{\mathbb{E}}[T(x,x,x)]$ is at most $1 + 1/\log n$ with high probability.

In Section 5 we give algorithms for finding a component given a pseudo-expectation $\tilde{\mathbb{E}}$ with $\tilde{\mathbb{E}}[T(x,x,x)] \approx 1$. The main idea of our algorithm is similar to the robust tensor decomposition algorithm in [10]: first we show there must be a component a_i such that $\tilde{\mathbb{E}}[\langle a_i, x\rangle^d]$ is large for a large d, then we use ideas in [10] to find the component a_i.

4 Certifying Injective Norm

Algorithm 1 Certifying Injective Norm

```
Input: A random 3-tensor $T$
Output: If $\|T\|_{\text{inj}} > 1 + 1/\log n$, return NO. If $T \sim \mathcal{D}_{m,n} (m \ll n^{3/2})$, then w.h.p. return
    YES.
    Solve the following optimization and obtain optimal value OPT
        Maximize      $\tilde{\mathbb{E}}[T(x,x,x)]$
        Subject to    $\tilde{\mathbb{E}}$ is a degree-12 pseudo-expectation                (1)
                      that satisfies $\{r(x) = \|x\|^2 - 1 = 0\}$                          (2)
    return YES if OPT $\le 1 + 1/\log n$ and NO otherwise.
```

In this section, we give Algorithm 1 based on SoS hierarchy that certifies the injective norm of random tensor. In particular, we will prove Theorem 5 which we restate in more details here.

▶ **Theorem 9.** *Algorithm 1 always returns NO when $\|T\|_{inj} > 1 + 1/\log n$. When $T \sim \mathcal{D}_{m,n}$ and $m \ll n^{3/2}$, Algorithm 1 returns YES with high probability over the randomness of T. Further, the same guarantee holds given an approximation $\tilde{T}$ where if $M \in \mathbb{R}^{n\times n^2}$ is an unfolding of $T - \tilde{T}$, $\|M\| \le 1/2\log n$.*

When $\|T\|_{\text{inj}} > 1 + 1/\log n$, then by definition there must be a vector x^* that satisfies $\|x^*\| = 1$ and $T(x^*, x^*, x^*) > 1/\log n$. We can take $\widetilde{\mathbb{E}}$ to be the expectation of a distribution that is only supported on x^* (i.e. with probability 1 $x = x^*$). Clearly this pseudo-expectation is valid, and OPT will be at least larger than $1/\log n$. Hence the algorithm returns NO.

For random tensor T, we hope to show that with high probability, the tensor norm is less than $1 + 1/\log n$ can be proved via SoS.

▶ **Theorem 10.** *With high probability over the randomness of the tensor T, for $r(x) = \|x\|^2 - 1$,*

$$T(x,x,x) \preceq_{r,12} 1 + \widetilde{O}(m/n^{3/2}) \tag{3}$$

Note that taking pseudo-expectation $\widetilde{\mathbb{E}}$ on both hand sides of (3), for any degree-12 pseudo-expectation $\widetilde{\mathbb{E}}$ that is consistent with $r(x)$,

$$\widetilde{\mathbb{E}}\left[T(x,x,x)\right] \le 1 + \widetilde{O}(m/n^{3/2})$$

That is, when $m \ll n^{3/2}$, the objective value of the convex program in Algorithm 1 is less than $1 + 1/\log n$ with high probability for random tensor.

Now we need to prove Theorem 10. We first use Cauchy-Schwarz inequality to transform LHS of (3) to a degree-4 polynomial, which would then correspond to 4th order tensors and enable non-trivial unfoldings.

▶ **Claim 11.**

$$[T(x,x,x)]^2 \preceq_{r,12} \underbrace{\sum_{i=1}^m \langle a_i, x\rangle^4}_{\text{2-4 norm}} + \underbrace{\sum_{i\neq j} \langle a_i, a_j\rangle \langle a_i, x\rangle^2 \langle a_j, x\rangle^2}_{:=p(x)}. \tag{4}$$

Proof. This is a direct application of Cauchy-Schwarz inequality:

$$\left(T\cdot x^{\otimes 3}\right)^2 = \left\langle \sum_{i=1}^m \langle a_i, x\rangle^2 a_i, x\right\rangle^2 \preceq \left\|\sum_{i=1}^m \langle a_i, x\rangle^2 a_i\right\|^2 \|x\|^2 \preceq_r \left\|\sum_{i=1}^m \langle a_i, x\rangle^2 a_i\right\|^2$$

Expanding this quantity, and using the fact that $\|a_i\| = 1$, we get

$$\left\|\sum_{i=1}^m \langle a_i, x\rangle^2 a_i\right\|^2 = \sum_{i=1}^m \langle a_i, x\rangle^4 + \sum_{i\neq j} \langle a_i, a_j\rangle \langle a_i, x\rangle^2 \langle a_j, x\rangle^2. \tag{5}$$

◀

The first term is closely related to 2-to-4 norm of random matrices: let $A \in \mathbb{R}^{m\times n}$ be a matrix whose rows are equal to a_i's, then $\|A\|_{2\to4} = \sup_{\|x\|=1}\|Ax\|_4$. Clearly, $\|A\|_{2\to4}^4 = \sup_{\|x\|=1}\sum_{i=1}^m \langle a_i, x\rangle^4$ is the maximum value of the first term. This is considered in [8] where they gave a SoS proof that when $m \ll n^2$ the first term is bounded by $O(1)$. Here we are in the regime $m \ll n^{3/2}$ so we can improve the bound to $1 + o(1)$ (The proof is deferred to Appendix A.1):

▶ **Lemma 12.** *With high probability over the randomness of a_i's,*

$$\sum_{i=1}^m \langle a_i, x\rangle^4 \preceq_{r,12} 1 + \widetilde{O}(m/n^{3/2}) \tag{6}$$

The harder part of the proof is to deal with the second term $p(x)$ on the RHS of (4). The naive idea would be to let $y = x^{\otimes 2}$ and view $p(x)$ as a degree-2 polynomial of y,

$$q(y) = \sum_{i\neq j}\langle a_i, a_j\rangle\langle a_i\otimes a_i, y\rangle\langle a_j\otimes a_j, y\rangle = y^T N y. \tag{7}$$

Here N is an n^2 by n^2 random matrix that depends on a_i's. Suppose N has spectral norm less than $o(1)$, then we have $y^T N y \preceq \|N\|\|y\|^2$, and by replacing $y = x\otimes x$ we obtain $p(x) = q(x\otimes x) \preceq o(1)$. However, in our case the matrix N have spectral norm much larger than $o(1)$.

Our key insight is that we could have different ways to unfold $p(x)$ into a degree-2 polynomial. In particular, we use the following way of unfolding:

$$q'(y) = \sum_{i\neq j}\langle a_i, a_j\rangle\langle a_i\otimes a_j, y\rangle\langle a_i\otimes a_j, y\rangle = y^T M y \tag{8}$$

where M is the n^2 by n^2 matrix that encodes the coefficients of $q'(y)$,

$$M = \sum_{i\neq j}\langle a_i, a_j\rangle(a_i\otimes a_j)(a_i\otimes a_j)^T$$

It turns out that $q'(y)$ still have the property that $q'(x\otimes x) = p(x)$. The matrix M has much better spectral norm bound, which leads us to the bound for $p(x)$.

▶ **Lemma 13.** *When $m \ll n^{3/2}$, the matrix $M = \sum_{i\neq j}\langle a_i, a_j\rangle(a_i\otimes a_j)(a_i\otimes a_j)^T$ has spectral norm at most $\widetilde{O}(m/n^{3/2})$ and as a direct consequence,*

$$p(x) \preceq_{r,4} \widetilde{O}(m/n^{3/2})$$

First we give an informal and suboptimal bound for intuition. Let B be the $n^2\times m^2$ matrix whose (i,j)-column ($i,j\in[m]$) is $a_i\otimes a_j$ (viewed as an n^2 dimensional vector). Then M can be written as $M = B\,\mathrm{diag}(\langle a_i, a_j\rangle)_{i\neq j}B^T$. Note that B can also be written as $A\otimes A$ where $\otimes$ is the Kronecker product of two matrices, so we have $\|B\| = \|A\|^2 \lesssim m/n$. Then we can bound the norm of M by $\|M\| \le \|B\|\|\,\mathrm{diag}(b)\|\|B\| \le (m/n)\cdot\max_{i,j}|\langle a_i, a_j\rangle|\cdot(m/n) \lesssim m^2/n^{5/2}$, where we used the incoherence of a_i's, that is, $|\langle a_i, a_j\rangle| \lesssim 1/\sqrt{n}$. This will only be $o(1)$ when $m \lesssim n^{1.25}$.

Intuitively, this proof is not tight because we ignored potential cancellation caused by the randomness of $\langle a_i, a_j\rangle$. Note that $\langle a_i, a_j\rangle$ have expectation 0, but we treated them all as positive $1/\sqrt{n}$. If we assume that $\langle a_i, a_j\rangle$'s are independent $\pm1/\sqrt{n}$, then $M =$

$\sum_{i\neq j}\langle a_i, a_j\rangle(a_i \otimes a_j)(a_i \otimes a_j)^T$ would be a sum of PSD matrices with random weights and we can apply more standard matrix concentration bounds to make sure cancellations happen.

However, $\langle a_i, a_j\rangle$ are of course not independent and our key idea is to decouple the randomness of $\langle a_i, a_j\rangle$.

Proof. (Sketch) We first replace the vectors a_i's with $\sigma_i a_i$ where σ_i is a random ± 1 variable. This is OK because the distribution of a_i and $\sigma_i a_i$ are the same. Now we first sample the a_i's, conditioned on the samples $M = \sum_{i\neq j}\sigma_i\sigma_j\langle a_i, a_j\rangle(a_i \otimes a_j)(a_i \otimes a_j)^T$ (where only σ_i's are still random). Now since the vectors a_i's are all fixed, the correlation between different terms only depends on scalar variables $\sigma_i\sigma_j$, and we never use the term σ_i^2 (because $i \neq j$).

By a result of [28], in this case we can decouple the product $\sigma_i\sigma_j$. In particular, in order to prove concentration properties for M, it suffices to prove concentration for a different matrix $\sum_{i\neq j}\sigma_i\tau_j\langle a_i, a_j\rangle(a_i \otimes a_j)(a_i \otimes a_j)^T$. Here $\tau \in \{\pm 1\}^m$ is an independent copy of σ_i's. In this way we have decoupled the randomness in σ_i and τ_i, and the rest of the Lemma can follow from careful matrix concentration analysis. ◀

We give the full proof of Lemma 13 in Appendix A.2.

Proof Sketch of Main Theorem

Theorem 10 follows directly from Lemma 12 and Lemma 13. Using Lemma 8, we get the main Theorem 9 in the noiseless case. When there is noise, since we have bounds on spectral norm of an unfolding of $\tilde{T} - T$, it implies (by Lemma 33) $[\tilde{T} - T](x, x, x) \preceq_{r,12} 1/2 \log n$.it is easy to verify that $\tilde{T}(x, x, x) = T(x, x, x) + [\tilde{T} - T](x, x, x) \preceq_{r,12} 1 + 1/\log n$, so Theorem 9 still holds. We give more details in Appendix A.3.

5 Quasi-polynomial Time Algorithm for Tensor Decomposition

In this section we give a quasi-polynomial time algorithm for decomposing random 3rd order tensors in distribution $\mathcal{D}_{m,n}$. In particular, we prove Theorem 2 which we restate with more details below:

▶ **Theorem 14.** *Let T be a tensor chosen from $\mathcal{D}_{m,n}$, when $m \ll n^{3/2}$ with high probability over the randomness of T Algorithm 2 returns $\{\hat{a}_i\}$ that is 0.1-close to $\{a_i\}$ in time $n^{O(\log n)}$. Further, the same guarantee holds given an approximation $\tilde{T}$ where if $M \in \mathbb{R}^{n\times n^2}$ is an unfolding of $T - \tilde{T}$, $\|M\| \leq 1/10 \log n$.*

A key component of our algorithm is a way of sampling pseudo-distributions given in [10]:

▶ **Theorem 15** (Theorem 5.1 in [10])**.** *For every $k \geq 0$, there exists a randomized algorithm with running time $n^{O(k)}$ and success probability $2^{-k/\operatorname{poly}(\epsilon)}$ for the following problem: Given a degree-k pseudo distribution $\{u\}$ over $\mathbb{R}^n$ that satisfies the polynomial constraint $\|u\|^2 = 1$ and the condition $\tilde{\mathbb{E}}[\langle c, u\rangle^k] \geq e^{-\epsilon k}$ for some unit vector $c \in \mathbb{R}^n$, output a unit vector $c' \in \mathbb{R}^n$ with $\langle c, c'\rangle \geq 1 - O(\epsilon)$.*

The basic idea of Algorithm 2 is as follows. At each iteration, the algorithm tries to find a new vector $\hat{a}_i$. As we discussed in Section 3, in order to find a vector close to a_i it finds a vector x with large $T(x, x, x)$ value. Moreover, It enforces that the new vector is different from all previous found vectors by the set of polynomial equations $\{\langle s, x\rangle^2 \leq 1/8 : s \in S\}$. Intuitively, if we haven't found all of the vectors a_i's any of the remaining a_i's will satisfy the set of constraints $\{\langle s, x\rangle^2 \leq 1/8 : s \in S\}$ and $T(x, x, x) \geq 1 - 1/\log n$. Therefore each time we can find a valid pseudo-expectation $\tilde{\mathbb{E}}$.

What we need to prove is for any pseudo-expectation $\widetilde{\mathbb{E}}$ we found, it always satisfies $\widetilde{\mathbb{E}}[\langle a_i, x\rangle^k] \geq e^{-\epsilon k}$ for some $k = O((\log n)/\epsilon)$ for some small enough constant ϵ. Then by Theorem 15we can obtain a new vector that is $O(\epsilon)$-close to one of the a_i's. We formalize this in the following lemma:

Algorithm 2 Overcomplete Random 3-Tensor Decomposition

```
Input:  Random 3-tensor $T = \sum_{i=1}^m a_i^{\otimes 3} \sim \mathcal{D}_{m,n}$.
Output:  $\hat{a}_1, \dots, \hat{a}_m \in \mathbb{R}^n$ s.t. $\{\hat{a}_i\}$ is 0.1-close to $\{a_i\}$
1: $S \leftarrow \emptyset$
2: repeat
3:     Using semidefinite programming to find a degree $k = O(\log n)$ pseudo-expectation $\widetilde{\mathbb{E}}$
       that satisfies the constraints $\{T(x,x,x) \geq 1 - 1/\log n, \|x\|^2 = 1\}$ and $\{\langle s, x\rangle^2 \leq 1/8 : s \in S\}$.
4:     Run the algorithm in Theorem 5.1 of [10] (for $n^{O(k)}$ times) with input $\widetilde{\mathbb{E}}$ and obtain
       vector $c$ such that $T(c,c,c) \geq 0.99$.
5:     add vector $c$ to $S$.
6: until $|S| = m$
7: return  $\{\hat{a}_i\} = S$.
```

► **Lemma 16.** *When T is chosen from $\mathcal{D}_{m,n}$ where $m \ll n^{3/2}$, with high probability over the randomness of T, the pseudo-expectation found in Step 3 of Algorithm 2 satisfies the following: there exists an a_i such that $\tilde{E}[\langle a_i, x\rangle^k] \geq e^{-\epsilon k}$ for sufficiently small constant ϵ (where the pseudo-expectation has degree $4k$ and $k = O((\log n)/\epsilon)$). In particular, applying Theorem 15, repeat the algorithm for $n^{O(k)}$ time will give a vector c such that $\langle c, a_i\rangle \geq 1 - O(\epsilon)$.*

The main intuition is to use Cauchy-Schwarz and Hölder inequalities (like what we used in Claim 11) to raise the power in the sum $\sum_{i=1}^m \langle a_i, x\rangle^d$ (we start with $d = 3$ and hope to get to $d = k$). When the degree is high enough we can afford to do an averaging argument and lose a factor of m to go from the sum to a individual vector, because $e^{-\epsilon k} = \text{poly}(m)$. The detailed proof is given in Appendix B.1.

Now we are ready to prove Theorem 14.

Proof. (sketch) We prove Theorem 14 by induction. Suppose s already contains a set of vectors $\hat{a}_i$'s, where for each $\hat{a}_i$ there is a corresponding a_j that satisfies $\|\hat{a}_i - a_j\| \leq 0.1$. We would like to show with high probability in the next iteration, the algorithm finds a new component that is different from all the previously found a_i's.

In order to do that, we need to show the following:

1. The SDP in Step 3 of Algorithm 2 is feasible and gives a valid pseudo-expectation.
2. For any valid pseudo-expectation, with high probability we get an unit vector c that satisfies $T(c,c,c) \geq 0.99$, and c is far from all the previously found a_i's.
3. For any unit vector c such that $T(c,c,c) \geq 0.99$, there must be a component a_i such that $\|a_i - c\| \leq 0.1$.

In these three steps, Step 1 follows because we can take $\widetilde{\mathbb{E}}$ to be the expectation of a true distribution: $x = a_i$ with probability 1 for some unfound a_i. Step 2 is basically Lemma 16, when we choose ϵ to be a small enough constant, it is easy to prove that all the vectors that satisfy $\langle c, a_i\rangle \geq 1 - O(\epsilon)$ must satisfy $T(c,c,c) \geq 0.99$. Step 3 is the second part of our observation in Section 3, which we prove in the appendix. ◄

The details in this proof can be found in Appendix B.2.

6 Conclusion

In this paper we give the first algorithm that can decompose an overcomplete 3rd order tensor when the rank m is almost $n^{3/2}$ that matches the $n^{p/2}$ bounds for even order tensors. Our argument is based on a special unfolding of the tensor and a decoupling argument for matrix concentration. We feel such techniques can be useful in other settings.

Tensor decompositions are widely applied in machine learning for learning latent variable models. Although the SoS based algorithm have poor dependency on the accuracy ϵ, in the case of tensor decomposition we can actually use SoS as an initialization algorithm. We hope such ideas can help solving more problems in machine learning.

Acknowledgements. We would like to thank Anima Anandkumar, Boaz Barak, Johnathan Kelner, David Steurer, Venkatesan Guruswami for helpful discussions at various stages of this work

References

1 Boris Alexeev, Michael A Forbes, and Jacob Tsimerman. Tensor rank: Some lower and upper bounds. In *Computational Complexity (CCC), 2011 IEEE 26th Annual Conference on*, pages 283–291. IEEE, 2011.

2 A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y. K. Liu. Two SVDs Suffice: Spectral Decompositions for Probabilistic Topic Modeling and Latent Dirichlet Allocation. *to appear in the special issue of Algorithmica on New Theoretical Challenges in Machine Learning*, July 2013.

3 A. Anandkumar, R. Ge, D. Hsu, and S. M. Kakade. A Tensor Spectral Approach to Learning Mixed Membership Community Models. In *Conference on Learning Theory (COLT)*, June 2013.

4 A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor Methods for Learning Latent Variable Models. *J. of Machine Learning Research*, 15:2773–2832, 2014.

5 A. Anandkumar, D. Hsu, and S. M. Kakade. A Method of Moments for Mixture Models and Hidden Markov Models. In *Proc. of Conf. on Learning Theory*, June 2012.

6 Anima Anandkumar, Rong Ge, and Majid Janzamin. Guaranteed Non-Orthogonal Tensor Decomposition via Alternating Rank-1 Updates. *arXiv preprint arXiv:1402.5180*, Feb. 2014.

7 Joseph Anderson, Mikhail Belkin, Navin Goyal, Luis Rademacher, and James Voss. The more, the merrier: the blessing of dimensionality for learning large gaussian mixtures. *arXiv preprint arXiv:1311.2891*, 2013.

8 Boaz Barak, Fernando G.S.L. Brandao, Aram W. Harrow, Jonathan Kelner, David Steurer, and Yuan Zhou. Hypercontractivity, sum-of-squares proofs, and their applications. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC'12, pages 307–326, New York, NY, USA, 2012. ACM.

9 Boaz Barak, Jonathan A. Kelner, and David Steurer. Rounding sum-of-squares relaxations. In *STOC*, pages 31–40, 2014.

10 Boaz Barak, Jonathan A. Kelner, and David Steurer. Dictionary learning and tensor decomposition via the sum-of-squares method. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC'15, 2015.

11 Boaz Barak and Ankur Moitra. Tensor prediction, rademacher complexity and random 3-XOR. `http://arxiv.org/abs/1501.06521`, 2015.

12 Boaz Barak and David Steurer. Sum-of-squares proofs and the quest toward optimal algorithms. In *Proceedings of International Congress of Mathematicians (ICM)*, 2014. To appear.

13 Aditya Bhaskara, Moses Charikar, Ankur Moitra, and Aravindan Vijayaraghavan. Smoothed analysis of tensor decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 594–603. ACM, 2014.

14 Joseph T. Chang. Full reconstruction of Markov models on evolutionary trees: Identifiability and consistency. *Mathematical Biosciences*, 137:51–73, 1996.

15 Pierre Comon. Tensor: a partial survey. *Signal Processing Magazine*, page 11, 2014.

16 Lieven De Lathauwer, Joséphine Castaing, and Jean-François Cardoso. Fourth-order cumulant-based blind identification of underdetermined mixtures. *Signal Processing, IEEE Transactions on*, 55(6):2965–2973, 2007.

17 Ignat Domanov and Lieven De Lathauwer. Canonical polyadic decomposition of third-order tensors: relaxed uniqueness conditions and algebraic algorithm. *arXiv preprint arXiv:1501.07251*, 2015.

18 Ignat Domanov and Lieven De Lathauwer. Canonical polyadic decomposition of third-order tensors: reduction to generalized eigenvalue decomposition. *SIAM Journal on Matrix Analysis and Applications*, 35(2):636–660, 2014.

19 Rong Ge, Qingqing Huang, and Sham M. Kakade. Learning mixtures of gaussians in high dimensions. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC'15, 2015.

20 Leonid Gurvits. Classical deterministic complexity of edmonds' problem and quantum entanglement. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC'03, pages 10–19, New York, NY, USA, 2003. ACM.

21 Aram W Harrow and Ashley Montanaro. Testing product states, quantum merlin-arthur games and tensor optimization. *Journal of the ACM (JACM)*, 60(1):3, 2013.

22 Johan Håstad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

23 Christopher J. Hillar and Lek-Heng Lim. Most tensor problems are NP hard. *arXiv preprint arXiv:0911.1393*, 2009.

24 J.B. Kruskal. Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138, 1977.

25 Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

26 Elchanan Mossel and Sébastian Roch. Learning nonsingular phylogenies and hidden Markov models. *Annals of Applied Probability*, 16(2):583–614, 2006.

27 Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization.* PhD thesis, California Institute of Technology, 2000.

28 Victor H. de la Pena and S. J. Montgomery-Smith. Decoupling inequalities for the tail probabilities of multivariate u-statistics. *The Annals of Probability*, 23(2):pp. 806–816, 1995.

29 Volker Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973.

30 Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

A Omitted Proofs in Section 4

A.1 Proof of Lemma 12

We first restate the lemma here.

▶ **Lemma 17.** *With high probability over the randomness of a_i's,*

$$\sum_{i=1}^{m}\langle a_i, x\rangle^4 \preceq_{r,12} 1+\widetilde{O}(m/n^{3/2}) \tag{9}$$

Recall [8] showed that when $m \ll n^2$,

$$\sum_{i=1}^{m}\langle a_i, x\rangle^4 \leq O(1) \tag{10}$$

Here in order to improve this bound, we consider the square of the LHS of (6) and apply Cauchy-Schwarz (similar to Claim 11),

$$\begin{aligned}\left(\sum_{i=1}^{m}\langle a_i,x\rangle^4\right)^2 &= \left\langle \sum_{i=1}^{m}\langle a_i,x\rangle^3 a_i, x\right\rangle^2 \\ &\preceq \left\|\sum_{i=1}^{m}\langle a_i,x\rangle^3 a_i\right\|^2 \|x\|^2 \qquad \text{by Cauchy-Schwarz} \\ &\preceq_r \left\|\sum_{i=1}^{m}\langle a_i,x\rangle^3 a_i\right\|^2 = \sum_{i=1}^{m}\langle a_i,x\rangle^6 + \sum_{i\neq j}\langle a_i,a_j\rangle\langle a_i,x\rangle^3\langle a_j,x\rangle^3 \end{aligned} \tag{11}$$

We will bound the first term of (11) by $1+o(1)$. We simply let $y = x^{\otimes 3}$ and let B be the matrix whose ith row is $a_i^{\otimes 3}$. Then $f(y) = \|By\|^2$ has the property that $f(x^{\otimes 3}) = \sum_{i=1}^{m}\langle a_i,x\rangle^6$. Therefore it suffices to prove that $f(y) \preceq (1+o(1)\|y\|^2$ or equivalently $\|B\| \leq 1+o(1)$.

Consider the matrix BB^T. It is a n by n matrix with diagonal entries 1 and off diagonal entries of the form $\langle a_i^{\otimes 3}, a_j^{\otimes 3}\rangle = \langle a_i, a_j\rangle^3$. By the incoherence of a_i's, we have $\langle a_i,a_j\rangle^3 \lesssim 1/n^{3/2}$. Then by Gershgorin disk theorem, we have $\|BB^T\| \leq 1+\widetilde{O}(m/n^{3/2}) = 1+\delta$. It follows that $\|B\| \leq 1+\widetilde{O}(m/n^{3/2})$. Therefore,

$$\sum_{i=1}^{m}\langle a_i,x\rangle^6 = \|Bx^{\otimes 3}\|^2 \preceq (1+\widetilde{O}(m/n^{3/2}))\|x^{\otimes 3}\| \leq_r 1+\widetilde{O}(m/n^{3/2}) \tag{12}$$

For the second term of (11), we apply Cauchy-Schwarz again:

$$\begin{aligned}\left(\sum_{i\neq j}\langle a_i,a_j\rangle\langle a_i,x\rangle^3\langle a_j,x\rangle^3\right)^2 &\preceq \left(\sum_{i\neq j}\langle a_i,a_j\rangle^2\langle a_i,x\rangle^2\langle a_j,x\rangle^2\right)\left(\sum_{i\neq j}\langle a_i,x\rangle^4\langle a_j,x\rangle^4\right) \\ &\preceq \left(\frac{1}{n}\cdot\sum_i\langle a_i,x\rangle^2\sum_j\langle a_j,x\rangle^2\right)\left(\sum_i\langle a_i,x\rangle^4\sum_j\langle a_j,x\rangle^4\right)\end{aligned} \tag{13}$$

Note that the matrix $A = [a_1|\dots|a_m]$ has spectral norm bound $\|A\| \lesssim \sqrt{m/n}$, and therefore

$$\sum_i \langle a_i, x\rangle^2 = \|A^T x\|^2 \preceq \|A\|^2\|x\|^2 \preceq_r \|A\|^2$$

Then using Equation 10, and the equation above, we have

$$\text{RHS of (13)} \;\preceq_r \frac{1}{n}\cdot\frac{m}{n}\cdot\frac{m}{n}\cdot O(1)\cdot O(1) \le \widetilde{O}(m^2/n^3) \tag{14}$$

Then by 13 and 14 and Lemma 34, we have that

$$\sum_{i\neq j}\langle a_i, a_j\rangle\langle a_i, x\rangle^3\langle a_j, x\rangle^3 \preceq_r \widetilde{O}(m^2/n^3) \tag{15}$$

Hence, combining equation (15), (12) and (11) we have that

$$\begin{aligned}\left(\sum_{i=1}^m \langle a_i, x\rangle^4\right)^2 &\preceq_r \sum_{i=1}^m \langle a_i, x\rangle^6 + \sum_{i\neq j}\langle a_i, a_j\rangle\langle a_i, x\rangle^3\langle a_j, x\rangle^3 \\ &\preceq_r 1 + \widetilde{O}(m/n^{3/2}) + \widetilde{O}(m/n^{3/2}) = 1 + \widetilde{O}(m/n^{3/2})\end{aligned} \tag{16}$$

Using Lemma 34 again, we complete the proof of Lemma 6.

A.2 Proof of Lemma 13

We first restate the lemma:

▶ **Lemma 18.** *When $m \ll n^{3/2}$, the matrix $M = \sum_{i\neq j}\langle a_i, a_j\rangle(a_i\otimes a_j)(a_i\otimes a_j)^T$ has spectral norm at most $\widetilde{O}(m/n^{3/2})$ and as a direct consequence,*

$$p(x) \preceq_{r,4} \widetilde{O}(m/n^{3/2})$$

Proof. As suggested in the proof sketch, we first use a simple symmetrization which allows us to focus on the randomness of signs of $\langle a_i, a_j\rangle$. For simplicity of notation, let $Q_{ij} := \langle a_i, a_j\rangle(a_i\otimes a_j)(a_i\otimes a_j)^T$. Let $\sigma \in \{\pm 1\}^m$ be uniform random ± 1 vector and define M' as

$$M' = \sum_{i\neq j}\sigma_i\sigma_j Q_{ij}.$$

We claim that M' has the same distribution as M, since a_i has the same distribution as $\sigma_i a_i$. Then from now on we condition on the event that a_i's have incoherence property and low spectral norm, that is, $\langle a_i, a_j\rangle \lesssim 1/\sqrt{n}$, $\|A\| = \|[a_1|a_2\dots|a_m]\| \lesssim \sqrt{m/n}$, and we will only focus on the randomness of σ. Ideally we want to write M' as a sum of independent random matrices so that we can apply matrix Bernstein inequality. However, now the random coefficients are $\sigma_i\sigma_j$, and they are not independent with each other.

A key observation here is that the sum is only over the indices (i, j) with $i \neq j$, therefore we can use Theorem 1 of [28] (restated as Theorem 29 in the end) to decouple the correlation first.

Theorem 29 basically says that to study the concentration of a sum of the form $\sum_{i\neq j} f_{ij}(X_i, X_j)$, it is up to constant factor similar to the concentration of the sum $\sum_{i\neq j} f_{ij}(X_i, Y_j)$ where Y_i is an independent copy of X_i. Applying the theorem to our situation, we have that there exists absolute constant C such that

$$\Pr[\|M'\| \ge t] \le C\Pr[M'' \ge t/C] \tag{17}$$

where

$$M'' := \sum_{i \neq j} \sigma_i \tau_j Q_{ij},$$

and σ, τ are independently uniform over $\{-1,+1\}^m$.

Now it suffices to bound the norm of M''. We proceed by rewriting M'' as

$$M'' = \sum_i \sigma_i \sum_{j \neq i} \tau_j Q_{ij} := \sum_i \sigma_i T_i,$$

where

$$T_i := \sum_{j \neq i} \tau_j Q_{ij} \tag{18}$$

We study the properties of T_i first.

▶ **Claim 19.** *With high probability over the randomness of a_i's, for all i, $T_i \preceq \tilde{O}(\sqrt{m}/n)(a_i a_i^T) \otimes I$.*

Proof. Recall that $Q_{ij} = \langle a_i, a_j \rangle (a_i \otimes a_j)(a_i \otimes a_j^T)$. In the definition 18 of T_i, the index i is fixed and we take sum over j. Therefore it will be convenient to write Q_{ij} as $Q_{ij} = \langle a_i, a_j \rangle (a_i a_i^T) \otimes (a_j a_j)^T$ where $\otimes$ is the Kronecker product between matrices. Then T_i can be written as

$$T_i = (a_i a_i^T) \otimes \left(\sum_j \tau_j \langle a_i, a_j \rangle a_j a_j^T \right).$$

We apply the Matrix Bernstein inequality (Theorem 30) on the right factor. Matrix Bernstein bound requires spectral norm bound for individual matrices, and a variance bound.

For the spectral norm of individual matrices, we check that $\|\tau_j \langle a_i, a_j \rangle a_j a_j^T\| \lesssim 1/\sqrt{n}$ (by incoherence). For variance we know

$$\| \mathbb{E}[\sum_j \tau_j^2 (\langle a_i, a_j \rangle a_j a_j^T)^2] \| = \|A \operatorname{diag}(\langle a_i, a_j \rangle^2)_{j \neq i} A^T\| \lesssim m/n^2,$$

where we used the spectral norm of A and the fact that $\langle a_i, a_j \rangle^2 \lesssim 1/n$.

Therefore by Matrix Bernstein's inequality (Theorem 30) we have that whp, over the randomness of τ,

$$\| \sum_j \tau_j \langle a_i, a_j \rangle a_j a_j^T \| \leq \widetilde{O}(\sqrt{m}/n).$$

Using the fact that for two matrices P and Q, if $P \preceq Q$ and R is PSD, then $R \otimes P \preceq R \otimes Q$ (see Claim 20), it follows that

$$T_i \preceq (a_i a_i^T) \otimes (\widetilde{O}(\sqrt{m}/n) \cdot I).$$

Finally we use union bound and conclude with high probability this is true for any i. ◀

Now we can apply matrix Bernstein for the sum $M'' = \sum_{i=1}^m \sigma_i T_i$. The individual spectral norm is bounded by $\tilde{O}(\sqrt{m}/n)$ by the Claim 19. The variance is

$$\| \sum_{i=1}^m T_i^2 \| \leq \tilde{O}(m/n^2) \| \sum_{i=1}^m ((a_i a_i^T) \otimes I)^2 \| = \tilde{O}(m/n^2) \|(AA^T) \otimes I\| = \tilde{O}(m^2/n^3).$$

Using matrix Bernstein inequality, we know with high probability $\|M''\| \leq \tilde{O}(m/n^{3/2})$.

Using (17), we get that whp, $\|M'\| \leq \widetilde{O}(m/n^{3/2})$. Since M' and M has the same distribution, we conclude that whp, $\|M\| \leq \widetilde{O}(m/n^{3/2})$. ◀

We complete the proof by providing the following claim about Kronecker products.

▶ **Claim 20.** *If $P \preceq Q$ and R is psd, then $R \otimes P \preceq R \otimes Q$.*

Proof. It suffices to prove this when $R = uu^T$ (as we can always decompose R as sum of rank one components). In that case, for any $y \in \mathbb{R}^{n^2}$, we can write $y = u \otimes v + z$ where z is orthogonal to $u \otimes e_i$ for all $i \in [n]$. Now $(R \otimes P)z = 0$, therefore

$$y^T(R \otimes P)y = (u \otimes v)^T(R \otimes P)(u \otimes v) = (u^T Ru)(v^T Pv) \leq (u^T Ru)(v^T Qv) = y^T(R \otimes Q)y.$$

Therefore $R \otimes P \preceq R \otimes Q$. ◀

A.3 Main Theorem for Certifying Injective Norm

Now we are ready to prove Theorem 9.

▶ **Theorem 21.** *Algorithm 1 always returns NO when $\|T\|_{inj} > 1+1/\log n$. When $T \sim \mathcal{D}_{m,n}$ and $m \ll n^{3/2}$, Algorithm 1 returns YES with high probability over the randomness of T. Further, the same guarantee holds given an approximation $\tilde{T}$ where if $M \in \mathbb{R}^{n \times n^2}$ is an unfolding of $T - \tilde{T}$, $\|M\| \leq 1/2 \log n$.*

Proof. We first prove whenever $\|T\|_{\mathrm{inj}} > 1 + 1/\log n$, the algorithm returns NO. This is because a large injective norm implies there exists an unit vector x^* with $T(x^*, x^*, x^*) = 1$. We can construct a pseudo-expectation $\widetilde{\mathbb{E}}$ as $\widetilde{\mathbb{E}}[p(x)] = p(x^*)$. Clearly this is a valid pseudo-expectation (it is even the expectation of a true distribution: $x = x^*$ with probability 1). Also, we know $\widetilde{\mathbb{E}}[T(x, x, x)] = T(x^*, x^*, x^*) > 1+1/\log n$, so in particular $OPT > 1+1/\log n$ and the algorithm must return NO.

Next we show the algorithm returns YES with high probability when T is chosen from $\mathcal{D}$. This follows directly from Theorem 10, which in turn follows from Lemmas 12 and 13. In particular, we know there is a degree-12 SoS proof that shows $T(x, x, x) \leq 1 + \tilde{O}(m/n^{3/2}) \leq 1 + 1/2 \log n$, so by Lemma 8 this must also hold for any pseudo-expectation.

When we are only given tensor $\tilde{T}$ such that the unfolding of $\tilde{T} - T$ has spectral norm $1/2 \log n$. Let M be the unfolding of $\tilde{T} - T$, and $y = x \otimes x$, then by Lemma 33 we know $(x^T My)^2 \preceq \|x\|^2\|M\|^2\|y\|^2$, which implies (by Lemma 33) $[\tilde{T} - T](x, x, x) = x^T My \preceq_{r,12} \|M\| \leq 1/2 \log n$. Combining the two terms we know

$$\tilde{T}(x, x, x) = T(x, x, x) + [\tilde{T} - T](x, x, x) \preceq_{r,12} 1 + 1/\log n.$$

◀

B Omitted Proof in Section 5

B.1 Proof of Lemma 16

We first restate the lemma here:

▶ **Lemma 22.** *When T is chosen from $\mathcal{D}_{m,n}$ where $m \ll n^{3/2}$, with high probability over the randomness of T, the pseudo-expectation found in Step 3 of Algorithm 2 satisfies the following: there exists an a_i such that $\tilde{E}[\langle a_i, x\rangle^k] \geq e^{-\epsilon k}$ for sufficiently small constant ϵ (where the pseudo-expectation has degree $4k$ and $k = O((\log n)/\epsilon)$). In particular, applying Theorem 15, repeat the algorithm for $n^{O(k)}$ time will give a vector c such that $\langle c, a_i\rangle \geq 1 - O(\epsilon)$.*

First we will show that for a valid pseudo-expectation, the sum of $\langle a_i, x\rangle^4$ and $\langle a_i, x\rangle^6$ are also bounded. This actually follows directly from the proof of Lemma 12 and 13.

▶ **Lemma 23.** *With high probability over the randomness of T, we have that for any degree-12 pseudo expectation $\widetilde{\mathbb{E}}$ that satisfies the constraints $\{\|x\|^2 = 1, T(x,x,x) \geq 1-\tau\}$, it also satisfies*

$$1 + \epsilon \geq \widetilde{\mathbb{E}}\left[\sum_{i=1}^{m} \langle a_i, x\rangle^4\right] \geq 1 - \epsilon \tag{19}$$

$$1 + \epsilon \geq \widetilde{\mathbb{E}}\left[\sum_{i=1}^{m} \langle a_i, x\rangle^6\right] \geq 1 - \epsilon \tag{20}$$

for $\epsilon = \widetilde{O}(m/n^{3/2}) + O(\tau)$.

Proof. We essentially just take pseudo-expectation on the SoS proofs for Lemma 12 and 13. The upper bounds follows directly by taking pseudo-expectation on equation (9) and (12). Fo the lower bounds, by taking pseudo-expectation over the SoS equation in Lemma 13, we have that $\widetilde{\mathbb{E}}[p(x)] \leq \widetilde{O}(m/n^{3/2})$. Taking pseudo-expectation over Claim 11, using the assumption that $\widetilde{\mathbb{E}}$ satisfies $T(x,x,x) \geq 1-\tau$, we have that

$$1 - \tau \leq \widetilde{\mathbb{E}}\left[[T(x,x,x)]^2\right] \leq \widetilde{\mathbb{E}}\left[\langle a_i, x\rangle^4\right] + \widetilde{\mathbb{E}}[p(x)] \leq \widetilde{\mathbb{E}}\left[\langle a_i, x\rangle^4\right] + \widetilde{O}(m/n^{3/2}) \tag{21}$$

which implies

$$\widetilde{\mathbb{E}}\left[\langle a_i, x\rangle^4\right] \geq 1 - \tau - \widetilde{O}(m/n^{3/2}). \tag{22}$$

For proving the lower bounds in (20), we first pseudo-expectation on equation 15, we have that

$$\widetilde{\mathbb{E}}\left[\sum_{i \neq j} \langle a_i, a_j\rangle \langle a_i, x\rangle^3 \langle a_j, x\rangle^3\right] \leq \widetilde{O}(m^2/n^3)$$

Then taking pseudo-expectation over equation (16), we obtain that

$$\widetilde{\mathbb{E}}\left[\left(\sum_{i=1}^{m} \langle a_i, x\rangle^4\right)^2\right] \leq \widetilde{\mathbb{E}}\left[\sum_{i=1}^{m} \langle a_i, x\rangle^6\right] + \widetilde{\mathbb{E}}\left[\sum_{i \neq j} \langle a_i, a_j\rangle \langle a_i, x\rangle^3 \langle a_j, x\rangle^3\right]$$

Note that by equation (22) and Cauchy-Schwarz, we have

$$\widetilde{\mathbb{E}}\left[\left(\sum_{i=1}^{m} \langle a_i, x\rangle^4\right)^2\right] \geq \left(\widetilde{\mathbb{E}}\left[\sum_{i=1}^{m} \langle a_i, x\rangle^4\right]\right)^2 \geq 1 - O(\tau) - \widetilde{O}(m/n^{3/2})$$

Combining the two equations above, we obtain that

$$\widetilde{\mathbb{E}}\left[\sum_{i=1}^{m} \langle a_i, x\rangle^6\right] \geq 1 - O(\tau) - \widetilde{O}(m/n^{3/2})$$

◀

Next we are going to prove that $\widetilde{\mathbb{E}}$ also satisfies the condition of Theorem 5.2 of [10].

▶ **Lemma 24.** *For $k = O((\log n)/\epsilon)$ with constant $\epsilon < 1$, If $\widetilde{\mathbb{E}}$ is a degree-k pseudo-expectation that satisfies equation (20) and (19), then there must exists $i \in [m]$ such that $\widetilde{\mathbb{E}}[\langle a_i, x\rangle^k] \geq e^{-(2\epsilon+\delta)k}$ with $\delta = \widetilde{O}(m/n^{3/2})$.*

Proof. By equation (2.5) of [10], we the following SoS version of Holder inequality. For any integer t, d and $k = t(d-2)$,

$$\|v\|_d^{dt} \preceq_k \|v\|_k^k \cdot \|v\|^{2t}$$

Let $v_i = \langle a_i, x\rangle^2$, we have

$$\left(\sum_{i=1}^m \langle a_i, x\rangle^{2d}\right)^t \preceq_k \sum_{i=1}^m \langle a_i, x\rangle^{2k} \cdot \left(\sum_{i=1}^m \langle a_i, x\rangle^4\right)^t \tag{23}$$

By Lemma 12, we have that with high probability over randomness of a_i's, $\sum_{i=1}^m \langle a_i, x\rangle^4 \preceq 1 + \widetilde{O}(m/n^{3/2})$, and it follows that

$$\left(\sum_{i=1}^m \langle a_i, x\rangle^4\right)^t \leq (1 + \widetilde{O}(m/n^{3/2}))^t \tag{24}$$

By picking $d = 3$, we have $t = k$. Taking $t = O(\log m/\epsilon)$ and combining equation (23) and (24), we have that

$$\left(\sum_{i=1}^m \langle a_i, x\rangle^6\right)^k \preceq_k \sum_{i=1}^m \langle a_i, x\rangle^{2k} \cdot \left(\sum_{i=1}^m \langle a_i, x\rangle^4\right)^k \preceq_k (1 + \widetilde{O}(m/n^{3/2}))^k \sum_{i=1}^m \langle a_i, x\rangle^{2k}$$

Applying pseudo-expectation on both hands, we obtain,

$$\widetilde{\mathbb{E}}\left[\left(\sum_{i=1}^m \langle a_i, x\rangle^6\right)^k\right] \leq (1 + \widetilde{O}(m/n^{3/2}))^k \cdot \widetilde{\mathbb{E}}\left[\sum_{i=1}^m \langle a_i, x\rangle^{2k}\right]$$

Note that by Cauchy-Schwarz and equation (20), we have

$$(1-\epsilon)^k \leq \widetilde{\mathbb{E}}\left[\sum_{i=1}^m \langle a_i, x\rangle^6\right]^k \leq \widetilde{\mathbb{E}}\left[\left(\sum_{i=1}^m \langle a_i, x\rangle^6\right)^k\right]$$

Combining the two equations above, we obtain that for $\delta = \widetilde{O}(m/n^{3/2})$,

$$\widetilde{\mathbb{E}}\left[\sum_{i=1}^m \langle a_i, x\rangle^{2k}\right] \geq (1-\delta)^k (1-\epsilon)^k \tag{25}$$

Therefore by averaging argument, there exists i such that

$$\widetilde{\mathbb{E}}[\langle a_i, x\rangle^{2k}] \geq (1-\delta)^k/m = e^{-\delta k - \log m - \epsilon k}$$

when $k \geq (\log m)/\epsilon$, we have that $\widetilde{\mathbb{E}}[\langle a_i, x\rangle^{2k}] \geq e^{-(2\epsilon+\delta)k}$ ◀

Lemma 16 follows directly from the two lemmas above.

B.2 Proof of Theorem 14

In this section we prove the main theorem in Section 5.

▶ **Theorem 25.** *Let T be a tensor chosen from $\mathcal{D}_{m,n}$, when $m \ll n^{3/2}$ with high probability over the randomness of T Algorithm 2 returns $\{\hat{a}_i\}$ that is 0.1-close to $\{a_i\}$ in time $n^{O(\log n)}$. Further, the same guarantee holds given an approximation $\tilde{T}$ where if $M \in \mathbb{R}^{n\times n^2}$ is an unfolding of $T - \tilde{T}$, $\|M\| \le 1/10 \log n$.*

As suggested in the proof sketch, we prove this theorem by induction. The induction hypothesis is that all vectors $s_i \in S$ are 0.1-close (in ℓ_2 norm) to distinct components a_i's. We break the proof into three claims:

▶ **Claim 26.** *With high probability over the tensor T, suppose all the previously found s_i's are 0.1-close (in ℓ_2 norm) to some components a_j's, then there exists a pseudo-expectation that satisfies Step 3 in Algorithm 2.*

Proof. We first prove that with high probability $T(a_i, a_i, a_i) \ge 1 - 1/\log n$ for all i. This is easy because $T(a_i, a_i, a_i) = 1 + \sum_{j\neq i}\langle a_i, a_j\rangle^3$. Conditioned on a_i, the values $\langle a_i, a_j\rangle$ are sub-Gaussian random variables with mean 0 and variance $1/n$, so by standard concentration bounds we know with high probability $\sum_{j\neq i}\langle a_i, a_j\rangle^3 \ge -1/\log n$. We can then take the union bound and conclude $T(a_i, a_i, a_i) \ge 1 - 1/\log n$ for all i.

Now for simplicity of notation, assume that $S = \{s_1, \dots, s_t\}$ for some $t < m$, where s_i is 0.1-close to a_i. We can construct a pseudo-expectation $\widetilde{\mathbb{E}}[p(x)] = p(a_{t+1})$. Clearly this is a valid pseudo-expectation that satisfies $\|x\|^2 = 1$. For the inequality constraints we also know $\langle a_{t+1}, s_i\rangle^2 \le 2(\langle a_{t+1}, a_i\rangle^2 + \langle a_{t+1}, a_i - s_i\rangle^2) < 1/8$ (where the whole proof only uses Cauchy-Schwarz and $(A+B)^2 \le 2(A^2+B^2)$, so the proof is SoS). Therefore the system in Step 3 must have a feasible solution. ◀

▶ **Claim 27.** *For any valid pseudo-expectation in Step 3, with high probability we get an unit vector c that satisfies $T(c,c,c) \ge 0.99$, and c is far from all the previously found a_i's.*

Proof. By Lemma 16 we know there must be a vector a_i such that $\widetilde{\mathbb{E}}[\langle a_i, x\rangle^k] \ge e^{-\epsilon k}$ for sufficiently small constant ϵ. We show that this vector a_i cannot be among the previously found ones. By Lemma 32 we know that for even number k,

$$(\langle s_i, x\rangle + \langle s_i - a_i, x\rangle)^k \le 2^{k-1}(\langle s_i - a_i, x\rangle^k + \langle s_i, x\rangle^k)$$

Taking pseudo-expectations over both sides, we have that

$$\widetilde{\mathbb{E}}[\langle a_i, x\rangle^k] \preceq_{2k} 2^{k-1}(\widetilde{\mathbb{E}}[\langle s_i, x\rangle^k] + k\widetilde{\mathbb{E}}[\langle s_i - a_i, x\rangle^k]) \preceq_{\|x|^2=1,2k} e^{-\epsilon k}$$

where we've used the constraint $\langle s_i, x\rangle^2 \le 1/8$ and induction hypothesis $\|s_i - a_i\| \le 0.1$.

Now applying Theorem 15 we get a vector c that is has inner-product $1 - O(\epsilon)$ with a_i. Therefore $T(c,c,c) = T(a_i,a_i,a_i) + T(c - a_i, a_i, a_i) + T(c, c - a_i, a_i) + T(c,c,a_i) \ge 1 - 1/\log n - 3\|T\|_{\mathrm{inj}}\|c - a_i\| \ge 0.99$. Here $T(x,y,z) = \sum_{i_1,i_2,i_3} T_{i_1,i_2,i_3}x_{i_1}y_{i_2}z_{i_3}$ is the multilinear form for the tensor, and note that this step of the proof does not need to be SoS because we already have the vector c from Theorem 15. ◀

▶ **Claim 28.** *For any unit vector c such that $T(c,c,c) \ge 0.99$, there must be a component a_i such that $\|a_i - c\| \le 0.1$.*

Proof. We define the following trivial pseudo-expectation $\widetilde{\mathbb{E}}^c$ defined by c: $\widetilde{\mathbb{E}}^c[p(x)] = p(c)$. Then we know that $\widetilde{\mathbb{E}}^c$ does satisfy equation $T(x,x,x) \geq 0.99$, and the degree of $\widetilde{\mathbb{E}}^c$ can be any finite number. Therefore, by Lemma 24, we have that $\widetilde{\mathbb{E}}^c\left[\langle a_i, x\rangle^k\right] \geq e^{-(2\epsilon+\delta)k}$ for $k = O(\log n)$. Therefore using the definition of $\widetilde{\mathbb{E}}^c$, we have that $\widetilde{\mathbb{E}}^c\left[\langle a_i, x\rangle^k\right] = \langle a_i, c\rangle^k \geq e^{-(2\epsilon+\delta)k}$. Taking $\epsilon = 0.001$ and then we have that $\langle a_i, c\rangle \geq 0.999 - \delta$ and it follows that $\|a_i - c\| \leq 0.99$. ◀

These three claims finishes the induction in the noiseless case. For the noisy case, we can handle it the same ways as Theorem 9: note that $[\tilde{T} - T](x,x,x) \preceq_{\|x\|^2=1,12} 1/2 \log n$ and this additional term does not change any part of the proof.

Finally, the runtime of Line 3 in Algorithm 2 is $n^{O(k)}$, and the run-time of line 4 is also $n^{O(k)}$. Therefore the total runtime is $n^{O(k)}$.

C Matrix Concentrations

In this section we introduce theorems used to prove matrix concentrations. First we need the following lemma for decoupling the randomness in the sum.

▶ **Theorem 29** (Special case of Theorem 1 of [28]). *Let $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ are independent random variables on a measurable space over S, where X_i and Y_i has the same distribution for $i = 1, \ldots, n$. Let $f_{ij}(\cdot,\cdot)$ be a family of functions taking $S \times S$ to a Banach space $(B, \|\cdot\|)$. Then there exists absolute constant C, such that for all $n \geq 2$, $t > 0$,*

$$\Pr\left[\left\|\sum_{i\neq j} f_{ij}(X_i, X_j)\right\| \geq t\right] \leq C \Pr\left[\left\|\sum_{i\neq j} f_{ij}(X_i, Y_j)\right\| \geq t/C\right]$$

We also need the Matrix Bernstein's Inequality:

▶ **Theorem 30** (Matrix Bernstein, [30]). *Consider a finite sequence $\{X_k\}$ of independent, random symmetric matrices with dimension d. Assume that each random matrix satisfies*

$$\mathbb{E}[X_k] = 0 \text{ and } \|X_k\| \leq R \text{ almost surely.}$$

Then, for all $t \geq 0$,

$$\Pr[\|\sum_k X_k\| \geq t] \leq d \cdot \exp\left(\frac{-t^2/2}{\sigma^2 + Rt/3}\right) \text{ where } \sigma^2 := \|\sum_k \mathbb{E}[X_k^2]\|.$$

D Sum-of-Square Proofs

In this section we state some lemmas that can be proved by low-degree SoS proofs. Most of these lemmas can be found in [12] and [9] but we still give the proofs here for completeness.

▶ **Lemma 31.** *[SoS proof for Cauchy-Schwarz] Cauchy-Schwarz inequality can be proved by degree-2 sum of squares proofs,*

$$\left(\sum_{i=1}^n a_i^2\right)\left(\sum_{i=1}^n b_i^2\right) - \left(\sum_i a_i b_i\right)^2 = \sum_{i,j}(a_i b_j - a_j b_i)^2$$

▶ **Lemma 32.** *For any vector x, y, we have that for even number k,*

$$\|x+y\|^k \preceq_k 2^{k-1}(\|x\|^k + \|y\|^k)$$

Proof. Note that it suffices to prove it for one dimensional vector x, y. We prove by induction. For $k = 2$, it just follows Cauchy-Schwarz. Suppose it is true for $k-2$ case, we have

$$(x+y)^k = (x+y)^{k-2}(x+y)^2 \preceq 2^{k-3}(x^{k-2}+y^{k-2}) \cdot 2(x^2+y^2)$$

Note that

$$2(x^k+y^k) - (x^{k-2}+y^{k-2})(x^2+y^2) = (x^2-y^2)^2(x^{k-4}+x^{k-6}y^2+\cdots+y^{k-4}) \succeq 0$$

Combing the two equations above we obtain the desired result. ◀

▶ **Lemma 33.** *Suppose M is $m \times n$ matrix with spectral norm $\|M\|$, then*

$$(x^T My)^2 \preceq_4 \|x\|^2\|y\|^2\|M\|^2$$

Proof. Assume $m \le n$ without loss of generality, and suppose M has singular decomposition $M = U\Sigma V^T$ where $\Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_m)$. Let $z = x^T U$ and $w = V^T y$. Then

$$(x^T My)^2 = \left(\sum_{i=1}^m \sigma_i z_i w_i\right)^2 \preceq_4 \left(\sum_{i=1}^m \sigma_i^2 z_i^2\right)\left(\sum_{i=1}^m w_i^2\right) \le \|M\|^2\|z\|^2\|w\|^2 = \|x\|^2\|y\|^2\|M\|^2$$

◀

▶ **Lemma 34.** *For a nonnegative real number a and a set of polynomial R and positive integer k, if a polynomial $p(x)$ satisfy $p(x) \preceq_{R,k} a^2$, then $p(x) \preceq_{R,k'} a$ for $k' = \max\{k, 2\deg(p)\}$.*

Proof. By a simple manipulation of algebra, we have that

$$p(x) - a \preceq_{R,k} \frac{1}{2a}(p(x)-a)^2 \preceq_{R,k'} 0.$$

◀

Negation-Limited Formulas*

Siyao Guo[1] and Ilan Komargodski[2]

1 **Chinese University of Hong Kong**
Shatin, New Territories, Hong Kong SAR, China
`syguo@cse.cuhk.edu.hk`

2 **Weizmann Institute of Science**
Rehovot, Israel
`ilan.komargodski@weizmann.ac.il`

Abstract

We give an efficient structural decomposition theorem for formulas that depends on their negation complexity and demonstrate its power with the following applications:

- We prove that every formula that contains t negation gates can be shrunk using a random restriction to a formula of size $O(t)$ with the shrinkage exponent of monotone formulas. As a result, the shrinkage exponent of formulas that contain a constant number of negation gates is equal to the shrinkage exponent of monotone formulas.
- We give an efficient transformation of formulas with t negation gates to circuits with $\log t$ negation gates. This transformation provides a generic way to cast results for negation-limited circuits to the setting of negation-limited formulas. For example, using a result of Rossman ([33]), we obtain an average-case lower bound for formulas of polynomial-size on n variables with $n^{1/2-\varepsilon}$ negations.

In addition, we prove a lower bound on the number of negations required to compute one-way permutations by polynomial-size formulas.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases Negation complexity, De Morgan formulas, Shrinkage

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.850

1 Introduction

Understanding the complexity of classical computational models for Boolean functions is the holy grail of theoretical computer science. We focus on one of the simplest and most well studied models known as *Boolean formulas* over the De Morgan basis. Such a formula is a Boolean formula over the basis that includes AND, OR and NOT gates, where the former two are of fan in two. The size of a formula is defined as the number of leaves it contains. A formula is said to be *monotone* if it does not contain any negation gate.

One of the things that makes it so difficult to prove lower bounds on the size of formulas is the presence of negation gates. The best such lower bound known for formulas is almost cubic (see [16, 37]), whereas in the setting of *monotone* formulas, exponential lower bounds

* This work was partially supported by RGC GRF grant CUHK 410111, by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307952, by a grant from the I-CORE Program of the Planning and Budgeting Committee, the Israel Science Foundation, BSF and the Israeli Ministry of Science and Technology.

are known (see [14, 12] and references therein).[1] Bridging this gap is a major challenge since even a super-polynomial lower bound on the size of formulas (for a function that is constructible deterministically in polynomial-time) would separate P from NC^1.

In 1962 Nechiporuk [27] considered the model of formulas with a limited number of negation gates and proved the following classical result: $\lceil n/2 \rceil$ negation gates are sufficient to compute any Boolean function on n variables by a formula, and moreover, any formula can be efficiently transformed into a formula that computes the same function but contains at most $\lceil n/2 \rceil$ negation gates (see [27, 25] and [22]).

In this paper, we continue this line of research and study negation-limited formulas with two main perspectives. The first perspective, which is motivated by bridging the gap between monotone and non-monotone formulas, is that we view negation-limited formulas as a natural extension of monotone formulas and try to extend various complexity properties of monotone formulas to the negation-limited setting. The second perspective, which is motivated by separating the power of circuits and formulas, is that we view negation-limited formulas as a restricted form of negation-limited circuits and ask natural questions about negation-limited circuits in the setting of formulas.

1.1 Our Contributions

The main tool: efficient decomposition of negation-limited formulas

We prove an *efficient* structural decomposition theorem for negation-limited formulas. Specifically, we prove that any function f that can be computed using a formula of size s that contains t negation gates can be decomposed (in polynomial-time) into $T+1$ functions $h, g_1, \ldots, g_T$ such that $f(x) \equiv h(g_1(x), \ldots, g_T(x))$, where $T = O(t)$, h is a read-once formula, each g_i is a monotone function and the total (monotone) formula size of all the g_i's is at most $2s$. That is, roughly speaking, we are able to (efficiently) push all the negation gates to the root of the formula while increasing its size only by a small constant factor (i.e., 2).

This decomposition theorem serves us as the main tool to extend results for *monotone* formulas to *negation-limited* formulas, and to leverage results concerning negation-limited *circuits* to negation-limited *formulas*. We give two applications to demonstrate the usage of our main tool.

Application 1: shrinkage of negation-limited formulas under random restrictions

One of the most successful methods for proving lower bounds in several computational models is the method of shrinkage under random restrictions.[2] This method was invented and first used by Subbotovskaya [35] who proved a lower bound of $\Omega(n^{\Gamma})$ on size of formulas that compute the parity function on n variables, where $\Gamma \geq 1.5$ is referred to as the *shrinkage exponent* of (De Morgan) formulas under random restrictions. Subsequent improvements on

[1] More precisely, there exists an explicit Boolean function on n inputs such that every formula that computes it must be of size $n^{3-o(1)}$ (see [16, 37]). Moreover, there exists an explicit monotone function on n inputs such that every monotone formula that computes it must be of size $2^{\Omega(n/\log n)}$ (see [12]).

[2] A *random restriction* with parameter $p \in [0,1]$ is a vector $\rho \in \{0,1,\star\}^n$ such that with probability p each entry gets the value $\star$ and with probability $1-p$ each entry is assigned, with equal probabilities, to 0 or 1. Given a function $f\colon \{0,1\}^n \to \{0,1\}$ and a random restriction ρ as above, the restricted function $f|_\rho$ is defined in the following way: if $\rho_i \in \{0,1\}$ then the i^{th} input variable of f is fixed to 0 or 1, respectively, and otherwise it is still an unfixed variable. We say that formulas have *shrinkage exponent* Γ if for every function f the expected formula size of $f|_\rho$ is at most $O(p^{\Gamma} \cdot L(f) + 1)$, where $L(f)$ is the formula size of f and the expectation is over the choice of ρ.

the constant Γ led to improved lower bounds on formula size. Impagliazzo and Nisan [20] and Paterson and Zwick [31] proved that $\Gamma \geq 1.55$ and $\Gamma \geq 1.63$, respectively, Håstad [16] proved that $\Gamma \geq 2 - o(1)$ and very recently Tal [37] closed the gap and proved that $\Gamma = 2$. Apart from being useful for proving lower bounds, shrinkage results have a broad scope of applications in other areas including pseudorandomness [19], Fourier concentration [18] and #SAT algorithms [7, 8].

A major open problem (mentioned e.g., in [31, 16, 37]) is to understand what is the shrinkage exponent of *monotone* formulas.[3] We study the related question of understanding the shrinkage exponent of negation-limited formulas and provide a trade-off between the number of negations and the shrinkage exponent. More precisely, we prove that every formula that contains t negation gates can be shrunk using a random restriction to size $O(t)$ with the shrinkage exponent of *monotone* formulas. As a simple instantiation of our result, we get that the shrinkage exponent of formulas that contain a constant number of negation gates is exactly the same as the shrinkage exponent of monotone formulas.

Application 2: efficient transformation from negation-limited formulas to circuits

The decomposition theorem gives a way to efficiently transform formulas with t negations into circuit with roughly $\log t$ negations. Specifically, we prove that a formula of size s that contains t negations can be transformed into a circuit of size $2s + O(t \cdot \log t)$ that contains only $\log t + O(1)$ negation gates.

This transformation also provides a generic way to cast results for negation-limited circuits to the setting of negation-limited formulas. Informally, algorithms for circuits with $\log t$ negation gates will apply for formulas with t negation gates (with almost the same size and depth), and lower bound for circuits with $\log t$ negations will imply lower bounds for formulas with t negation gates (with almost the same size and depth). As an example, this allows us to cast the recent average-case lower bound for mNC^1 [33], lower bounds for several cryptographic primitives [13], and the upper bound on learning circuits with few negations [5] to the setting of negation-limited formulas as we elaborate in Section 4.2.1.

More Results

Lower bound on negation complexity of one-way permutations

We prove a lower bound for implementing one-way permutations by negation-limited formulas. Specifically, we show that every permutation on n bits that can be computed by a formula of size s that contains t negation gates can be inverted (on every image) in time $2^{2t} \cdot s$. This implies, in particular, that every implementation of a one-way permutation as a polynomial-size formula must contain at least $\omega(\log n)$ negation gates. As a comparison, Guo et al. [13] left open the question of whether one-way permutations are computable by circuits that contain a single negation gate.

Upper bound on the total influence

Total influence has many applications in various areas of theoretical computer science. Most relevant to our context, it serves as the main tool in recent studies of negation-limited circuits in computational learning [5] and cryptography [13].

[3] It is conjectured that the shrinkage exponent of monotone formulas is equal to 3.27, the shrinkage exponent of read-once formulas (see Conjecture 3 in [31]).

The literature on negation complexity defines a measure, $a(\cdot)$, called "alternation complexity" which denotes the maximal number of times a function $f\colon \{0,1\}^n \to \{0,1\}$ changes its value along a chain (i.e., a monotone sequence of strings) starting at 0^n and ending at 1^n. Blais et al. [5] proved (using their inefficient decomposition theorem) that for any function f it holds that $\mathsf{Inf}(f) \le O(a(f) \cdot \sqrt{n})$. We give a simple direct probabilistic argument for this fact.

1.2 Related Work

An inefficient decomposition theorem for negation-limited *circuits* into monotone circuits explicitly appeared in [5]. They proved that any function f that can be computed using a circuit with t negations can be decomposed into $T+1$ functions $h, g_1, \ldots, g_T$ such that $f(x) \equiv h(g_1(x), \ldots, g_T(x))$, where $T = O(2^t)$, h is either the parity function or its negation and each g_i is a monotone function.[4] An efficient version of this decomposition theorem (with related parameters) appeared explicitly in [13] and implicitly in [2, 33].

Besides the above, the power of negations in different models has been studied in many works including [24, 27, 10, 34, 32, 39, 4, 36, 2, 25, 21]. For more information on negations in complexity theory we refer to Jukna's book [22, §10] and references therein.

1.3 Overview of Our Techniques

In this section we present a high-level overview of the techniques used to obtain some of our results.

Efficient decomposition of negation-limited formulas

Using the theorem of Nechiporuk [27] it is quite straightforward to cast the decomposition theorem of [5] to the setting of negation-limited formulas which results in the same statement except that $T = O(t)$ (rather than $T = O(2^t)$). More precisely, it gives that and function f can be rewritten as $f(x) \equiv h(g_1(x), \ldots, g_T(x))$, where $T = O(t)$, h is either the parity function or its negation and each g_i is a monotone function. Unfortunately, such a decomposition is not enough for us since it is inefficient, in particular, it does not preserve the size or depth of the original formula. Note that the efficiency of the decomposition was mostly not an issue in [5, 13], whereas for us it is crucial since, for example, shrinkage is a combinatorial property that general circuits do not have (unlike formulas).

To overcome this we prove an *efficient* version in which the resulting formula has almost the same size and depth as the original one.[5] Technically, our decomposition is more involved than the inefficient version and is influenced by ideas and techniques used in recent papers on De Morgan formulas [19, 23, 37].

As an applications of this theorem, we prove the shrinkage result and the transformation from negation-limited formulas to circuits. The shrinkage theorem relies on two properties of the decomposition: it does not introduce much overhead in the formula size and the g_i's are monotone, and thus, shrink as well as monotone formulas. To get the transformation result,

[4] We refer to this decomposition as "inefficient" since the decomposed monotone components (i.e., the g_i's) may have exponential size.

[5] We note that our transformation is efficient in a strong sense: (1) it can be implemented in polynomial-time in the size of the input formula, and (2) it results with a formula of polynomial-size (close to the size of the input formula).

we use our efficient decomposition theorem for formulas, view h as a circuit on t inputs, and apply Fischer's transformation [10] (see also [4]) to implement h with $\lceil \log_2(t+1) \rceil$ negations.

Negation complexity of one-way permutations

Our lower bound on the number of negations required to compute one-way permutations relies crucially on the fact that the fan-out of formulas is 1. We take advantage of this fact together with Talagrand's inequality [38] in a way that might be of independent interest. We emphasize that previously it was known that one-way permutations cannot be computed by a monotone circuit.

1.4 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we provide an overview of the notation, definitions, and tools underlying our proofs. In Section 3 we present our central tool: the decomposition theorem for negation-limited formulas. In Sections 4.1 to 4.3 we give the statements of the shrinkage result, the transformation from negation-limited formulas to negation-limited circuits, the lower bound for one-way permutations, and the influence bound for negation-limited formulas.

2 Preliminaries

In this section we present the notation and basic definitions that are used in this work. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. Unless explicitly stated, we assume that the underlying probability distribution in our equations is the uniform distribution over the appropriate set. Further, we let $\mathcal{U}_\ell$ denote the uniform distribution over $\{0,1\}^\ell$. We use $\log x$ to denote a logarithm in base 2. We denote by $\mathsf{wt}(x)$ the Hamming weight of a string $x \in \{0,1\}^n$ (i.e., the number of 1's in the string).

Boolean Formulas

We recall some standard definitions and notation regarding formulas. We refer to [22] for a thorough introduction. We consider formulas over the De Morgan basis $B_{\mathsf{DM}} = \{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$, where the AND and OR gates are of fan-in two. Whenever we refer to formulas we actually refer to De Morgan formulas.

A Boolean formula is a Boolean circuit whose fan-out is at most one. A De Morgan formula is represented by a tree such that every leaf is labeled by an input variable and every internal node is labeled by an operation from B_2. A formula is said to compute a function $f\colon \{0,1\}^n \to \{0,1\}$ if on input $x \in \{0,1\}^n$ it outputs $f(x)$. The computation is done in the natural way from the leaves to the root. The *size* of a formula F, denoted by $L(F)$, is defined as the number of leaves it contains. For a function f, we denote by $L(f)$ the size of the smallest formula that computes the function f.

A formula is called *read-once* if every input variable labels at most one leaf. A formula F that does not contain negation gates is called a *monotone formula.* We say that a formula F is *anti-monotone* if F is the negation of a monotone formula.

Consider a formula F. Let q be a node in F (q can be either an internal node or a leaf). We refer to the tree rooted at q as a *subformula* of F or a subtree of F.

Let $f\colon \{0,1\}^n \to \{0,1\}^m$ be a Boolean multi-bit output function. Such a function can be computed by m formulas $F_1,\dots,F_m$ such that F_i computes the i^{th} output bit of f. The *size* of the formula that computes f is the sum of the sizes of $F_1,\dots,F_m$. Moreover, the number of negation gates in f is the sum of the number of negation gates in $F_1,\dots,F_m$.

Decrease, Alternating and Inversion Complexity

For two strings $x,y \in \{0,1\}^n$, we write $x \preceq y$ if $x_i \le y_i$ for every $i \in [n]$. If $x \preceq y$ and $x \neq y$, then we write $x \prec y$. A *chain* $\mathcal{X} = (x^1,\dots,x^t)$ is a monotone sequence of strings over $\{0,1\}^n$, i.e., $x^i \preceq x^{i+1}$ for every $i \in [t]$. We say i is a *jump-down position* of f along a chain $\mathcal{X} = (x^1,x^2,\dots,x^t)$ if $f(x^i) = 1$ and $f(x^{i+1}) = 0$. We let $d(f,\mathcal{X})$ be the number of all jump-down positions of f on chain $\mathcal{X}$ We say a chain $\mathcal{X} = (x^1,x^2,\dots,x^t)$ is k-*alternating* with respect to a function f if there exist indexes $i_0 < i_1 < \ldots < i_k$ such that $f(x^{i_j}) \neq f(x^{i_{j+1}})$, for every $j \in [0,k-1]$. We let $a(f,\mathcal{X})$ be the size of the largest set of indexes satisfying this condition. The *decrease* of a Boolean function f is given by $d(f) \overset{\text{def}}{=} \max_{\mathcal{X}} d(f,\mathcal{X})$ and the alternating complexity of a Boolean function f is given by $a(f) \overset{\text{def}}{=} \max_{\mathcal{X}} a(f,\mathcal{X})$, where $\mathcal{X}$ is a chain over $\{0,1\}^n$. Note that $a(f) \le 2d(f)+1$.

For a Boolean function f, we define the *inversion complexity* of f, denoted by $I(f)$, as the minimum number of NOT gates in any formula that computes f. The relation between the inversion complexity and decrease complexity is stated in the following theorem.

▶ **Theorem 1** ([27, 25]). *For every Boolean function f it holds that*

$$I(f) = d(f),$$

where $I(f)$ is the inversion complexity of f and $d(f)$ is the decrease of f.

Fourier Basis and Influence

For each $S \subseteq [n]$, define $\chi_S : \{0,1\}^n \to \{-1,1\}$ as $\chi_S(x) = \prod_{i\in S}(-1)^{x_i}$. It is well known that the set $\{\chi_S\}_{S\subseteq[n]}$ is an orthonormal basis (called the Fourier basis) for the space of all functions $f\colon \{0,1\}^n \to \mathbb{R}$. It follows that every function $f\colon \{0,1\}^n \to \mathbb{R}$ can be represented as

$$f(x) = \sum_{S\subseteq[n]} \widehat{f}(S)\chi_S(x),$$

where $\widehat{f}\colon \{0,1\}^n \to \mathbb{R}$, and $\widehat{f}(S) \overset{\text{def}}{=} \mathbb{E}_x[(-1)^{\sum_{i\in S} x_i + f(x)}]$ is called the Fourier coefficient of f at $S \subseteq [n]$. We use $\mathsf{Inf}_i(f)$ to denote the *influence* of the i-th input variable on f, i.e.,

$$\mathsf{Inf}_i(f) \overset{\text{def}}{=} \Pr_x[f(x) \neq f(x^{\oplus i})],$$

where $x^{\oplus i}$ denotes the string obtained from x by flipping its i-th coordinate. The *influence* of f (also known as *average-sensitivity*) is defined as $\mathsf{Inf}(f) \overset{\text{def}}{=} \sum_{i\in[n]} \mathsf{Inf}_i(f)$. We refer to O'Donnell's book [30] for an introduction to Fourier analysis.

Some of our proofs rely on the following inequality for monotone Boolean functions.

▶ **Proposition 2** (Talagrand [38]). *For any pair of monotone Boolean functions $f,g\colon \{0,1\}^n \to \{0,1\}$, it holds that*

$$\Pr_x[f(x) = 1 \wedge g(x) = 1] \ge \Pr_x[f(x) = 1] \cdot \Pr_x[g(x) = 1] + \psi\Big(\sum_{i\in[n]} \mathsf{Inf}_i(f)\cdot \mathsf{Inf}_i(g)\Big),$$

where $\psi(x) \overset{def}{=} c \cdot x/\log(e/x)$, e is the base of the natural logarithm and $c > 0$ is a fixed constant independent of n.

One-Way Functions and One-Way Permutations

We say that a function $f\colon \{0,1\}^n \to \{0,1\}^m$ is an (s,ε)-secure *one-way function* (OWF) if for every circuit C of size at most s,

$$\Pr_{x\leftarrow\{0,1\}^n,\, y=f(x)}[C(y) \in f^{-1}(y)] \le \varepsilon.$$

If $m = n$, we say that f is *length-preserving*. If f is an (s,ε)-secure one-way function that is lengh-preserving and one-to-one, we say that f is an (s,ε)-secure *one-way permutation* (OWP).

3 Efficient Decomposition for Negation-Limited Formulas

In this section we present our main tool, an efficient structural decomposition theorem for formulas which, intuitively, pushes all negation gates to the root of the formula.

▶ **Theorem 3.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a formula F of size s containing $t > 0$ negation gates. Then, there exist $T \le 15(t+1)$ functions $g_1,\dots,g_T\colon \{0,1\}^n \to \{0,1\}$ and a function $h : \{0,1\}^T \to \{0,1\}$ such that $f(x) = h(g_1(x),\dots,g_T(x))$, h is computable by a read-once* formula *and $g_1,\dots g_T$ are computable by monotone formulas of total size at most $2s$.*

We first need the following claim that states that any formula that has t negation gates can be decomposed into $2(t+1)$ subformulas such that each of them is monotone or anti-monotone (i.e., either it has zero negations or it has one negation in the root). Moreover, each such subformula has at most two "special" children which are subformulas by themselves. We note that the proof of Theorem 3 draws ideas from a proof of a different decomposition theorem used by Tal [37] which, in turn, is partially built on ideas that were used before in [19] and then in [23]. However, since the properties of our decomposition are very different, we cannot use the other theorems as a black-box.

▶ **Claim 4.** *Let F be a formula of size s that contains $t > 0$ negations. Then, F can be decomposed into at most $2(t+1)$ subformulas of total size s, such that (1) each subformula has at most one negation gate in its root, and (2) each subformula has at most two "special" children which are other subformulas.*

Proof. Execute the following step t times: let $g_1,\dots,g_s$ be the nodes of the formula F sorted by their distance from the root g_s. For any $i = 1,\dots s$ if $g_i = \mathsf{NOT}$ we set F_i to be the subformula rooted at g_i and set $F = F \setminus F_i$. This process results with $T = t+1$ subformulas $F_1,\dots,F_T$ whose total size is s and each is either monotone (i.e., does not include a NOT gate) or includes one NOT gate located at its root (i.e., it is anti-monotone). This process results with at most $t+1$ subformulas.

For each subformula F_i with more than two subformula children, find a subformula F_i' of F_i with exactly two subformula children, and divide F_i into F_i' and $F_i \setminus F_i'$. Note that $F_i \setminus F_i'$ now has one fewer subformula children. Continue doing this until all subformulas have at most two subformula children. This process results with the desired number of subformulas, $2(t+1)$, since the above process can happen at most the original number of subformulas. ◀

Proof of Theorem 3. Let F be as in the lemma. Apply the decomposition from Claim 4 on F to get the subformulas $F_1,\dots,F_{T'}$, where $T' = 2(t+1)$. We show by induction on T' that one can construct a read-once formula H of size $T \le 7T'$ and T monotone formulas

$G_1, \dots, G_T$ of size s such that $F(x) = H(G_1(x), \dots, G_T(x))$. For $t = 0$ (and $T' = 1$) the statement holds trivially.

Assume that the root of the formula F is a node in the subformula F_1, and that the subformula F_1 has two subformula children F_2 and F_3. (The case in which F_1 has only one subformula child is handled similarly). Denote by $k_2^{(1)}, k_3^{(1)} \in F_1$, $k_1^{(2)} \in F_2$ and $k_1^{(3)} \in F_3$ the nodes such that $k_1^{(2)}$ and $k_1^{(3)}$ are the roots of F_2 and F_3, respectively, $k_2^{(1)}$ is the father of $k_1^{(2)}$, and $k_3^{(1)}$ is the father of $k_1^{(3)}$. Disconnect F_2 and F_3 from F_1 and add two new leaves labeled by z_2 and z_3 to F_1 as a child of $k_2^{(1)}$ and $k_3^{(1)}$, respectively.

Call the formula F_1 with the two new leaves F'. Notice that by Claim 4, F' is either monotone or anti-monotone, namely a negation of a monotone function. We prove the case when F' is anti-monotone and the argument for monotone case is similar. Let F_1' be the minimal subformula of F' that contain both z_2 and z_3 and let F_2' and F_3' be the corresponding subformulas such that $F_1' = F_2' \ \mathsf{gate}\ F_3'$, where $\mathsf{gate} \in \{\mathsf{AND}, \mathsf{OR}\}$, and F_2' contains z_2 (but not z_3) and F_3' contains z_3 (but not z_2). We will construct a formula which is equivalent to F_1'.

We observe that $F_2' = F_2'|_{z_2=0}\ \mathsf{OR}\ (F_2'|_{z_2=1}\ \mathsf{AND}\ z_2)$. This is true since F_2' is monotone (i.e., does not contain any negation gates). Similarly, $F_3' = F_3'|_{z_3=0}\ \mathsf{OR}\ (F_3'|_{z_3=1}\ \mathsf{AND}\ z_3)$. Thus,

$$F_1' = (F_2'|_{z_2=0}\ \mathsf{OR}\ (F_2'|_{z_2=1}\ \mathsf{AND}\ z_2))\ \mathsf{gate}\ (F_3'|_{z_3=0}\ \mathsf{OR}\ (F_3'|_{z_3=1}\ \mathsf{AND}\ z_3)).$$

Replacing F_1' with a new leaf z (where z is a new special variable) we have (by a similar argument) that $F_1 = F_1|_{z=1}\ \mathsf{OR}\ (F_1|_{z=0}\ \mathsf{AND}\ z)$ (this follows by the anti-monotonicity of F_1). By expanding according to the definition of z we get that

$$\begin{aligned} F_1 = F_1|_{z=1}\ \mathsf{OR}\ (F_1|_{z=0}\ \mathsf{AND}\ ((F_2'|_{z_2=0}\ \mathsf{OR}\ (F_2'|_{z_2=1}\ \mathsf{AND}\ z_2))\ \mathsf{gate}\\ (F_3'|_{z_3=0}\ \mathsf{OR}\ (F_3'|_{z_3=1}\ \mathsf{AND}\ z_3)))). \end{aligned}$$

Now, we observe that the right hand side can be rewritten as $F''(G_1, \dots, G_6, z_2, z_3)$, where F'' is read-once and $G_1, \dots, G_6$ are formulas of size at most s (defined over the same set of variables as the initial F).

Let t_2 and t_3 be the number of subformulas which are descendants of F_2 and F_3 in the formula decomposition, respectively. By induction the subformula of F rooted at $k_1^{(2)}$ is equivalent to $F_2'(G_1^{(2)}(x), \dots, G_{6t_2}^{(2)}(x))$, where F_2' is read-once and $G_i^{(2)}$ is of size at most s. Similarly, the subformula of F rooted at $k_1^{(3)}$ is equivalent to $F_3'(G_1^{(3)}(x), \dots, G_{6t_3}^{(3)}(x))$, where F_3' is read-once and $G_i^{(3)}$ is of size at most s. Thus,

$$F(x) = F''(G_1(x), \dots, G_6(x), F_2'(G_1^{(2)}(x), \dots, G_{6t_2}^{(2)}(x)), G_1^{(3)}(x), \dots, G_{6t_3}^{(3)}(x))).$$

By rearranging the right hand size we get a read-once formula of size $T \leq 6 + 6t_2 + 6t_3 \leq 7T'$ and T monotone subformulas each of size at most s such that their composition is equivalent to F. To see that the total size of the subformulas is bounded by $2s$ notice that every subformula was duplicated at most once. ◀

4 The Complexity of Negation-Limited Formulas

4.1 Shrinkage under Random Restrictions

A well known property of formulas is called *shrinkage*. We begin with several definitions. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. A restriction ρ is a vector of length n of elements from $\{0,1,\star\}$. We denote by $f|_\rho$ the function f restricted according to ρ in the following sense: if $\rho_i = \star$ then the i-th input bit of f is unassigned and otherwise the i-th input bit of f is assigned to be ρ_i. A p-random restriction is a restriction as above that is sampled as follows. For every $i \in [n]$, independently with probability p set $\rho_i = \star$ and with probability $\frac{1-p}{2}$ set ρ_i to be 0 and 1, respectively. We denote this distribution of restrictions by $\mathcal{R}_p$.

▶ **Definition 5** (Shrinkage exponent). Let $\mathcal{F}$ be a class of formulas. The shrinkage exponent of $\mathcal{F}$ is said to be Γ if for any $F \in \mathcal{F}$

$$\mathop{\mathbb{E}}_{\rho \leftarrow \mathcal{R}_p}[L(F|_\rho)] \leq O\left(p^\Gamma \cdot L(F) + 1\right).$$

Denote by $\Gamma, \Gamma_0, \Gamma^*$ the shrinkage exponent of (De Morgan) formulas, monotone formulas and read-once formulas, respectively. Denote by Γ_t the shrinkage exponent of formulas that contain at most t negation gates. It is known that (1) $\Gamma = 2$ [16, 37], (2) $\Gamma^* = \log_{\sqrt{5}-1} 2 \approx 3.27$ [9, 17], and (3) for every $t \geq 0$ it holds that $\Gamma^* \geq \Gamma_t \geq \Gamma_{t+1} \geq \Gamma = 2$. Figuring out the value of Γ_0, the shrinkage exponent of *monotone* Boolean formulas, is a major open problem [31, 16, 37].

Our main theorem of this section is a trade-off between the number of negations in the formula and its shrinkage exponent. In particular, we get that the shrinkage exponent of formulas that contain a constant number of negation gates is equal to Γ_0.

▶ **Theorem 6.** *Let F be a formula that contains $t > 0$ negation gates. It holds that*

$$\mathop{\mathbb{E}}_{\rho \leftarrow \mathcal{R}_p}[L(F|_\rho)] \leq O\left(p^{\Gamma_0} \cdot L(F) + t\right).$$

Proof of Theorem 6. Given a formula F we decompose it using Theorem 3 to get $H, G_1, \ldots, G_T$, where $T \leq 15(t+1)$, $\sum_{i=1}^{T} L(G_i) \leq 2 \cdot L(F)$ and $F(x) = H(G_1(x), \ldots, G_T(x))$. Clearly we have that the formula size of F is at most the sum of the sizes of the G_i's. Namely,

$$L(F) \leq \sum_{i=1}^{T} L(G_i) \leq 2 \cdot L(F), \tag{1}$$

where the second inequality is true by the guarantee of the decomposition from Theorem 3. Let $\rho \leftarrow \mathcal{R}_p$ be a random restriction. For each $i \in [T]$ since G_i is monotone, we have that $\mathbb{E}_\rho[L(G_i|_\rho)] \leq O(p^{\Gamma_0} \cdot L(G_i) + 1)$. Thus, the expected size of $L(F)$ after applying ρ is

$$\begin{aligned}
\mathop{\mathbb{E}}_{\rho}[L(F|_\rho)] \leq & \sum_{i=1}^{T} \mathop{\mathbb{E}}_{\rho}\left[L\left(G_i|_\rho\right)\right] && \text{(Linearity of expectation)} \\
\leq & \sum_{i=1}^{T} O\left(p^{\Gamma_0} \cdot L\left(G_i\right) + 1\right) && \text{(Each } G_i \text{ is monotone)} \\
\leq & O\left(p^{\Gamma_0} \cdot L(F) + t\right). && \text{(Equation (1))}
\end{aligned}$$

◀

Notice that when $t = O(1)$ we get that $\mathbb{E}_\rho[L(F)|_\rho] \leq O(p^{\Gamma_0} \cdot L(F) + 1)$ which means that the shrinkage exponent of such formulas is exactly equal to the shrinkage exponent of monotone formulas. More generally, Theorem 6 implies that every formula F that contains $t > 0$ negation gates can be shrunk in two steps of random restrictions such that in the first step the formula F shrinks to size $O(t)$ as monotone formulas shrink (i.e., with Γ_0 as the shrinkage exponent) and in the second step the formula (of size $O(t)$) shrinks as formulas shrink (with Γ as the shrinkage exponent). To be more precise, F can be restricted with a random restriction $\rho_1 \leftarrow \mathcal{R}_{p_1}$, where $p_1 = \sqrt[\Gamma_0]{t/L(F)}$, to get a formula F_1 of size $O(t)$ and then it can be restricted with a random restriction $\rho_2 \leftarrow \mathcal{R}_p$ for any p to get a formula F_2 of size $O(p^{\Gamma} \cdot t + 1)$. In the following corollary we state a shrinkage result parameterized by t, the number of negations, p, the restriction parameter, and $L(F)$, the formula size.

▶ **Corollary 7.** *Let F be a formula that contains $t = t(L(F)) > 0$ negations and let $c > 0$ be a constant. Then, for $p \geq \sqrt[\Gamma_0]{(c \cdot t)/L(F)}$, it holds that*

$$\mathop{\mathbb{E}}_{\rho \leftarrow \mathcal{R}_p} [L(F|_\rho)] \leq O\left(p^{\Gamma_0} \cdot L(F)\right).$$

4.2 Efficient Transformation from Negation-Limited Formulas to Circuits

In this section we show that negation-limited formulas can be transformed into negation-limited circuits with exponentially smaller number of negations with almost linear blowup in the size and depth. An inefficient transformation was previously known due to the theorems of Markov [24] and Nechiporuk [27].[6]

▶ **Theorem 8.** *Let $F\colon \{0,1\}^n \to \{0,1\}$ be a formula of size s and depth d and t negations, then there is a circuit C of size s', depth d' and t' negations computing F such that $s' = 2s + O(t \log t)$, $d' = d + O(\log t)$ and $t' = \log t + O(1)$.*

Fischer's theorem [10] can efficiently transform negation-limited formulas with t negations into negation-limited circuits with $\log n$ negations. Our theorem combines Fischer's theorem and our decomposition theorem (Theorem 3) to efficiently transform the negation-limited formulas with t negations into negation-limited circuits with $\log t$ negations.

Proof. Our decomposition theorem (Theorem 3) states that the function f computed by F can be written as $f(x) = h(g_1(x), \ldots, g_T(x))$ where $T \leq 15(t+1)$, $g_1, \ldots, g_T\colon \{0,1\}^n \to \{0,1\}$ are computable by monotone formulas of total size at most $2s$ (also depth at most d) and $h\colon \{0,1\}^T \to \{0,1\}$ is computable by a read-once formula. We use the efficient version of Markov's theorem to get a circuit with few negations that compute h.

▶ **Proposition 9** ([10, 4]). *If a function on n variables can be computed by circuit over a basis that includes* AND, OR *and* NOT *gates of size s and depth d, then it can be computed by a circuit of size at most $2s + O(n \log n)$ and depth $d + O(\log n)$ using at most $\lceil \log (n+1) \rceil$ negations.*

The read-once formula computing h has input size T so that size is at most T and depth is at most $\log T$. By the above theorem, we conclude that h can be computed by a circuit of size at most $2T + O(T \log T) = O(t \log t)$ and depth $O(\log T)$ using at most $\lceil \log (T+1) \rceil = \log t + O(1)$ negations. It is easy to see we can compose the circuit for h with formulas for $g_1, \ldots, g_T$ to compute f. Since $g_1, \ldots, g_T$ are computable by monotone formulas of total size at most $2s$ and depth at most d, we can further conclude that f are computable by a circuit of size at most $2s + O(t \log t)$, depth $d + O(\log t)$ and $O(\log t)$ negations. ◀

[6] By the theorem of Nechiporuk [27], the decrease of a function computable by a formula with t negations is t. Then, by the theorem of Markov [24], any function with decrease t is computable by a circuit with $\lceil \log (t+1) \rceil$ negations. The size of the resulting circuit, however, is unbounded (i.e., it can be exponential in the number of inputs).

4.2.1 Applications

In this section we exemplify the usefulness of Theorem 8.

Average-case lower bounds for negation-limited formulas

An average-case computation (a.k.a. approximate computation) of a function $f\colon \{0,1\}^n \to \{0,1\}$ is a computation that is required to agree with f only on a $1/2+\delta$ fraction of the inputs. Besides being interesting in their own right, average-case lower bounds (a.k.a. correlation bounds) have proved useful in many fields of complexity theory, such as derandomization (e.g., [28, 29]).

Recently, Rossman [33] proved the first average-case lower bound for mNC^1, the class of polynomial-size logarithmic-depth monotone circuits, or equivalently, polynomial-size monotone formulas. More precisely, for every $\varepsilon > 0$, Rossman gives an explicit monotone function on n variables which is $(1/2 + n^{-1/2+\varepsilon})$-hard to approximate in mNC^1 under the uniform distribution. His bound for mNC^1 extends to circuits in NC^1 with at most $(1/2 - \varepsilon)\log n$ negations. Using Theorem 8 and [33], we get the following corollary.

▶ **Corollary 10.** *For every $\varepsilon > 0$, there is an explicit function $f\colon \{0,1\}^n \to \{0,1\}$ such that for every polynomial-size formula F with $n^{1/2-\varepsilon}$ negations, it holds that $\Pr_{x\leftarrow\{0,1\}^n}[F(x) = f(x)] \leq 1/2 + o(1)$.*

We remark that Theorem 10 crucially relies on that the transformation in Theorem 8 is efficient.

Cryptography in negation-limited formulas

One of the goals of cryptography is to study how simple cryptographic primitives can be, where simplicity can be measured by e.g., the required assumptions, the circuit depth and more. Recently, Guo et al. [13] (following on [11]) proved lower bounds on the number of negations required to represent many cryptographic primitives as circuits. The simplicity of a cryptographic primitive can also be measured by the simplicity of the model in which it can be implemented (see e.g., [3] and concrete examples in [15, 26]). Using Theorem 8, one can easily cast some of the results of [13] to the setting of negation-limited formulas and obtain *exponentially higher* lower bounds on several primitives including pseudorandom functions, hardcore predicates and extractors. (We refer the reader to [13] for the relevant notation and definitions.)

▶ **Corollary 11.** *If $f\colon \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}$ is a $(\mathsf{poly}(n), 1/3)$-secure pseudorandom function, then any Boolean formula computing f contains at least $\Omega(n)$ negation gates.*

▶ **Corollary 12.** *Assume that there exists a family $f = \{f_n\}_{n\in\mathbb{N}}$ of $(\mathsf{poly}(n), n^{-\omega(1)})$-secure one-way functions, where each $f_n\colon \{0,1\}^n \to \{0,1\}^n$. Then, for every $\varepsilon > 0$, there exists a family $g_\varepsilon = \{g_n\}_{n\in\mathbb{N}}$ of (length-preserving) $(\mathsf{poly}(n), n^{-\omega(1)})$-secure one-way functions for which the following holds. If $h = \{h_n\}_{n\in\mathbb{N}}$ is a $(\mathsf{poly}(n), n^{-\omega(1)})$-secure hardcore predicate for g_ε, then for every n sufficiently large, any formula computing h_n contains at least $\Omega(n^{1/2-\varepsilon})$ negations.*

▶ **Corollary 13.** *Let $0 < \alpha < 1/2$ be a constant, and $m = m(n) \geq 100$. Further, suppose that $\mathcal{H} \subseteq \{h \mid h\colon \{0,1\}^n \to \{0,1\}^m\}$ is a family of functions such that each output bit $h_i\colon \{0,1\}^n \to \{0,1\}$ of a function $h \in \mathcal{H}$ is computed by a formula and the total number of negations of a function $h \in \mathcal{H}$ is at most t. If $\mathcal{H}$ is an $(n^{\frac{1}{2}-\alpha}, 1/2)$-extractor, then $t = \Omega(n^\alpha)$.*[7]

[7] We remark that the above bound can be further improved to $\Omega(m \cdot n^\alpha)$ if one combines the proof of

Uniform-distribution learnability of negation-limited formulas

Monotone functions are known to be somewhat efficiently learnable with high accuracy given uniformly distributed examples. Namely, Bshouty and Tamon [6] showed that any monotone Boolean function on n variables can be learned from uniformly distributed examples to error ε in time $O(n^{\sqrt{n}/\varepsilon})$. Recently, Blais et al. [5] studied the question of learning negation-limited *circuits*. They showed that any function on n variables that can be computed by a circuit with t negations can be learned from uniformly distributed examples to error ε in time $n^{O(2^t \cdot \sqrt{n}/\varepsilon)}$. Using Theorem 8 we obtain the following corollary.

▶ **Corollary 14.** *There is a uniform-distribution learning algorithm that learns any Boolean function f on n variables that can be computed by a formula with t negations to error ε in time $n^{O(t \cdot \sqrt{n}/\varepsilon)}$.*

4.3 One-Way Functions and Permutations in Negation-Limited Formulas

In this section we study the negation-limited complexity of one-way functions and one-way permutations in the model of Boolean formulas. We start with a simple observation (see Observation 15) that if one-way functions in NC^1 exist, then there exist one-way functions that can be computed by monotone logarithmic-depth formulas. Then, in Theorem 16, we show that any one-way permutation is not computable by a formula that has $O(\log n)$ negations.

▶ **Observation 15.** *Assume that there is a one-way function in NC^1. Then, there is a one-way function computable by a logarithmic-depth monotone formula.*

Proof. Recall the transformation of Goldreich and Izsak [11] that transformed every one-way function into a monotone one-way function. Let C be a circuit that computes a one-way function and let C' be a circuit obtained from C by pushing all negation gates to the leaves and replacing negated variables by auxiliary variables, namely, $C(x) = C'(x, \bar{x})$, where $\bar{x}_i = \neg x_i$. Let $\mathsf{Th}_k : \{0,1\}^n \to \{0,1\}$ be a function such that $\mathsf{Th}_k(x) = 1$ if and only if the hamming weight of x is at least k. Notice that for any x of hamming weight k it holds that $\neg x_i = \mathsf{Th}_k(x \wedge 1^{i-1}01^{n-i})$. Therefore, $N(x) = (\mathsf{Th}_k(x \wedge 01^{n-1}), \ldots, \mathsf{Th}_k(x \wedge 1^{n-1}0))$ and we get that

$$C''(x) = (\mathsf{Th}_{n/2} \wedge C'(x, N(x))) \vee \mathsf{Th}_{(n/2)+1}(x)$$

is a monotone function which is efficiently computable and weakly one-way. Then, applying the standard hardness amplification process they obtain a one-way function (we refer to [11] for the exact detail).

We observe that if we start with a one-way functions in NC^1, then the reduction of [11] results with a monotone one-way function which is in NC^1. Then, we use the standard transformation from circuits in NC^1 to formulas. Since this transformation preserves monotonicity and depth, we complete the proof. We note that the above transformation of [11] uses threshold functions which are computable by (uniform) formula of logarithmic depth (using sorting networks [1]). ◀

[13] (with slight modifications) and the total influence upper bound given in Theorem 17.

▶ **Theorem 16.** *Let $f\colon \{0,1\}^n \to \{0,1\}^n$ be a permutation. If f is computable by a formula of size s that contains t negations, then there exists a deterministic algorithm whose running time is $2^{2t} \cdot s$ such that given as input any $y = f(x)$ outputs x. In particular, if $s \in \mathsf{poly}(n)$ and $t = O(\log n)$, then the algorithm runs in polynomial-time.*

Proof. Let $f_i\colon \{0,1\}^n \to \{0,1\}$ be the Boolean function corresponding to the i-th output bit of f and $F_i\colon \{0,1\}^n \to \{0,1\}$ be a formula computing f_i. Let $S = \{i \in [n] \mid F_i \text{ is monotone}\}$, i.e., the collection of indices $i \in [n]$ for which F_i contains no negations. Since f has $t < n$ negations, we obtain $|S| \geq n - t$. Let $S_1 = \{i \in S \mid \exists j \in [n], \forall x \in \{0,1\}^n : \ F_i(x) = x_j\}$, i.e., the collection of indices $i \in S$ for which F_i is a dictator function. Let $I_i = \{j \in [n] \mid \mathsf{Inf}_j(f_i) \neq 0\}$, i.e., the set of input variables that f_i depends on.

Consider functions f_ℓ and f_k, where $\ell \neq k \in S$. By Talagrand's inequality (Proposition 2),

$$\Pr_x[f_\ell(x) = 1 \wedge f_k(x) = 1] \ \geq \ \Pr_x[f_\ell(x) = 1] \cdot \Pr_x[f_k(x) = 1] + \psi\Big(\sum_{i\in[n]} \mathsf{Inf}_i(f_\ell) \cdot \mathsf{Inf}_i(f_k)\Big).$$

Since f is a permutation, $\Pr_x[f_\ell(x) = 1 \wedge f_k(x) = 1] = 1/4$ and $\Pr x[f_\ell(x) = 1] = \Pr_x[f_k(x) = 1] = 1/2$. Thus, since f_ℓ and f_k are monotone and using the definition of ψ, we get that

$$\sum_{i\in[n]} \mathsf{Inf}_i(f_\ell) \cdot \mathsf{Inf}_i(f_k) = 0.$$

Therefore, $I_\ell \cap I_k = \emptyset$, i.e., f_ℓ and f_k depend on a disjoint set of input variables. Since the above holds for every pair ℓ, k such that $\ell \neq k \in S$, we obtain

$$n \geq \Big|\bigcup_{i\in S} I_i\Big| = \sum_{i\in S} |I_i| = \sum_{i\in S_1} |I_i| + \sum_{i\in S\setminus S_1} |I_i| \,. \tag{2}$$

For $i \in S \setminus S_1$, since the function f_i is non-constant we have that $|I_i| \geq 2$. Plugging this into Equation (2), we obtain

$$n \geq \sum_{i\in S_1} 1 + \sum_{i\in S\setminus S_1} 2 = 2\,|S| - |S_1| \geq 2(n-t) - |S_1|\,,$$

which implies that $|S_1| \geq n - 2t$.

Given $y = f(x)$, we can invert y and find $x' = x$ using following algorithm:

1. For every $i \in S_1$, we set x'_j to be y_i where j is the only element in the set I_i.
2. Go over all possible assignments on the unassigned variables in x' until $f(x') = y$,
3. Output x'.

After the first step, $|S_1| \geq n - 2t$ variables are assigned correctly. The number of unassigned variables is at most $2t$, so that we can try all possible assignments on the remaining unassigned variables in time $2^{2t} \cdot s$, where s is the evaluation time of the permutation. If $s \in \mathsf{poly}(n)$ and $t = O(\log n)$, we get that the above algorithm runs in polynomial-time. ◀

4.4 Total Influence of Negation-Limited Formulas

In this section we prove a general connection between total influence and negation complexity of Boolean functions.

▶ **Theorem 17.** *For any function $f\colon \{0,1\}^n \to \{0,1\}$, it holds that $\mathsf{Inf}(f) \leq O\left(a(f) \cdot \sqrt{n}\right)$, where $\mathsf{Inf}(f)$ is the total influence of f and $a(f)$ is the alternating complexity of f.*

In particular, if f can be computed by a formula with t negations, then $\mathsf{Inf}(f) \leq O\left(t \cdot \sqrt{n}\right)$.

A proof of Theorem 17 was given (somewhat implicitly in this full generality) in [5] using their (inefficient) decomposition theorem. We show a direct probabilistic proof for Theorem 17 which arguably simplifies their arguments and might be of independent interest.

We note that the bound in Theorem 17 is tight up to constants. Indeed, for any $n \in \mathbb{N}$, any constant $c \in \mathbb{N}$ (independent of n) and any $t \leq c \cdot \sqrt{n}$ consider the function $f\colon \{0,1\}^n \to \{0,1\}$ defined as

$$f(x) = \begin{cases} \mathsf{wt}(x) \bmod 2 & \text{if } |\mathsf{wt}(x) - n/2| \leq t/2, \\ 0 & \text{otherwise.} \end{cases}$$

First, it is easy to see that $t-1 \leq a(f) \leq t+1$. Moreover, a simple analysis shows that $\mathsf{Inf}(f) \geq \Omega(t \cdot \sqrt{n})$. To see this observe that since $t \leq O(\sqrt{n})$, then $\Pr_{x \leftarrow \{0,1\}^n}[|\mathsf{wt}(x) - n/2| \leq t/2] \geq \Omega(t/\sqrt{n})$, and that if x satisfies that $|\mathsf{wt}(x) - n/2| \leq t/2$, then changing each of its n coordinates will flip the value of the function.

Proof of Theorem 17. The "In particular" part of the above theorem follows by Nechiporuk's theorem (see Theorem 1). We proceed with the main part.

Denote by D the set of all pairs of points in $\{0,1\}^n$ that differ at one coordinate. Namely, $(x,y) \in D$ if and only if there exists an $i \in [n]$ such that $x^{\oplus i} = y$.

We define two ways to sample edges from D and show that they define the same distribution. The first way to sample an edge from D is by first sampling a point $x \in \{0,1\}^n$ and then sampling a random direction $i \in [n]$. This gives rise to the edge $(x, x^{\oplus i})$. Notice that for any edge $e \in D$ it holds that $\Pr_{x \leftarrow \{0,1\}^n, i \leftarrow [n]}[(x, x^{\oplus i}) = e] = \frac{1}{n \cdot 2^{n-1}}$. Moreover, observe that by the definition of total influence, we have that

$$\frac{\mathsf{Inf}(f)}{n} = \Pr_{x \leftarrow \{0,1\}^n, i \leftarrow [n]}[f(x) \neq f(x^{\oplus i})]. \tag{3}$$

The second way is defined as follows. Denote by $\mathcal{C}$ the set of all valid chains starting from 0^n and ending at 1^n. First, we sample a random chain $\mathcal{X} = (x_0 = 0^n, x_1, \ldots, x_{n-1}, x_n = 1^n)$ from $\mathcal{C}$. Notice that $\mathsf{wt}(x_i) = i$ for all $i \in [n] \cup \{0\}$. Then, we pick the edge $e^{(i)} = (x_i, x_{i+1})$ for $i \in [n-1] \cup \{0\}$ on the chain with probability $\binom{n-1}{i}/2^{n-1}$. Notice that this is a probability distribution since we have that $\sum_{i=0}^{n-1} \binom{n-1}{i}/2^{n-1} = 1$. Also, observe that a random chain $\mathcal{X}$ from $\mathcal{C}$ contains an arbitrary edge $(x, x') \in D$ with probability $1/(\binom{n-1}{\mathsf{wt}(x)} \cdot n)$. In total, using the above process, the probability to pick an edge $e \in D$ is

$$\begin{aligned}\Pr_{\mathcal{X} \leftarrow \mathcal{C}, (x_i, x_{i+1}) \leftarrow \mathcal{X}}[(x_i, x_{i+1}) = e] &= \Pr_{(x_i, x_{i+1}) \leftarrow \mathcal{X}}[(x_i, x_{i+1}) = e \mid e \in \mathcal{X}] \cdot \Pr_{\mathcal{X} \leftarrow \mathcal{C}}[e \in \mathcal{X}] \\ &= \frac{\binom{n-1}{\mathsf{wt}(x)}}{2^{n-1}} \cdot \frac{1}{\binom{n-1}{\mathsf{wt}(x)} \cdot n} = \frac{1}{n \cdot 2^{n-1}}.\end{aligned}$$

Therefore, we got that the two ways to sample an edge on the cube have the same distribution. Thus, using Equation (3), we get that

$$\frac{\mathsf{Inf}(f)}{n} = \Pr_{\mathcal{X} \leftarrow \mathcal{C}, (x_i, x_{i+1}) \leftarrow \mathcal{X}}[f(x_i) \neq f(x_{i+1})]. \tag{4}$$

However, notice that for any $\mathcal{X} \in \mathcal{C}$ it holds that

$$\Pr_{(x_i, x_{i+1}) \leftarrow \mathcal{X}}[f(x_i) \neq f(x_{i+1})] \leq a(f) \cdot \max_{i \in [n-1] \cup \{0\}} \left\{ \frac{\binom{n-1}{i}}{2^{n-1}} \right\} \leq O\left(a(f)/\sqrt{n}\right),$$

where the first inequality follows by the definition of $a(f)$ (the maximum number of alternations at any chain) and the second inequality holds since the second term is maximized roughly when $i \approx n/2$ and it is known (by e.g., Stirling's approximation) that $\binom{n}{n/2} = O(2^n/\sqrt{n})$. Plugging this back into Equation (4) we get that $\mathsf{Inf}(f) \leq O(a(f) \cdot \sqrt{n})$. ◀

5 Open Problems

In this paper we study the power of negation gates in the model of Boolean De Morgan formulas. Our shrinkage result (Theorem 6) implies that as long as $t \ll L(F)$, the shrinkage exponent of F is essentially Γ_0, the shrinkage exponent of monotone formulas. In addition, we showed that formulas with t negation gates can be efficiently transformed into circuits with roughly $\log t$ negation gates without incurring significant blow-up in size or depth.

Morizumi [25] showed that any formula F can be transformed into a formula F' that has only $\lceil n/2 \rceil$ negations and such that $L(F') \leq L(F) \cdot O(n^{6.3})$. His transformation uses as a building block the monotone formula that compute the threshold function of Valiant [40] which gives a short but non-explicit construction. We leave open the question whether one can come up with an explicit and efficient transformation from any formula to a formula with few negations.

Lastly, we mention the important open problem of determining the shrinkage exponent of monotone formulas.

Acknowledgments. We thank Avishay Tal for helpful discussions and, in particular, for a discussion that led to the proof of Theorem 3. We thank Andrej Bogdanov, Moni Naor, Alon Rosen and Eylon Yogev for inspiring discussions. Finally, we thank the reviewers for their constructive comments.

References

1 Miklós Ajtai, János Komlós, and Endre Szemerédi. An $o(n \log n)$ sorting network. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, STOC*, pages 1–9, 1983.

2 Kazuyuki Amano and Akira Maruoka. A superpolynomial lower bound for a circuit computing the clique function with at most $(1/6) \log \log n$ negation gates. *SIAM J. Comput.*, 35(1):201–216, 2005.

3 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

4 Robert Beals, Tetsuro Nishino, and Keisuke Tanaka. On the complexity of negation-limited boolean networks. *SIAM J. Comput.*, 27(5):1334–1347, 1998.

5 Eric Blais, Clément L. Canonne, Igor Carboni Oliveira, Rocco A. Servedio, and Li-Yang Tan. Learning circuits with few negations. *To appear in Proceedings of the 19th International Workshop on Randomization and Computation, RANDOM*, 2015. Available at `http://arxiv.org/abs/1410.8420`.

6 Nader H. Bshouty and Christino Tamon. On the Fourier spectrum of monotone functions. *J. ACM*, 43(4):747–770, 1996.

7 Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. In *Proceedings of the 29th Conference on Computational Complexity, CCC*, pages 262–273, 2014.

8 Ruiwen Chen, Valentine Kabanets, and Nitin Saurabh. An improved deterministic #SAT algorithm for small De Morgan formulas. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science, MFCS*, pages 165–176, 2014.

9 Moshe Dubiner and Uri Zwick. How do read-once formulae shrink? *Combinatorics, Probability & Computing*, 3:455–469, 1994.

10 Michael. J. Fischer. The complexity of negation-limited networks–a brief survey. *Automata Theory and Formal Languages*, 33:71–82, 1975.

11 Oded Goldreich and Rani Izsak. Monotone circuits: One-way functions versus pseudorandom generators. *Theory of Computing*, 8(1):231–238, 2012.

12 Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. In *Proceedings of the 46th Annual Symposium on Theory of Computing, STOC*, pages 847–856, 2014.

13 Siyao Guo, Tal Malkin, Igor Carboni Oliveira, and Alon Rosen. The power of negations in cryptography. In *Proceedings of the 12th Theory of Cryptography Conference, TCC*, pages 36–65, 2015.

14 Danny Harnik and Ran Raz. Higher lower bounds on monotone size. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, STOC*, pages 378–387, 2000.

15 Johan Håstad. One-way permutations in NC0. *Inf. Process. Lett.*, 26(3):153–155, 1987.

16 Johan Håstad. The shrinkage exponent of de Morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998.

17 Johan Håstad, Alexander A. Razborov, and Andrew Chi-Chih Yao. On the shrinkage exponent for read-once formulae. *Theor. Comput. Sci.*, 141(1&2):269–282, 1995.

18 Russell Impagliazzo and Valentine Kabanets. Fourier concentration from shrinkage. In *Proceedings of the 29th Conference on Computational Complexity, CCC*, pages 321–332, 2014.

19 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 111–119, 2012.

20 Russell Impagliazzo and Noam Nisan. The effect of random restrictions on formula size. *Random Struct. Algorithms*, 4(2):121–134, 1993.

21 Kazuo Iwama, Hiroki Morizumi, and Jun Tarui. Negation-limited complexity of parity and inverters. *Algorithmica*, 54(2):256–267, 2009.

22 Stasys Jukna. *Boolean Function Complexity – Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012.

23 Ilan Komargodski, Ran Raz, and Avishay Tal. Improved average-case lower bounds for DeMorgan formula size. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 588–597, 2013.

24 Andrey A. Markov. On the inversion complexity of a system of functions. *J. ACM*, 5(4):331–334, 1958.

25 Hiroki Morizumi. Limiting negations in formulas. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming, ICALP*, pages 701–712, 2009.

26 Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.

27 Eduard I. Nechiporuk. On the complexity of schemes in some bases containing nontrivial elements with zero weights. *Problemy Kibernetiki*, 8:123–160, 1962. In Russian.

28 Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

29 Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

30 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

31 Mike Paterson and Uri Zwick. Shrinkage of de Morgan formulae under restriction. *Random Struct. Algorithms*, 4(2):135–150, 1993.

32 Ran Raz and Avi Wigderson. Probabilistic communication complexity of boolean relations (extended abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science, FOCS*, pages 562–567, 1989.

33 Benjamin Rossman. Correlation bounds against monotone NC^1. In *Proceedings of the 30th Conference on Computational Complexity, CCC*, pages 392–411, 2015.

34 Miklos Santha and Christopher B. Wilson. Polynomial size constant depth circuits with a limited number of negations. In *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science, STACS*, pages 228–237, 1991.

35 Bella A. Subbotovskaya. Realizations of linear function by formulas using $+, \cdot, -$. *Doklady Akademii Nauk SSSR*, 136:3:553–555, 1961. In Russian.

36 Shao Chin Sung and Keisuke Tanaka. Limiting negations in bounded-depth circuits: An extension of markov's theorem. *Inf. Process. Lett.*, 90(1):15–20, 2004.

37 Avishay Tal. Shrinkage of De Morgan formulae by spectral techniques. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 551–560, 2014.

38 Michel Talagrand. How much are increasing sets positively correlated? *Combinatorica*, 16(2):243–258, 1996.

39 Keisuke Tanaka, Tetsuro Nishino, and Robert Beals. Negation-limited circuit complexity of symmetric functions. *Inf. Process. Lett.*, 59(5):273–279, 1996.

40 Leslie G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5(3):363–366, 1984.

Deletion Codes in the High-noise and High-rate Regimes[*]

Venkatesan Guruswami and Carol Wang

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA
`guruswami@cmu.edu`, `wangc@cs.cmu.edu`

Abstract

The noise model of deletions poses significant challenges in coding theory, with basic questions like the capacity of the binary deletion channel still being open. In this paper, we study the harder model of *worst-case* deletions, with a focus on constructing efficiently encodable and decodable codes for the two extreme regimes of high-noise and high-rate. Specifically, we construct polynomial-time decodable codes with the following trade-offs (for any $\varepsilon > 0$):

(i) Codes that can correct a fraction $1 - \varepsilon$ of deletions with rate poly(ε) over an alphabet of size poly$(1/\varepsilon)$;
(ii) Binary codes of rate $1 - \tilde{O}(\sqrt{\varepsilon})$ that can correct a fraction ε of deletions; and
(iii) Binary codes that can be *list decoded* from a fraction $(1/2 - \varepsilon)$ of deletions with rate poly(ε).

Our work is the first to achieve the qualitative goals of correcting a deletion fraction approaching 1 over bounded alphabets, and correcting a constant fraction of bit deletions with rate approaching 1 over a fixed alphabet. The above results bring our understanding of deletion code constructions in these regimes to a similar level as worst-case errors.

1998 ACM Subject Classification E.4 Coding and Information Theory

Keywords and phrases algorithmic coding theory, deletion codes, list decoding, probabilistic method, explicit constructions

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.867

1 Introduction

This work addresses the problem of constructing codes which can be efficiently corrected from a constant fraction of worst-case deletions. In contrast to erasures, the locations of deleted symbols are *not* known to the decoder, who receives only a subsequence of the original codeword. The deletions can be thought of as corresponding to errors in synchronization during communication. The loss of position information makes deletions a very challenging model to cope with, and our understanding of the power and limitations of codes in this model significantly lags behind what is known for worst-case errors.

The problem of communicating over the *binary deletion channel*, in which each transmitted bit is deleted independently with a fixed probability p, has been a subject of much study (see the excellent survey by Mitzenmacher [17] for more background and references). However, even this easier case is not well-understood. In particular, the capacity of the binary deletion channel remains open, although it is known to approach $1 - h(p)$ as p goes to 0, where $h(p)$

[*] Research supported in part by NSF grants CCF-0963975 and CCF-1422045. The second author is supported by an NSF Graduate Research Fellowship.

is the binary entropy function (see [5, 6, 25] for lower bounds and [12, 13] for upper bounds), and it is known to be positive (at least $(1-p)/9$) [18]) even as $p \to 1$.

The more difficult problem of correcting from adversarial rather than random deletions has also been studied, but with a focus on correcting a constant *number* (rather than fraction) of deletions [16]. Codes that can correct a single deletion have received a fair bit of attention (see the survey [23]), but it turns out that even correcting two deletions poses significant challenges and is not well understood, with efficient codes with low redundancy discovered only very recently [2].

Coding for a constant *fraction* of adversarial deletions, which is the focus of this work, has been considered previously by Schulman and Zuckerman [21]. They construct constant-rate binary codes which are efficiently decodable from a small constant fraction of worst-case deletions and insertions, and can also handle a small fraction of transpositions. The rate of these codes are bounded away from 1, whereas existentially one can hope to achieve a rate approaching 1 for a small deletion fraction.

The central theoretical goal in error-correction against any specific noise model is to understand the combinatorial trade-off between the rate of the code and noise rate that can be corrected, and to construct codes with efficient error-correction algorithms that ideally approach this optimal trade-off. While this challenge is open in general even for the well-studied and simpler model of errors and erasures, in the case of worst-case deletions, our knowledge has even larger gaps. (For instance, we do not know the largest deletion fraction which can be corrected with positive rate for any fixed alphabet size.) Over large alphabets that can grow with the length of the code, we can include the position of each codeword symbol as a header that is part of the symbol. This reduces the model of deletions to that of erasures, where simple optimal constructions (eg. Reed-Solomon codes) are known.

Given that we are far from an understanding of the best rate achievable for any specified deletion fraction, in this work we focus on the two extreme regimes — when the deletion fraction is small (and the code rate can be high), and when the deletion fraction approaches the maximum tolerable value (and the code rate is small). Our emphasis is on constructing codes that can be efficiently encoded and decoded, with trade-offs not much worse than random/inefficient codes (whose parameters we compute in Section 2). Our results, described next, bring the level of knowledge on efficient deletion codes in these regimes to a roughly similar level as worst-case errors. There are numerous open questions, both combinatorial and algorithmic, that remain open, and it is our hope that the systematic study of codes for worst-case deletions undertaken in this work will spur further research on good constructions beyond the extremes of low-noise and high-noise.

1.1 Our results

The best achievable rate against a fraction p of deletions cannot exceed $1-p$, as we need to be able to recover the message from the first $(1-p)$ fraction of codeword symbols. As mentioned above, over large (growing) alphabets this trade-off can in fact be achieved by a simple reduction to the model of erasures. Existentially, as we show in Section 2, for any $\gamma > 0$, it is easy to show that there are codes of rate $1-p-\gamma$ to correct a fraction p of deletions over an alphabet size that depends only on γ. For the weaker model of erasures, where the receiver knows the locations of erased symbols, we know explicit codes, namely certain algebraic-geometric codes [22] or expander based constructions [1, 8], achieving the optimal trade-off (rate $1-p-\gamma$ to correct a fraction p of erasures) over alphabets growing only as a function of $1/\gamma$. For deletions, we do not know how to construct codes with such a trade-off efficiently. However, in the high-noise regime when the deletion fraction is $p = 1-\varepsilon$

for some small $\varepsilon > 0$, we are able to construct codes of rate $\mathrm{poly}(\varepsilon)$ over an alphabet of size $\mathrm{poly}(1/\varepsilon)$. Note that an alphabet of size at least $1/\varepsilon$ is needed, and the rate can be at most ε, even for the simpler model of erasures, so we are off only by polynomial factors.

▶ **Theorem** (Theorem 7). *Let $1/2 > \varepsilon > 0$. There is an explicit code of rate $\Omega(\varepsilon^2)$ and alphabet size $\mathrm{poly}(1/\varepsilon)$ which can be corrected from a $1 - \varepsilon$ fraction of worst-case deletions.*

Moreover, this code can be constructed, encoded, and decoded in time $N^{\mathrm{poly}(1/\varepsilon)}$, where N is the block length of the code.

The above handles the case of very large fraction of deletions. At the other extreme, when the deletion fraction is small, the following result shows that we achieve high rate (approaching one) even over the binary alphabet.

▶ **Theorem** (Theorem 11). *Let $\varepsilon > 0$. There is an explicit binary code $C \subseteq \{0,1\}^N$ which is decodable from an ε fraction of deletions with rate $1 - \tilde{O}(\sqrt{\varepsilon})$ in time $N^{\mathrm{poly}(1/\varepsilon)}$.*

Moreover, C can be constructed and encoded in time $N^{\mathrm{poly}(1/\varepsilon)}$.

▶ Remark. For both of the above results, the construction and encoding/decoding complexity can be improved to $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\varepsilon)}$ at the expense of slightly worse parameters. See Theorems 16 and 10.

The next question is motivated by constructing binary codes for the "high noise" regime. In this case, we do not know (even non-constructively) the minimum fraction of deletions that forces the rate of the code to approach zero. (Contrast this with the situation for erasures (resp. errors), where we know the zero-rate threshold to be an erasure fraction 1/2 (resp. error fraction 1/4).) Clearly, if the adversary can delete half of the bits, he can always ensure that the decoder receives $0^{n/2}$ or $1^{n/2}$, so at most two strings can be communicated. Surprisingly, in the model of *list decoding*, where the decoder is allowed to output a small list consisting of all codewords which contain the received string as a subsequence, one can in fact decode from an deletion fraction arbitrarily close to 1/2, as our third construction shows:

▶ **Theorem** (Theorem 19). *Let $0 < \varepsilon < 1/2$. There is an explicit binary code $C \subseteq \{0,1\}^N$ of rate $\tilde{\Omega}(\varepsilon^3)$ which is list-decodable from a $1/2 - \varepsilon$ fraction of deletions with list size $(1/\varepsilon)^{O(\log\log(1/\varepsilon))}$.*

This code can be constructed, encoded, and list-decoded in time $N^{\mathrm{poly}(1/\varepsilon)}$.

We should note that it is not known if list decoding is required to correct deletion fractions close to 1/2, or if one can get by with unique decoding. Our guess would be that the largest deletion fraction unique decodable with binary codes is bounded away from 1/2. The cubic dependence on ε in the rate in the above theorem is similar to what has been achieved for correcting $1/2 - \varepsilon$ fraction of errors [9]. We anticipate (but have not formally checked) that a similar result holds over any fixed alphabet size k for list decoding from a fraction $(1 - 1/k - \varepsilon)$ of symbol deletions.

Construction approach

Our codes, like many considered in the past, including those of [3, 4, 20] in the random setting and particularly [21] in the adversarial setting, are based on concatenating a good error-correcting code (in our case, Reed-Solomon or Parvaresh-Vardy codes) with an inner deletion code over a much smaller block length. This smaller block length allows us to find and decode the inner code using brute force. The core of the analysis lies in showing that the adversary can only affect the decoding of a bounded fraction of blocks of the inner code, allowing the outer code to decode using the remaining blocks.

While our proofs only rely on elementary combinatorial arguments, some care is needed to execute them without losing in rate (in the case of Theorem 11) or in the deletion fraction we can handle (in the case of Theorems 7 and 19). In particular, for handling close to fraction 1 of deletions, we have to carefully account for errors and erasures of outer Reed-Solomon symbols caused by the inner decoder. To get binary codes of rate approaching 1, we separate inner codeword blocks with (not too long) buffers of 0's and we exploit some additional structural properties of inner codewords that necessitate many deletions to make them resemble buffers. The difficulty in both these results is unique identification of enough inner codeword boundaries so that the Reed-Solomon decoder will find the correct message. The list decoding result is easier to establish, as we can try many different boundaries and use a "list recovery" algorithm for the outer algebraic code. To optimize the rate, we use the Parvaresh-Vardy codes [19] as the outer algebraic code.

1.2 Organization

In Section 2, we consider the performance of certain random and greedily constructed codes. These serve both as benchmarks and as starting points for our efficient constructions. In Section 3, we construct codes in the high deletion regime. In Section 4, we give high-rate binary codes which can correct a small constant fraction of deletions. In Section 5, we give list-decodable binary codes up to the optimal error fraction. Some open problems appear in Section 6.

2 Existential bounds for deletion codes

A quick recap of standard coding terminology: a code C of block length m over an alphabet Σ is a subset $C \subseteq \Sigma^m$. The rate of C is defined as $\frac{\log |C|}{m \log |\Sigma|}$. The encoding function of a code is a map $E : [|C|] \to \Sigma^m$ whose image equals C (with messages identified with $[|C|]$ in some canonical way). Our constructions all exploit the simple but powerful idea of code concatenation: If $C_{\text{out}} \subseteq \Sigma_{\text{out}}^n$ is an "outer" code with encoding function E_{out}, and $C_{\text{in}} \subseteq \Sigma_{\text{in}}^m$ is an "inner" code encoding function $E_{\text{in}} : \Sigma_{\text{out}} \to \Sigma_{\text{in}}^m$, the the concatenated code $C_{\text{out}} \circ C_{\text{in}} \subseteq \Sigma_{\text{in}}^{nm}$ is a code whose encoding function first applies E_{out} to the message, and then applies E_{in} to each symbol of the resulting outer codeword.

In this section, we show the existence of deletion codes in certain ranges of parameters, without the requirement of efficient encoding or decoding. The proofs (found in the full version of this paper [11]) follow from standard probabilistic arguments, but to the best of our knowledge, these bounds were not known previously. The codes of Theorem 4 will be used as inner codes in our final concatenated constructions.

Throughout, we will write $[k]$ for the set $\{1, \ldots, k\}$. We will also use the binary entropy function, defined for $\delta \in [0, 1]$ as $h(\delta) = \delta \log \frac{1}{\delta} + (1 - \delta) \log \frac{1}{1-\delta}$. All logarithms in the paper are to base 2.

We note that constructing a large code in $[k]^m$ which can correct from a δ fraction of deletions is equivalent to constructing a large set of strings such that for each pair, their longest common subsequence (LCS) has length less than $(1 - \delta)m$.

We first consider how well a random code performs, using the following theorem from [15], which upper bounds the probability that a pair of randomly chosen strings has a long LCS.

▶ **Theorem 1** ([15], Theorem 1). *For every $\gamma > 0$, there exists $c > 0$ such that if k and $m/\sqrt{k}$ are sufficiently large, and u, v are chosen independently and uniformly from $[k]^m$, then*

$$\Pr\left[\left|\mathrm{LCS}(u,v) - 2m/\sqrt{k}\right| \geq \frac{\gamma m}{\sqrt{k}}\right] \leq e^{-cm/\sqrt{k}}.$$

Fixing γ to be 1, we obtain the following.

▶ **Proposition 2.** *Let $\varepsilon > 0$ be sufficiently small and let $k = (4/\varepsilon)^2$. There exists a code $C \subseteq [k]^m$ of rate $R = \Omega\big(\varepsilon/\log(1/\varepsilon)\big)$ which can correct a $1 - \varepsilon = 1 - 4/\sqrt{k}$ fraction of deletions.*

The following results, and in particular Corollary 6, show that we can nearly match the performance of random codes using a simple greedy algorithm.

We first bound the number of strings which can have a fixed string s as a subsequence.

▶ **Lemma 3.** *Let $\delta \in (0, 1/k)$, set $\ell = (1-\delta)m$, and let $s \in [k]^\ell$. The number of strings $s' \in [k]^m$ containing s as a subsequence is at most*

$$\sum_{t=\ell}^{m} \binom{t-1}{\ell-1} k^{m-t}(k-1)^{t-\ell} \leq k^{m-\ell}\binom{m}{\ell}.$$

When $k = 2$, we have the estimate

$$\sum_{t=\ell}^{m} \binom{t-1}{\ell-1} 2^{m-t} \leq \delta m \binom{m}{\ell}.$$

▶ **Theorem 4.** *Let $\delta, \gamma > 0$. Then for every m, there exists a code $C \subseteq [k]^m$ of rate $R = 1 - \delta - \gamma$ such that:*

- *C can be corrected from a δ fraction of worst-case deletions, provided $k \geq 2^{2h(\delta)/\gamma}$.*
- *C can be found, encoded, and decoded in time $k^{O(m)}$.*

Moreover, when $k = 2$, we may take $R = 1 - 2h(\delta) - \log(\delta m)/m$.

▶ Remark. The authors of [14] show a similar result for the binary case, but use the weaker bound in Lemma 3 to get a rate of $1 - \delta - 2h(\delta)$.

With a slight modification to the proof of Theorem 4, we obtain the following construction, which will be used in Section 4. The so-called "β-dense" property will help us to distinguish codewords, which have high Hamming weight, from long strings of zeroes.

▶ **Proposition 5.** *Let $\delta, \beta \in (0, 1)$. Then for every m, there exists a code $C \subseteq \{0,1\}^m$ of rate $R = 1 - 2h(\delta) - O(\log(\delta m)/m) - 2^{-\Omega(\beta m)}/m$ such that:*

- *For every string $s \in C$, s is "β-dense": every interval of length βm in s contains at least $\beta m/10$ ones,*
- *C can be corrected from a δ fraction of worst-case deletions, and*
- *C can be found, encoded, and decoded in time $2^{O(m)}$.*

In the high-deletion regime, we have the following corollary to Theorem 4, obtained by setting $\delta = 1 - \varepsilon$ and $\gamma = (1-\theta)\varepsilon$, and noting that $h(\varepsilon) \leq \varepsilon \log(1/\varepsilon) + 2\varepsilon$ when $\varepsilon < 1/2$.

▶ **Corollary 6.** *Let $1/2 > \varepsilon > 0$ and $\theta \in (0, 1/3]$. There for every m, there exists a code $C \subseteq [k]^m$ of rate $R = \varepsilon \cdot \theta$ which can correct a $1 - \varepsilon$ fraction of deletions in time $k^{O(m)}$, provided $k \geq 64/\varepsilon^{\frac{2}{1-\theta}}$.*

3 Coding against $1 - \varepsilon$ deletions

In this section, we construct codes for the high-deletion regime. We will use a concatenated coding approach, with an enlarged alphabet to help us determine the location of inner codewords. By choosing the parameters carefully, we are able to correct a large fraction of deletions. More precisely, we have the following theorem.

▶ **Theorem 7.** *Let $1/2 > \varepsilon > 0$. There is an explicit code of rate $\Omega(\varepsilon^2)$ and alphabet size $\mathrm{poly}(1/\varepsilon)$ which can be corrected from a $1 - \varepsilon$ fraction of worst-case deletions.*

Moreover, this code can be constructed, encoded, and decoded in time $N^{\mathrm{poly}(1/\varepsilon)}$, where N is the block length of the code.

We first define the code. Theorem 7 is then a direct corollary of Lemmas 8 and 9.

The code: Our code will be over the alphabet $\{0, 1, \ldots, D-1\} \times [k]$, where $D = 8/\varepsilon$ and $k = O(1/\varepsilon^3)$.

We first define a code C' over the alphabet $[k]$ by concatenating a Reed-Solomon code with an inner code over $[k]$ which can correct a slightly higher fraction of deletions.

More specifically, let $\mathbb{F}_q$ be a finite field. For any $n' \le n \le q$, the Reed-Solomon code of length $n \le q$ and dimension n' is a subset of $\mathbb{F}_q^n$ which admits an efficient algorithm to uniquely decode from t errors and r erasures, provided $r + 2t < n - n'$ (see, for example, [24]).

In our construction, we will take $n = q = 2n'/\varepsilon$. We first encode our message to a codeword $c = (c_1, \ldots, c_n)$ of the Reed-Solomon code. For each i, we then encode the pair (i, c_i) using an inner code over some alphabet $[k]$ which can correct a $1 - \varepsilon/2$ fraction of deletions.

To obtain our final code C, we replace every symbol s in C' which encodes the ith RS coordinate by the pair $\big(i \pmod D, s\big) \in \{0, 1, \ldots, D-1\} \times [k]$. The first coordinate, $i \pmod D$, contains the location of the codeword symbol modulo D, and we will refer to it as a **header**.

In order to obtain the parameters stated in Theorem 7, we will instantiate the inner code using Corollary 6, setting $\theta = 1/3$. This gives an inner code $C_1 \colon [n] \times \mathbb{F}_q \to [k]^m$, where $m = 12 \log q/\varepsilon$ and $k = O(1/\varepsilon^3)$, which can correct a $1 - \varepsilon/2$ fraction of deletions.

▶ **Lemma 8.** *For an inner code of rate R_{in}, the rate of C is $\Omega(\varepsilon R_{in})$. In particular, the rate of C can be taken to be $\Omega(\varepsilon^2)$.*

Proof. The rate of the outer Reed-Solomon code, labeled with indices, is at least $\varepsilon/4$. Finally, the alphabet increase in transforming C' to C decreases the rate by a factor of $\frac{\log(k)}{\log(Dk)} = \Omega(1)$.

By Corollary 6, the rate of the inner code can be taken to be $\Omega(\varepsilon)$.This gives us a final rate of $\Omega(\varepsilon^2)$. ◀

▶ **Lemma 9.** *Let the inner code have block length m and be decodable from a $1-\varepsilon/2$ fraction of worst-case deletions in time $T(m)$. Then the concatenated code C can be decoded from a $1 - \varepsilon$ fraction of worst-case deletions in time $\mathrm{poly}(N) \cdot T(m)$, where N is the block length of C.*

In particular, the concatenated code using Corollary 6 can be decoded in time $N^{O(\mathrm{poly}\, 1/\varepsilon)}$.

Proof. We apply the following algorithm to decode C.

- We partition the received word into *blocks* as follows: The first block begins at the first coordinate, and each subsequent block begins at the next coordinate whose header differs from its predecessor. This takes time $\mathrm{poly}(N)$.

- We begin with an empty set L.
  For each block which is of length between $\varepsilon m/2$ and m, we remove the headers by replacing each symbol (a, b) with the second coordinate b. We then apply the decoder from Corollary 6 to the block. If this succeeds, outputting a pair (i, r_i), then we add (i, r_i) to L. This takes time $\mathrm{poly}(N) \cdot T(m)$.
- If for any i, L contains multiple pairs with first coordinate i, we remove all such pairs from L. L thus contains at most one pair (i, r_i) for each index i. We apply the Reed-Solomon decoding algorithm to the string r whose ith coordinate is r_i if $(i, r_i) \in L$ and erased otherwise. This takes time $\mathrm{poly}(N)$.

Analysis: For any i, we will decode a correct coordinate $\big(i, c_i\big)$ if there is a block of length at least $\varepsilon m/2$ which is a subsequence of $C_1(i, c_i)$. (Here and in what follows we abuse notation by disregarding headers on codeword symbols.)

Thus, the Reed-Solomon decoder will receive the correct value of the ith coordinate unless one of the following occurs:

1. (Erasure) The adversary deletes a $\geq 1 - \varepsilon/2$ fraction of $C_1(i, c_i)$.
2. (Merge) The block containing (part of) $C_1(i, c_i)$ also contains symbols from other codewords of C_1, because the adversary has erased the codewords separating $C_1(i, c_i)$ from its neighbors with the same header.
3. (Conflict) Another block decodes to (i, r) for some r. Note that an erasure cannot cause a coordinate to decode incorrectly, so a conflict can only occur from a merge.

We would now like to bound the number of errors and erasures the adversary can cause.

- If the adversary causes an erasure without causing a merge, this requires at least $(1-\varepsilon/2)m$ deletions within the block which is erased, and no other block is affected.
- If the adversary merges t inner codewords with the same label, this requires at least $(t-1)(D-1)m$ deletions, of the intervening codewords with different labels. The merge causes the fully deleted inner codewords to be erased, and causes the t merged codewords to resolve into at most one (possibly incorrect) value. This value, if incorrect, could also cause one conflict.
  In summary, in order to cause one error and $r \leq (t-1)D + 2$ erasures, the adversary must introduce at least $(t-1)(D-1)m \geq (2+r)(1-\varepsilon/2)m$ deletions.

In particular, if the adversary causes s errors and r_1 erasures by merging, and r_2 erasures without merging, this requires at least

$$\geq (2s + r_1)(1 - \varepsilon/2)m + r_2(1 - \varepsilon/2)m = (2s + r)(1 - \varepsilon/2)m$$

deletions. Thus, when the adversary deletes at most a $(1 - \varepsilon)$ fraction of codeword symbols, we have that $2s + r$ is at most $(1 - \varepsilon)mn/(1 - \varepsilon/2)m < n(1 - \varepsilon/2)$. Recalling that the Reed-Solomon decoder in the final step will succeed as long as $2s + r < n(1 - \varepsilon/2)$, we conclude that the decoding algorithm will output the correct message. ◀

▶ Remark (Improving the encoding and decoding complexity). Our decoding algorithm requires only that the inner code C_1 be correctable from a $1 - \varepsilon/2$ fraction of deletions. By using the concatenated code of Theorem 7 as the inner code in our construction (that is, with two levels of concatenation), we can reduce the time complexity significantly, at the cost of a polynomial reduction in other parameters of the code. This is summarized in the following theorem.

▶ **Theorem 10.** *Let $1/2 > \varepsilon > 0$. There is an explicit code of rate $\Omega(\varepsilon^3)$ and alphabet size* $\mathrm{poly}(1/\varepsilon)$ *which can be corrected from a $1-\varepsilon$ fraction of worst-case deletions. Moreover, this code can be constructed, encoded, and decoded in time* $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\varepsilon)}$*, where N is the block length of the code.*

4 Binary codes against ε deletions

4.1 Construction overview

The goal in our constructions is to allow the decoder to approximately locate the boundaries between codewords of the inner code, in order to recover the symbols of the outer code. In the previous section, we were able to achieve this by augmenting the alphabet and letting each symbol encode some information about the block to which it belongs. In the binary case, we no longer have this luxury.

The basic idea of our code is to insert long runs of zeros, or "buffers," between adjacent inner codewords. The buffers are long enough that the adversary cannot destroy many of them. If we then choose the inner code to be dense (in the sense of Proposition 5), it is also difficult for a long interval in any codeword to be confused for a buffer. This approach optimizes that of [21], which uses an inner code of rate 1/2 and thus has final rate bounded away from 1.

The balance of buffer length and inner codeword density seems to make buffered codes unsuited for high deletion fractions, and indeed our results only hold as the deletion fraction goes to zero.

4.2 Our construction

We now give the details of our construction. For simplicity, we will not optimize constants in the analysis.

▶ **Theorem 11.** *Let $\varepsilon > 0$. There is an explicit binary code $C \subseteq \{0,1\}^N$ which is decodable from an ε fraction of deletions with rate $1 - \tilde{O}(\sqrt{\varepsilon})$ in time $N^{\mathrm{poly}(1/\varepsilon)}$.*

Moreover, C can be constructed and encoded in time $N^{\mathrm{poly}(1/\varepsilon)}$.

The code: We again use a concatenated construction with a Reed-Solomon code as the outer code, choosing one which can correct a $12\sqrt{\varepsilon}$ fraction of errors and erasures. For each i, we replace the ith coordinate c_i with the pair (i, c_i). In order to ensure that the rate stays high, we use a RS code over $\mathbb{F}_{q^h}$, with block length $n = q$, where we will take $h = 1/\varepsilon$.

The inner code will be a good binary deletion code C_1 of block length m correcting a $\delta = 40\sqrt{\varepsilon}$ fraction of deletions. We will also require the codewords of C_1 to be β-dense, for $\beta = \delta/4$. Recall that a string of length m is β-dense if any interval of length βm contains at least $\beta m/10$ 1's. We will assume each codeword begins and ends with a 1.

Now, between each pair of adjacent inner codewords of C_1, we insert a *buffer* of δm zeros. This gives us our final code C.

In order to obtain the final parameters stated in Theorem 11, we will construct the inner code C_1 using Proposition 5. This gives a code of rate $1 - 2h(\delta) - o(1)$ satisfying the requirements of our construction.

▶ **Lemma 12.** *For an inner code of rate R_{in}, the rate of the concatenated code C is $R_{in} \cdot (1 - O(\sqrt{\varepsilon})$.*

In particular, the rate of the concatenated code using Proposition 5 is $1 - \tilde{O}(\sqrt{\varepsilon}))$.

Proof. The rate of the outer (labeled) Reed-Solomon code is $(1 - 24\sqrt{\varepsilon}) \cdot \frac{h}{h+1}$. Finally, adding buffers reduces the rate by a factor of $\frac{1}{1+\delta}$.

Combining these with our choice of δ, we get that the rate of C is $R_i(1 - \tilde{O}(\sqrt{\varepsilon}))$.

The rate of the inner code C_1 can be taken to be $1 - 2h(\delta) - o(1)$, by Proposition 5, giving a final rate of $1 - \tilde{O}(\sqrt{\varepsilon})$. ◀

▶ **Lemma 13.** *Let the inner code have block length m and be decodable from a δ fraction of worst-case deletions in time $T(m)$. Then the concatenated code C can be decoded from a ε fraction of worst-case deletions in time $\mathrm{poly}(N) \cdot T(m)$, where N is the block length of C.*

In particular, the concatenated code with inner code constructed using Proposition 5 can be decoded in time $N^{O(\mathrm{poly}\, 1/\varepsilon)}$.

The algorithm:

- The decoder first locates all runs of at least $\delta m/2$ contiguous zeroes in the received word. These runs ("buffers") are removed, dividing the codeword into blocks of contiguous symbols which we will call *decoding windows*. Any leading zeroes of the first decoding window and trailing zeroes of the last decoding window are also removed. This takes time $\mathrm{poly}(N)$.
- We begin with an empty set L.
  For each decoding window, we apply the decoder from Proposition 5 to attempt to recover a pair (i, r_i). If we succeed, this pair is added to L. This takes time $\mathrm{poly}(N) \cdot T(m)$.
- If for any i, L contains multiple pairs with first coordinate i, we remove all such pairs from L. L thus contains at most one pair (i, r_i) for each index i. We apply the Reed-Solomon decoding algorithm to the string r whose ith coordinate is r_i if $(i, r_i) \in L$ and erased otherwise, attempting to recover from a $12\sqrt{\varepsilon}$ fraction of errors and erasures. This takes time $\mathrm{poly}(N)$.

Analysis: Notice that if no deletions occur, the decoding windows will all be codewords of the inner code C_1, which will be correctly decoded. At a high level, we will show that the adversary cannot corrupt many of these decoding windows, even with an ε fraction of deletions.

We first show that the number of decoding windows considered by our algorithm is close to n, the number of windows if there are no deletions.

▶ **Lemma 14.** *If an ε fraction of deletions have occurred, then the number of decoding windows considered by our algorithm is between $(1 - 2\sqrt{\varepsilon})n$ and $(1 + 2\sqrt{\varepsilon})n$.*

Proof. Recall that the adversary can cause at most $\varepsilon nm(1+\delta) \leq 2\varepsilon nm$ deletions.

Upper bound: The adversary can increase the number of decoding windows only by creating new runs of $\delta m/2$ zeroes (that are not contained within a buffer). Such a new run must be contained entirely within an inner codeword $w \in C_1$. However, as w is $\delta/4$-dense, in order to create a run of zeroes of length $\delta m/2$, at least $\delta m/20 = 2\sqrt{\varepsilon}$ 1's must be deleted for each such run. In particular, at most $\sqrt{\varepsilon} n$ blocks can be added.

Lower bound: The adversary can decrease the number of decoding windows only by decreasing the number of buffers. He can achieve this either by removing a buffer, or by merging two buffers. Removing a buffer requires deleting $\delta m/2 = 20\sqrt{\varepsilon} m$ zeroes from the original buffer. Merging two buffers requires deleting all 1's in the inner codewords between them. As inner codewords are $\delta/4$-dense, this requires at least $\sqrt{\varepsilon} m$ deletions for each merged buffer. In particular, at most $2\sqrt{\varepsilon} n$ buffers can be removed. ◀

We now show that almost all of the decoding windows being considered are decoded correctly by the inner decoder.

▶ **Lemma 15.** *The number of decoding windows which are incorrectly decoded is at most $4\sqrt{\varepsilon}n$.*

Proof. The inner decoder will succeed on each decoding window which is a subsequence of a valid inner codeword $w \in C_1$ of length at least $(1-\delta)m$. This will happen unless:

1. The window is too short:
   (a) a subsequence of w has been marked as a (new) buffer, or
   (b) a ρ fraction of w has been marked as part of the adjacent buffers, combined with a $\delta - \rho$ fraction of deletions within w.
2. The window is not a subsequence of a valid inner codeword: the window contains buffer symbols and/or a subsequence of multiple inner codewords.

We first show that (1) holds for at most $3\sqrt{\varepsilon}n$ windows.

From the proof of Lemma 14, there can be at most $\sqrt{\varepsilon}n$ new buffers introduced, thus handling Case 1(a). In Case 1(b), if $\rho < \delta/2$, then there must be $\delta/2$ deletions within w. On the other hand, if $\rho \geq \delta/2$, one of two buffers adjacent to w must have absorbed at least $\delta m/4$ symbols of w, so as w is $\delta/4$-dense, this requires $\delta m/40 = \sqrt{\varepsilon}m$ deletions, so can occur in at most $2\sqrt{\varepsilon}n$ windows.

We also have that (2) holds for at most $\sqrt{\varepsilon}n$ windows, as at least $\delta m/2$ symbols must be deleted from a buffer in order to prevent the algorithm from marking it as a buffer. As in Lemma 14, this requires $20\sqrt{\varepsilon}$ deletions for each merged window, and so there are at most $\sqrt{\varepsilon n}$ windows satisfying case (2). ◀

We now have that the inner decoder outputs $(1-6\sqrt{\varepsilon})n$ correct values. After removing possible conflicts in the last step of the algorithm, we have at least $(1-12\sqrt{\varepsilon})n$ correct values, so that the Reed-Solomon decoder will succeed and output the correct message.

▶ Remark (Improving the encoding and decoding efficiency). Our decoding algorithm succeeds as long as the inner code can correct a δ fraction of deletions, and consists of codewords which are $\delta/4$-dense. As in the high deletion case, the time complexity of Theorem 11 can be improved using a more efficient inner code, at the cost of a reduction in rate.

Because of the addition of buffers, the code of Theorem 11 may not be dense enough to use as an inner code. However, we can modify the construction to obtain a dense inner code (details can be found in the full version [11]). In particular, these modifications give us the following.

▶ **Theorem 16.** *Let $\varepsilon > 0$. There is an explicit binary code $C \subseteq \{0,1\}^N$ which is decodable from an ε fraction of deletions with rate $1 - \tilde{O}(\sqrt[4]{\varepsilon})$ in time $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\varepsilon)}$.*

Moreover, C can be constructed and encoded in time $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\varepsilon)}$.

5 List-decoding binary deletion codes

The results of Section 4 show that we can have good explicit binary codes when the deletion fraction is low. In this section, we address the opposite regime, of high deletion fraction. As a first step, notice that in any reasonable model, including list-decoding, we cannot hope to efficiently decode from a $1/2$ deletion fraction with a polynomial list size and constant rate. With block length n and $n/2$ deletions, the adversary can ensure that what is received is either $n/2$ 1's or $n/2$ 0's.

Thus, for binary codes and $\varepsilon > 0$, we will consider the question of whether it is possible to list decode from a fraction $1/2 - \varepsilon$ of deletions.

▶ **Definition 17.** We say that a code $C \subseteq \{0,1\}^m$ is list-decodable from a δ deletion fraction with list size L if every sequence of length $(1-\delta)m$ is a subsequence of at most L codewords. If this is the case, we will call C (δ, L) *list-decodable from deletions.*

▶ Remark. Although the results of this section are proven in the setting of list-decoding, it is *not* known that we cannot have unique decoding of binary codes up to deletion fraction $1/2 - \varepsilon$. See the first open problem in Section 6.

5.1 List-decodable binary deletion codes (existential)

In this section, we show that good list-decodable codes exist. This construction will be the basis of our explicit construction of list-decodable binary codes. The proof appears in the appendix.

▶ **Theorem 18.** *Let $\delta, L > 0$. Let $C \subseteq \{0,1\}^m$ consist of 2^{Rm} independently, uniformly chosen strings, where $R \leq 1 - h(\delta) - 3/L$. Then C is $\big(\delta, L\big)$ list-decodable from deletions with probability at least $1 - 2^{-m}$.*

Moreover, such a code can be constructed and decoded in time $2^{\mathrm{poly}(mL)}$.

In particular, when $\delta = 1/2 - \varepsilon$, we can construct and decode in time $2^{\mathrm{poly}(m/\varepsilon)}$ a code $C \subseteq \{0,1\}^m$ of rate $\Omega(\varepsilon^2)$ which is $\big(\delta, O(1/\varepsilon^2)\big)$ list-decodable from deletions.

5.2 List-decodable binary deletion codes (explicit)

We now use the existential construction of Theorem 18 to give an explicit construction of constant-rate list-decodable binary codes. Our code construction uses Parvaresh-Vardy codes ([19]) as outer codes, and an inner code constructed using Section 5.1.

The idea is to list-decode "enough" windows and then apply the list recovery algorithm of Theorem 20.

▶ **Theorem 19.** *Let $0 < \varepsilon < 1/2$. There is an explicit binary code $C \subseteq \{0,1\}^N$ of rate $\tilde{\Omega}(\varepsilon^3)$ which is list-decodable from a $1/2 - \varepsilon$ fraction of deletions with list size $(1/\varepsilon)^{O(\log\log\varepsilon)}$.*

This code can be constructed, encoded, and list-decoded in time $N^{\mathrm{poly}(1/\varepsilon)}$.

We will appeal in our analysis to the following theorem, which can be found in [10].

▶ **Theorem 20** ([10], Corollary 5)**.** *For all integers $s \geq 1$, for all prime powers r, every pair of integers $1 < K \leq N \leq q$, there is an explicit $\mathbb{F}_r$-linear map $E\colon \mathbb{F}_q^K \to \mathbb{F}_{q^s}^N$ whose image C' is a code satisfying:*

- *There is an algorithm which, given a collection of subsets $S_i \subseteq \mathbb{F}_{q^s}$ for $i \in [N]$ with $\sum_i |S_i| \leq N\ell$, runs in $\mathrm{poly}\big((rs)^s, q, \ell\big)$ time, and outputs a list of size $O\big((rs)^s N\ell/K\big)$ that includes precisely the set of codewords $(c_1, \ldots, c_N) \in C'$ that satisfy $c_i \in S_i$ for at least αN values of i, provided*

$$\alpha > (s+1)(K/N)^{s/(s+1)}\ell^{1/(s+1)}.$$

The code: We set $s = O(\log 1/\varepsilon)$, $r = O(1)$, and $N = K\,\mathrm{poly}\big(\log(1/\varepsilon)\big)/\varepsilon$ in Theorem 20 in order to obtain a code $C' \subseteq \mathbb{F}_{q^s}^N$. We modify the code, replacing the ith coordinate c_i with the pair (i, c_i) for each i, in order to obtain a code C''. This latter step only reduces the rate by a constant factor.

Recall that we are trying to recover from a $1/2-\varepsilon$ fraction of deletions. We use Theorem 18 to construct an inner code $C_1\colon [N] \times \mathbb{F}_q^s \to \{0,1\}^m$ of rate $\Omega(\varepsilon^2)$ which recovers from a $1/2 - \delta$ deletion fraction (where we will set $\delta = \varepsilon/4$). Our final code C is a concatenation of C'' with C_1, which has rate $\tilde{\Omega}(\varepsilon^3)$.

▶ **Theorem 21.** *C is list-decodable from a $1/2 - \varepsilon$ fraction of deletions in time $N^{\mathrm{poly}(1/\varepsilon)}$.*

Proof. Our algorithm first defines a set of "decoding windows". These are intervals of length $(1/2 + \delta)m$ in the received codeword which start at positions $1 + t\delta m$ for $t = 0, 1, \ldots, N/\delta - (1/2 + \delta)/\delta$, in addition to one interval consisting of the last $(1/2 + \delta)m$ symbols in the received codeword.

We use the algorithm of Theorem 18 to list-decode each decoding window, and let $\mathcal{L}$ be the union of the lists for each window. Finally, we apply the algorithm of Theorem 20 to $\mathcal{L}$ to obtain a list containing the original message.

Correctness: Let $c = (c_1, \ldots, c_N)$ be the originally transmitted codeword of C'. If an inner codeword $C_1(i, c_i)$ has suffered fewer than a $1/2 - 2\delta$ fraction of deletions, then one of the decoding windows is a substring of $C_1(i, c_i)$, and $\mathcal{L}$ will contain the correct pair (i, c_i).

When $\delta = \varepsilon/4$, by a simple averaging argument, we have that an ε fraction of inner codewords have at most $1/2 - 2\delta$ fraction of positions deleted. For these inner codewords, $\mathcal{L}$ contains a correct decoding of the corresponding symbol of c.

In summary, we have list-decoded at most N/δ windows, with a list size of $O(1/\delta^2)$ each. We also have that an ε fraction of symbols in the outer codeword of C' is correct. Setting $\ell = O(1/\delta^3)$ in the algorithm of Theorem 20, we can take $\alpha = \varepsilon$. Theorem 20 then guarantees that the decoder will output a list of $\mathrm{poly}(1/\varepsilon)$ codewords, including the correct codeword c. ◀

6 Conclusion and open problems

In this work, we initiated a systematic study of codes for the adversarial deletion model, with an eye towards constructing codes achieving more-or-less the correct trade-offs at the high-noise and high-rate regimes. There are still several major gaps in our understanding of deletion codes, and below we highlight some of them (focusing only on the worst-case model):

1. For binary codes, what is the supremum p^* of all fractions p of adversarial deletions for which one can have positive rate? Clearly $p^* \le 1/2$; could it be that $p^* = 1/2$ and this trivial limit can be matched? Or is it the case that p^* is strictly less than $1/2$?
2. Can one construct codes of rate $1 - p - \gamma$ to efficiently correct a fraction p of deletions over an alphabet size that only depends on γ?

   Note that this requires a relative distance of p, and currently we only know algebraic-geometric and expander-based codes which achieve such a tradeoff between rate and relative distance.
3. Can one improve the rate of the binary code construction to correct a fraction ε of deletions to $1 - \varepsilon\,\mathrm{poly}(\log(1/\varepsilon))$, approaching more closely the existential $1 - O(\varepsilon \log(1/\varepsilon))$ bound? In the case of errors, an approach using expanders gives the analogous tradeoff (see [7] and references therein). Could such an approach be adapted to the setting of deletions?

References

1 N. Alon, J. Edmonds, and M. Luby. Linear time erasure codes with nearly optimal recovery (extended abstract). In *FOCS*, pages 512–519. IEEE Computer Society, 1995.

2 J. Brakensiek, V. Guruswami, and S. Zbarsky. Efficient low-redundancy codes for correcting multiple deletions. In preparation, 2015.

3 J. Chen, M. Mitzenmacher, C. Ng, and N. Varnica. Concatenated codes for deletion channels. In *IEEE International Symposium on Information Theory, 2003*, pages 218–218, June 2003.

4 M. Davey and D. J. C. MacKay. Reliable communication over channels with insertions, deletions, and substitutions. *IEEE Transactions on Information Theory*, 47(2):687–698, Feb 2001.

5 S. Diggavi and M. Grossglauser. On information transmission over a finite buffer channel. *IEEE Transactions on Information Theory*, 52(3):1226–1237, March 2006.

6 R. Gallager. Sequential decoding for binary channels with noise and synchronization errors, October 1961. Lincoln Lab. Group Report.

7 V. Guruswami. Guest column: error-correcting codes and expander graphs. *SIGACT News*, 35(3):25–41, 2004.

8 V. Guruswami and P. Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, 2005.

9 V. Guruswami and A. Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008.

10 V. Guruswami and A. Rudra. Soft decoding, dual BCH codes, and better list-decodable e-biased codes. In *Proceedings of the 2008 IEEE 23rd Annual Conference on Computational Complexity*, CCC '08, pages 163–174, Washington, DC, USA, 2008. IEEE Computer Society.

11 Venkatesan Guruswami and Carol Wang. Deletion codes in the high-noise and high-rate regimes. *CoRR*, abs/1411.6667, 2014.

12 A. Kalai, M. Mitzenmacher, and M. Sudan. Tight asymptotic bounds for the deletion channel with small deletion probabilities. In *ISIT*, pages 997–1001, 2010.

13 Y. Kanoria and A. Montanari. Optimal coding for the binary deletion channel with small deletion probability. *IEEE Transactions on Information Theory*, 59(10):6192–6219, Oct 2013.

14 I. Kash, M. Mitzenmacher, J. Thaler, and J. Ullman. On the zero-error capacity threshold for deletion channels. In *Information Theory and Applications Workshop (ITA), 2011*, pages 1–5, Feb 2011.

15 M. Kiwi, M. Loebl, and J. Matoušek. Expected length of the longest common subsequence for large alphabets. *Advances in Mathematics*, 197:480–498, November 2004.

16 V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics - Doklady*, 10(8):707–710, 1966.

17 M. Mitzenmacher. A survey of results for deletion channels and related synchronization channels. *Probability Surveys*, 6:1–33, 2009.

18 M. Mitzenmacher and E. Drinea. A simple lower bound for the capacity of the deletion channel. *IEEE Transactions on Information Theory*, 52(10):4657–4660, 2006.

19 F. Parvaresh and A. Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 285–294, 2005.

20 E. Ratzer. Marker codes for channels with insertions and deletions. *Annals of Telecommunications*, 60(1–2):29–44, Jan–Feb 2005.

21 L. Schulman and D. Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Transactions on Information Theory*, 45(7):2552–2557, November 1999.

22 K. W. Shum, I. Aleshnikov, P. V. Kumar, H. Stichtenoth, and V. Deolalikar. A low-complexity algorithm for the construction of algebraic-geometric codes better than the Gilbert-Varshamov bound. *IEEE Transactions on Information Theory*, 47(6):2225–2241, 2001.

23 N. J. A. Sloane. On single-deletion-correcting codes. *CoRR*, arxiv.org/abs/math/0207197, 2002.

24 L. R. Welch and E. R. Berlekamp. Error correction of algebraic block codes. *US Patent Number 4,633,470*, December 1986.

25 K. Zigangirov. Sequential decoding for a binary channel with drop-outs and insertions. *Problemy Peredachi Informatsii*, 5(2):23–30, 1969.

Communication with Partial Noiseless Feedback

Bernhard Haeupler[1], Pritish Kamath[2], and Ameya Velingker[1]

1 **Computer Science Department**
Carnegie Mellon University, USA
`{haeupler,avelingk}@cs.cmu.edu`

2 **Computer Science and Artificial Intelligence Laboratory**
Massachusetts Institute of Technology, USA
`pritish@mit.edu`

Abstract

We introduce the notion of *one-way communication schemes with partial noiseless feedback*. In this setting, Alice wishes to communicate a message to Bob by using a communication scheme that involves sending a sequence of bits over a channel while receiving feedback bits from Bob for δ fraction of the transmissions. An adversary is allowed to corrupt up to a constant fraction of Alice's transmissions, while the feedback is always uncorrupted. Motivated by questions related to coding for interactive communication, we seek to determine the maximum error rate, as a function of $0 \leq \delta \leq 1$, such that Alice can send a message to Bob via some protocol with δ fraction of noiseless feedback. The case $\delta = 1$ corresponds to *full feedback*, in which the result of [1] implies that the maximum tolerable error rate is $1/3$, while the case $\delta = 0$ corresponds to *no feedback*, in which the maximum tolerable error rate is $1/4$, achievable by use of a binary error-correcting code.

In this work, we show that for any $\delta \in (0, 1]$ and $\gamma \in [0, 1/3)$, there exists a *randomized* communication scheme with noiseless δ-feedback, such that the probability of miscommunication is low, as long as no more than a γ fraction of the rounds are corrupted. Moreover, we show that for any $\delta \in (0, 1]$ and $\gamma < f(\delta)$, there exists a *deterministic* communication scheme with noiseless δ-feedback that always decodes correctly as long as no more than a γ fraction of rounds are corrupted. Here f is a monotonically increasing, piecewise linear, continuous function with $f(0) = 1/4$ and $f(1) = 1/3$. Also, the rate of communication in both cases is constant (dependent on δ and γ but independent of the input length).

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Communication with feedback, Interactive communication, Coding theory

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.881

1 Introduction

Motivated by questions in interactive coding, we introduce the model of *communication with partial noiseless feedback*. Alice wishes to communicate a message, say in $\{0, 1\}^k$, to Bob. Alice sends a total of N bits to Bob, and she receives δN bits of feedback from Bob for some fixed $\delta > 0$. We have an adversary who can corrupt γN of the bits sent by Alice, but the feedback bits are left uncorrupted. We wish to find the maximal tolerable error fraction γ (on Alice's transmissions) under which we can guarantee that Bob is able to receive Alice's message correctly. This problem is summarized in Figure 1.

We introduce this problem as a generalization of the problem of communication with complete noiseless feedback (corresponds to $\delta = 1$ above), where after each bit sent by Alice,

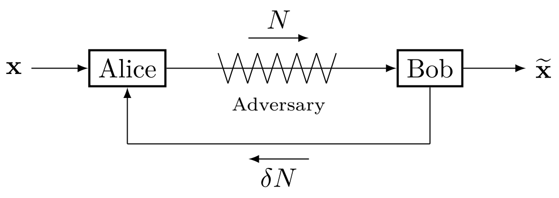


Figure 1 Communication with partial noiseless feedback.

Bob sends the received bit as feedback, so that Alice can use her knowledge of exactly what Bob has received thus far to possibly adapt her future transmissions. Indeed, Berlekamp showed that in this setting, one can tolerate any error rate less than $1/3$ with non-vanishing communication rate if we require that any possible error pattern of up to the error rate be corrected [1], and moreover, this is the maximal error fraction one can hope to correct with constant rate. This bound also follows from the game of questions with liars [20].

On the other extreme, if there were no feedback at all (i.e. $\delta = 0$), then this is equivalent to error correcting codes, for which it is known that one can tolerate up to $1/4$ error fraction while still achieving positive communication rate [12, 21]. And an error fraction of $\geq 1/4$ necessarily results in zero asymptotic rate, due to the Plotkin bound [15]. Thus, feedback increases the set of achievable communication rates in the adversarial error model. This is in contrast to the random-error setting in which random error patterns need to be corrected only with high probability, as Shannon showed that feedback does not increase the capacity of a discrete memoryless channel [19].

1.1 Coding for interactive communication

The problem of communication with noiseless feedback has garnered further interest recently in the context of coding for interactive communication. In this setting, Alice and Bob are given inputs $\mathbf{x}$ and $\mathbf{y}$, respectively, and they are required to compute some function $f(\mathbf{x}, \mathbf{y})$ by exchanging messages over a noisy channel. In particular, up to a γ fraction of the total transmitted bits may get flipped by the channel, and one requires a coding scheme that allows successful computation even in the presence of noise, preferably with a only constant blow up in communication. Schulman was the first to investigate the problem, and in a series of works, he gave the first constant rate scheme that could tolerate an error fraction of up to $\gamma = 1/240$ [16, 17, 18]. Subsequently, in an influential work, Braverman and Rao [6] showed a coding scheme that works for any error rate $\gamma < 1/4$, and moreover, they showed that any error rate of $\geq 1/4$ cannot be tolerated as long as the encoded protocol is *non-adaptive* (meaning that whose turn it is to speak during each round of communication is predetermined). There have been a lot of subsequent works since, which deal with computational efficiency [2, 4, 3, 9], allowing adaptivity [11], list-decoding [11, 10, 5], interactive channel capacity under random noise [14] and adversarial noise [13] etc.

Approaching an error fraction of $1/4$ in the non-adaptive setting requires communicating symbols from a growing alphabet size. If we restrict ourselves to communicating bits, then the coding scheme of Braverman and Rao [6] tolerates any error fraction $\gamma < 1/8$. Determining the maximum tolerable noise for interactive coding with symbols from a binary alphabet is still an open question. However, the optimality of $1/3$ as the maximal tolerable error fraction in the noiseless feedback problem can be used to establish an upper bound of $1/6$ for the maximal tolerable error fraction for interactive coding over binary alphabets! Furthermore, Efremenko, Gelles and Haeupler [7] show a coding scheme over binary alphabets that tolerates any error fraction $\gamma < 1/6$ if noiseless feedback is allowed in the interactive setting as well. Also, Gelles and Haeupler [8] show that for an error fraction of ε, any *alternating* interactive protocol can be encoded with rate $1 - \Theta(H(\varepsilon))$ over channels with noiseless feedback as well as erasure channels.

1.2 Our results

In this work, we show that for any $\delta \in (0, 1]$ and $\gamma \in [0, 1/3)$, there exists a *randomized* communication scheme with noiseless δ-feedback, such that the probability of miscommunication is low, as long as no more than a γ fraction of the rounds are corrupted. Moreover, we show that for any $\delta \in (0, 1]$ and $\gamma < f(\delta)$, there exists a *deterministic* communication scheme with noiseless δ-feedback that always decodes correctly as long as no more than a γ fraction of rounds are corrupted. Here f is a monotonically increasing, piecewise linear, continuous function with $f(0) = 1/4$ and $f(1) = 1/3$. Also, the rate of communication in both cases is constant (dependent on δ and γ but independent of the input length).

1.3 Organization of this paper

In Section 2, we give some of the basic definitions and notations as well as the statements of the two main theorems in this work. In Section 3, we describe a simple deterministic communication scheme with 1-feedback that tolerates up to $1/3$ fraction errors. In Section 4, we describe our randomized communication scheme with partial noiseless feedback. In Section 5, we describe our deterministic communication schemes which comes about by a de-randomization of the randomized communication scheme. Finally, we give a summary of our results and suggest possible future directions in Section 6.

2 Preliminaries and results

In this section, we describe our problem set-up and state our results.

2.1 One-way communication schemes with partial noiseless feedback

We consider the problem of *one-way communication with partial noiseless feedback*, which we define as follows:

▶ **Definition 1.** For all $\delta \in [0, 1]$, a *"one-way communication scheme with noiseless δ-feedback"* is defined as follows (summarized in Figure 1):

- Alice wishes to send a message in $\mathbf{x} \in \Sigma^k$ to Bob.
- Alice and Bob engage in a communication protocol of length $N + \delta N$, out of which N symbols are sent by Alice (forward rounds) and δN symbols are sent by Bob (feedback rounds). At most one of the parties can send a symbol in any round.
- The adversary can corrupt at most γN of the forward rounds, but none of the feedback rounds.
- At the end of the protocol, Bob is required to decode the message $\mathbf{x}$ from the transcript of the protocol.

We call N as the *length* of the communication scheme. The *'rate'* of the scheme is k/N, and γ is the error fraction tolerated. (We will often drop the words 'one-way' and 'noiseless'. All feedback in this paper will be assumed to be noiseless.)

The protocol can be *deterministic* or *randomized* (but with only private randomness). In deterministic schemes, we require that Bob is always able to recover $\mathbf{x}$ correctly. In randomized schemes, we require that Bob is able to recover $\mathbf{x}$ correctly with probability at least $1 - o_k(1)$, where the probability is over the private randomness of Alice and Bob.

Note. All the results in this paper will only be for $\Sigma = \{0, 1\}$. Thus, for the rest of the paper, we will work with communication schemes over a binary alphabet.

An important remark about randomized communication schemes is that the adversary is not aware of the random bits being used by the parties in advance. The adversary can only infer the random bits after they are used. We emphasize that it is this remark that makes *de-randomizing* such communication schemes very challenging!

In this work, we wish to fix δ and find an infinite sequence of communication schemes with noiseless δ-feedback, for increasing values of k, where the communication schemes have length $N(k)$. The *asymptotic rate* of the sequence of communication schemes is defined to be $\lim_{k\to\infty} k/N(k)$. For the remainder of the paper, we will often say "communication scheme" with a particular "rate" as shorthand for an *infinite sequence* of communication schemes for increasing message lengths with a particular *asymptotic* rate.

The main question we seek to answer in this work is: For a fixed $\delta \in [0, 1]$, what is the largest error fraction that can be tolerated by an infinite sequence of communication schemes with feedback fraction at most δ? We can ask this question for both deterministic as well as randomized communication schemes.

▶ **Definition 2.** For any $\delta \in [0, 1]$, we define $\Gamma^{\text{det}}(\delta)$ and $\Gamma^{\text{rand}}(\delta)$ as follows:

- $\Gamma^{\text{det}}(\delta)$ is the supremum over γ such that there exists a deterministic communication scheme with δ-feedback that tolerates an error fraction of γ and has constant rate.[1]
- $\Gamma^{\text{rand}}(\delta)$ is the supremum over γ such that there exists a randomized communication scheme with δ-feedback that tolerates an error fraction of γ and has constant rate.

We know that error correcting codes with distance $1/2 - \varepsilon$ exist for all $\varepsilon > 0$. Thus, we get that an error fraction of $\gamma = 1/4 - \varepsilon$ can be tolerated even without having any feedback, and thus, $\Gamma^{\text{rand}}(\delta) \geq \Gamma^{\text{det}}(\delta) \geq 1/4$ for all $\delta \geq 0$.

2.2 Upper bounds on the tolerable error fraction

It is known that for $\delta = 1$, if the communication scheme uses the *mirror feedback* structure, then no communication scheme can tolerate a $1/3$ error fraction for arbitrarily large message length [1]. The mirror feedback structure means that the communication protocol consists of alternating forward and feedback rounds, where each feedback bit sent by Bob is simply the bit that he has received from Alice in the preceding round.

▶ **Observation 3.** *If $\delta = 1$, we can assume without loss of generality that any deterministic one-way communication scheme with noiseless δ-feedback has only* mirror feedback*, namely, after every bit sent by Alice, Bob simply sends back the (potentially corrupted) bit he received.*

The observation follows because if Bob were to send back the precise bits that he receives from Alice, then Alice can compute any deterministic function of the same and thus any 1-feedback protocol can be simulated by using only mirror feedback. Combined with the

[1] rate that can depend on γ and δ, but not on the length of the input $\mathbf{x}$.

upper limit of $1/3$ from [1], we get that for any $\delta > 0$, no *deterministic* communication scheme with δ-feedback can tolerate an error fraction of $1/3$. For completeness, we give a proof of this result in Appendix A.

▶ **Theorem 4.** *For any $\delta \geq 0$, we have that $\Gamma^{\mathrm{det}}(\delta) \leq 1/3$.*

2.3 Main results

We prove two results that provide lower bounds on $\Gamma^{\mathrm{rand}}(\delta)$ and $\Gamma^{\mathrm{det}}(\delta)$, respectively. Our first result presents a randomized communication scheme that tolerates any error fraction $\gamma < 1/3$ for any $\delta > 0$.

▶ **Theorem 5.** *For any $\delta > 0$, we have that $\Gamma^{\mathrm{rand}}(\delta) \geq 1/3$. Namely, for any $\delta > 0$ and for all $\varepsilon > 0$, and $\gamma = 1/3 - \varepsilon$, there is a randomized communication scheme with noiseless δ-feedback that tolerates an error fraction of γ. Furthermore, one can achieve a rate of communication of $\Omega(\varepsilon\delta)$ with failure probability $\exp(-\Omega(k))$, where k is the length of the message being transmitted.*

Our second result presents a 'derandomization' of the underlying randomized communication scheme of Theorem 5 that beats the $1/4$ bound achieved by error correcting codes for all $\delta > 0$. The tolerable error fraction becomes $1/3$ for $\delta \geq 2/3$, which is optimal.

▶ **Theorem 6.** *Define $f : (0, 1] \to \mathbb{R}$ as follows:*

$$f(\delta) \overset{\mathrm{def}}{=} \begin{cases} \frac{1}{3}, & \text{if } \frac{2}{3} \leq \delta \leq 1 \\ \max\left\{\frac{\delta(r+1)}{2}, \frac{r+2}{4r+7}\right\}, & \text{if } 0 < \delta < \frac{2}{3} \end{cases} \qquad \text{where } r = r(\delta) \overset{\mathrm{def}}{=} \left\lfloor \frac{1}{2\delta} - \frac{3}{4} \right\rfloor$$

Then, for any $\delta \in (0, 1]$, $\Gamma^{\mathrm{det}}(\delta) \geq f(\delta)$. Namely, for any $\delta > 0$ and for all $\epsilon > 0$, there is a deterministic communication scheme with δ-feedback that tolerates an error fraction of $\gamma = f(\delta) - \varepsilon$ such that the rate of communication is $\Omega(\varepsilon\delta)$.

▶ Remark. The function f defined in Theorem 6 is a monotonically increasing piecewise linear function that is continuous on the interval in which it is defined (see Figure 2). Moreover $\lim_{\delta \to 0^+} f(\delta) = 1/4$, and one can easily tolerate any error fraction less than $\frac{1}{4}$ with zero feedback by simply using a binary error-correcting code with relative distance of twice the desired error fraction. For the other extremal case, namely $\delta = 1$, the protocol $\pi_1^{\mathrm{det}}(\gamma)$ (Figure 3 adapted from [7]) can be used to tolerate any error fraction less than $f(1) = 1/3$. The main contribution of this work is to establish the achievability of error fractions up to $f(\delta)$ for *intermediate* values of $\delta \in (0, 1)$. This shows that any non-zero feedback fraction allows us to beat the $1/4$ limit on the tolerable error fraction in the presence of no feedback.

Note that $\Gamma^{\mathrm{det}}(\delta) \leq 1/3$ for all $\delta \in [0, 1]$ (as implied by Theorem 4), and Theorem 5 shows us that randomized communication schemes are able to get arbitrarily close to this limit for *all* non-zero feedback fractions. We do not know whether the same is true for deterministic communication schemes, as the positive result of Theorem 6 exhibits a gap to the $1/3$ upper bound for $0 < \delta < 2/3$. We leave the question of whether $\Gamma^{\mathrm{det}}(\delta) = 1/3$ for all δ as an open problem. The error fractions tolerated by our communication schemes are summarized in Figure 2.

3 Deterministic communication scheme with full feedback

For completeness, we first present a communication scheme $\pi_1^{\mathrm{det}}(\gamma)$ with 1-feedback that tolerates an error fraction of $\gamma = 1/3 - \varepsilon$ (where $\varepsilon > 0$). Such a protocol was obtained

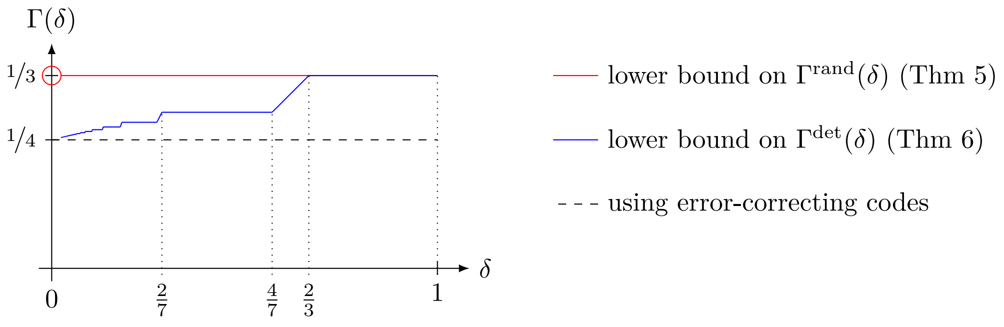


Figure 2 Maximum error fraction tolerated as function of δ.

previously by Berlekamp [1]. We present a very simple scheme (in Figure 3) that was implicit in [7]. It is easy to see that the communication scheme presented has rate $\Theta(\varepsilon)$.

Correctness of $\pi_1^{\text{det}}(\gamma)$

We first introduce a couple of notations. Firstly, for all $s \in \{0,1\}^*$, define len(s) to be the length of s. Next, for strings s which do not contain consecutive 0's, we define the *weight* of s as follows.

▶ **Definition 7** (Weight of a string). Given string $s \in \{0,1\}^*$, such that s has no consecutive 0's, we define the *weight* of s as follows: Suppose s breaks into a 0's, b 1's and c 10's, with the smallest number of pieces. We define $\text{wt}(s) \overset{\text{def}}{=} 2a + 2b + c$.

For example,

- wt('0110101') $= 2 \cdot 1 + 2 \cdot 2 + 1 \cdot 2 = 8$, since '0110101' = '0' + '1' + '10' + '10' + '1'.
- wt('111010') $= 2 \cdot 0 + 2 \cdot 2 + 1 \cdot 2 = 6$, since '111010' = '1' + '1' + '10' + '10'.

Note that since s does not contain consecutive 0's, it follows that $a = 1$ if s starts with a '0' and $a = 0$ otherwise.

To prove that the communication scheme $\pi_1^{\text{det}}(\gamma)$ in Figure 3 tolerates an error fraction of γ, we define a potential function as $\Phi = \Phi(T) \overset{\text{def}}{=} \text{len}(T_{\text{right}}) - \text{wt}(T_{\text{wrong}})$. Note that the scheme in Figure 3 ensures that T_{wrong} never has consecutive 0's, and thus $\text{wt}(T_{\text{wrong}})$ is always well defined.

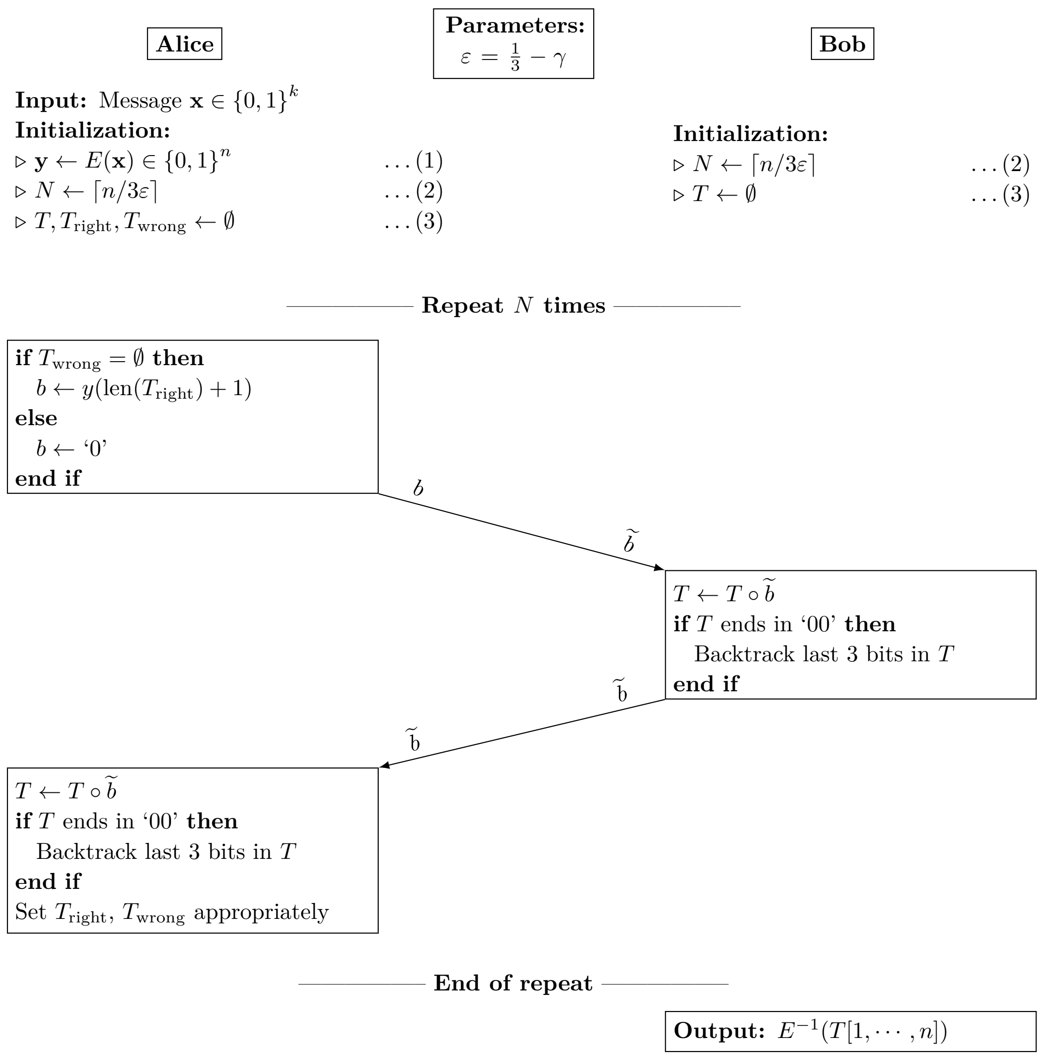
The following easy proposition shows how Φ changes after each round of communication. The proof appears in Appendix B.

▶ **Proposition 8.** *After each round of communication, if Alice's bit is received correctly by Bob, then Φ increases by at least 1. On the other hand, if Alice's bit is received incorrectly by Bob, then Φ decreases by at most 2.*

As an easy corollary of Proposition 8, we get a lower bound on Φ at the end of the protocol, as a function of the error fraction.

▶ **Corollary 9.** *If γ fraction of Alice's transmissions in communication scheme $\pi_1^{\text{det}}(\gamma)$ (given in Figure 3) are corrupted, then at the end of the protocol, $\Phi \geq (1 - 3\gamma) N$.*

Proof. We have that there are $(1 - \gamma)N$ forward rounds of the protocol which are not corrupted in which Φ increases by 1, while γN rounds which are corrupted in which Φ



Alice | **Parameters:** $\varepsilon = \frac{1}{3} - \gamma$ | **Bob**

```
Alice
Input: Message $\mathbf{x} \in \{0,1\}^k$
Initialization:
▷ $\mathbf{y} \leftarrow E(\mathbf{x}) \in \{0,1\}^n$                ...(1)
▷ $N \leftarrow \lceil n/3\varepsilon \rceil$                          ...(2)
▷ $T, T_{\text{right}}, T_{\text{wrong}} \leftarrow \emptyset$        ...(3)

Bob
Initialization:
▷ $N \leftarrow \lceil n/3\varepsilon \rceil$                          ...(2)
▷ $T \leftarrow \emptyset$                                             ...(3)

—— Repeat $N$ times ——

Alice:
if $T_{\text{wrong}} = \emptyset$ then
    $b \leftarrow y(\text{len}(T_{\text{right}}) + 1)$
else
    $b \leftarrow$ '0'
end if

Alice sends $b$; Bob receives $\widetilde{b}$

Bob:
$T \leftarrow T \circ \widetilde{b}$
if $T$ ends in '00' then
    Backtrack last 3 bits in $T$
end if

Bob sends $\widetilde{b}$; Alice receives $\widetilde{b}$

Alice:
$T \leftarrow T \circ \widetilde{b}$
if $T$ ends in '00' then
    Backtrack last 3 bits in $T$
end if
Set $T_{\text{right}}, T_{\text{wrong}}$ appropriately

—— End of repeat ——

Bob:
Output: $E^{-1}(T[1, \cdots, n])$
```

(1) $E(\mathbf{x})$ is a simple encoding of $\mathbf{x}$ such that $E(\mathbf{x})$ does not contain any consecutive '0's. One way to do this: add a '1' between two consecutive bits, making $n = 2k$. We will refer to the bits of $\mathbf{y}$ as $y(i)$. For $i > n$, we will assume $y(i) =$ '1'.

(2) N is the number of rounds

(3) T is the transcript as maintained by Bob. However, Alice is able to decompose T as $T = T_{\text{right}} \circ T_{\text{wrong}}$, where T_{right} is the largest prefix of T which exactly matches the prefix of $\mathbf{y}$ of the same length, and T_{wrong} is the remaining part in T. Basically, T_{wrong} is the part of the transcript which starts with an incorrectly received bit, and so all the following bits have to be erased before proceeding further.

Figure 3 Communication scheme with complete feedback : $\pi_1^{\text{det}}(\gamma)$.

decreases by at most 2. Thus, by Proposition 8, we see that at the end of the protocol,

$$\Phi(T) \geq 1 \cdot (1-\gamma)N - 2 \cdot \gamma N = (1-3\gamma)\, N$$

◀

Thus, by the above Corollary, if $\gamma = {}^1\!/\!{}_3 - \varepsilon$ fraction of the forward rounds of protocol $\pi_1^{\text{det}}(\gamma)$ are corrupted, then at the end of the protocol we will have,

$$\text{len}(T_{\text{right}}) \geq \Phi(T) \geq 3\varepsilon N$$

By our choice of N, we have that $3\varepsilon N \geq n$. Therefore, at the end of the protocol, $\text{len}(T_{\text{right}}) \geq n$, meaning that the first n bits of T are exactly $\mathbf{y}$. Hence, Bob is able to decode $\mathbf{x}$ correctly by just applying E^{-1} on the first n bits of T.

4 Randomized communication scheme with partial feedback

In this section, we prove Theorem 5 by giving a randomized protocol $\pi_\delta^{\text{rand}}(\gamma)$ with δ-feedback (for any $\delta \in (0,1]$), that tolerates an error fraction of $\gamma = {}^1\!/\!{}_3 - \varepsilon$ (where $\varepsilon > 0$) and has a rate of $\Theta(\varepsilon\delta)$. The full details of $\pi_\delta^{\text{rand}}(\gamma)$ can be found in Figure 4. The main idea is as follows:

We wish to simulate $\pi_1^{\text{det}}(\gamma)$ with a smaller feedback fraction. We break the protocol into $N_0 = \lceil 2n/3\varepsilon \rceil$ iterations, where each iteration roughly corresponds to one forward and feedback round of $\pi_1^{\text{det}}(\gamma)$. In each iteration, Alice sends $D = \lceil 2/\delta \rceil$ bits, namely $\mathbf{c} = (c_1, \cdots, c_D) = b^D$ (i.e. D copies of bit b she would have sent in π_1^{det}). Bob receives a set of symbols $\widetilde{\mathbf{c}} = (\widetilde{c}_1, \cdots, \widetilde{c}_D)$. He uses a 'soft decoding' scheme to obtain $\widetilde{b}$ which is his interpretation of what b must have been. Let m be the number of '1's present in $\widetilde{\mathbf{c}}$. If $m \leq D/2$, then he interprets $\widetilde{b}$ as '0' with probability $1 - 2m/D$, and if $m > D/2$, then he interprets $\widetilde{b}$ to be '1' with probability $2m/D - 1$. In other cases, Bob interprets $\widetilde{b}$ to be '?'. If $\widetilde{b} \neq$ '?', then Bob then makes appropriate progress on the protocol $\pi_1^{\text{det}}(\gamma)$. As feedback, Bob sends back the value of $\widetilde{b}$ (which takes 2 bits of feedback). Thus, the entire protocol uses $N = \lceil 2/\delta \rceil\, N_0$ number of forward bits of communication and $2N_0$ bits of feedback, and thus it uses $(2/\lceil 2/\delta \rceil)$-feedback (which is at most δ-feedback), and has rate $\Theta(\varepsilon\delta)$. All that remains to show now is that this protocol tolerates an error fraction of $\gamma = {}^1\!/\!{}_3 - \varepsilon$.

Correctness of $\pi_\delta^{\text{rand}}(\gamma)$

We show that for any $\delta \in (0,1]$ and $\gamma < {}^1\!/\!{}_3$, the communication scheme $\pi_\delta^{\text{rand}}(\gamma)$ tolerates an error fraction of γ with constant rate. This immediately implies Theorem 5.

▶ **Theorem 10.** *For any $\delta \in (0,1]$ and $\gamma = {}^1\!/\!{}_3 - \varepsilon$ (where $\varepsilon > 0$), the communication scheme $\pi_\delta^{\text{rand}}(\gamma)$ (from Figure 4) tolerates an error fraction of γ with rate being $\Theta(\varepsilon\delta)$.*

Proof. Suppose that an adversary corrupts at most $\gamma N = ({}^1\!/\!{}_3 - \varepsilon)\, N$ of Alice's transmissions in $\pi_\delta^{\text{rand}}(\gamma)$ (recall that $N = \lceil 2/\delta \rceil\, N_0$). We will show that for any fixed error pattern, Bob can successfully recover Alice's message with high probability.

Consider the potential function Φ that was used for proving the correctness of protocol $\pi_1^{\text{det}}(\gamma)$. In any iteration $1 \leq i \leq N_0$ of the simulating protocol $\pi_\delta^{\text{rand}}(\gamma)$ (in Figure 4) above, with e fraction of errors (i.e. with eD errors), the potential function Φ changes by an amount X_i given as follows (see Proposition 8):

- *if* $e_i \leq 1/2$, then with probability $1 - 2e_i$, Φ increases by at least $X_i = 1$, and with probability $2e_i$ the potential function remains unchanged ($X_i = 0$). In expectation, Φ increases by at least $\mathbb{E}[X_i] = (1 - 2e_i)$.

Alice | **Parameters:** $\varepsilon = \frac{1}{3} - \gamma$ | **Bob**

Input: Message $\mathbf{x} \in \{0,1\}^k$

Initialization: (Alice)

```
▷ $\mathbf{y} \leftarrow E(\mathbf{x}) \in \{0,1\}^n$            ...(1)
▷ $N_0 \leftarrow \lceil 2n/3\varepsilon \rceil$                 ...(2)
▷ $T, T_{\text{right}}, T_{\text{wrong}} \leftarrow \emptyset$   ...(3)
▷ $D \leftarrow \lceil 2/\delta \rceil$                          ...(4)
```

Initialization: (Bob)

```
▷ $N_0 \leftarrow \lceil 2n/3\varepsilon \rceil$   ...(2)
▷ $T \leftarrow \emptyset$                         ...(3)
▷ $D \leftarrow \lceil 2/\delta \rceil$            ...(4)
```

Repeat N_0 times

Alice:

```
if $T_{\text{wrong}} = \emptyset$ then
  $b \leftarrow y(\text{len}(T_{\text{right}}) + 1)$
else
  $b \leftarrow$ '0'
end if
$\mathbf{c} \leftarrow b^D$
```

$\mathbf{c} \longrightarrow \widetilde{\mathbf{c}}$

Bob:

```
$m \leftarrow \left|\{i : \widetilde{c}_i = 1\}\right|$
if $m \le D/2$ then
  $\widetilde{b} \leftarrow \begin{cases} \text{'0'} & \text{w.p. } 1 - 2m/D \\ \text{'?'} & \text{w.p. } 2m/D \end{cases}$
else
  $\widetilde{b} \leftarrow \begin{cases} \text{'1'} & \text{w.p. } 2m/D - 1 \\ \text{'?'} & \text{w.p. } 2 - 2m/D \end{cases}$
end if
if $\widetilde{b} \neq$ '?' then
  $T \leftarrow T \circ \widetilde{b}$
  if $T$ ends in '00' then
    Backtrack last 3 bits in $T$
  end if
  Set $T_{\text{right}}, T_{\text{wrong}}$ appropriately
end if
```

$\widetilde{b} \longrightarrow \widetilde{b}$

Alice:

```
if $\widetilde{b} \neq$ '?' then
  $T \leftarrow T \circ \widetilde{b}$
  if $T$ ends in '00' then
    Backtrack last 3 bits in $T$
  end if
  Set $T_{\text{right}}, T_{\text{wrong}}$ appropriately
end if
```

End of repeat

Output: $E^{-1}(T[1, \cdots, n])$

(1), (2) and (3) as defined in the scheme $\pi_1^{\text{det}}(\gamma)$ (Figure 3)

(4) D is the number of bits sent in every iteration.

Figure 4 Randomized communication scheme with complete feedback : $\pi_\delta^{\text{rand}}(\gamma)$.

- *if* $e_i > 1/2$, then with probability $2e_i - 1$, Φ decreases by at most 2, i.e. increases by at least $X_i = -2$, and with probability $2 - 2e_i$ the potential function remains unchanged ($X_i = 0$). In expectation, Φ increases by at least $\mathbb{E}[X_i] = -(4e_i - 2)$.

Suppose we use N_0 phases of the above protocol and suppose that the fraction of errors that the adversary makes in each of these N_0 phases is $e_1, e_2, \cdots, e_{N_0}$, respectively. Let $S_1 = \{i : e_i \leq 1/2\}$ and $S_2 = \{i : e_i > 1/2\}$.

Thus, the expected value of the potential function at the end of N_0 phases will be,

$$\begin{aligned}
\mathbb{E}[\Phi] &\geq \sum_{i=1}^{N} \mathbb{E}[X_i] \\
&= \sum_{i \in S_1} (1 - 2e_i) - \sum_{j \in S_2} (4e_j - 2) \\
&= \sum_{i \in S_1 \cup S_2} (1 - 2e_i) - \sum_{j \in S_2} (2e_j - 1) \\
&= N_0 - \sum_{i=1}^{N_0} 2e_i - \sum_{j \in S_2} (2e_j - 1)
\end{aligned}$$

We want to bound $\mathbb{E}[\Phi]$ from below. Firstly, $\sum_{i=1}^{N_0} e_i \leq N_0 (1/3 - \varepsilon)$. Also, since each $e_j \leq 1$, we have that $|S_2| \geq \sum_{j \in S_2} e_j$, and hence $\sum_{j \in S_2} (2e_j - 1) \leq \sum_{j \in S_2} e_j \leq \sum_{i=1}^{N_0} e_i \leq N_0 (1/3 - \varepsilon)$. Thus, from above equations we have that,

$$\mathbb{E}[\Phi] \geq N_0 - 2N_0 \left(\frac{1}{3} - \varepsilon \right) - N_0 \left(\frac{1}{3} - \varepsilon \right) = 3\varepsilon N_0.$$

Since we choose $N_0 = \lceil 2n/3\varepsilon \rceil$, we have that $\mathbb{E}[\Phi] \geq 2n$. Also, note that either $X_i \in [0, 2]$ or $X_i \in [-2, 0]$ for all i. Thus, using Hoeffding's concentration inequality, we have

$$\begin{aligned}
\Pr[\Phi \geq n] &\geq \Pr[X_1 + X_2 + \cdots + X_{N_0} \geq n] \\
&\geq 1 - \Pr\left[\left| X_1 + X_2 + \cdots + X_{N_0} - \mathbb{E}\left[\textstyle\sum_{i=1}^{N_0} X_i \right] \right| \geq n \right] \\
&\geq 1 - 2e^{-2n^2/4N_0} \\
&\geq 1 - 2e^{-3\varepsilon n/4} \qquad [\text{putting } N_0 = \lceil 2n/3\varepsilon \rceil]
\end{aligned}$$

Recall that if $\Phi \geq n$ at the end of the protocol, then Bob is able to decode Alice's message correctly. Thus, we conclude that $\pi_\delta^{\text{rand}}(\gamma)$ works with a failure probability of at most $\exp(-\Omega(k))$ (since $n = \Theta(k)$). ◀

▶ Remark. In the application of the Hoeffding's inequality we required the error pattern to be fixed, that is, the adversary pre-commits to the error pattern (although unknown to Alice and Bob). We feel that this is only a technical difficulty and it should be generalizable to adaptive adversaries. Nevertheless, in the next section, we de-randomize this protocol, although with a smaller error fraction. For deterministic protocols, it does not matter whether the errors are adaptive or not, as we require a worst-case guarantee.

5 Deterministic communication schemes with partial feedback

In this section, we prove Theorem 6 by giving a deterministic protocol $\pi_\delta^{\text{det}}(\gamma)$ with δ-feedback (for any $\delta \in (0, 1]$). We first give a deterministic communication scheme with δ-feedback for $\delta = 2/(4r + 3)$ for any $r \in \mathbb{N}$ (this scheme is described in Section 5.1), which we obtain by a certain derandomization of π_δ^{rand}. Next, for $\frac{2}{4r+7} < \delta < \frac{2}{4r+3}$, we give a communication scheme that *interpolates* between $\pi^{\text{det}}_{2/(4r+3)}$ and $\pi^{\text{det}}_{2/(4r+7)}$ (the full details are described in Section 5.2).

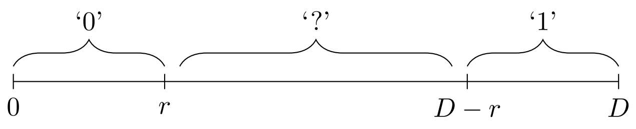
5.1 Deterministic communication scheme for $\delta = 2/(4r+3)$

In this section, we present a deterministic communication scheme $\pi^{\text{det}}_{\delta}(\gamma)$, where $\delta = 2/(4r+3)$ for any $r \in \mathbb{N}$. We show that this scheme tolerates an error fraction of $\gamma = (r+1)/(4r+3) - \varepsilon$ and has rate $\Theta(\varepsilon\delta)$. We obtain this by instantiating the communication scheme $\pi^{\text{det}}_{\delta}(\gamma)$ in Figure 5, with $D_i = (4r+3)$ and $r_i = r$ for all i. The main idea behind the protocol is as follows:

We will consider a protocol identical to $\pi^{\text{rand}}_{\delta}$, except that in each of the N_0 iterations, Bob chooses a value ('0,' '1,' or '?') for $\widetilde{b}$ in a *deterministic* fashion. In particular, in any iteration, Alice sends $D = (4r+3) = 2/\delta$ bits (say $\mathbf{c}$ given by b^D), which Bob then receives as $\widetilde{\mathbf{c}}$. Let m be the number of 1's in $\widetilde{\mathbf{c}}$. Bob chooses $\widetilde{b}$ as follows:

$$\widetilde{b} \quad \leftarrow \quad \begin{cases} \text{‘0’} & \text{if } m \le r \\ \text{‘1’} & \text{if } m \ge D - r \\ \text{‘?’} & \text{if } r < m < D - r \end{cases}$$

Thus, $r+1$ is the minimum number of bits of $\mathbf{c}$ that an adversary must corrupt in order to force Bob to interpret the round as a '?', and $D - r$ is the minimum number of bits of $\mathbf{c}$ that must be corrupted in order to force Bob to interpret the round as opposite of the bit that Alice intended to send. The decoding strategy of Bob is summarized in the following figure:



Thus, the entire protocol uses $N = DN_0 = (2/\delta)N_0$ number of forward bits of communication and $2N_0$ bits of feedback, and thus it uses δ-feedback. We will choose $N_0 = \lceil n/3\varepsilon \rceil$, and hence the rate is $\Theta(\varepsilon\delta)$. So all that remains to show now is that this protocol tolerates an error fraction of $\gamma = (r+1)/(4r+3) - \varepsilon$. The following proposition (which is an analogue of Corollary 9) will be useful in proving the same.

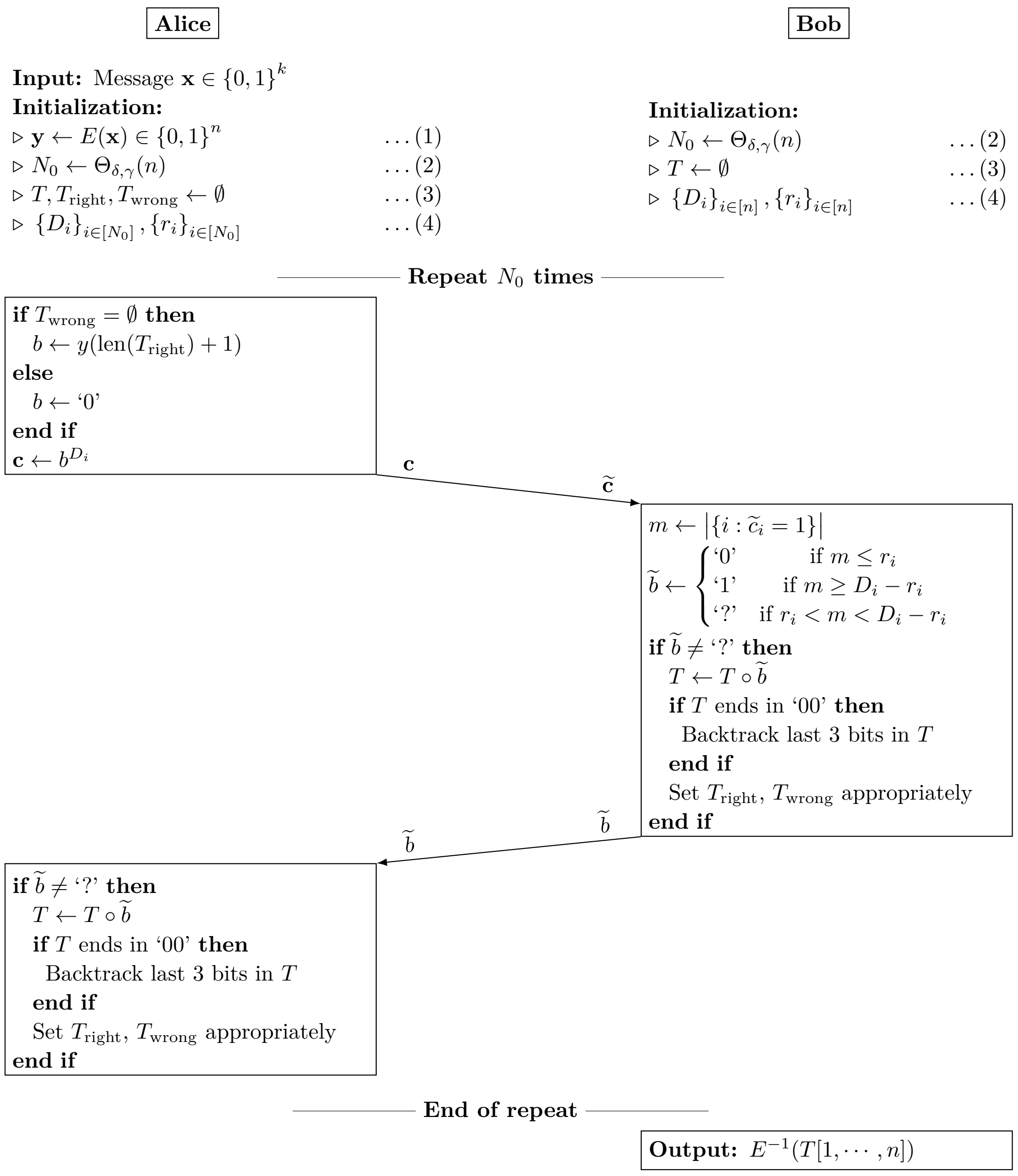
▶ **Proposition 11.** *For $\delta = 2/(4r+3)$, if γ fraction of Alice's transmissions in communication scheme $\pi^{\text{det}}_{\delta}(\gamma)$ (from Figure 5) are corrupted, then at the end of the protocol, $\Phi \ge \left(1 - \frac{(4r+3)\gamma}{r+1}\right) N_0$.*

Proof. In the communication scheme $\pi^{\text{det}}_{\delta}(\gamma)$, any iteration is considered *correctly decoded* if $\widetilde{b}$ is set to be the bit that appears in $\mathbf{c}$, while it is said to be *incorrectly decoded* if $\widetilde{b}$ is set to the opposite bit. An iteration is considered *ambiguous* if $\widetilde{b}$ is set to be '?'.

Now, suppose an adversary corrupts the scheme such that AN_0, BN_0, and CN_0 are the number of iterations that are ambiguous, correctly decoded and incorrectly decoded respectively. Note that the adversary has a total budget of γDN_0 corruptions, where γ is the overall error fraction of the protocol. Thus, we have the following constraints:

$$A + B + C = 1 \tag{1}$$

$$(r+1)A + (D-r)C \le \gamma D \tag{2}$$



Alice

Input: Message $\mathbf{x} \in \{0,1\}^k$

Initialization:

```
▷ $\mathbf{y} \leftarrow E(\mathbf{x}) \in \{0,1\}^n$                              ...(1)
▷ $N_0 \leftarrow \Theta_{\delta,\gamma}(n)$                                        ...(2)
▷ $T, T_{\text{right}}, T_{\text{wrong}} \leftarrow \emptyset$                      ...(3)
▷ $\{D_i\}_{i\in[N_0]}, \{r_i\}_{i\in[N_0]}$                                        ...(4)
```

Bob

Initialization:

```
▷ $N_0 \leftarrow \Theta_{\delta,\gamma}(n)$                                        ...(2)
▷ $T \leftarrow \emptyset$                                                          ...(3)
▷ $\{D_i\}_{i\in[n]}, \{r_i\}_{i\in[n]}$                                            ...(4)
```

Repeat N_0 times

Alice:

```
if $T_{\text{wrong}} = \emptyset$ then
  $b \leftarrow y(\text{len}(T_{\text{right}}) + 1)$
else
  $b \leftarrow$ '0'
end if
$\mathbf{c} \leftarrow b^{D_i}$
```

$\mathbf{c}$ → $\widetilde{\mathbf{c}}$

Bob:

```
$m \leftarrow |\{i : \widetilde{c}_i = 1\}|$
$\widetilde{b} \leftarrow \begin{cases} \text{'0'} & \text{if } m \le r_i \\ \text{'1'} & \text{if } m \ge D_i - r_i \\ \text{'?'} & \text{if } r_i < m < D_i - r_i \end{cases}$
if $\widetilde{b} \neq$ '?' then
  $T \leftarrow T \circ \widetilde{b}$
  if $T$ ends in '00' then
    Backtrack last 3 bits in $T$
  end if
  Set $T_{\text{right}}, T_{\text{wrong}}$ appropriately
end if
```

$\widetilde{b}$ → $\widetilde{b}$

Alice:

```
if $\widetilde{b} \neq$ '?' then
  $T \leftarrow T \circ \widetilde{b}$
  if $T$ ends in '00' then
    Backtrack last 3 bits in $T$
  end if
  Set $T_{\text{right}}, T_{\text{wrong}}$ appropriately
end if
```

End of repeat

Output: $E^{-1}(T[1,\cdots,n])$

(1), (2) and (3) as defined in the scheme π_1^{det} (Figure 3)

(4) D_i is the number of forward bits sent in iteration i. r_i is the deterministic threshold used by Bob to interpret the received stream as either '0', '?' or '1' in iteration i. The total number of forward bits is $\sum_i D_i$, and total number of feedback bits is $2N_0$. The exact choice of D_i's and r_i's will depend on δ and γ.

Figure 5 Template for deterministic communication scheme with complete feedback : $\pi_\delta^{\text{det}}(\gamma)$.

Equation 1 follows because the total number of iterations is N_0. Inequality 2 follows because, we may assume without loss of generality, that an adversary corrupts 0 bits for an iteration that is correctly decoded, $D-r$ bits for an iteration that is incorrectly decoded, and $r+1$ bits for an iteration that is ambiguous. This is because these are the minimum number of bits needed to be corrupted for each category, and an adversary cannot possibly gain by corrupting more than the minimum.

From Proposition 8, we have that $\Phi \geq (B-2C)N_0$. And so, we wish to lower bound $B-2C$. For this, we subtract $\frac{3}{D-r}$ times Inequality (2) from Equation (1) to obtain

$$B - 2C - A\left(\frac{3(r+1)}{D-r} - 1\right) \geq 1 - \frac{3\gamma D}{D-r}$$

Since $D = 4r+3$, it follows that $\frac{3(r+1)}{D-r} = 1$, and so,

$$B - 2C \geq 1 - \frac{3\gamma D}{D-r} = 1 - \frac{(4r+3)\gamma}{r+1}$$

Thus, we get that at the end of the protocol, $\Phi \geq \left(1 - \frac{(4r+3)\gamma}{r+1}\right) N_0$. ◀

By the above Proposition, if $N_0 = \lceil n/3\varepsilon \rceil$ and $\gamma = \frac{r+1}{4r+3} - \epsilon$ fraction of Alice's transmissions in $\pi_\delta^{\det}(\gamma)$ are corrupted, then at the end of the protocol we will have,

$$\operatorname{len}(T_{\text{right}}) \geq \Phi \geq \left(1 - \frac{4r+3}{r+1}\left(\frac{r+1}{4r+3} - \epsilon\right)\right) N_0 = \frac{(4r+3)\epsilon}{r+1} N_0 \geq n$$

This guarantees that at the end of the protocol $\operatorname{len}(T_{\text{right}}) \geq n$ meaning the first n bits of T are exactly $\mathbf{y}$ and Bob can correctly decode Alice's message by applying E^{-1} on the first n bits of T.

5.2 Deterministic communication schemes for all $\delta \in (0, 1]$

In the previous section we gave a deterministic communication scheme $\pi_\delta^{\det}(\gamma)$ for $\delta = 2/(4r+3)$, and showed that one can tolerate an error fraction of up to $\frac{r+1}{4r+3}$. In the section, we give a communication scheme $\pi_\delta^{\det}(\gamma)$ for any feedback fraction $\delta \in \left(\frac{2}{4r+7}, \frac{2}{4r+3}\right)$, that can tolerate an error fraction of $\gamma = \frac{\delta(r+1)}{2} - \varepsilon$ (where $\varepsilon > 0$).

The key is to "interpolate" the protocols $\pi_\delta^{\det}$ that we obtain for $\delta = \frac{2}{D}$ between $D = 4r+3$ and $D = 4(r+1)+3$. In particular, we will have that for the first qN_0 iterations, we use $D_i = 4r+7$ and $r_i = r+1$, and the later $(1-q)N_0$ iterations, we use $D_i = 4r+3$ and $r_i = r$. We let $N_0 \geq (r+1)\delta n/2\varepsilon$. This also gives that the rate is $\Theta(\varepsilon\delta)$ (since $\delta \geq 2/(4r+7)$).

Let A_1N_0, B_1N_0, and C_1N_0 be the number of iterations that are ambiguous, correctly decoded and incorrectly decoded respectively in the first qN_0 iterations. Similarly, let A_2N_0, B_2N_0, and C_2N_0 be the number of iterations that are ambiguous, correctly decoded and incorrectly decoded respectively in the later $(1-q)N_0$ iterations. Also note that the adversary has a total budget of $\gamma(q(4r+7)+(1-q)(4r+3))N_0$ corruptions, where γ is the overall error fraction of the protocol.

Note that $\delta = {}^2/_{(q(4r+7)+(1-q)(4r+3))} = {}^2/_{(4q+4r+3)}$ and hence $q = \frac{1}{2\delta} - \frac{(4r+3)}{4}$. We have the following constraints:

$$\begin{aligned} A_1 + B_1 + C_1 &= q && (3)\\ A_2 + B_2 + C_2 &= 1-q && (4)\\ (r+2)A_1 + (r+1)A_2 + (3r+6)C_1 + (3r+3)C_2 &\leq \gamma(4q + (4r+3)) && (5) \end{aligned}$$

From Proposition 8, we have that the potential at the end of the protocol satisfies $\Phi \geq (B_1 + B_2 - 2C_1 - 2C_2)N_0$. Thus, we wish to lower bound $(B_1 + B_2 - 2C_1 - 2C_2)$. We add equations 3 and 4 and subtract (1/(r+1)) times inequality 5, to get,

$$B_1 + B_2 - 2C_1 - 2C_2 - \frac{A_1}{r+1} - \frac{3C_1}{r+1} \geq 1 - \frac{\gamma(4q+4r+3)}{r+1} = 1 - \frac{2\gamma}{(r+1)\delta}$$

Since $A_1, C_1 \geq 0$ and $\gamma = (r+1)\delta/2 - \varepsilon$, we get that,

$$\Phi \geq (B_1 + B_2 - 2C_1 - 2C_2)N_0 \geq \frac{2\varepsilon N_0}{(r+1)\delta} \geq n$$

where the last inequality follows because we chose $N_0 \geq (r+1)\delta n/2\varepsilon$. This guarantees that at the end of the protocol $\text{len}(T_{\text{right}}) \geq \Phi \geq n$ meaning the first n bits of T are exactly $\mathbf{y}$ and Bob can correctly decode Alice's message by applying E^{-1} on the first n bits of T.

5.3 Putting it all together

Proof of Theorem 6. In Section 5.2, we showed that when $\frac{2}{4r+7} \leq \delta \leq \frac{2}{4r+3}$, we have that $\Gamma^{\text{det}}(\delta) \geq (r+1)\delta/2$. But from Section 5.1, we obtained that $\Gamma^{\text{det}}\left(\frac{2}{4r+7}\right) \geq \frac{r+2}{4r+7}$, and thus by monotonicity of $\Gamma^{\text{det}}(\delta)$, we have that for all $\delta \geq \frac{2}{4r+7}$, $\Gamma^{\text{det}}(\delta) \geq \frac{r+2}{4r+7}$. Combining the two results we get that,

$$\forall r \in \mathbb{Z}_{\geq 0} \quad \forall \delta \in \left[\frac{2}{4r+7}, \frac{2}{4r+3}\right] \quad \Gamma^{\text{det}}(\delta) \geq \max\left\{\frac{(r+1)\delta}{2}, \frac{r+2}{4r+7}\right\}$$

◀

6 Discussion

We have introduced the notion of communication schemes under partial noiseless feedback as a natural interpolation between two familiar settings, namely, the problem of transmission over a binary channel with adversarial errors (achievable by the use of error-correcting codes) as well as the problem of transmission over a binary feedback channel (achievable by the protocol in [1]). The results of this work show that the availability of a non-zero fraction of feedback, however small, allows Alice to communicate a message to Bob in a way that tolerates an adversarial error fraction of more than 1/4, the limit for error-correcting codes. An upper bound of 1/3 on the tolerable error fraction for a deterministic communication scheme holds for all feedback fractions $0 \leq \delta \leq 1$, and we show how to obtain a randomized communication scheme that tolerates any error fraction up to 1/3. Furthermore, we have shown deterministic communication schemes that tolerates error fractions of up to $f(\delta)$, where f is a monotonically increasing, piecewise linear, continuous function with $f(0) = 1/4$ and $f(1) = 1/3$. In particular, we have shown that our deterministic scheme can tolerate any error fraction less than 1/3 for all $\delta \geq 2/3$.

Our work points to several interesting directions for further investigation.

- Is the bound $\Gamma^{\text{det}}(\delta) \geq f(\delta)$ provided by Theorem 6 is tight? Currently we only know that $f(\delta) \leq \Gamma^{\text{det}}(\delta) \leq 1/3$ for $\delta < 2/3$. In particular, is it possible for a deterministic communication scheme to tolerate error fractions up to 1/3 for *all* δ in the way that the randomized scheme π_δ^{rand} can. One possible direction is to derandomize π_δ^{rand} in a more clever way that avoids loss in the error fraction tolerance. Otherwise, is it possible to prove better upper bounds than 1/3 on $\Gamma^{\text{det}}(\delta)$?

- In this work, we have considered only protocols over binary alphabets. It will be interesting to determine the limits on the tolerable error fraction for communication schemes with partial feedback that use symbols from non-binary alphabets as well as to find explicit communication schemes in this setting. Over an alphabet of size q, we know that error correcting codes can tolerate an error fraction of $(1 - 1/q)/2$, whereas, with noiseless feedback, one can tolerate an error fraction of up to $1/2$ (see [7] for example).
- In this work we only studied the model of noiseless feedback. It will be interesting to understand what bounds could be proved for the noisy feedback model, where the adversary is allowed to corrupt the feedback as well. An immediate question is whether it is even possible to correct more than $1/4$ fraction errors in this model (for any amount of feedback - where we measure the error budget as a fraction of the length of the entire protocol and not just Alice's transmissions).

Acknowledgements. We would like to thank Madhu Sudan for many helpful discussions and comments on the draft. PK would like to thank Dana Moshkovitz for general guidance and encouragement. AV would like to thank Venkatesan Guruswami for helpful conversations.

PK was supported in part by NSF grants CCF-1218547 and CCF-1420956. Part of this work was done while AV was visiting Microsoft Research New England, Cambridge, MA. AV was supported in part by NSF grant CCF-0963975.

References

1 Elwyn R. Berlekamp. *Block Coding with Noiseless Feedback*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1964.

2 Zvika Brakerski and Yael Tauman Kalai. Efficient interactive coding against adversarial noise. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 160–166, 2012.

3 Zvika Brakerski, Yael Tauman Kalai, and Moni Naor. Fast interactive coding against adversarial noise. *J. ACM*, 61(6):35, 2014.

4 Zvika Brakerski and Moni Naor. Fast algorithms for interactive coding. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 443–456, 2013.

5 Mark Braverman and Klim Efremenko. List and unique coding for interactive communication in the presence of adversarial noise. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 236–245, 2014.

6 Mark Braverman and Anup Rao. Toward coding for maximum errors in interactive communication. *IEEE Transactions on Information Theory*, 60(11):7248–7255, 2014.

7 Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Maximal noise in interactive communication over erasure channels and channels with feedback. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 11–20, 2015.

8 Ran Gelles and Bernhard Haeupler. Capacity of interactive communication over erasure channels and channels with feedback. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1296–1311, 2015.

9 Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient coding for interactive communication. *IEEE Transactions on Information Theory*, 60(3):1899–1913, 2014.

10 Mohsen Ghaffari and Bernhard Haeupler. Optimal error rates for interactive coding II: efficiency and list decoding. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 394–403, 2014.

11 Mohsen Ghaffari, Bernhard Haeupler, and Madhu Sudan. Optimal error rates for interactive coding I: adaptivity and other settings. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 – June 03, 2014*, pages 794–803, 2014.

12 E. N. Gilbert. A comparison of signalling alphabets. *Bell System Technical Journal*, 31:504–522, 1952.

13 Bernhard Haeupler. Interactive channel capacity revisited. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 226–235, 2014.

14 Gillat Kol and Ran Raz. Interactive channel capacity. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 715–724, 2013.

15 Morris Plotkin. Binary codes with specified minimum distance. *IRE Transactions on Information Theory*, 6(4):445–450, 1960.

16 Leonard J. Schulman. Communication on noisy channels: A coding theorem for computation. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 724–733, 1992.

17 Leonard J. Schulman. Deterministic coding for interactive communication. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 747–756, 1993.

18 Leonard J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996.

19 Claude E. Shannon. The zero error capacity of a noisy channel. *IRE Transactions on Information Theory*, 2(3):8–19, 1956.

20 Joel Spencer and Peter Winkler. Three thresholds for a liar. *Combinatorics, Probability & Computing*, 1:81–93, 1992.

21 R. R. Varshamov. Estimate of the number of signals in error correcting codes. *Dokl. Acad. Nauk SSSR*, 117:739–741, 1957.

A Upper bound on tolerable error fraction for mirror feedback

For completeness we give a proof of the following theorem, which was already proved by Berlekamp [1] and also later by Spencer-Winkler in the context of questions with liars [20].

▶ **Theorem 12.** *Any one-way communication scheme with noiseless feedback which uses the mirror feedback structure (that is, each feedback bit sent by Bob is simply the bit that he receives from Alice), cannot tolerate* $1/3$ *fraction of errors, as long as the input space of Alice has at least three elements.*

Proof. Let A, B and C be three possible inputs that Alice receives. Consider three parallel executions of any communication scheme with mirror feedback. We will show that there exists an adversary who can ensure that the view of Bob in two out of these three executions are the same by using only $1/3$ fraction errors in each of the executions. We describe the adversary below.

Let a_i, b_i and c_i be the bits sent by Alice in the i-th round of these three executions. Clearly at least two out of these three bits have to be the same. Thus, if one of these three bits is different from the other two, then the adversary will corrupt that bit, otherwise he will not corrupt any of the bits. This ensures that up to any round i, Bob's view of the protocol in all the three executions are the same.

Eventually, it might happen that the adversary has committed $1/3$ fraction errors on one of three executions. In this case, the adversary ignore that execution and focusses only on the other two. Suppose without loss of generality that the executions of A and B are still surviving. In the future rounds, whenever $a_i \neq b_i$, the adversary chooses to corrupt the execution where the number of error so far have been fewer.

Since the adversary makes at most one error in any round of the three executions, it is clear that the adversary never makes more than $1/3$ fraction of errors on any of the executions. Moreover, at the end of the executions, Bob will have identical views of the transcript in the executions corresponding to both A and B. ◀

B Changes in potential function

Proof of Proposition 8. We consider the following four exhaustive cases. Recall that T does not contain consecutive '0's.

Case 1 ($T_{\text{wrong}} = \emptyset$): Suppose Alice sends a bit b. In the case, that it is correctly received by Bob, $\text{len}(T_{\text{right}})$ increases by 1 and $\text{wt}(T_{\text{wrong}})$ remains 0. Thus, Φ increases by 1. Suppose it is incorrectly received by Bob. If $b = 1$ and the last bit of T_{right} is 0, then note that Bob would have received two consecutive 0's and hence will backtrack two symbols and $\text{len}(T_{\text{right}})$ decreases by 2, while T_{wrong} remains $\emptyset$. In all other cases, T_{right} remains unchanged after the transmission, while T_{wrong} is either '1' or '0', which means $\text{len}(T_{\text{right}})$ remains unchanged and $\text{wt}(T_{\text{wrong}})$ becomes 2. In either case, Φ decreases by 2.

Case 2 (T_{wrong} ends in a '1'): In this case, the scheme ensures that Alice sends a '0'. If Bob correctly receives the bit, then the unit '1' is now converted to '10'. Thus, $\text{wt}(T_{\text{wrong}})$ decreases by 1, causing Φ to increase by 1. On the other hand, if Bob does not receive the correct bit, another '1' is added to T_{wrong} which means $\text{wt}(T_{\text{wrong}})$ increases by 2, causing Φ to decrease by 2.

Case 3 (T_{wrong} is '0'): In this case, the scheme ensures that Alice sends a '0'. If Bob correctly receives the '0', then $\text{len}(T_{\text{right}})$ goes down by at most 1, but $\text{wt}(|T_{\text{wrong}}|)$ goes down by 2, implying that Φ increases by at least 1. On the other hand, if Bob incorrectly receives a '1', then $\text{wt}(T_{\text{wrong}})$ increases by 2 and thus Φ goes down by 2.

Case 4 (T_{wrong} ends in '10'): In this case, the scheme ensures that Alice sends a '0'. If Bob correctly receives the '0', then he will backtrack the '10', and thus $\text{wt}(T_{\text{wrong}})$ decreases by 1, implying that Φ increases by 1. On the other hand, if Bob receives a '1', then $\text{wt}(T_{\text{wrong}})$ increases by 2, and thus Φ decreases by 2. ◀

Spectral Norm of Random Kernel Matrices with Applications to Privacy

Shiva Prasad Kasiviswanathan[1] and Mark Rudelson[2]

1 **Samsung Research America**
Mountain View, CA, USA
`kasivisw@gmail.com`

2 **Department of Mathematics, University of Michigan**
Ann Arbor, MI, USA
`rudelson@umich.edu`

Abstract

Kernel methods are an extremely popular set of techniques used for many important machine learning and data analysis applications. In addition to having good practical performance, these methods are supported by a well-developed theory. Kernel methods use an implicit mapping of the input data into a high dimensional feature space defined by a kernel function, i.e., a function returning the inner product between the images of two data points in the feature space. Central to any kernel method is the kernel matrix, which is built by evaluating the kernel function on a given sample dataset.

In this paper, we initiate the study of non-asymptotic spectral properties of random kernel matrices. These are $n \times n$ random matrices whose (i, j)th entry is obtained by evaluating the kernel function on $\mathbf{x}_i$ and $\mathbf{x}_j$, where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are a set of n independent random high-dimensional vectors. Our main contribution is to obtain tight upper bounds on the spectral norm (largest eigenvalue) of random kernel matrices constructed by using common kernel functions such as polynomials and Gaussian radial basis.

As an application of these results, we provide lower bounds on the distortion needed for releasing the coefficients of kernel ridge regression under attribute privacy, a general privacy notion which captures a large class of privacy definitions. Kernel ridge regression is standard method for performing non-parametric regression that regularly outperforms traditional regression approaches in various domains. Our privacy distortion lower bounds are the first for any kernel technique, and our analysis assumes realistic scenarios for the input, unlike all previous lower bounds for other release problems which only hold under very restrictive input settings.

1998 ACM Subject Classification F. Theory of Computation

Keywords and phrases Random Kernel Matrices, Spectral Norm, Subguassian Distribution, Data Privacy, Reconstruction Attacks

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.898

1 Introduction

In recent years there has been significant progress in the development and application of kernel methods for many practical machine learning and data analysis problems. Kernel methods are regularly used for a range of problems such as classification (binary/multiclass), regression, ranking, and unsupervised learning, where they are known to almost always outperform "traditional" statistical techniques [23, 24]. At the heart of kernel methods is the notion of *kernel function*, which is a real-valued function of two variables. The power of kernel methods stems from the fact for every (positive definite) kernel function it is possible to define an inner-product and a lifting (which could be nonlinear) such that inner-product between any two lifted datapoints can be quickly computed using the kernel function evaluated at those two datapoints. This allows for introduction of nonlinearity into the traditional optimization problems (such as Ridge Regression, Support Vector Machines, Principal Component Analysis) without unduly complicating them.

The main ingredient of any kernel method is the *kernel matrix*, which is built using the kernel function, evaluated at given sample points. Formally, given a kernel function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and a sample set $\mathbf{x}_1, \dots, \mathbf{x}_n$, the kernel matrix K is an $n \times n$ matrix with its (i,j)th entry $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. Common choices of kernel functions include the polynomial kernel ($\kappa(\mathbf{x}_i, \mathbf{x}_j) = (a\langle \mathbf{x}_i, \mathbf{x}_j \rangle + b)^p$, for $p \in \mathbb{N}$) and the Gaussian kernel ($\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-a\|\mathbf{x}_i - \mathbf{x}_j\|^2)$, for $a > 0$) [23, 24].

In this paper, we initiate the study of non-asymptotic spectral properties of *random kernel matrices*. A random kernel matrix, for a kernel function κ, is the kernel matrix K formed by n independent random vectors $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$. The prior work on random kernel matrices [13, 2, 6] have established various interesting properties of the spectral distributions of these matrices in the asymptotic sense (as $n, d \to \infty$). However, analyzing algorithms based on kernel methods typically requires understanding of the spectral properties of these random kernel matrices for *large, but fixed* n, d. A similar parallel also holds in the study of the spectral properties of "traditional" random matrices, where recent developments in the non-asymptotic theory of random matrices have complemented the classical random matrix theory that was mostly focused on asymptotic spectral properties [27, 20].

We investigate upper bounds on the largest eigenvalue (spectral norm) of random kernel matrices for polynomial and Gaussian kernels. We show that for inputs $\mathbf{x}_1, \dots, \mathbf{x}_n$ drawn independently from a wide class of probability distributions over $\mathbb{R}^d$ (satisfying the subgaussian property), the spectral norm of a random kernel matrix constructed using a polynomial kernel of degree p, with high probability, is roughly bounded by $O(d^p n)$. In a similar setting, we show that the spectral norm of a random kernel matrix constructed using a Gaussian kernel is bounded by $O(n)$, and with high probability, this bound reduces to $O(1)$ under some stronger assumptions on the subgaussian distributions. These bounds are almost tight. Since the entries of a random kernel matrix are highly correlated, the existing techniques prevalent in random matrix theory cannot be directly applied. We overcome this problem by careful splitting and conditioning arguments on the random kernel matrix. Combining these with subgaussian norm concentrations form the basis of our proofs.

1.1 Applications

Largest eigenvalue of kernel matrices plays an important role in the analysis of many machine learning algorithms. Some examples include, bounding the Rademacher complexity for multiple kernel learning [16], analyzing the convergence rate of conjugate gradient technique for matrix-valued kernel learning [26], and establishing the concentration bounds for eigenvalues of kernel matrices [12, 25].

In this paper, we focus on an application of these eigenvalue bounds to an important problem arising while analyzing sensitive data. Consider a curator who manages a database of sensitive information but wants to release statistics about how a *sensitive* attribute (say, disease) in the database relates with some *nonsensitive* attributes (e.g., postal code, age, gender, etc). This setting is widely considered in the applied data privacy literature, partly since it arises with medical and retail data. Ridge regression is a well-known approach for solving these problems due to its good generalization performance. Kernel ridge regression is a powerful technique for building nonlinear regression models that operate by combining ridge regression with kernel methods [21].[1] We present a *linear reconstruction attack* that reconstructs, with high probability, almost all the sensitive attribute entries given sufficiently accurate approximation of the kernel ridge regression coefficients. In a linear reconstruction attack, given the released information ρ, the attacker constructs a system of approximate linear equalities of the form $A\mathbf{z} \approx \rho$ for a matrix A and attempts to solve for $\mathbf{z}$.

We consider reconstruction attacks against *attribute privacy*, a loose notion of privacy, where the goal is to just avoid any gross violation of privacy. Concretely, the input is assumed to be a database whose ith row (record for individual i) is $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i \in \mathbb{R}^d$ is assumed to be known to the attacker (public information) and $y_i \in \{0,1\}$ is the sensitive attribute, and a privacy mechanism is *attribute non-private* if the attacker can consistently reconstruct a large fraction of the sensitive attribute $(y_1, \dots, y_n)$. We show that any privacy mechanism that always adds $\approx o(1/(d^p n))$ noise[2] to each coefficient of a polynomial kernel ridge regression model is attribute non-private. Similarly any privacy mechanism that always adds $\approx o(1)$ noise[2] to each coefficient of a Gaussian kernel ridge regression model is attribute non-private. As we later discuss, there exists natural settings of inputs under which these kernel ridge regression coefficients, even without the privacy constraint, have the same magnitude as these noise bounds, implying that privacy comes at a steep price. While the linear reconstruction attacks employed in this paper themselves are well-known [9, 15, 14], these are the first attribute privacy lower bounds that: (i) are applicable to any kernel method and (ii) work for any d-dimensional data, analyses of all previous attacks (for other release problems) require d to be comparable to n. Additionally, unlike previous reconstruction attack analyses, our bounds hold for a wide class of realistic distributional assumptions on the data.

1.2 Comparison with Related Work

In this paper, we study the largest eigenvalue of an $n \times n$ random kernel matrix in the non-asymptotic sense. The general goal with studying non-asymptotic theory of random matrices is to understand the spectral properties of random matrices, which are valid with high probability for matrices of a large fixed size. This is contrast with the existing theory on random kernel matrices which have focused on the asymptotics of various spectral characteristics of these random matrices, when the dimensions of the matrices tend to infinity. Let $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ be n i.i.d. random vectors. For any $F : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$, symmetric in the first two variables, consider the random kernel matrix K with (i,j)th entry $K_{ij} = F(\mathbf{x}_i, \mathbf{x}_j, d)$. El Karoui [13] considered the case where K is generated by either the *inner-product kernels* (i.e., $F(\mathbf{x}_i, \mathbf{x}_j, d) = f(\langle \mathbf{x}_i, \mathbf{x}_j \rangle, d)$) or the *distance kernels* (i.e., $F(\mathbf{x}_i, \mathbf{x}_j, d) = f(\|\mathbf{x}_i - \mathbf{x}_j\|^2, d)$). It was shown there that under some assumptions on f and on the distributions of $\mathbf{x}_i$'s, and in the "large d, large n" limit (i.e., and $d, n \to \infty$ and $d/n \to (0, \infty)$): a) the non-linear kernel matrix converges asymptotically in spectral norm to a linear kernel matrix, and b) there is a weak convergence of the limiting spectral density. These results were recently strengthened in different directions by Cheng *et al.* [2] and Do *et al.* [6]. To the best of our knowledge, ours is the first paper investigating the non-asymptotic spectral properties of a random kernel matrix.

Like the development of non-asymptotic theory of traditional random matrices has found multitude of applications in areas including statistics, geometric functional analysis, and compressed sensing [27], we believe that the growth of a non-asymptotic theory of random

[1] We provide a brief coverage of the basics of kernel ridge regression in Section 4.
[2] Ignoring the dependence on other parameters, including the regularization parameter of ridge regression.

kernel matrices will help in better understanding of many machine learning applications that utilize kernel techniques.

The goal of *private data analysis* is to release global, statistical properties of a database while protecting the privacy of the individuals whose information the database contains. Differential privacy [7] is a formal notion of privacy tailored to private data analysis. Differential privacy requires, roughly, that any single individual's data have little effect on the outcome of the analysis. A lot of recent research has gone in developing differentially private algorithms for various applications, including kernel methods [11]. A typical objective here is to release as accurate an approximation as possible to some function f evaluated on a database D.

In this paper, we follow a complementary line of work that seeks to understand how much distortion (noise) is necessary to privately release some particular function f evaluated on a database containing sensitive information [5, 8, 9, 15, 4, 18, 3, 19, 14]. The general idea here, is to provide *reconstruction attacks*, which are attacks that can reconstruct (almost all of) the sensitive part of database D given sufficiently accurate approximations to $f(D)$. Reconstruction attacks violate any *reasonable* notion of privacy (including, differential privacy), and the existence of these attacks directly translate into lower bounds on distortion needed for privacy.

Linear reconstruction attacks were first considered in the context of data privacy by Dinur and Nissim [5], who showed that any mechanism which answers $\approx n \log n$ random inner product queries on a database in $\{0,1\}^n$ with $o(\sqrt{n})$ noise per query is not private. Their attack was subsequently extended in various directions by [8, 9, 18, 3].

The results that are closest to our work are the attribute privacy lower bounds analyzed for releasing k-way marginals [15, 4], linear/logistic regression parameters [14], and a subclass of statistical M-estimators [14]. Kasiviswanathan *et al.* [15] showed that, if $d = \tilde{\Omega}(n^{1/(k-1)})$, then any mechanism which releases all k-way marginal tables with $o(\sqrt{n})$ noise per entry is attribute non-private.[3] These noise bounds were improved by De [4], who presented an attack that can tolerate a constant fraction of entries with arbitrarily high noise, as long as the remaining entries have $o(\sqrt{n})$ noise. Kasiviswanathan *et al.* [14] recently showed that, if $d = \Omega(n)$, then any mechanism which releases d different linear or logistic regression estimators each with $o(1/\sqrt{n})$ noise is attribute non-private. They also showed that this lower bound extends to a subclass of statistical M-estimator release problems. A point to observe is that in all the above referenced results, d has to be comparable to n, and this dependency looks unavoidable in those results due to their use of least singular value bounds. However, in this paper, our privacy lower bounds hold for all values of d, n (d could be $\ll n$). Additionally, all the previous reconstruction attack analyses critically require the $\mathbf{x}_i$'s to be drawn from product of univariate subgaussian distributions, whereas our analysis here holds for any d-dimensional subgaussian distributions (not necessarily product distributions), thereby is more widely applicable. The subgaussian assumption on the input data is quite common in the analysis of machine learning algorithms [1].

2 Preliminaries

2.1 Notation

We use $[n]$ to denote the set $\{1, \dots, n\}$. $d_H(\cdot, \cdot)$ measures the Hamming distance. Vectors used in the paper are by default column vectors and are denoted by boldface letters. For

[3] The $\tilde{\Omega}$ notation hides polylogarithmic factors.

a vector $\mathbf{v}$, $\mathbf{v}^\top$ denotes its transpose and $\|\mathbf{v}\|$ denotes its Euclidean norm. For two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle$ denotes the inner product of $\mathbf{v}_1$ and $\mathbf{v}_2$. For a matrix M, $\|M\|$ denotes its spectral norm, $\|M\|_F$ denotes its Frobenius norm, and M_{ij} denotes its (i, j)th entry. $\mathbb{I}_n$ represents the identity matrix in dimension n. The unit sphere in d dimensions centered at origin is denoted by $S^{d-1} = \{\mathbf{z} : \|\mathbf{z}\| = 1, \mathbf{z} \in \mathbb{R}^d\}$. Throughout this paper C, c, C', also with subscripts, denote absolute constants (i.e., independent of d and n), whose value may change from line to line.

2.2 Background on Kernel Methods

We provide a very brief introduction to the theory of kernel methods; see the many books on the topic [23, 24] for further details.

▶ **Definition 1** (Kernel Function). Let $\mathcal{X}$ be a non-empty set. Then a function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel function on $\mathcal{X}$ if there exists a Hilbert space $\mathcal{H}$ over $\mathbb{R}$ and a map $\phi : \mathcal{X} \to \mathcal{H}$ such that for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, we have

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_{\mathcal{H}}.$$

For any *symmetric and positive semidefinite*[4] kernel κ, by Mercer's theorem [17] there exists: (i) a unique functional Hilbert space $\mathcal{H}$ (referred to as the reproducing kernel Hilbert space, Definition 2) on $\mathcal{X}$ such that $\kappa(\cdot, \cdot)$ is the inner product in the space and (ii) a map ϕ defined as $\phi(\mathbf{x}) := \kappa(\cdot, \mathbf{x})$[5] that satisfies Definition 1. The function ϕ is called the *feature map* and the space $\mathcal{H}$ is called the *feature space*.

▶ **Definition 2** (Reproducing Kernel Hilbert Space). A kernel $\kappa(\cdot, \cdot)$ is a reproducing kernel of a Hilbert space $\mathcal{H}$ if $\forall f \in \mathcal{H}$, $f(\mathbf{x}) = \langle \kappa(\cdot, \mathbf{x}), f(\cdot) \rangle_{\mathcal{H}}$. For a (compact) $\mathcal{X} \subseteq \mathbb{R}^d$, and a Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$, we say $\mathcal{H}$ is a Reproducing Kernel Hilbert Space if there $\exists \kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, s.t.: a) κ has the reproducing property, and b) κ spans $\mathcal{H} = \overline{\text{span}\{\kappa(\cdot, \mathbf{x}) : \mathbf{x} \in \mathcal{X}\}}$.

A standard idea used in the machine-learning community (commonly referred to as the "kernel trick") is that kernels allow for the computation of inner-products in high-dimensional feature spaces ($\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_{\mathcal{H}}$) using simple functions defined on pairs of input patterns ($\kappa(\mathbf{x}, \mathbf{y})$), without knowing the ϕ mapping explicitly. This trick allows one to efficiently solve a variety of non-linear optimization problems. Note that there is no restriction on the dimension of the feature maps ($\phi(\mathbf{x})$), i.e., it could be of infinite dimension.

Polynomial and Gaussian are two popular kernel functions that are used in many machine learning and data mining tasks such as classification, regression, ranking, and structured prediction. Let the input space $\mathcal{X} = \mathbb{R}^d$. For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, these kernels are defined as:

1. **Polynomial Kernel:** $\kappa(\mathbf{x}, \mathbf{y}) = (a\langle \mathbf{x}, \mathbf{y} \rangle + b)^p$, with parameters $a, b \in \mathbb{R}$ and $p \in \mathbb{N}$. Here a is referred to as the slope parameter, $b \geq 0$ trades off the influence of higher-order versus lower-order terms in the polynomial, and p is the polynomial degree. For an input $\mathbf{x} \in \mathbb{R}^d$, the feature map $\phi(\mathbf{x})$ of the polynomial kernel is a vector with a polynomial in d number of dimensions [23].

[4] A positive definite kernel is a function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that for any $n \geq 1$, for any finite set of points $\{\mathbf{x}_i\}_{i=1}^n$ in $\mathcal{X}$ and real numbers $\{a_i\}_{i=1}^n$, we have $\sum_{i,j=1}^n a_i a_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 0$.

[5] $\kappa(\cdot, \mathbf{x})$ is a vector with entries $\kappa(\mathbf{x}', \mathbf{x})$ for all $\mathbf{x}' \in \mathcal{X}$.

2. **Gaussian Kernel:** (frequently referred to as the *radial basis kernel*): $\kappa(\mathbf{x},\mathbf{y}) = \exp\left(-a\|\mathbf{x}-\mathbf{y}\|^2\right)$ with real parameter $a > 0$. The value of a controls the locality of the kernel with low values indicating that the influence of a single point is "far" and vice-versa [23]. An equivalent popular formulation, is to set $a = 1/2\sigma^2$, and hence, $\kappa(\mathbf{x},\mathbf{y}) = \exp\left(-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2\right)$. For an input $\mathbf{x} \in \mathbb{R}^d$, the feature map $\phi(\mathbf{x})$ of the Gaussian kernel is a vector of infinite dimensions [23]. Note that while we focus on the Gaussian kernel in this paper, the extension of our results to other exponential kernels such as the *Laplacian kernel* (where $\kappa(\mathbf{x},\mathbf{y}) = \exp\left(-a\|\mathbf{x}-\mathbf{y}\|_1\right)$), is quite straightforward.

2.3 Background on Subgaussian Random Variables

Let us start by formally defining subgaussian random variables and vectors.

▶ **Definition 3** (Subgaussian Random Variable and Vector)**.** We call a random variable $x \in \mathbb{R}$ subgaussian if there exists a constant $C > 0$ if $\Pr[|x| > t] \leq 2\exp(-t^2/C^2)$ for all $t > 0$. We say that a random vector $\mathbf{x} \in \mathbb{R}^d$ is subgaussian if the one-dimensional marginals $\langle \mathbf{x}, \mathbf{y}\rangle$ are subgaussian random variables for all $\mathbf{y} \in \mathbb{R}^d$.

The class of subgaussian random variables includes many random variables that arise naturally in data analysis, such as standard normal, Bernoulli, spherical, bounded (where the random variable x satisfies $|x| \leq M$ *almost surely* for some fixed M). The natural generalizations of these random variables to higher dimension are all subgaussian random vectors. For many *isotropic convex sets*[6] $\mathcal{K}$ (such as the hypercube), a random vector $\mathbf{x}$ uniformly distributed in $\mathcal{K}$ is subgaussian.

▶ **Definition 4** (Norm of Subgaussian Random Variable and Vector)**.** The ψ_2-norm of a subgaussian random variable $x \in \mathbb{R}$, denoted by $\|x\|_{\psi_2}$ is:

$$\|x\|_{\psi_2} = \inf\left\{t > 0 \,:\, \mathbb{E}[\exp(|x|^2/t^2)] \leq 2\right\}.$$

The ψ_2-norm of a subgaussian random vector $\mathbf{x} \in \mathbb{R}^d$ is:

$$\|\mathbf{x}\|_{\psi_2} = \sup_{\mathbf{y}\in S^{d-1}} \|\langle \mathbf{x},\mathbf{y}\rangle\|_{\psi_2}.$$

▶ **Claim 5** (Vershynin [27])**.** *Let $x \in \mathbb{R}$ be a subgaussian random variable. Then there exists a constant $C > 0$, such that $\Pr[|x| > t] \leq 2\exp(-Ct^2/\|x\|_{\psi_2}^2)$.*

Consider a subset T of $\mathbb{R}^d$, and let $\epsilon > 0$. An ϵ-net of T is a subset $\mathcal{N} \subseteq T$ such that for every $\mathbf{x} \in T$, there exists a $\mathbf{z} \in \mathcal{N}$ such that $\|\mathbf{x}-\mathbf{z}\| \leq \epsilon$. We would use the following well-known result about the size of ϵ-nets.

▶ **Proposition 6** (Bounding the size of an ϵ-Net [27])**.** *Let T be a subset of S^{d-1} and let $\epsilon > 0$. Then there exists an ϵ-net of T of cardinality at most $(1+2/\epsilon)^d$.*

The proof of the following claim follows by standard techniques.

▶ **Claim 7** (Vershynin [27])**.** *Let $\mathcal{N}$ be a 1/2-net of S^{d-1}. Then for any $\mathbf{x} \in \mathbb{R}^d$, $\|\mathbf{x}\| \leq 2\max_{\mathbf{y}\in\mathcal{N}} \langle \mathbf{x},\mathbf{y}\rangle$.*

[6] A convex set $\mathcal{K}$ in $\mathbb{R}^d$ is called isotropic if a random vector chosen uniformly from $\mathcal{K}$ according to the volume is isotropic. A random vector $\mathbf{x} \in \mathbb{R}^d$ is isotropic if for all $\mathbf{y} \in \mathbb{R}^d$, $\mathbb{E}[\langle \mathbf{x},\mathbf{y}\rangle^2] = \|\mathbf{y}\|^2$.

3 Largest Eigenvalue of Random Kernel Matrices

In this section, we provide the upper bound on the largest eigenvalue of a random kernel matrix, constructed using polynomial or Gaussian kernels. Notice that the entries of a random kernel matrix are dependent. For example any triplet of entries (i, j), (j, k) and (k, i) are mutually dependent. Additionally, we deal with vectors drawn from general subgaussian distributions, and therefore, the coordinates within a random vector need not be independent.

We start off with a simple lemma, to bound the Euclidean norm of a subgaussian random vector. A random vector $\mathbf{x}$ is *centered* if $\mathbb{E}[\mathbf{x}] = 0$.

▶ **Lemma 8.** *Let $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ be independent centered subgaussian vectors. Then for all $i \in [n]$, $\Pr[\|\mathbf{x}_i\| \geq C\sqrt{d}] \leq \exp(-C'd)$ for constants C, C'.*

Proof. To this end, note that since $\mathbf{x}_i$ is a subgaussian vector (from Definition 3)

$$\Pr\left[|\langle \mathbf{x}_i, \mathbf{y}\rangle| \geq C\sqrt{d}/2\right] \leq 2\exp(-C_2 d),$$

for constants C and C_2, any unit vector $\mathbf{y} \in S^{d-1}$. Taking the union bound over a (1/2)-net ($\mathcal{N}$) in S^{d-1}, and using Proposition 6 for the size of the nets (which is at most 5^d as $\epsilon = 1/2$), we get that

$$\Pr\left[\max_{\mathbf{y}\in\mathcal{N}} |\langle \mathbf{x}_i, \mathbf{y}\rangle| \geq C\sqrt{d}/2\right] \leq \exp(-C_3 d),$$

From Claim 7, we know that $\|\mathbf{x}_i\| \leq 2\max_{\mathbf{y}\in\mathcal{N}} \langle \mathbf{x}_i, \mathbf{y}\rangle$. Hence, $\Pr\left[\|\mathbf{x}_i\| \geq C\sqrt{d}\right] \leq \exp(-C'd)$. ◀

3.1 Polynomial Kernel

We now establish the bound on the spectral norm of a polynomial kernel random matrix. We assume $\mathbf{x}_1, \dots, \mathbf{x}_n$ are independent vectors drawn according to a centered subgaussian distribution over $\mathbb{R}^d$. Let K_p denote the kernel matrix obtained using $\mathbf{x}_1, \dots, \mathbf{x}_n$ in a polynomial kernel. Our idea to split the kernel matrix K_p into its diagonal and off-diagonal parts, and then bound the spectral norms of these two matrices separately. The diagonal part contains independent entries of the form $(a\|\mathbf{x}_i\|^2 + b)^p$, and we use Lemma 8 to bound its spectral norm. Dealing with the off-diagonal part of K_p is trickier because of the dependence between the entries, and here we bound the spectral norm by its Frobenius norm. We also verify the upper bounds provided in the following theorem by conducting numerical experiments (see Figure 1a).

▶ **Theorem 9.** *Let $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ be independent centered subgaussian vectors. Let $p \in \mathbb{N}$, and let K_p be the $n \times n$ matrix with (i, j)th entry $K_{p_{ij}} = (a\langle \mathbf{x}_i, \mathbf{x}_j\rangle + b)^p$. Assume that $n \leq \exp(C_1 d)$ for a constant C_1. Then there exists constants C_0, C_0' such that*

$$\Pr\left[\|K_p\| \geq C_0^p |a|^p d^p n + 2^{p+1}|b|^p n\right] \leq \exp(-C_0' d).$$

Proof. To prove the theorem, we split the kernel matrix K_p into the diagonal and off-diagonal parts. Let $K_p = D + W$, where D represents the diagonal part of K_p and W the off-diagonal part of K_p. Note that

$$\|K_p\| \leq \|D\| + \|W\| \leq \|D\| + \|W\|_F.$$

Let us estimate the norm of the diagonal part D first. From Lemma 8, we know that for all $i \in [n]$ with $C_3 = C'$,

$$\Pr\left[\|\mathbf{x}_i\| \geq C\sqrt{d}\right] = \Pr\left[\|\mathbf{x}_i\|^2 \geq (C\sqrt{d})^2\right] \leq \exp(-C_3 d).$$

Instead of $\|\mathbf{x}\|_i^2$, we are interested in bounding $(a\|\mathbf{x}_i\|^2 + b)^p$.

$$\Pr\left[\|\mathbf{x}_i\|^2 \geq (C\sqrt{d})^2\right] = \Pr\left[(a\|\mathbf{x}_i\|^2 + b)^p \geq (a(C\sqrt{d})^2 + b)^p\right]. \tag{1}$$

Consider $(a(C\sqrt{d})^2 + b)^p$. A simple inequality to bound $(a(C\sqrt{d})^2 + b)^p$ is[7]

$$(a(C\sqrt{d})^2 + b)^p \leq 2^p(|a|^p(C\sqrt{d})^{2p} + |b|^p).$$

Therefore,

$$\Pr\left[(a\|\mathbf{x}_i\|^2 + b)^p \geq 2^p(|a|^p(C\sqrt{d})^{2p} + |b|^p)\right] \leq \Pr\left[(a\|\mathbf{x}_i\|^2 + b)^p \geq (a(C\sqrt{d})^2 + b)^p\right].$$

Using (1) and substituting in the above equation, for any $i \in [n]$

$$\Pr\left[(a\|\mathbf{x}_i\|^2 + b)^p \geq 2^p(|a|^p C^{2p} d^p + |b|^p)\right] \leq \Pr\left[\|\mathbf{x}_i\| \geq C\sqrt{d}\right] \leq \exp(-C_3 d).$$

By applying a union bound over all n non-zero entries in D, we get that for all $i \in [n]$

$$\begin{aligned}\Pr\left[(a\|\mathbf{x}_i\|^2 + b)^p \geq 2^p(|a|^p C^{2p} d^p + |b|^p)\right] \leq n \cdot \exp(-C_3 d) &\leq \exp(C_1 d) \cdot \exp(-C_3 d)\\ &\leq \exp(-C_4 d),\end{aligned}$$

as we assumed that $n \leq \exp(C_1 d)$. This implies that

$$\Pr[\|D\| \geq 2^p(|a|^p C^{2p} d^p + |b|^p)] \leq \exp(-C_4 d). \tag{2}$$

We now bound the spectral norm of the off-diagonal part W using Frobenius norm as an upper bound on the spectral norm. Firstly note, by definition, for any $\mathbf{y} \in \mathbb{R}^d$, the random variable $\langle \mathbf{x}_i, \mathbf{y}\rangle$ is subgaussian with its ψ_2-norm at most $C_5\|\mathbf{y}\|$ for some constant C_5. This follows as:

$$\|\langle \mathbf{x}_i, \mathbf{y}\rangle\|_{\psi_2} := \inf\left\{t > 0 \,:\, \mathbb{E}[\exp(\langle \mathbf{x}_i, \mathbf{y}\rangle^2/t^2)] \leq 2\right\} \leq C_5\|\mathbf{y}\|.$$

Therefore, for a fixed $\mathbf{x}_j$, $\|\langle \mathbf{x}_i, \mathbf{x}_j\rangle\|_{\psi_2} \leq C_5\|\mathbf{x}_j\|$. For $i \neq j$, conditioning on $\mathbf{x}_j$,

$$\Pr\left[|\langle \mathbf{x}_i, \mathbf{x}_j\rangle| \geq \tau\right] = \mathbb{E}_{\mathbf{x}_j}\left[\Pr\left[|\langle \mathbf{x}_i, \mathbf{x}_j\rangle| \geq \tau \mid \mathbf{x}_j\right]\right].$$

From Claim 5,

$$\begin{aligned}\mathbb{E}_{\mathbf{x}_j}\left[\Pr\left[|\langle \mathbf{x}_i, \mathbf{x}_j\rangle| \geq \tau \mid \mathbf{x}_j\right]\right] \leq \mathbb{E}_{\mathbf{x}_j}\left[\exp\left(\frac{-C_6\tau^2}{\|\langle \mathbf{x}_i, \mathbf{x}_j\rangle\|_{\psi_2}^2}\right)\right] &\leq \mathbb{E}_{\mathbf{x}_j}\left[\exp\left(\frac{-C_6\tau^2}{(C_5\|\mathbf{x}_j\|)^2}\right)\right]\\ &= \mathbb{E}_{\mathbf{x}_j}\left[\exp\left(\frac{-C_7\tau^2}{\|\mathbf{x}_j\|^2}\right)\right],\end{aligned}$$

[7] For any $a, b, m \in \mathbb{R}$ and $p \in \mathbb{N}$, $(a \cdot m + b)^p \leq 2^p(|a|^p|m|^p + |b|^p)$.

where the last inequality uses the fact that $\|\langle \mathbf{x}_i, \mathbf{x}_j\rangle\|_{\psi_2} \leq C_5\|\mathbf{x}_j\|$. Now let us condition the above expectation on the value of $\|\mathbf{x}_j\|$ based on whether $\|\mathbf{x}_j\| \geq C\sqrt{d}$ or $\|\mathbf{x}_j\| < C\sqrt{d}$. We can rewrite

$$\begin{aligned}\mathbb{E}_{\mathbf{x}_j}\left[\frac{-C_7\tau^2}{\|\mathbf{x}_j\|^2}\right] \leq \mathbb{E}_{\mathbf{x}_j}&\left[\exp\left(\frac{-C_7\tau^2}{C^2 d}\right)\,\middle|\, \|\mathbf{x}_j\| < C\sqrt{d}\right]\Pr[\|\mathbf{x}_j\| < C\sqrt{d}] \\ &+ \mathbb{E}_{\mathbf{x}_j}\left[\exp\left(\frac{-C_7\tau^2}{\|\mathbf{x}_j\|^2}\right)\,\middle|\, \|\mathbf{x}_j\| \geq C\sqrt{d}\right]\Pr[\|\mathbf{x}_j\| \geq C\sqrt{d}].\end{aligned}$$

The above equation can be easily be simplified as:

$$\mathbb{E}_{\mathbf{x}_j}\left[\frac{-C_7\tau^2}{\|\mathbf{x}_j\|^2}\right] \leq \exp\left(\frac{-C_8\tau^2}{d}\right) + \mathbb{E}_{\mathbf{x}_j}\left[\exp\left(\frac{-C_7\tau^2}{\|\mathbf{x}_j\|^2}\right)\,\middle|\, \|\mathbf{x}_j\| \geq C\sqrt{d}\right]\Pr[\|\mathbf{x}_j\| \geq C\sqrt{d}].$$

From Lemma 8, $\Pr[\|\mathbf{x}_j\| \geq C\sqrt{d}] \leq \exp(-C_3 d)$, and

$$\mathbb{E}_{\mathbf{x}_j}\left[\exp\left(\frac{-C_7\tau^2}{\|\mathbf{x}_j\|^2}\right)\,\middle|\, \|\mathbf{x}_j\| \geq C\sqrt{d}\right] \leq 1.$$

This implies that as $\Pr[\|\mathbf{x}_j\| \geq C\sqrt{d}] \leq \exp(-C_3 d))$,

$$\mathbb{E}_{\mathbf{x}_j}\left[\exp\left(\frac{-C_7\tau^2}{\|\mathbf{x}_j\|^2}\right)\,\middle|\, \|\mathbf{x}_j\| \geq C\sqrt{d}\right]\Pr[\|\mathbf{x}_j\| \geq C\sqrt{d}] \leq \exp(-C_3 d).$$

Putting the above arguments together,

$$\Pr\left[|\langle \mathbf{x}_i, \mathbf{x}_j\rangle| \geq \tau\right] = \mathbb{E}_{\mathbf{x}_j}\left[\Pr\left[|\langle \mathbf{x}_i, \mathbf{x}_j\rangle| \geq \tau \mid \mathbf{x}_j\right]\right] \leq \exp\left(\frac{-C_8\tau^2}{d}\right) + \exp(-C_3 d).$$

Taking a union bound over all $(n^2 - n) < n^2$ non-zero entries in W,

$$\Pr\left[\max_{i\neq j}|\langle \mathbf{x}_i, \mathbf{x}_j\rangle| \geq \tau\right] \leq n^2\left(\exp\left(\frac{-C_8\tau^2}{d}\right) + \exp(-C_3 d)\right).$$

Setting $\tau = C\cdot d$ in the above and using the fact that $n \leq \exp(C_1 d)$,

$$\Pr\left[\max_{i\neq j}|\langle \mathbf{x}_i, \mathbf{x}_j\rangle| \geq C\cdot d\right] \leq \exp(-C_9 d). \tag{3}$$

We are now ready to bound the Frobenius norm of W.

$$\begin{aligned}\|W\|_F = \left(\sum_{i\neq j}(a\langle \mathbf{x}_i, \mathbf{x}_j\rangle + b)^{2p}\right)^{1/2} &\leq \left(n^2 2^{2p}\left(|a|^{2p}\langle \mathbf{x}_i, \mathbf{x}_j\rangle^{2p} + |b|^{2p}\right)\right)^{1/2} \\ &\leq n2^p\left(|a|^p|\langle \mathbf{x}_i, \mathbf{x}_j\rangle|^p + |b|^p\right).\end{aligned}$$

Plugging in the probabilistic bound on $|\langle \mathbf{x}_i, \mathbf{x}_j\rangle|$ from (3) gives,

$$\begin{aligned}\Pr\left[\|W\|_F \geq n2^p\left(|a|^p|C^p d^p + |b|^p\right)\right] &\leq \Pr\left[n2^p\left(|a|^p|\langle \mathbf{x}_i, \mathbf{x}_j\rangle|^p + |b|^p\right) \geq n2^p\left(|a|^p|C^p d^p + |b|^p\right)\right] \\ \Pr\left[\|W\|_F \geq n2^p\left(|a|^p|C^p d^p + |b|^p\right)\right] &\leq \Pr\left[n2^p\left(|a|^p|\langle \mathbf{x}_i, \mathbf{x}_j\rangle|^p + |b|^p\right) \geq n2^p\left(|a|^p|C^p d^p + |b|^p\right)\right] \\ &\leq \exp(-C_9 d). \end{aligned} \tag{4}$$

Plugging bounds on $\|D\|$ (from (2)) and $\|W\|_F$ (from (4)) to upper bound $\|K_p\| \leq \|D\| + \|W\|_F$ yields that there exists constants C_0 and C_0' such that,

$$\begin{aligned}\Pr\left[\|K_p\| \geq C_0^p|a|^p d^p n + 2^{p+1}|b|^p n\right] &\leq \Pr\left[\|D\| + \|W\|_F \geq C_0^p|a|^p d^p n + 2^{p+1}|b|^p n\right] \\ &\leq \exp(-C_0' d).\end{aligned}$$

This completes the proof of the theorem. The chain of constants can easily be estimated starting with the constant in the definition of the subgaussian random variable. ◀

▶ Remark. Note that for our proofs it is only necessary that $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are independent random vectors, but they need not be identically distributed.

The above spectral norm upper bound on K_p (again with exponentially high probability) could be improved to

$$O\left(C_0^p|a|^p(d^p + d^{p/2}n) + 2^{p+1}n|b|^p\right),$$

with a slightly more involved analysis (omitted here). For an even p, the expectation of every individual entry of the matrix K_p is positive, which provides tight examples for this bound.

3.2 Gaussian Kernel

We now establish the bound on the spectral norm of a Gaussian kernel random matrix. Again assume $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are independent vectors drawn according to a centered subgaussian distribution over $\mathbb{R}^d$. Let K_g denote the kernel matrix obtained using $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in a Gaussian kernel. Here an upper bound of n on the spectral norm on the kernel matrix follows trivially as all entries of K_g are less than equal to 1. We show that this bound is tight, in that for small values of a, with high probability the spectral norm is at least $\Omega(n)$ (Theorem 10 (Part 2)).

In fact, even for large a's, it is impossible to obtain better than $O(n)$ upper bound on the spectral norm of K_g without additional assumptions on the subgaussian distribution, as illustrated by this example: Consider a distribution over $\mathbb{R}^d$, such that a random vector drawn from this distribution is a zero vector $(0)^d$ with probability 1/2 and uniformly distributed over the sphere in $\mathbb{R}^d$ of radius $2\sqrt{d}$ with probability 1/2. A random vector $\mathbf{x}$ drawn from this distribution is isotropic and subgaussian, but $\Pr[\mathbf{x} = (0)^d] = 1/2$. Therefore, in $\mathbf{x}_1, \ldots, \mathbf{x}_n$ drawn from this distribution, with high probability more than a constant fraction of the vectors will be $(0)^d$. This means that a proportional number of entries of the matrix K_g will be 1, and the norm will be $O(n)$ regardless of a.

This situation changes, however, when we add the additional assumption that $\mathbf{x}_1, \ldots, \mathbf{x}_n$ have independent centered subgaussian coordinates[8] (i.e., each $\mathbf{x}_i$ is drawn from a product distribution formed from some d centered univariate subgaussian distributions). In that case, the kernel matrix K_g is a small perturbation of the identity matrix, and we show that the spectral norm of K_g is with high probability bounded by an absolute constant (for $a = \Omega(\log n/d)$). For this proof, similar to Theorem 9, we split the kernel matrix into its diagonal and off-diagonal parts. The spectral norm of the off-diagonal part is again bounded by its Frobenius norm. We also verify the upper bounds presented in the following theorem by conducting numerical experiments (see Figure 1b).

[8] Some of the commonly used subgaussian random vectors such as the standard normal, Bernoulli satisfy this additional assumption.

▶ **Theorem 10.** *Let $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ be independent centered subgaussian vectors. Let $a > 0$, and let K_g be the $n \times n$ matrix with (i,j)th entry $K_{g_{ij}} = \exp(-a\|\mathbf{x}_i - \mathbf{x}_j\|^2)$. Then there exists constants c, c_0, c_0', c_1 such that*

1. *$\|K_g\| \leq n$.*
2. *If $a < c_1/d$, $\Pr[\|K_g\| \geq c_0 n] \geq 1 - \exp(-c_0' n)$.*
3. *If all the vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$ satisfy the additional assumption of having independent centered subgaussian coordinates, and assume $n \leq \exp(C_1 d)$ for a constant C_1. Then for any $\delta > 0$ and $a \geq (2+\delta)\frac{\log n}{d}$, $\Pr[\|K_g\| \geq 2] \leq \exp(-c\zeta^2 d)$ with $\zeta > 0$ depending only on δ.*

Proof. Proof of Part 1 is straightforward as all entries of K_g do not exceed 1.

Let us prove the lower estimate for the norm in Part 2. For $i = 1, \dots, n$ define

$$Z_i = \sum_{j=\frac{n}{2}+1}^{n} K_{g_{ij}}.$$

From Lemma 8 for all $i \in [n]$, $\Pr\left[\|\mathbf{x}_i\| \geq C\sqrt{d}\right] \leq \exp(-C'd)$. In other words, $\|\mathbf{x}_i\|$ is less than $C\sqrt{d}$ for all $i \in [d]$ with probability at least $1 - \exp(-C'd)$. Let us call this event $\mathcal{E}_1$. Under $\mathcal{E}_1$ and assumption $a < c_1/d$, $\mathbb{E}[Z_i] \geq c_2 n$ and $\mathbb{E}[Z_i^2] \leq c_3 n^2$. Therefore, by Paley-Zygmund inequality (under event $\mathcal{E}_1$),

$$\Pr[Z_i \geq c_4 n] \geq c_5. \tag{5}$$

Now $Z_1, \dots, Z_n$ are not independent random variables. But if we condition on $\mathbf{x}_{n/2+1}, \dots, \mathbf{x}_n$, then $Z_1, \dots, Z_{n/2}$ become independent (for simplicity, assume that n is divisible by 2). Thereafter, an application of Chernoff bound on $Z_1, \dots, Z_{n/2}$ using the probability bound from (5) (under conditioning on $\mathbf{x}_{n/2+1}, \dots, \mathbf{x}_n$ and event $\mathcal{E}_1$) gives:

$$\Pr\left[Z_i \geq c_4 n \text{ for at least } c_5 n \text{ entries } Z_i \in \{Z_1, \dots, Z_{n/2}\}\right] \geq 1 - \exp(-c_6 n).$$

The first conditioning can be removed by taking the expectation with respect to $\mathbf{x}_{n/2+1}, \dots, \mathbf{x}_n$ without disturbing the exponential probability bound. Similarly, conditioning on event $\mathcal{E}_1$ can also be easily removed.

Let K_g' be the submatrix of K_g consisting of rows $1 \leq i \leq n/2$ and columns $n/2+1 \leq j \leq n$. Note that $\|K_g'\| \geq \mathbf{u}^\top K_g' \mathbf{u}$, where $\mathbf{u} = \left(\sqrt{\frac{2}{n}}, \dots, \sqrt{\frac{2}{n}}\right)$ (of dimension $n/2$). Then

$$\Pr[\|K_g\| \leq c_0 n] \leq \Pr[\|K_g'\| \leq c_7 n] \leq \Pr[\mathbf{u}^\top K_g' \mathbf{u} \leq c_7 n]$$
$$\Pr\left[\frac{2}{n}\sum_{i=1}^{n/2} Z_i \leq c_7 n\right] \leq \exp(-c_0' n).$$

The last line follows as from above arguments with exponentially high probability above more than $\Omega(n)$ entries in $Z_1, \dots, Z_{n/2}$ are greater than $\Omega(n)$, and by readjusting the constants.

Proof of Part 3: As in Theorem 9, we split the matrix K_g into the diagonal (D) and the off-diagonal part (W) (i.e., $K_g = D + W$). It is simple to observe that $D = \mathbb{I}_n$, therefore we just concentrate on W. The (i,j)th entry in W is $\exp(-a\|\mathbf{x}_i - \mathbf{x}_j\|^2)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are independent vectors with independent centered subgaussian coordinates. Therefore, we can use Hoeffding's inequality, for fixed i, j,

$$\Pr\left[\exp(-a\|\mathbf{x}_i - \mathbf{x}_j\|^2) \geq \exp(-a(1-\zeta)d)\right] = \Pr\left[\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{d} \leq (1-\zeta)\right] \leq \exp(-c_8\zeta^2 d), \tag{6}$$

where we used the fact that if a random variable is subgaussian then its square is a subexponential random variable [27].[9] To estimate the norm of W, we bound it by its Frobenius norm. If $a \geq (2+\delta)\frac{\log n}{d}$, then we can choose $\zeta > 0$ depending on δ such that $n^2 \exp(-a(1-\zeta)d) \leq 1$. Hence,

$$\begin{aligned}
\Pr[\|K_g\| \geq 2] &\leq \Pr[\|D\| + \|W\|_F \geq 2] = \Pr[\|W\|_F \geq 1] \\
&= \Pr\left[\sum_{1\leq i,j\leq n, i\neq j} \exp(-a\|\mathbf{x}_i - \mathbf{x}_j\|^2) \geq 1\right] \\
&\leq \Pr\left[\sum_{1\leq i,j\leq n, i\neq j} \exp(-a\|\mathbf{x}_i - \mathbf{x}_j\|^2) \geq n^2\exp(-a(1-\zeta)d)\right] \\
&\leq \Pr\left[\sum_{1\leq i,j\leq n} \exp(-a\|\mathbf{x}_i - \mathbf{x}_j\|^2) \geq n^2\exp(-a(1-\zeta)d)\right] \\
&\leq n^2 \Pr\left[\max_{1\leq i,j\leq n} \exp(-a\|\mathbf{x}_i - \mathbf{x}_j\|^2) \geq \exp(-a(1-\zeta)d)\right] \\
&\leq n^2 \exp(-c_8\zeta^2 d) \\
&\leq \exp(-c\zeta^2 d) \text{ for some constant } c.
\end{aligned}$$

The first equality follows as $\|D\| = 1$, and the second-last inequality follows from (6). This completes the proof of the theorem. Again the long chain of constants can easily be estimated starting with the constant in the definition of the subgaussian random variable. ◀

▶ Remark. Note that again the $\mathbf{x}_i$'s need not be identically distributed.

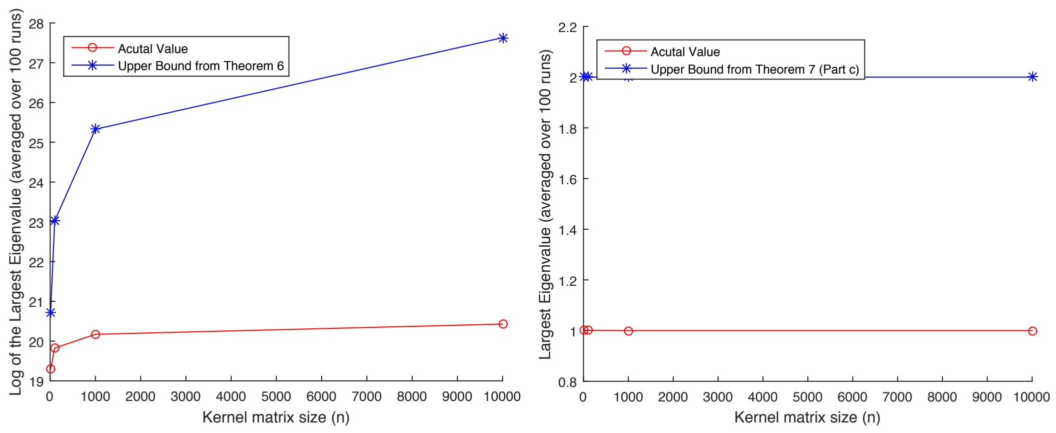
The analysis in Theorem 10 could easily be reworked to handle other exponential kernels such as the Laplacian kernel.

4 Privately Releasing Kernel Ridge Regression Coefficients

We consider an application of Theorems 9 and 10 to obtain noise lower bounds for privately releasing coefficients of kernel ridge regression. For privacy violation, we consider a generalization of *blatant non-privacy* [5] referred to as attribute non-privacy (formalized in [15]). Consider a database $D \in \mathbb{R}^{n\times d+1}$ that contains, for each individual i, a sensitive attribute $y_i \in \{0,1\}$ as well as some other information $\mathbf{x}_i \in \mathbb{R}^d$ which is assumed to be known to the attacker. The ith record is thus $(\mathbf{x}_i, y_i)$. Let $X \in \mathbb{R}^{n\times d}$ be a matrix whose ith row is $\mathbf{x}_i$, and let $\mathbf{y} = (y_1,\dots,y_n)$. We denote the entire database $D = (X|\mathbf{y})$ where $|$ represents vertical concatenation. Given some released information ρ, the attacker constructs an estimate $\hat{\mathbf{y}}$ that she hopes is close to $\mathbf{y}$. We measure the attack's success in terms of the Hamming distance $d_H(\mathbf{y},\hat{\mathbf{y}})$. A scheme is *not* attribute private if an attacker can consistently get an estimate that is within distance $o(n)$. Formally:

▶ **Definition 11** (Failure of Attribute Privacy [15]). A (randomized) mechanism $\mathcal{M} : \mathbb{R}^{n\times d+1} \to \mathbb{R}^l$ is said to allow (θ,γ)-attribute reconstruction if there exists a setting of the nonsensitive

[9] We call a random variable $x \in \mathbb{R}$ subexponential if there exists a constant $C > 0$ if $\Pr[|x| > t] \leq 2\exp(-t/C)$ for all $t > 0$.



(a) Polynomial Kernel **(b)** Gaussian Kernel

Figure 1 Largest eigenvalue distribution for random kernel matrices constructed with a polynomial kernel (left plot) and a Gaussian kernel (right plot). The actual value plots are constructed by averaging over 100 runs, and in each run we draw n independent standard Gaussian vectors in $d = 100$ dimensions. The predicted values are computed from bounds in Theorems 9 and 10 (Part 3). The kernel matrix size n is varied from 10 to 10000 in multiples of 10. For the polynomial kernel, we set $a = 1, b = 1$, and $p = 4$, and for the Gaussian kernel $a = 3\log(n)/d$. Note that our upper bounds are fairly close to the actual results. For the Gaussian kernel, the actual values are very close to 1.

attributes $X \in \mathbb{R}^{n\times d}$ and an algorithm (adversary) $\mathcal{A} : \mathbb{R}^{n\times d} \times \mathbb{R}^{l} \to \mathbb{R}^{n}$ such that for every $\mathbf{y} \in \{0,1\}^n$,

$$\Pr_{\rho \leftarrow \mathcal{M}((X|\mathbf{y}))}[\mathcal{A}(X,\rho) = \hat{\mathbf{y}} \,:\, d_H(\mathbf{y},\hat{\mathbf{y}}) \le \theta] \ge 1-\gamma.$$

Asymptotically, we say that a mechanism is *attribute nonprivate* if there is an infinite sequence of n for which $\mathcal{M}$ allows $(o(n), o(1))$-attribute reconstruction. Here $d = d(n)$ is a function of n. We say the attack $\mathcal{A}$ is *efficient* if it runs in time $\mathrm{poly}(n,d)$.

4.1 Kernel Ridge Regression Background

One of the most basic regression formulation is that of ridge regression [10]. Suppose that we are given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ consisting of n points with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Here $\mathbf{x}_i$'s are referred to as the *regressors* and y_i's are the *response variables*. In linear regression the task is to find a linear function that models the dependencies between $\mathbf{x}_i$'s and the y_i's. A common way to prevent overfitting in linear regression is by adding a penalty regularization term (also known as *shrinkage* in statistics). In kernel ridge regression [21], we assume a model of form $y = f(\mathbf{x}) + \xi$, where we are trying to estimate the regression function f and ξ is some unknown vector that accounts for discrepancy between the actual response (y) and predicted outcome ($f(\mathbf{x})$). Given a reproducing kernel Hilbert space $\mathcal{H}$ with kernel κ, the goal of ridge regression kernel ridge regression is to estimate the unknown function $f^\star$ such the least-squares loss defined over the dataset with a weighted penalty based on the squared Hilbert norm is minimized.

$$\text{Kernel Ridge Regression:} \qquad \operatorname{argmin}_{f\in\mathcal{H}} \left(\frac{1}{n}\sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 + \lambda \|f\|_{\mathcal{H}}^2 \right), \tag{7}$$

where $\lambda > 0$ is a regularization parameter. By representer theorem [22], any solution $f^\star$ for (7), takes the form

$$f^\star(\cdot) = \sum_{i=1}^{n} \alpha_i \kappa(\cdot, \mathbf{x}_i), \tag{8}$$

where $\alpha = (\alpha_1, \dots, \alpha_n)$ is known as the kernel ridge regression coefficient vector. Plugging this representation into (7) and solving the resulting optimization problem (in terms of α now), we get that the minimum value is achieved for $\alpha = \alpha^\star$, where

$$\alpha^\star = (K + \lambda \mathbb{I}_n)^{-1}\mathbf{y}, \text{ where } K \text{ is the kernel matrix with } K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j) \text{ and } \mathbf{y} = (y_1, \dots, y_n). \tag{9}$$

Plugging this $\alpha^\star$ from (9) in to (8), gives the final form for estimate $f^\star(\cdot)$. For a new point $\mathbf{x} \in \mathbb{R}^d$, the predicted response is $f^\star(\mathbf{x}) = \sum_{i=1}^n \alpha_i^\star \kappa(\mathbf{x}, \mathbf{x}_i)$ where $\alpha^\star = (K + \lambda \mathbb{I}_n)^{-1}\mathbf{y}$ and $\alpha^\star = (\alpha_1^\star, \dots, \alpha_n^\star)$. Therefore, knowledge of $\alpha^\star$ and $\mathbf{x}_1, \dots, \mathbf{x}_n$ suffices for making future predictions.

If K is constructed using a polynomial kernel (defined in 1. in Section 2.2) then the above procedure is referred to as the *polynomial kernel ridge regression*, and similarly if K is constructed using a Gaussian kernel (defined in 2. in Section 2.2) then the above procedure is referred to as the *Gaussian kernel ridge regression*.

4.2 Reconstruction Attack from Noisy α^*

Algorithm 1 outlines the attack. The privacy mechanism releases a noisy approximation to $\alpha^\star$. Let $\tilde{\alpha}$ be this noisy approximation, i.e., $\tilde{\alpha} = \alpha^\star + \mathbf{e}$ where $\mathbf{e}$ is some unknown noise vector. The adversary tries to reconstruct an approximation $\hat{\mathbf{y}}$ of $\mathbf{y}$ from $\tilde{\alpha}$. The adversary solves the following ℓ_2-minimization problem to construct $\hat{\mathbf{y}}$:

$$\min_{\mathbf{z} \in \mathbb{R}^n} \|\tilde{\alpha} - (K + \lambda \mathbb{I}_n)^{-1}\mathbf{z}\|. \tag{10}$$

In the setting of attribute privacy, the database $D = (X|\mathbf{y})$. Let $\mathbf{x}_1, \dots, \mathbf{x}_n$ be the rows of X, using which the adversary can construct K to carry out the attack. Since the matrix $K + \lambda \mathbb{I}_n$ is invertible for $\lambda > 0$ as K is a positive semidefinite matrix, the solution to (10) is simply $\mathbf{z} = (K + \lambda \mathbb{I}_n)\tilde{\alpha}$, element-wise rounding of which to closest $0, 1$ gives $\hat{\mathbf{y}}$.

Algorithm 1 Reconstruction Attack from Noisy Kernel Ridge Regression Coefficients

Input: Public information $X \in \mathbb{R}^{n \times d}$, regularization parameter λ, and $\tilde{\alpha}$ (noisy $\alpha^\star$).

```
1: Let $\mathbf{x}_1, \dots, \mathbf{x}_n$ be the rows of $X$, construct the kernel matrix $K$ with $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$
2: Return $\hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_n)$ defined as follows:
       $\hat{y}_i = \begin{cases} 0 & \text{if } i\text{th entry in } (K + \lambda \mathbb{I}_n)\tilde{\alpha} < 1/2 \\ 1 & \text{otherwise} \end{cases}$
```

▶ **Lemma 12.** *Let $\tilde{\alpha} = \alpha^\star + \mathbf{e}$, where $\mathbf{e} \in \mathbb{R}^n$ is some unknown (noise) vector. If $\|\mathbf{e}\|_\infty \le \beta$ (absolute value of all entries in $\mathbf{e}$ is less than β), then $\hat{\mathbf{y}}$ returned by Algorithm 1 satisfies, $d_H(\mathbf{y}, \hat{\mathbf{y}}) \le 4(K + \lambda)^2\beta^2 n$. In particular, if $\beta = o\left(\frac{1}{\|K\|+\lambda}\right)$, then $d_H(\mathbf{y}, \hat{\mathbf{y}}) = o(n)$.*

Proof. Since $\alpha^\star = (K + \lambda\mathbb{I}_n)^{-1}\mathbf{y}$, $\tilde{\alpha} = (K + \lambda\mathbb{I}_n)^{-1}\mathbf{y} + \mathbf{e}$. Now multiplying $(K + \lambda\mathbb{I}_n)$ on both sides gives,

$$(K + \lambda\mathbb{I}_n)\tilde{\alpha} = \mathbf{y} + (K + \lambda\mathbb{I}_n)\mathbf{e}.$$

Concentrate on $\|(K + \lambda\mathbb{I}_n)\mathbf{e}\|$. This can be bound as

$$\|(K + \lambda\mathbb{I}_n)\mathbf{e}\| \leq \|(K + \lambda\mathbb{I}_n)\|\|\mathbf{e}\| = (\|K\| + \lambda)\|\mathbf{e}\|.$$

If the absolute value of all the entries in $\mathbf{e}$ are less than β then $\|\mathbf{e}\| \leq \beta\sqrt{n}$. A simple manipulation then shows that if the above hold then $(K + \lambda\mathbb{I}_n)\mathbf{e}$ cannot have more than $4(\|K\| + \lambda)^2\beta^2 n$ entries with absolute value above $1/2$. Since $\hat{\mathbf{y}}$ and $\mathbf{y}$ only differ in those entries where $(K + \lambda\mathbb{I}_n)\mathbf{e}$ is greater than $1/2$, it follows that $d_H(\mathbf{y}, \hat{\mathbf{y}}) \leq 4(\|K\| + \lambda)^2\beta^2 n$. Setting $\beta = o(\frac{1}{\|K\|+\lambda})$ implies $d_H(\mathbf{y}, \hat{\mathbf{y}}) = o(n)$. ◀

For a privacy mechanism to be attribute non-private, the adversary has to be able reconstruct an $1 - o(1)$ fraction of $\mathbf{y}$ with high probability. Using the above lemma, and the different bounds on $\|K\|$ established in Theorems 9 and 10, we get the following lower bounds for privately releasing kernel ridge regression coefficients.

▶ **Theorem 13.**

1. *Any privacy mechanism which for every database $D = (X|\mathbf{y})$ where $X \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \{0,1\}^n$ releases the coefficient vector of a polynomial kennel ridge regression model (for constants a, b, and p) fitted between X (matrix of regressor values) and $\mathbf{y}$ (response vector), by adding $o(\frac{1}{d^p n+\lambda})$ noise to each coordinate is attribute non-private. The attack that achieves this attribute privacy violation operates in $O(dn^2)$ time.*
2. *Any privacy mechanism which for every database $D = (X|\mathbf{y})$ where $X \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \{0,1\}^n$ releases the coefficient vector of a Gaussian kennel ridge regression model (for constant a) fitted between X (matrix of regressor values) and $\mathbf{y}$ (response vector), by adding $o(\frac{1}{2+\lambda})$ noise to each coordinate is attribute non-private. The attack that achieves this attribute privacy violation operates in $O(dn^2)$ time.*

Proof. For Part 1, draw each individual i's non-sensitive attribute vector $\mathbf{x}_i$ independently from any d-dimensional subgaussian distribution, and use Lemma 12 in conjunction with Theorem 9.

For Part 2, draw each individual i's non-sensitive attribute vector $\mathbf{x}_i$ independently from any product distribution formed from some d centered univariate subgaussian distributions, and use Lemma 12 in conjunction with Theorem 10 (Part 3).[10]

The time needed to construct the kernel matrix K is $O(dn^2)$, which dominates the overall computation time. ◀

We can ask how the above distortion needed for privacy compares to typical entries in $\alpha^\star$. The answer is not simple, but there are natural settings of inputs, where the noise needed for privacy becomes comparable with coordinates of $\alpha^\star$, implying that the privacy comes at a steep price. One such example is if the $\mathbf{x}_i$'s are drawn from the standard normal distribution, $\mathbf{y} = (1)^n$, and all other kernel parameters are constant, then the expected value of the corresponding $\alpha^\star$ coordinates match the noise bounds obtained in Theorem 13.

[10] Note that it is not critical for $\mathbf{x}_i$'s to be drawn from a product distribution. It is possible to analyze the attack even under a (weaker) assumption that each individual i's non-sensitive attribute vector $\mathbf{x}_i$ is drawn independently from a d-dimensional subgaussian distribution, by using Lemma 12 in conjunction with Theorem 10 (Part 1).

Note that Theorem 13 makes no assumptions on the dimension d of the data, and holds for all values of n, d. This is different from all other previous lower bounds for attribute privacy [15, 4, 14], all of which require d to be comparable to n, thereby holding only either when the non-sensitive data (the $\mathbf{x}_i$'s) are very high-dimensional or for very small n. Also all the previous lower bound analyses [15, 4, 14] critically rely on the fact that the individual coordinates of each of the $\mathbf{x}_i$'s are independent[11], which is not essential for Theorem 13.

4.3 Note on using ℓ_1-reconstruction Attacks

A natural alternative to (10) is to use ℓ_1-minimization (also known as "LP decoding"). This gives rise to the following linear program:

$$\min_{\mathbf{z}\in\mathbb{R}^n} \|\tilde{\alpha} - (K + \lambda\mathbb{I}_n)^{-1}\mathbf{z}\|_1. \tag{11}$$

In the context of privacy, the ℓ_1-minimization approach was first proposed by Dwork *et al.* [8], and recently reanalyzed in different contexts by [4, 14]. These results have shown that, for some settings, the ℓ_1-minimization can handle considerably more complex noise patterns than the ℓ_2-minimization. However, in our setting, since the solutions for (11) and (10) are exactly the same ($\mathbf{z} = (K + \lambda\mathbb{I}_n)\tilde{\alpha}$), there is no inherent advantage of using the ℓ_1-minimization.

5 Concluding Remarks

We initiate the study of non-asymptotic spectral properties of random kernel matrices, and provide tight bounds on the spectral norm of these matrices when constructed using kernel functions such as polynomials and Gaussian radial basis. Using these results, we provide lower bounds on the distortion needed for releasing coefficients of kernel ridge regression under attribute privacy, a general privacy notion that captures a large class of privacy definitions.

We believe that developing a non-asymptotic spectral theory for random kernel matrices is an interesting research direction that would provide deep insights into the workings of many kernel-based machine learning algorithms.

Acknowledgements. We are grateful for helpful initial discussions with Adam Smith and Ambuj Tewari.

References

1 Olivier Bousquet, Ulrike von Luxburg, and G Rätsch. Advanced Lectures on Machine Learning. In *ML Summer Schools 2003*, 2004.

2 Xiuyuan Cheng and Amit Singer. The Spectrum of Random Inner-Product Kernel Matrices. *Random Matrices: Theory and Applications*, 2(04), 2013.

3 Krzysztof Choromanski and Tal Malkin. The Power of the Dinur-Nissim Algorithm: Breaking Privacy of Statistical and Graph Databases. In *PODS*, pages 65–76. ACM, 2012.

4 Anindya De. Lower Bounds in Differential Privacy. In *TCC*, pages 321–338, 2012.

5 Irit Dinur and Kobbi Nissim. Revealing Information while Preserving Privacy. In *PODS*, pages 202–210. ACM, 2003.

6 Yen Do and Van Vu. The Spectrum of Random Kernel Matrices: Universality Results for Rough and Varying Kernels. *Random Matrices: Theory and Applications*, 2(03), 2013.

[11] This may not be a realistic assumption in many practical scenarios. For example, an individual's salary and postal address code are typically correlated.

7 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*, volume 3876 of *LNCS*, pages 265–284. Springer, 2006.

8 Cynthia Dwork, Frank McSherry, and Kunal Talwar. The Price of Privacy and the Limits of LP Decoding. In *STOC*, pages 85–94. ACM, 2007.

9 Cynthia Dwork and Sergey Yekhanin. New Efficient Attacks on Statistical Disclosure Control Mechanisms. In *CRYPTO*, pages 469–480. Springer, 2008.

10 Arthur E Hoerl and Robert W Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, 1970.

11 Prateek Jain and Abhradeep Thakurta. Differentially Private Learning with Kernels. In *ICML*, pages 118–126, 2013.

12 Lei Jia and Shizhong Liao. Accurate Probabilistic Error Bound for Eigenvalues of Kernel Matrix. In *Advances in Machine Learning*, pages 162–175. Springer, 2009.

13 Noureddine El Karoui. The Spectrum of Kernel Random Matrices. *The Annals of Statistics*, pages 1–50, 2010.

14 Shiva Prasad Kasiviswanathan, Mark Rudelson, and Adam Smith. The Power of Linear Reconstruction Attacks. In *SODA*, pages 1415–1433, 2013.

15 Shiva Prasad Kasiviswanathan, Mark Rudelson, Adam Smith, and Jonathan Ullman. The Price of Privately Releasing Contingency Tables and the Spectra of Random Matrices with Correlated Rows. In *STOC*, pages 775–784, 2010.

16 Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the Kernel Matrix with Semidefinite Programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

17 James Mercer. Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, pages 415–446, 1909.

18 Martin M Merener. Polynomial-time Attack on Output Perturbation Sanitizers for Real-valued Databases. *Journal of Privacy and Confidentiality*, 2(2):5, 2011.

19 S. Muthukrishnan and Aleksandar Nikolov. Optimal Private Halfspace Counting via Discrepancy. In *STOC*, pages 1285–1292, 2012.

20 Mark Rudelson. Recent Developments in Non-asymptotic Theory of Random Matrices. *Modern Aspects of Random Matrix Theory*, 72:83, 2014.

21 Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge Regression Learning Algorithm in Dual Variables. In *ICML*, pages 515–521, 1998.

22 Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A Generalized Representer Theorem. In *COLT*, pages 416–426, 2001.

23 Bernhard Scholkopf and Alexander J Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2001.

24 John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

25 John Shawe-Taylor, Christopher KI Williams, Nello Cristianini, and Jaz Kandola. On the Eigenspectrum of the Gram matrix and the Generalization Error of Kernel-PCA. *Information Theory, IEEE Transactions on*, 51(7):2510–2522, 2005.

26 Vikas Sindhwani, Minh Ha Quang, and Aurélie C Lozano. Scalable Matrix-valued Kernel Learning for High-dimensional Nonlinear Multivariate Regression and Granger Causality. *arXiv preprint arXiv:1210.4792*, 2012.

27 Roman Vershynin. Introduction to the Non-asymptotic Analysis of Random Matrices. *arXiv preprint arXiv:1011.3027*, 2010.

Separating Decision Tree Complexity from Subcube Partition Complexity*

Robin Kothari[1,2], David Racicot-Desloges[3,4], and Miklos Santha[4,5]

1 Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, MA, USA
rkothari@mit.edu

2 David R. Cheriton School of Computer Science and the Institute for Quantum Computing, University of Waterloo, Waterloo, ON, Canada

3 Département de Mathématiques, Faculté des Sciences, Université de Sherbrooke, Sherbrooke, QC, Canada
David.Racicot-Desloges@USherbrooke.ca

4 Centre for Quantum Technologies, National University of Singapore, Singapore

5 LIAFA, CNRS, Université Paris Diderot, Paris, France
miklos.santha@liafa.univ-paris-diderot.fr

Abstract

The subcube partition model of computation is at least as powerful as decision trees but no separation between these models was known. We show that there exists a function whose deterministic subcube partition complexity is asymptotically smaller than its randomized decision tree complexity, resolving an open problem of Friedgut, Kahn, and Wigderson ([7]). Our lower bound is based on the information-theoretic techniques first introduced to lower bound the randomized decision tree complexity of the recursive majority function.

We also show that the public-coin partition bound, the best known lower bound method for randomized decision tree complexity subsuming other general techniques such as block sensitivity, approximate degree, randomized certificate complexity, and the classical adversary bound, also lower bounds randomized subcube partition complexity. This shows that all these lower bound techniques cannot prove optimal lower bounds for randomized decision tree complexity, which answers an open question of Jain and Klauck ([9]) and Jain, Lee, and Vishnoi ([10]).

1998 ACM Subject Classification F.1.2 Modes of Computation

Keywords and phrases Decision tree complexity, query complexity, randomized algorithms, subcube partition complexity

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.915

1 Introduction

The decision tree is a widely studied model of computation. While we have made significant progress in understanding this model (e.g., see the survey by Buhrman and de Wolf [5]), questions from over 40 years ago still remain unsolved [19].

* The research is partially funded by the Singapore Ministry of Education and the National Research Foundation, also through the Tier 3 Grant "Random numbers from quantum processes," MOE2012-T3-1-009; the European Commission IST STREP project Quantum Algorithms (QALGO) 600700; the French ANR Blanc program under contract ANR-12-BS02-005 (RDAM project); and the ARO grant Contract Number W911NF-12-1-0486.

In the decision tree model, we wish to compute a function $f : \{0,1\}^n \to \{0,1\}$ on an input $x \in \{0,1\}^n$, but we only have access to the input via a black box. The black box can be queried with an index $i \in [n]$, where $[n] = \{1,2,\ldots,n\}$, and will respond with the value of x_i, the ith bit of x. The goal is to compute $f(x)$, while minimizing the number of queries made to the black box.

For a function $f : \{0,1\}^n \to \{0,1\}$, let $D(f)$ denote the deterministic query complexity (or decision tree complexity) of computing f, the minimum number of queries made by a deterministic algorithm that computes f correctly on all inputs. Let $R_0(f)$ denote the zero-error randomized query complexity of computing f, the minimum expected cost of a zero-error randomized algorithm that computes f correctly on all inputs. Finally, let $R(f)$ denote the bounded-error randomized query complexity of computing f, the number of queries made in the worst case by a randomized algorithm that outputs $f(x)$ on input x with probability at least $2/3$. More precise definitions can be found in Section 2.

Several lower bound techniques have been developed for query complexity over the years, most of which are based on the following observation: A decision tree that computes f and makes d queries partitions the set of all inputs, the hypercube $\{0,1\}^n$, into a set of monochromatic subcubes where each subcube has at most d fixed variables. A subcube is a restriction of the hypercube in which the values of some subset of the variables have been fixed. For example, the set of n-bit strings in which the first variable is set to 0 is a subcube of $\{0,1\}^n$ with one fixed variable. A subcube is monochromatic if f takes the same value on all inputs in the subcube. This idea is also the basis of many lower bound techniques in communication complexity [13], where a valid protocol partitions the space of inputs into monochromatic rectangles.

However, not all subcube partitions arise from decision trees, which naturally leads to a potentially more powerful model of computation. This model is called the subcube partition model in [7], but has been studied before under different names (see e.g., [4]). The deterministic subcube partition complexity of f, denoted by $D^{\mathrm{sc}}(f)$, is the minimum d such that there is a partition of the hypercube into a set of monochromatic subcubes in which each subcube has at most d fixed variables. Since a decision tree making d queries always gives rise to such a partition, we have $D^{\mathrm{sc}}(f) \le D(f)$. Similarly, we define zero-error and bounded-error versions of subcube partition complexity, denoted by $R_0^{\mathrm{sc}}(f)$ and $R^{\mathrm{sc}}(f)$, respectively, and obtain the inequalities $R_0^{\mathrm{sc}}(f) \le R_0(f)$ and $R^{\mathrm{sc}}(f) \le R(f)$. As expected, we also have $R_0^{\mathrm{sc}}(f) \le D^{\mathrm{sc}}(f)$ and $R^{\mathrm{sc}}(f) \le D^{\mathrm{sc}}(f)$.

This brings up the obvious question of whether these models are equivalent. Separating them is difficult, precisely because most lower bound techniques for query complexity also lower bound subcube partition complexity. The analogous question in communication complexity is also a long-standing open problem (see [13, Open Problem 2.10] or [12, Chapter 3.2]). In fact, Friedgut, Kahn, and Wigderson [7, Question 1.1] explicitly ask if these measures are asymptotically different in the randomized model with zero error:

▶ **Question 1.** *Is there a function (family) $f = (f_n)$ such that $R_0^{\mathrm{sc}}(f) = o(R_0(f))$?*

Similarly, one can ask the same question for bounded-error randomized query complexity. The main result of this paper resolves these questions:

▶ **Theorem 2.** *There exists a function $f = (f_h)$, with $f_h : \{0,1\}^{4^h} \to \{0,1\}$, such that $D^{\mathrm{sc}}(f) \le 3^h$, but $D(f) = 4^h$, $R_0(f) \ge 3.2^h$, and $R(f) = \Omega(3.2^h)$.*

This shows that query complexity and subcube partition complexity are asymptotically different in the deterministic, zero-error, and bounded-error settings. Besides resolving this question, our result has another application. We know several techniques to lower bound bounded-error randomized query complexity, such as approximate polynomial degree [18], block sensitivity [17], randomized certificate complexity [1] and the classical adversary bound [15, 22, 2]. All these techniques are subsumed by the partition bound of Jain and Klauck [9], which in turn is subsumed by the public-coin partition bound of Jain, Lee, and Vishnoi [10]. Additionally, this new lower bound is within a quadratic factor of randomized query complexity. In other words, if $\mathrm{PPRT}(f)$ denotes the bounded-error public-coin partition bound for a function f, we have $\mathrm{PPRT}(f) \leq R(f)$ and also $R(f) = O(\mathrm{PPRT}(f)^2)$. This leaves open the intriguing possibility that this technique is optimal and is asymptotically equal to bounded-error randomized query complexity. Jain, Lee, and Vishnoi [10] indeed ask the following question:

▶ **Question 3.** *Is there a function (family) $f = (f_n)$ such that $\mathrm{PPRT}(f) = o(R(f))$?*

Our result also answers this question, because, as we show in Section 2, $\mathrm{PPRT}(f) \leq R^{\mathrm{sc}}(f)$. Thus, our asymptotic separation between $R^{\mathrm{sc}}(f)$ and $R(f)$ also separates $\mathrm{PPRT}(f)$ from $R(f)$.

We now provide a high-level overview of the techniques used in this paper. The main result is based on establishing the various complexities of a certain function. The function we choose is based on the quaternary majority function $\mathsf{4\text{-}MAJ} : \{0,1\}^4 \to \{0,1\}$, defined as the majority of the four input bits, with ties broken by the first bit. This function has low deterministic subcube complexity, $D^{\mathrm{sc}}(\mathsf{4\text{-}MAJ}) \leq 3$, but has deterministic query complexity $D(\mathsf{4\text{-}MAJ}) = 4$. From this function, we define an iterated function $\mathsf{4\text{-}MAJ}_h$ on 4^h variables by composing the function with itself h times, which gives us a function on 4^h bits. Since deterministic query complexity and deterministic subcube complexity behave nicely under composition, we have $D(\mathsf{4\text{-}MAJ}_h) = 4^h$ and $D^{\mathrm{sc}}(\mathsf{4\text{-}MAJ}_h) \leq 3^h$. The same function was also used by Savický [21], who studied this question in terms of decision tree size, as opposed to decision tree depth. These results are further discussed in Section 3. To prove Theorem 2, it remains to show that the randomized query complexity of this function is $\Omega(3.2^h)$.

We lower bound the randomized query complexity of $\mathsf{4\text{-}MAJ}_h$ using a strategy similar to the information-theoretic technique of Jayram, Kumar, and Sivakumar [11] and its simplification by Landau, Nachmias, Peres, and Vanniasegaram [14]. However, the original strategy was applied to lower bound a symmetric function (iterated $\mathsf{3\text{-}MAJ}$), whereas our function is not symmetric since the first variable of $\mathsf{4\text{-}MAJ}$ is different from the rest. We modify the technique to apply it to asymmetric functions and establish the claimed lower bound. The lower bound relies on choosing a "hard distribution" of inputs and establishing a recurrence relation between the complexities of the function and its subfunctions on this distribution. Unlike $\mathsf{3\text{-}MAJ}$, where there is a natural candidate for a hard distribution, our chosen distribution is not obvious and is constrained by the fact that it must fit nicely into these recurrence relations. We prove this lower bound in Section 4. We end with some discussion and open problems in Section 5.

Subsequent work

Independent of our work, Göös, Pitassi, and Watson [8] improved the separation between deterministic query complexity and deterministic subcube complexity by exhibiting a function on n bits with deterministic query complexity $\tilde{\Omega}(n)$ and deterministic subcube complexity $\tilde{O}(n^{2/3})$. It may be possible to adapt their ideas, as done in [3], to improve the separation between randomized query complexity and subcube complexity.

2 Preliminaries

In this section, we formally define the various models of query complexity and subcube partition complexity, and the partition bound [9] and public-coin partition bound [10]. We then study the relationships between these quantities.

For the remainder of the paper, let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function on n bits and $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ be any input. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$ and let the support of a probability distribution p be denoted by $\mathrm{supp}(p)$. Lastly, we require the notion of composing two Boolean functions. If $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$ are two Boolean functions, the *composed function* $f \circ g : \{0,1\}^{nm} \to \{0,1\}$ acts on the Boolean string $y = (y_{11}, \ldots, y_{1m}, y_{21}, \ldots, y_{nm})$ as $f \circ g(y) = f(g(y_{11}, \ldots, y_{1m}), \ldots, g(y_{n1}, \ldots, y_{nm}))$.

2.1 Decision tree or query complexity

The deterministic query complexity of a function f, $D(f)$, is the minimum number of queries made by a deterministic algorithm that computes f correctly.

Formally, a *deterministic decision tree A on n variables* is a binary tree in which each leaf is labeled by either a 0 or a 1, and each internal node is labeled with a value $i \in [n]$. For every internal node of A, one of the two outgoing edges is labeled 0 and the other edge is labeled 1. On an input x, the algorithm A follows the unique path from the root to one of its leaves in the natural way: for an internal node labeled with the value i, it follows the outgoing edge labeled by x_i. The output $A(x)$ of the algorithm A on input x is the label of the leaf of this path. We say that the decision tree A *computes* f if $A(x) = f(x)$ for all x.

We define the *cost* of algorithm A on input x, denoted by $C(A, x)$, to be the number of bits queried by A on x, that is the number of internal nodes evaluated by A on x. The cost of an algorithm A, denoted $C(A)$, is the worst-case cost of the algorithm over all inputs x, that is $C(A) = \max_x C(A, x)$. Now, let $\mathcal{D}_n$ denote the set of all deterministic decision trees on n variables and let $\mathcal{D}(f) \subseteq \mathcal{D}_n$ be the set of all deterministic decision trees that compute f. We define the *deterministic query complexity of f* as $D(f) = \min_{A \in \mathcal{D}(f)} C(A)$.

One of the features of deterministic query complexity that we use in this paper is its composition property [23, 16]. This property is very intuitive: it asserts that the best way to compute the composition of f and g is to use optimal algorithms for f and g independently.

▶ **Proposition 4.** *For any two Boolean functions f and g, $D(f \circ g) = D(f)D(g)$.*

We can now move on to randomized analogs of deterministic query complexity. In a randomized algorithm, the choice of the queries might also depend on some randomness. Formally, a *randomized decision tree B on n variables* is defined by a probability distribution b over $\mathcal{D}_n$, that is by a function $b : \mathcal{D}_n \to [0,1]$ such that $\sum_{A \in \mathcal{D}_n} b(A) = 1$. On an input x, the algorithm B picks a deterministic decision tree A with probability $b(A)$ and outputs $A(x)$. Thus, for every x, the value $B(x)$ of B on x is a random variable.

We say that a randomized algorithm B computes f with *error* $\varepsilon \geq 0$ if $\Pr[B(x) = f(x)] \geq 1 - \varepsilon$ for all x, that is if $\sum_{A(x)=f(x)} b(A) \geq 1 - \varepsilon$ for all x. Let $\mathcal{R}_n$ be the set of all randomized decision trees over n bits and let $\mathcal{R}_\varepsilon(f) \subseteq \mathcal{R}_n$ be the set of all randomized decision trees that compute f with error ε. A randomized algorithm B then computes f with zero error if $\mathrm{supp}(b) \subseteq \mathcal{D}(f)$, that is the probability distribution b is completely supported on the set of deterministic decision trees that compute f. A zero-error randomized algorithm, also known as a Las Vegas algorithm, always outputs the correct answer. The cost of a zero-error randomized algorithm B on x is defined as $C(B, x) = \sum_{A \in \mathcal{D}_n} b(A)C(A, x) = \mathbb{E}[C(A, x)]$, the expected number of queries made on input x. The *zero-error randomized query complexity of*

f, denoted by $R_0(f)$, is defined as $R_0(f) = \min_{B\in\mathcal{R}_0(f)} \max_x C(B,x)$. From the definition of zero-error randomized query complexity, it is clear that $R_0(f) \le D(f)$. The complexity $R_0(f)$ can be of strictly smaller order of growth than $D(f)$: there exists a function f for which $R_0(f) = o(D(f))$, e.g., the iterated NAND-function [20].

Randomized algorithms with error $\varepsilon > 0$ might give incorrect answer on their inputs with probability ε. We say that a randomized algorithm is of *bounded-error* (sometimes called a Monte Carlo algorithm) if on any input x, the probabilistic output is incorrect with probability at most 1/3. The constant 1/3 is not important and replacing it with any constant strictly between 0 and 1/2 will only change the complexity by a constant multiplicative factor. For $\varepsilon > 0$, the cost of an ε-error randomized algorithm B on x is defined as $C(B,x) = \max_{A\in\mathrm{supp}(b)} C(A,x)$, the maximum number of queries made on input x by an algorithm in the support of b. Note how this definition differs from the one given for the zero-error case. We define the *ε-error randomized complexity of f* as $R_\varepsilon(f) = \min_{B\in\mathcal{R}_\varepsilon(f)} \max_x C(B,x)$, and the *bounded-error randomized query complexity of f* as $R(f) = R_{1/3}(f)$. Note that this definition is valid only for $\varepsilon > 0$ and does not coincide with $R_0(f)$ defined above for $\varepsilon = 0$. Setting $\varepsilon = 0$ in this definition simply gives us the deterministic query complexity $D(f)$. Nonetheless, it is true that $R(f) = O(R_0(f))$. This distinction is discussed in more detail in Section 5. Lastly, note that for all $\varepsilon > 0$, we have $R_\varepsilon(f) \le D(f)$, and that there exist functions for which $R(f) = o(D(f))$ [20].

In order to establish lower bounds on randomized query complexity, it is useful to take a distributional view of randomized algorithms [24], that is to consider the performance of randomized algorithms on a chosen distribution over inputs. Let μ be a probability distribution over all possible inputs of length n, and let B be a randomized decision tree algorithm. The cost of B under μ is $C(B,\mu) = \sum_{x\in\{0,1\}^n} \mu(x)C(B,x) = \mathbb{E}[C(B,x)]$. We define the *$\varepsilon$-error distributional complexity of f under μ* as $\Delta^\mu_\varepsilon(f) = \min_{B\in\mathcal{R}_\varepsilon(f)} C(B,\mu)$. The following simple fact is the basis of many lower bound arguments.

▶ **Proposition 5.** *For every distribution μ over $\{0,1\}^n$, and for all $\varepsilon \ge 0$, we have $\Delta^\mu_\varepsilon(f) \le R_\varepsilon(f)$.*

Proof. This follows by expanding out the definitions and using the simple inequality between expectation and maximum:

$$\Delta^\mu_\varepsilon(f) = \min_{B\in\mathcal{R}_\varepsilon(f)} C(B,\mu) = \min_{B\in\mathcal{R}_\varepsilon(f)} \mathbb{E}[C(B,x)] \le \min_{B\in\mathcal{R}_\varepsilon(f)} \max_x C(B,x) = R_\varepsilon(f). \tag{1}$$

◀

2.2 Subcube partition complexity

A subcube of the hypercube $\{0,1\}^n$ is a set of n-bit strings obtained by fixing the values of some subset of the variables. In other words, a subcube is the set of all inputs consistent with a partial assignment of n bits. Formally, a *partial assignment on n variables* is a function $a : I_a \to \{0,1\}$, with $I_a \subseteq [n]$. Given a partial assignment a, we call $S(a) = \{y \in \{0,1\}^n : y_i = a(i) \text{ for all } i \in I_a\}$ the *subcube generated by a*. A set $S \subseteq \{0,1\}^n$ is a *subcube* of the hypercube $\{0,1\}^n$ if $S = S(a)$ for some partial assignment on n variables a. Clearly, for every subcube, there exists exactly one such a. We denote by I_S the domain $I_a \subseteq [n]$ of a where $S = S(a)$. For example, the set $\{0100, 0101, 0110, 0111\}$ is a subcube of $\{0,1\}^4$. It is generated by the partial assignment $a : \{1,2\} \to \{0,1\}$, where $a(1) = 0$ and $a(2) = 1$. An alternative representation of a partial assignment is by an n-bit string where a position i takes the value $a(i)$ if $i \in I_a$ and takes the value $*$ otherwise. For this example, the subcube

$\{0100, 0101, 0110, 0111\}$ is generated by the partial assignment $01**$. Finally, another useful representation is in terms of a conjunction of literals, that is satisfied by all strings in the subcube. For example, the subcube $\{0100, 0101, 0110, 0111\}$ consists exactly of all 4-bit strings that satisfy the formula $\overline{x_1} \wedge x_2$.

The subcube partition model of computation, studied previously in [7, 4, 6], is a generalization of the decision tree model. A *partition* $\{S_1, \ldots, S_\ell\}$ of $\{0,1\}^n$ is a set of pairwise disjoint subsets of $\{0,1\}^n$ that together cover the entire hypercube, that is $\bigcup_i S_i = \{0,1\}^n$ and $S_i \cap S_j = \emptyset$ for $i \neq j$.

A *deterministic subcube partition P on n variables* is a partition of $\{0,1\}^n$ with a Boolean value $s \in \{0,1\}$ associated to each subcube, that is $P = \{(S_1, s_1), (S_2, s_2), \ldots, (S_\ell, s_\ell)\}$, where each S_i is a subcube and $\{S_1, \ldots, S_\ell\}$ is a partition of $\{0,1\}^n$. If the assignment a generates S_i for some i, we call a a *generating assignment for P*. For any x, we let S^x denote the subcube containing x, that is, if $x \in S_i$, then $S^x = S_i$. We define the value $P(x)$ of P on x as s_i.

We say that a deterministic subcube partition P computes f if $P(x) = f(x)$ for all x. Note that every deterministic decision tree algorithm A computing f induces a subcube partition computing f that consists of the subcubes generated by the partial assignments defined by the root–leaf paths of the tree and the Boolean values of the corresponding leaves. We define the cost of P on x as $C(P, x) = |I_{S^x}|$, analogous to the number of queries made on input x in query complexity. We define the worst-case cost as $C(P) = \max_x C(P, x)$. Let $\mathcal{D}_n^{\mathrm{sc}}$ be the set of all deterministic subcube partitions on n variables and let $\mathcal{D}^{\mathrm{sc}}(f) \subseteq \mathcal{D}_n^{\mathrm{sc}}$ be those partitions that compute f. We define the *deterministic subcube partition complexity of f* as $D^{\mathrm{sc}}(f) = \min_{P \in \mathcal{D}^{\mathrm{sc}}(f)} C(P)$. Deterministic subcube partition complexity also satisfies a composition theorem.

▶ **Proposition 6.** *For any $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$, $D^{\mathrm{sc}}(f \circ g) \leq D^{\mathrm{sc}}(f) D^{\mathrm{sc}}(g)$.*

Proof. Let $P = \{(S_1, s_1), (S_2, s_2), \ldots, (S_p, s_p)\}$ and $Q = \{(T_1, t_1), \ldots, (T_q, t_q)\}$ be optimal deterministic subcube partitions computing f and g respectively. Suppose that S_h is generated by a_h for $h \in [p]$, and that T_j is generated by b_j for $j \in [q]$. Let $I_{a_h} = \{i_1, \ldots, i_{c_h}\}$. We define the deterministic subcube partition $P \circ Q$ on nm variables as follows. The generating assignments for $P \circ Q$ are $a_h \circ (b_{j_1}, \ldots, b_{j_{c_h}})$, for all $h \in [p]$, and $j_1, \ldots, j_{c_h} \in [q]$ that satisfy $a(i_k) = t_{j_k}$ for $k \in [c_h]$. When $|I_{b_{j_k}}| = d_k$, the assignment $e = a_h \circ (b_{j_1}, \ldots, b_{j_{c_h}})$ is defined by $I_e = \{(1,1), \ldots, (1, d_1), (2, 1), \ldots, (c_h, d_{c_h})\}$, and $e(k, r) = b_{j_k}(r)$ for $1 \leq r \leq d_k$. The Boolean value associated with e is s_h. It is easy to check that $P \circ Q$ computes $f \circ g$ and that $C(P \circ Q) \leq C(P)C(Q)$. ◀

As in the case of query complexity, we extend deterministic subcube complexity to the randomized setting. A *randomized subcube partition R on n variables* is given by a distribution r over all deterministic subcube partitions on n variables. As for randomized decision trees, $R(x)$ is a random variable and we say that R computes f with error $\varepsilon \geq 0$ if $\Pr[R(x) = f(x)] \geq 1 - \varepsilon$ for all x. Let $\mathcal{R}_n^{\mathrm{sc}}$ be the set of all randomized subcube partitions over n variables and $\mathcal{R}_\varepsilon^{\mathrm{sc}}(f) \subseteq \mathcal{R}_n^{\mathrm{sc}}$ be the set of all randomized subcube partitions that compute f with error ε.

The cost of a zero-error randomized subcube partition R on x is defined by $C(R, x) = \mathbb{E}[C(P, x)]$, where the expectation is taken over R. For an ε-error subcube partition R, with $\varepsilon > 0$, the cost on x is $C(R, x) = \max_{P \in \mathrm{supp}(r)} C(P, x)$. For $\varepsilon \geq 0$, we define the *ε-error randomized subcube complexity of f* by $R_\varepsilon^{\mathrm{sc}}(f) = \min_{R \in \mathcal{R}_\varepsilon^{\mathrm{sc}}(f)} \max_x C(R, x)$.

As mentioned before, a deterministic decision tree induces a deterministic subcube partition with the same cost and thus a randomized decision tree induces a randomized subcube partition with the same cost, which yields the following.

▶ **Proposition 7.** *For an n-bit Boolean function $f : \{0,1\}^n \to \{0,1\}$, we have that $D^{\mathrm{sc}}(f) \leq D(f)$ and, for all $\varepsilon \geq 0$, we have that $R^{\mathrm{sc}}_{\varepsilon}(f) \leq R_{\varepsilon}(f)$.*

2.3 Partition bounds

In 2010, Jain and Klauck [9] introduced a linear programming based lower bound technique for randomized query complexity called the partition bound. They showed that it subsumes all known general lower bound methods for randomized query complexity, including approximate polynomial degree [18], block sensitivity [17], randomized certificate complexity [1], and the classical adversary bound [15, 22, 2].

Recently, Jain, Lee, and Vishnoi [10] presented a modification of this method called the public-coin partition bound, which is easily seen to be stronger than the partition bound. Furthermore, they were able to show that the gap between this new lower bound and randomized query complexity can be at most quadratic. We define these lower bounds formally.

▶ **Definition 8** (Partition bound). Let $f : \{0,1\}^n \to \{0,1\}$ be an n-bit Boolean function and let $\mathcal{S}_n$ denote the set of all subcubes of $\{0,1\}^n$. Then, for any $\varepsilon \geq 0$, let $\mathrm{prt}_{\varepsilon}(f)$ be the optimal value of the following linear program:

$$\underset{w_{S,z}}{\text{minimize:}} \quad \sum_{z=0}^{1} \sum_{S \in \mathcal{S}_n} w_{S,z} \cdot 2^{|I_S|} \tag{2}$$

$$\text{subject to:} \quad \sum_{S : x \in S} w_{S,f(x)} \geq 1 - \varepsilon \qquad (\text{for all } x \in \{0,1\}^n), \tag{3}$$

$$\sum_{S : x \in S} \sum_{z=0}^{1} w_{S,z} = 1 \qquad (\text{for all } x \in \{0,1\}^n), \tag{4}$$

$$w_{S,z} \geq 0 \qquad (\text{for all } S \in \mathcal{S}_n \text{ and } z \in \{0,1\}). \tag{5}$$

The ε-*partition bound* of f is defined as $\mathrm{PRT}_{\varepsilon}(f) = \frac{1}{2}\log_2(\mathrm{prt}_{\varepsilon}(f))$.

We now define the public-coin partition bound. Although our definition differs from the original definition [10], it is not too hard to see that they are equivalent. Before presenting the definition, recall that $\mathcal{D}^{\mathrm{sc}}_n$ is the set of deterministic subcube partitions on n variables, and $\mathcal{R}^{\mathrm{sc}}_{\varepsilon}(f)$ is the set of randomized subcube partitions that compute f with error at most $\varepsilon \geq 0$. For a randomized subcube partition $R \in \mathcal{R}^{\mathrm{sc}}_{\varepsilon}(f)$, we let r be the probability distribution over deterministic subcube partitions corresponding to R.

▶ **Definition 9** (Public-coin partition bound). Let $f : \{0,1\}^n \to \{0,1\}$ be an n-bit Boolean function. Then, for any $\varepsilon \geq 0$, let $\mathrm{pprt}_{\varepsilon}(f)$ be the optimal value of the following linear program:

$$\underset{R}{\text{minimize:}} \quad \sum_{z=0}^{1} \sum_{S \in \mathcal{S}_n} \sum_{P : (S,z) \in P} r(P) \cdot 2^{|I_S|} \tag{6}$$

$$\text{subject to:} \quad R \in \mathcal{R}^{\mathrm{sc}}_{\varepsilon}(f). \tag{7}$$

The ε-*public-coin partition bound of* f is defined as $\mathrm{PPRT}_{\varepsilon}(f) = \frac{1}{2}\log_2(\mathrm{pprt}_{\varepsilon}(f))$.

Using the original definition, it is trivial that $\mathrm{prt}_\varepsilon(f) \leq \mathrm{pprt}_\varepsilon(f)$, since the public-coin partition bound is defined using the same linear program, with additional constraints. This statement also holds with the definitions given above, as we now prove.

▶ **Proposition 10.** *For any Boolean function f and for all $\varepsilon \geq 0$, we have that $\mathrm{prt}_\varepsilon(f) \leq \mathrm{pprt}_\varepsilon(f)$.*

Proof. Let R' be a randomized subcube partition achieving the optimal value for the linear program of $\mathrm{pprt}_\varepsilon(f)$ and r' be the corresponding probability distribution over deterministic subcube partitions. Then, for all (S, z) where S is a subcube and $z \in \{0, 1\}$, let

$$w'_{S,z} = \sum_{P:(S,z)\in P} r'(P). \tag{8}$$

This family of variables satisfies the conditions of the $\mathrm{pprt}_\varepsilon(f)$ linear program and is such that

$$\sum_{z=0}^{1} \sum_{S\in\mathcal{S}_n} w'_{S,z} \cdot 2^{|I_S|} = \sum_{z=0}^{1} \sum_{S\in\mathcal{S}_n} \sum_{P:(S,z)\in P} r'(P) \cdot 2^{|I_S|}. \tag{9}$$

◀

Recall that both partition bounds lower bound randomized query complexity, as shown in [10]. In particular, for all $\varepsilon > 0$, $\mathrm{PRT}_\varepsilon(f) \leq \mathrm{PPRT}_\varepsilon(f) \leq R_\varepsilon(f)$ and, when $\varepsilon = 0$, we have that $\mathrm{PRT}_0(f) \leq \mathrm{PPRT}_0(f) \leq D(f)$. It is not known if the zero-error partition bound also lower bounds zero-error randomized query complexity. However, as mentioned, the partition bounds also lower bound subcube partition complexity, which implies that they lower bound query complexity. The proof for query complexity easily extends to subcube partition complexity.

▶ **Proposition 11.** *For every Boolean function f and for all $\varepsilon > 0$, we have that $\mathrm{PPRT}_\varepsilon(f) \leq R^{\mathrm{sc}}_\varepsilon(f)$ and $\mathrm{PPRT}_0(f) \leq D^{\mathrm{sc}}(f)$.*

Proof. Let $R' \in \mathcal{R}^{\mathrm{sc}}_\varepsilon(f)$ be a randomized subcube partition that achieves $R^{\mathrm{sc}}_\varepsilon(f)$ and let r' be its corresponding probability distribution over deterministic subcube partitions. Let $P \in \mathrm{supp}(r')$. By definition, for every $(S, z) \in P$, we have that $|I_S| \leq C(P)$. Also by definition, $C(P) \leq R^{\mathrm{sc}}_\varepsilon(f)$. Furthermore, if $P = \{(S_1, z_1), (S_2, z_2), \ldots, (S_m, z_m)\}$, then

$$|P| \cdot 2^{n-C(P)} = m \cdot 2^{n-C(P)} \leq \sum_{i=1}^{m} 2^{n-|I_{S_i}|} = 2^n. \tag{10}$$

This implies that $|P| \leq 2^{C(P)} \leq 2^{R^{\mathrm{sc}}_\varepsilon(f)}$ and, therefore, that

$$\begin{aligned} \mathrm{pprt}_\varepsilon(f) &= \sum_{z=0}^{1} \sum_{S\in\mathcal{S}_n} \sum_{P:(S,z)\in P} r'(P) \cdot 2^{|I_S|} \;\leq\; 2^{R^{\mathrm{sc}}_\varepsilon(f)} \sum_{z=0}^{1} \sum_{S\in\mathcal{S}_n} \sum_{P:(S,z)\in P} r'(P) && (11)\\ &= 2^{R^{\mathrm{sc}}_\varepsilon(f)} \sum_{P\in\mathrm{supp}(r')} r'(P) \cdot |P| \;\leq\; 2^{R^{\mathrm{sc}}_\varepsilon(f)} \cdot 2^{R^{\mathrm{sc}}_\varepsilon(f)} \sum_{P\in\mathrm{supp}(r')} r'(P) && (12)\\ &= 2^{2R^{\mathrm{sc}}_\varepsilon(f)}. && (13) \end{aligned}$$

The first inequality holds since $|I_S| \leq R^{\mathrm{sc}}_\varepsilon(f)$, and the second inequality uses the fact that $|P| \leq 2^{R^{\mathrm{sc}}_\varepsilon(f)}$. Setting $\varepsilon = 0$ gives $\mathrm{PPRT}_0(f) \leq D^{\mathrm{sc}}(f)$. ◀

$$\begin{tikzcd}
 & D(f) & \\
R_\varepsilon(f) \arrow[ur] & & R_0(f) \arrow[ul] \\
 & D^{\mathrm{sc}}(f) \arrow[uu] & \\
R_\varepsilon^{\mathrm{sc}}(f) \arrow[uu] \arrow[ur] & & R_0^{\mathrm{sc}}(f) \arrow[uu] \arrow[ul] \\
 & \mathrm{PPRT}_0(f) \arrow[uu] & \\
\mathrm{PPRT}_\varepsilon(f) \arrow[uu] \arrow[ur] & &
\end{tikzcd}$$

Figure 1 Relationships between the complexity measures introduced. An arrow from X to Y represents $X \le Y$. For example, $D^{\mathrm{sc}}(f) \to D(f)$ means $D^{\mathrm{sc}}(f) \le D(f)$.

The following theorem summarizes the known relations between the introduced complexity measures.

▶ **Theorem 12.** *For any Boolean function $f : \{0,1\}^n \to \{0,1\}$ and for all $\varepsilon > 0$, the relations indicated in Figure 1 hold.*

Proof. The upper three vertical arrows represent the relations between query complexity and subcube partition complexity established in Proposition 7. The remaining vertical arrows represent the relations between the public-coin partition bounds and subcube partition complexity established in Proposition 11. The other inequalities are immediate and follow from their definitions. ◀

3 Iterated quaternary majority function

We now introduce the function we use to separate randomized query complexity from subcube partition complexity and establish some of its properties.

Let MAJ denote the Boolean majority function of its input bits when the number of bits is odd. The quaternary majority function 4-MAJ : $\{0,1\}^4 \to \{0,1\}$ is defined by 4-MAJ$(x_1, x_2, x_3, x_4) = x_1(x_2 \vee x_3 \vee x_4) \vee x_2x_3x_4$. This function was introduced in [21]. We call it 4-MAJ, because the output of the function is the majority of its input bits, with the first variable breaking equality in its favor. In other words, the first variable has two votes, while the others have one, that is 4-MAJ(x_1, x_2, x_3, x_4) = MAJ$(x_1, x_1, x_2, x_3, x_4)$. This function has previously been used to separate deterministic decision tree size from deterministic subcube partition size [21]. We use this function because its subcube partition complexity is smaller than its query complexity.

▶ **Proposition 13.** *We have $D^{\mathrm{sc}}(\text{4-MAJ}) = 3$ and $D(\text{4-MAJ}) = 4$.*

Proof. Observe that, for any choice of $w \in \{0,1\}$, we have that

$$\text{4-MAJ}(0,0,1,w) = \text{4-MAJ}(0,w,0,1) = \text{4-MAJ}(0,1,w,0) = \text{4-MAJ}(w,0,0,0) = 0$$

and that

$$\text{4-MAJ}(1,1,0,w) = \text{4-MAJ}(1,w,1,0) = \text{4-MAJ}(1,0,w,1) = \text{4-MAJ}(w,1,1,1) = 1.$$

The subcubes generated by these 8 partial assignments are disjoint and of size two, forming a partition of $\{0,1\}^4$. Thus, with the right Boolean values, they form a deterministic subcube partition that computes 4-MAJ. Since all partial assignments have length 3, $D^{\mathrm{sc}}(\text{4-MAJ}) \leq 3$. Although we do not use the inequality $D^{\mathrm{sc}}(\text{4-MAJ}) \geq 3$ in our results, this can be verified by enumerating all deterministic subcube partitions with complexity 2. Furthermore, $D(\text{4-MAJ}) \leq 4$ since any function can be computed by querying all input bits. $D(\text{4-MAJ}) \geq 4$ can be shown either by enumerating all decision trees that make 3 queries or by using the lower bound in the next section. ◀

While our results only require us to show lower bounds on the randomized query complexity of 4-MAJ, we want to mention that the randomized query complexity of 4-MAJ is indeed smaller than its deterministic query complexity.

▶ **Proposition 14.** *For the* 4-MAJ *function,* $R_0(\text{4-MAJ}) \leq 13/4 = 3.25$.

Proof. The randomized algorithm achieving this complexity is simple: with probability 1/4, the algorithm queries the first variable and then it checks if the other variables all have the opposite value; with probability 3/4, it checks if the last three variables have all the same value and, if not, it queries the first variable. ◀

Since the 4-MAJ function separates deterministic subcube complexity from deterministic query complexity, a natural candidate for a function family that separates these measures is the iterated quaternary majority function, 4-MAJ_h, defined recursively on 4^h variables, for $h \geq 0$. In the base case, 4-MAJ_0 is the identity function on one bit. For $h > 0$, we define $\text{4-MAJ}_h = \text{4-MAJ} \circ \text{4-MAJ}_{h-1}$. In other words, for $h > 0$, let x be an input of length 4^h, and for $i \in \{1, 2, 3, 4\}$, let $x^{(i)}$ denote the i^{th} quarter of x, that is $|x^{(i)}| = 4^{h-1}$ and $x = x^{(1)}x^{(2)}x^{(3)}x^{(4)}$. Then, we have that $\text{4-MAJ}_h(x) = \text{4-MAJ}(\text{4-MAJ}_{h-1}(x^{(1)}), \text{4-MAJ}_{h-1}(x^{(2)}), \text{4-MAJ}_{h-1}(x^{(3)}), \text{4-MAJ}_{h-1}(x^{(4)}))$.

The function 4-MAJ_h inherits several properties from 4-MAJ. It has low deterministic subcube complexity, but high deterministic query complexity:

▶ **Proposition 15.** *For all* $h \geq 0$, $D^{\mathrm{sc}}(\text{4-MAJ}_h) \leq 3^h$ *and* $D(\text{4-MAJ}_h) = 4^h$.

Proof. For $h = 0$, the statement is trivial and for $h = 1$, the statement is Proposition 13. Proposition 4 and Proposition 6 used recursively imply the result. ◀

We now introduce terminology that we use to refer to this function. We view 4-MAJ_h as defined by the read-once formula on the complete quaternary tree T_h of height h in which every internal node is a 4-MAJ gate. We identify the leaves of T_h from left to right with the integers $1, \ldots, 4^h$. For an input $x \in \{0,1\}^{4^h}$, the bit x_i defines the *value* of the leaf i. We then evaluate recursively the values of the internal nodes. The value of the root is $\text{4-MAJ}_h(x)$. For every internal node v in T_h, we denote its children by v_1, v_2, v_3 and v_4, from left to right. For any node v in T_h, let $Z(v)$ denote the set of variables associated with the leaves in the subtree rooted at v. We say that a node v is at level ℓ in T_h if the distance between v and the leaves is ℓ. The root is therefore at level h, and the leaves are at level 0. For $0 \leq \ell \leq h$, the set nodes at level ℓ is denoted by $\mathrm{T}_h(\ell)$.

4 Randomized query complexity of 4-MAJ_h

In this section, we prove our main technical result, a lower bound on the randomized query complexity of 4-MAJ_h. We prove this by using distributional complexity, that is by using the inequality in Proposition 5. First, we define a "hard distribution" d_h for which we will show that $\Delta_\varepsilon^{d_h}(\text{4-MAJ}_h) \geq (1 - 2\varepsilon)(16/5)^h$, which implies our main result (Theorem 2).

4.1 The hard distribution

Intuitively, the distribution we use in our lower bound has to be one on which it is difficult to compute $\mathsf{4\text{-}MAJ}_h$. We start by defining a hard distribution for $\mathsf{4\text{-}MAJ}$ and extend it to $\mathsf{4\text{-}MAJ}_h$ in the natural way: by composing it with itself.

The *hard distribution* d on inputs of length 4 is defined from d^0 and d^1, the respective hard distributions for 0-inputs and 1-inputs of length 4, by setting $d(x) = \frac{1}{2}d^b(x)$ when $\mathsf{4\text{-}MAJ}(x) = b$. We define d^0 as

$$\begin{aligned} &d^0(1000) = \frac{2}{5}, \quad d^0(0011) = d^0(0101) = d^0(0110) = \frac{1}{6}, \\ &d^0(0001) = d^0(0010) = d^0(0100) = \frac{1}{30}, \ \text{ and } \ d^0(0000) = 0. \end{aligned} \tag{14}$$

The definition of d^1 is analogous, or can be defined by $d^1(x_1, x_2, x_3, x_4) = d^0(1 - x_1, 1 - x_2, 1 - x_3, 1 - x_4)$. Given that the function $\mathsf{4\text{-}MAJ}$ is symmetric in x_2, x_3, and x_4, there are only 4 equivalence classes of 0-inputs, to which we have assigned probability masses $2/5, 1/2, 1/10$, and 0, and then distributed the probabilities uniformly inside each class. The probabilities were chosen to make the recurrence relations in Lemma 17 and Lemma 18 work, while putting more weight on the intuitively difficult inputs. For example $x = 0000$ seems like an easy input since all inputs that are Hamming distance 1 from it are also 0-inputs, and thus reading any 3 bits of this input is sufficient to compute the function. In Lemma 18 we will give an equivalent characterisation of the hard distribution which is more directly related to the recurrence relations in the lemmas.

From this distribution we recursively define, for $h \geq 0$, the *hard distribution* d_h on inputs of length 4^h. In the base case, $d_0(0) = d_0(1) = \frac{1}{2}$. For $h > 0$, as for d, the distribution d_h is defined from d_h^0 and d_h^1, the respective hard distributions for 0-inputs and 1-inputs of length 4^h, by setting $d_h(x) = \frac{1}{2}d_h^b(x)$ when $\mathsf{4\text{-}MAJ}(x) = b$. Let $x = x^{(1)}x^{(2)}x^{(3)}x^{(4)}$ be a b-input, where $x^{(i)}$ is a b_i-input of length 4^{h-1}, for $i \in \{1, 2, 3, 4\}$. Then, $d_h^b(x) = d^b(b_1b_2b_3b_4)\cdot\Pi_{i=1}^4 d_{h-1}^{b_i}(x^{(i)})$. It is easily seen that according to d_h, for each node v in T_h, if the value of v is b, then the children of v have values distributed according to d^b. With the additional constraints that the root has uniform distribution over $\{0, 1\}$, this actually makes an alternative definition of d_h.

We will also require the notion of a minority path in our proof. For a given input, a minority path is a path from the root to a leaf in which each node has a value different from its parent's value. (Recall that the value of a node is the function $\mathsf{4\text{-}MAJ}$ evaluated on the values of its children.) For example, for the $\mathsf{4\text{-}MAJ}$ function, on input 1000 the unique minority path is the edge from the root to the first variable, whereas on input 1001 there are two minority paths from the root to the second and third variable. In general, since there may be multiple such paths, the minority path is defined to be a random variable over all root–leaf paths. Formally, for every input $x \in \{0, 1\}^{4^h}$, we define the *minority path* $M(x)$ as a random variable over all root–leaf paths in T_h as follows. First, the root is always in $M(x)$. Then, for any node v in $M(x)$, if there is a unique child w of v with value different from that of v, then $w \in M(x)$. Otherwise, there are exactly two children with different values, and we put each of them in $M(x)$ with probability $\frac{1}{2}$. Note that with this definition, if x is chosen from the hard distribution d_h, conditioned on the node v being in $M(x)$, the first child v_1 is in the minority path with probability $\frac{2}{5}$, and the child v_i is in the minority path with probability $\frac{1}{5}$, for $i \in \{2, 3, 4\}$.

4.2 Complexity of 4-MAJ$_h$ under the hard distribution

We can now lower bound the distributional complexity of 4-MAJ$_h$ under the hard distribution.

► **Theorem 16.** *For all $\varepsilon \geq 0$ and $h \geq 0$, we have $\Delta_\varepsilon^{d_h}(\text{4-MAJ}_h) \geq (1-2\varepsilon)(16/5)^h$.*

To show this, we need to define some quantities. For a deterministic decision tree algorithm A computing 4-MAJ$_h$, let $L_A(x)$ denote the set of variables queried by A on input x. Let B be a randomized decision tree algorithm that computes 4-MAJ$_h$ with error ε, and let b be its probability distribution over deterministic algorithms. For any two (not necessarily distinct) nodes of T_h, u and v, we define the function $E_B(v,u)$ as $E_B(v,u) = \mathbb{E}\big[|Z(v) \cap L_A(x)|\big|u \in M(x)\big]$, where the expectation is taken over b, d_h and the randomness in $M(x)$. In words, $E_B(v,u)$ is the expected number of queries below the node v over the randomness of B, the hard distribution and the randomness for the choice of the minority path, under the condition that u is in the minority path. For $0 \leq \ell \leq h$, we also define the functions $J_B^\varepsilon(h,\ell)$, $K_B^\varepsilon(h,\ell)$, $J^\varepsilon(h,\ell)$, and $K^\varepsilon(h,\ell)$ by

$$J_B^\varepsilon(h,\ell) = \sum_{v \in \mathrm{T}_h(\ell)} E_B(v,v), \tag{15}$$

$$K_B^\varepsilon(h,\ell) = \sum_{v \in \mathrm{T}_h(\ell)} \left(\frac{2}{5} \sum_{i=2}^{4} E_B(v_i, v_1) + \frac{1}{5} \sum_{j=2}^{4} \sum_{i \neq j} E_B(v_i, v_j) \right), \tag{16}$$

$$J^\varepsilon(h,\ell) = \min_{B \in \mathcal{R}_\varepsilon(\text{4-MAJ}_h)} J_B^\varepsilon(h,\ell) \quad \text{and} \quad K^\varepsilon(h,\ell) = \min_{B \in \mathcal{R}_\varepsilon(\text{4-MAJ}_h)} K_B^\varepsilon(h,\ell). \tag{17}$$

Observe that $J^\varepsilon(h,h) = \min_{B \in \mathcal{R}_\varepsilon(\text{4-MAJ}_h)} \mathbb{E}[C(B,x)] = \Delta_\varepsilon^{d_h}(\text{4-MAJ}_h)$.

The proof of Theorem 16 essentially follows from the following two lemmas.

► **Lemma 17.** *For all $0 < l \leq h$, we have that $J^\varepsilon(h,\ell) \geq K^\varepsilon(h,\ell) + \frac{1}{5} J^\varepsilon(h,\ell-1)$.*

Proof. This proof mainly involves expanding the quantity $E_B(v,v)$ in terms of $E_B(v_i,v_j)$, where v_1, v_2, v_3, and v_4 are the children of v. Since, for every node v, the set of leaves below v is the disjoint union of the sets of leaves below its children, for every B we have that

$$J_B^\varepsilon(h,\ell) = \sum_{v \in \mathrm{T}_h(\ell)} \sum_{i=1}^{4} E_B(v_i, v). \tag{18}$$

By conditioning on the minority child of v, we get that

$$J_B^\varepsilon(h,\ell) = \sum_{v \in \mathrm{T}_h(l)} \sum_{i=1}^{4} \sum_{j=1}^{4} E_B(v_i, v_j) \Pr[v_j \in M(x) | v \in M(x)] \ . \tag{19}$$

As mentioned before, if x is chosen according to the distribution d_h, if $v \in M(x)$, then $v_1 \in M(x)$ with probability $\frac{2}{5}$ and $v_i \in M(x)$ with probability $\frac{1}{5}$, for $i \in \{2,3,4\}$. Substituting these values we get

$$J_B^\varepsilon(h,\ell) = K_B^\varepsilon(h,\ell) + \frac{1}{5} J_B^\varepsilon(h,\ell-1) + \frac{1}{5} E_B(v_1,v_1). \tag{20}$$

Discarding the last term on the right hand side, which is always non-negative, and taking the minimum over B for all remaining terms gives the result. ◄

Having established this, we need to relate $K^\varepsilon(h,\ell)$ with $J^\varepsilon(h-1,\ell-1)$. Informally, given a randomized algorithm that performs well on $\mathsf{4\text{-}MAJ}_h$ at depth ℓ, we construct another algorithm that performs well on $\mathsf{4\text{-}MAJ}_{h-1}$ at depth $\ell-1$.

▶ **Lemma 18.** *For all $0 < \ell \le h$, we have that $K^\varepsilon(h,\ell) \ge 3J^\varepsilon(h-1,\ell-1)$.*

Proof. For any $B \in \mathcal{R}_\varepsilon(\mathsf{4\text{-}MAJ}_h)$, we will construct $B' \in \mathcal{R}_\varepsilon(\mathsf{4\text{-}MAJ}_{h-1})$ such that

$$\frac{1}{3}K^\varepsilon_B(h,\ell) = J^\varepsilon_{B'}(h-1,\ell-1). \tag{21}$$

Taking the minimum over all $B \in \mathcal{R}_\varepsilon(\mathsf{4\text{-}MAJ}_h)$ implies the statement.

We start by giving a high level description of our construction of B' from B. First B' will choose a random injective mapping from $\{x_1,\ldots,x_{4^{h-1}}\}$ to $\{x_1,\ldots,x_{4^h}\}$, identifying each variable of T_{h-1} with some variable of T_h. Then, it will choose a random restriction for the remaining variables of T_h. Note that these choices are not made uniformly. Let B_r denote the algorithm for 4^{h-1} variables defined by B after the identification and the restriction according to randomness r. B' then simply executes B_r. Our embedding of the smaller instance into the larger instance is done in a way that preserves the output.

We now describe the random identification and restriction in detail. First, observe that there is a natural correspondence between the nodes of $\mathrm{T}_{h-1}(\ell-1)$ and $\mathrm{T}_h(\ell)$ (since they are of the same size): we simply map the ith node of $\mathrm{T}_{h-1}(\ell-1)$ from the left to the ith node of $\mathrm{T}_h(\ell)$ from the left. For every node $u \in \mathrm{T}_{h-1}(\ell-1)$, let $v \in \mathrm{T}_h(\ell)$ be its corresponding node. The algorithm B' makes the following independent random choices. To generate the random identification, B' randomly chooses a child w of v, where $w = v_1$ with probability $\frac{1}{5}$, and $w = v_i$ with probability $\frac{4}{15}$, for $i \in \{2,3,4\}$. Then, the variables of $Z(u)$ and the variables of $Z(w)$ are identified naturally, again from left to right.

For generating the random restriction, B' first generates random values for the three siblings of w. If $w = v_1$, then it chooses for (v_2,v_3,v_4) one of the six strings from the set $\{001,010,100,110,101,011\}$ uniformly at random. If $w \in \{v_2,v_3,v_4\}$, it chooses for v_1, a uniformly random value from $\{0,1\}$, and for the remaining two siblings, it picks the opposite value. From this, the restriction is generated as follows: for each sibling w' of w with value $b \in \{0,1\}$, a random string of length $4^{\ell-1}$ is generated according to $d^b_{\ell-1}$, and the variables in $Z(w')$ receive the values of this string. This finishes the description of B'.

We now show that $B' \in \mathcal{R}_\varepsilon(\mathsf{4\text{-}MAJ}_{h-1})$. Because of the identification of the variables of $Z(u)$ and $Z(w)$, for every $x \in \{0,1\}^{4^{h-1}}$, the value of u coincides with the value of w. The random values chosen for the siblings of w are such that whatever value w gets, it is always a majority child of v. Therefore, for every input x, and for every randomness r, the value of u is the same as the value of v. This implies that for every x and every randomness r, the value of the roots of $\mathrm{T}_{h-1}(\ell-1)$ and $\mathrm{T}_h(\ell)$ are the same. Since B is an algorithm which computes $\mathsf{4\text{-}MAJ}_h$ with error at most ε, this means that B_r is an algorithm which computes $\mathsf{4\text{-}MAJ}_{h-1}$ with error at most ε, for every randomness r. From this, it follows that $B' \in \mathcal{R}_\varepsilon(\mathsf{4\text{-}MAJ}_{h-1})$.

Finally we prove the equality in (21). For this, the main observation (which can be checked by direct calculation) is that when w gets a random Boolean value, the distribution of values generated by B' on the children of v is exactly the hard distribution d. Therefore,

$E_{B'}(u,u) = E_B(w,v)$. Consequently, we have that

$$\begin{aligned} J^{\varepsilon}_{B'}(h-1,\ell-1) &= \sum_{v\in \mathrm{T}_h(\ell)} E_B(w,v) = \sum_{v\in \mathrm{T}_h(\ell)} \sum_{i=1}^{4} E_B(v_i,v)\Pr[w=v_i|v\in M(x)] \\ &= \sum_{v\in \mathrm{T}_h(\ell)} \sum_{i=1}^{4}\sum_{j=1}^{4} E_B(v_i,v_j)\Pr[w=v_i]\Pr[v_j\in M(x)|w=v_i, v\in M(x)] \\ &= \frac{1}{3}K^{\varepsilon}_B(h,\ell). \end{aligned} \tag{22}$$

The third equality holds since the choice of w is independent from the fact that v is in the minority path. For the last equality, we used that the conditional probabilities evaluate to the following values:

$$\begin{aligned} &\Pr[v_j\in M(x)|w=v_j, v\in M(x)] = 0, && \text{for } j\in\{1,2,3,4\};\\ &\Pr[v_j\in M(x)|w=v_1, v\in M(x)] = \frac{1}{3}, && \text{for } j\neq 1;\\ &\Pr[v_1\in M(x)|w=v_i, v\in M(x)] = \frac{1}{2}, && \text{for } i\neq 1;\\ &\Pr[v_j\in M(x)|w=v_i, v\in M(x)] = \frac{1}{4}, && \text{for } i,j\in\{2,3,4\} \text{ and } i\neq j. \end{aligned}$$ ◀

We can now return to proving Theorem 16.

Proof of Theorem 16. We claim that, for all $0\leq \ell\leq h$, we have that

$$J^{\varepsilon}(h,\ell) \geq (1-2\varepsilon)(16/5)^{\ell}. \tag{23}$$

The proof is done by induction on ℓ. For the base case $\ell=0$, let $B\in\mathcal{R}_\varepsilon(\text{4-MAJ}_h)$. Then, we have that

$$J^{\varepsilon}_B(h,0) = \sum_{v\in \mathrm{T}_h(0)} \Pr[B \text{ queries } v\big|v\in M(x)]. \tag{24}$$

Observe that any randomized decision tree algorithm computing a nonconstant function with error at most ε must make at least one query with probability at least $1-2\varepsilon$, since otherwise it would output 0 or 1 with probability greater than ε, and thus on some input would err too much. Let therefore A be a deterministic algorithm from the support of B which makes at least one query. Then

$$\sum_{v\in \mathrm{T}_h(0)} \Pr[A \text{ queries } v\big|v\in M(x)] \geq \sum_{v\in \mathrm{T}_h(0)} \Pr[A \text{ first query is } v\big|v\in M(x)] = 1, \tag{25}$$

since in the summation the term corresponding to the first query of A is 1, whereas all other terms are 0. Thus, $J^{\varepsilon}(h,0)\geq 1-2\varepsilon$ for all $h\geq 0$.

Now let $\ell>0$, and assume the statement holds for $\ell-1$. For $h\geq\ell$, using Lemma 17 and Lemma 18, we get that $J^{\varepsilon}(h,\ell)\geq 3J^{\varepsilon}(h-1,\ell-1)+\frac{1}{5}J^{\varepsilon}(h,\ell-1)$. Therefore, by the induction hypothesis, we have that

$$J^{\varepsilon}(h,\ell) \geq 3(1-2\varepsilon)\left(\frac{16}{5}\right)^{\ell-1} + \frac{1}{5}(1-2\varepsilon)\left(\frac{16}{5}\right)^{\ell-1} = (1-2\varepsilon)\left(\frac{16}{5}\right)^{\ell}. \tag{26}$$

The theorem follows when we set $h=\ell$ by noting that $J^{\varepsilon}(h,h)\leq \Delta^{d_h}_{\varepsilon}(\text{4-MAJ}_h)$. ◀

Combining Proposition 15 and Theorem 16 gives us our main result, an asymptotic separation between deterministic subcube partition complexity and randomized query complexity:

▶ **Theorem 2.** *There exists a function* $f = (f_h)$*, with* $f_h : \{0,1\}^{4^h} \to \{0,1\}$*, such that* $D^{\mathrm{sc}}(f) \leq 3^h$*, but* $D(f) = 4^h$*,* $R_0(f) \geq 3.2^h$*, and* $R(f) = \Omega(3.2^h)$*.*

We can also immediately deduce that the $\mathsf{4\text{-}MAJ}_h$ function positively answers both Question 1 and Question 3.

▶ **Corollary 19.** *We have that* $R_0^{\mathrm{sc}}(\mathsf{4\text{-}MAJ}_h) = o(R_0(\mathsf{4\text{-}MAJ}_h))$*.*

▶ **Corollary 20.** *For* $0 \leq \varepsilon \leq 1/3$*, we have that* $\mathrm{PPRT}_\varepsilon(\mathsf{4\text{-}MAJ}_h) = o(R_\varepsilon(\mathsf{4\text{-}MAJ}_h))$*.*

5 Discussion and open problems

Our main result is actually stronger than stated. In addition to the zero-error and ε-error randomized query complexities we defined, we can also define ε-error expected randomized complexity. In this model, we only charge for the expected number of queries made by the randomized algorithm, like in the zero-error case, but we also allow the algorithm to err. Formally, the *ε-error expected randomized query complexity of f* is $R_\varepsilon^{\exp}(f) = \min_{B \in \mathcal{R}_\varepsilon(f)} \max_x C(B,x))$. Observe that since this generalizes zero-error randomized query complexity, $R_0^{\exp}(f) = R_0(f)$, and it is immediate that, for all $\varepsilon \geq 0$, we have that $R_\varepsilon^{\exp}(f) \leq R_\varepsilon(f) \leq D(f)$.

Randomized query complexity is usually defined in the worst case [5], that is as $R_\varepsilon(f)$ instead of $R_\varepsilon^{\exp}(f)$. The main reason for not dealing with these measures separately is that worst case and expected randomized complexities are closely related. We have already observed that (obviously), in expectation, one can not make more queries than in the worst case. On the other hand, if for some constant $\eta > 0$ we let the randomized algorithm that achieves $R_\varepsilon^{\exp}(f)$ make $\frac{1}{2\eta}R_\varepsilon^{\exp}(f)$ queries, and give a random answer in case the computation is not finished, we get an algorithm of error $\varepsilon + \eta$ which never makes more than $\frac{1}{2\eta}R_\varepsilon^{\exp}(f)$ queries. Therefore, for all $\varepsilon \geq 0$ and $\eta > 0$, we have that $R_{\varepsilon+\eta}(f) \leq \frac{1}{2\eta}R_\varepsilon^{\exp}(f)$.

The result we show actually lower bounds $R_\varepsilon^{\exp}(f)$ as well. Thus, a stronger version of our result is the following: For all $\varepsilon \geq 0$, $R_\varepsilon^{\exp}(\mathsf{4\text{-}MAJ}_h) \geq (1-2\varepsilon)(3.2)^h$.

We end with some open problems. It would be interesting to exactly pin down the randomized query complexity of $\mathsf{4\text{-}MAJ}_h$. For example we know that $R_0(\mathsf{4\text{-}MAJ}_h) \geq 3.2^h$ and $R_0(\mathsf{4\text{-}MAJ}_h) \leq 3.25^h$. The best separation between subcube partition complexity and query complexity remains open, even in the deterministic case.

Finally it would be interesting to know if the partition bounds also lower bound expected randomized query complexity, and in particular whether the zero-error partition bound lower bounds zero-error randomized query complexity.

References

1 S. Aaronson. Quantum certificate complexity. *SIAM Journal on Computing*, 35(4):804–824, 2006.

2 S. Aaronson. Lower bounds for local search by quantum arguments. *Journal of Computer and System Sciences*, 74(3):313–332, 2008.

3 A. Ambainis, K. Balodis, A. Belovs, T. Lee, M. Santha, and J. Smotrovs. Separations in query complexity based on pointer functions. *arXiv preprint* arXiv:1506.04719, 2015.

4 Y. Brandman, A. Orlitsky, and J. Hennessy. A spectral lower bound technique for the size of decision trees and two-level AND/OR circuits. *IEEE Transactons on Computers*, 39(2):282–287, February 1990.

5 H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

6 S. Chakraborty, R. Kulkarni, S. V. Lokam, and N. Saurabh. Upper bounds on Fourier entropy. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:52, 2013.

7 E. Friedgut, J. Kahn, and A. Wigderson. Computing graph properties by randomized subcube partitions. In *Randomization and Approximation Techniques in Computer Science*, volume 2483 of *Lecture Notes in Computer Science*, pages 105–113. Springer Berlin Heidelberg, 2002.

8 M. Göös, T. Pitassi, and T. Watson. Deterministic communication vs. partition number. *To appear in the Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2015.

9 R. Jain and H. Klauck. The partition bound for classical communication complexity and query complexity. In *Proceedings of the 2010 IEEE 25th Annual Conference on Computational Complexity*, CCC'10, pages 247–258, 2010.

10 R. Jain, T. Lee, and N. Vishnoi. A quadratically tight partition bound for classical communication complexity and query complexity. *arXiv preprint* arXiv:1401.4512, 2014.

11 T. S. Jayram, R. Kumar, and D. Sivakumar. Two applications of information complexity. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing (STOC)*, STOC'03, pages 673–682, New York, NY, USA, 2003. ACM.

12 S. Jukna. *Boolean Function Complexity: Advances and Frontiers.* Algorithms and Combinatorics. Springer, 2012.

13 E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 2006.

14 I. Landau, A. Nachmias, Y. Peres, and S. Vanniasegaram. The lower bound for evaluating a recursive ternary majority function: an entropy-free proof. Undergraduate Research Reports, Department of Statistics, University of California, Berkeley, 2006.

15 S. Laplante and F. Magniez. Lower bounds for randomized and quantum query complexity using Kolmogorov arguments. *SIAM Journal on Computing*, 38(1):46–62, 2008.

16 A. Montanaro. A composition theorem for decision tree complexity. *Chicago Journal of Theoretical Computer Science*, 2014(6), July 2014.

17 N. Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991.

18 N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 15(4):557–565, 1995.

19 A. L. Rosenberg. On the time required to recognize properties of graphs: a problem. *SIGACT News*, 5(4):15–16, 1973.

20 M. Saks and A. Wigderson. Probabilistic boolean decision trees and the complexity of evaluating game trees. *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 29–38, 1986.

21 P. Savický. On determinism versus unambiguous nondeterminism for decision trees. *ECCC*, TR02-009, 2002.

22 R. Špalek and M. Szegedy. All quantum adversary methods are equivalent. *Theory of Computing*, 2(1):1–18, 2006.

23 A. Tal. Properties and applications of boolean function composition. In *Proc. of the 4th Conf. on Innovations in Theoretical Computer Science*, ITCS'13, pages 441–454, 2013.

24 A. Yao. Probabilistic computations: Toward a unified measure of complexity. *Proc. of the 18th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

Distance-based Species Tree Estimation: Information-Theoretic Trade-off between Number of Loci and Sequence Length under the Coalescent[*]

Elchanan Mossel[1] and Sebastien Roch[2]

1 **U. Penn and UC Berkeley, USA**
`mossel@wharton.upenn.edu`

2 **UW–Madison, USA**
`roch@math.wisc.edu`

Abstract

We consider the reconstruction of a phylogeny from multiple genes under the multispecies coalescent. We establish a connection with the sparse signal detection problem, where one seeks to distinguish between a distribution and a mixture of the distribution and a sparse signal. Using this connection, we derive an information-theoretic trade-off between the number of genes, m, needed for an accurate reconstruction and the sequence length, k, of the genes. Specifically, we show that to detect a branch of length f, one needs $m = \Theta(1/[f^2\sqrt{k}])$.

1998 ACM Subject Classification F.2.0 Analysis of Algorithms and Problem Complexity – General, J.3 Life and Medical Sciences

Keywords and phrases phylogenetic reconstruction, multispecies coalescent, sequence length requirement

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.931

1 Introduction

In the **sparse signal detection problem**, one is given m i.i.d. samples $X_1, \ldots, X_m$ and the goal is to distinguish between a distribution $\mathbb{P}_0^{(m)}$

$$H_0^{(m)} : X_i \sim \mathbb{P}_0^{(m)},$$

and the same distribution corrupted by a sparse signal $\mathbb{P}_1^{(m)}$

$$H_1^{(m)} : X_i \sim \mathbb{Q}^{(m)} := (1 - \sigma_m)\,\mathbb{P}_0^{(m)} + \sigma_m\,\mathbb{P}_1^{(m)}.$$

Typically one takes $\sigma_m = m^{-\beta}$, where $\beta \in (0,1)$. This problem arises in a number of applications [19, 27, 7, 30]. The Gaussian case in particular is well-studied [26, 20, 5]. For instance it is established in [26, 20] that, in the case $\mathbb{P}_0^{(m)} \sim N(0,1)$ and $\mathbb{P}_1^{(m)} \sim N(\lambda_m, 1)$

* S.R. is supported by NSF grants DMS-1007144, DMS-1149312 (CAREER), and an Alfred P. Sloan Research Fellowship. E.M. is supported by NSF grants DMS-1106999 and CCF 1320105 and DOD ONR grants N000141110140 and N00014-14-1-0823 and grant 328025 from the Simons Foundation. E.M. and S.R. thank the Simons Institute for the Theory of Computing at U.C. Berkeley where this work was done.

18th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'15) / 19th Int'l Workshop on Randomization and Computation (RANDOM'15).
Editors: Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim; pp. 931–942
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

with $\lambda_m = \sqrt{2r \log m}$, a test with vanishing error probability exists if and only if r exceeds an explicitly known *detection boundary* $r^*(\beta)$.

In this paper, we establish a connection between sparse signal detection and the reconstruction of phylogenies from multiple genes or loci under a population-genetic model known as the multispecies coalescent [43]. The latter problem is of great practical interest in computational evolutionary biology and is currently the subject of intense study. See e.g. [31, 17, 2, 42] for surveys. There is in particular a growing body of theoretical results [15, 16, 14, 38, 32, 1, 45, 10, 46, 47], although much remains to be understood. The problem is also closely related to another very active area of research, the reconstruction of demographic history in population genetics. See e.g. [41, 4, 29] for some recent theoretical results.

By taking advantage of the connection to sparse signal detection, we derive a detection boundary for the multilocus phylogeny estimation problem and use it to characterize the trade-off between the number of genes needed to accurately reconstruct a phylogeny and the quality of the signal that can be extracted from each separate gene. Our results apply to an important class of reconstruction methods known as distance-based methods. Before stating our results more formally, we begin with some background. See e.g. [48] for a general introduction to mathematical phylogenetics.

1.1 Species tree estimation

An evolutionary tree, or phylogeny, is a graphical representation of the evolutionary relationships between a group of species. Each leaf in the tree corresponds to a current species while internal vertices indicate past speciation events. In the classical phylogeny estimation problem, one sequences a *single* common gene (or other *locus* such as pseudogenes, introns, etc.) from a representative individual of each species of interest. One then seeks to reconstruct the phylogeny by comparing the genes across species. The basic principle is simple: because mutations accumulate over time during evolution, more distantly related species tend to have more differences between their genes.

Formally, phylogeny estimation boils down to *learning the structure of a latent tree graphical model from i.i.d. samples at the leaves.* Let $T = (V, E, L, r)$ be a rooted leaf-labelled binary tree, with n leaves denoted by $L = \{1, \ldots, n\}$ and a root denoted by r. In the Jukes-Cantor model [28], one of the simplest Markovian models of molecular evolution, we associate to each edge $e \in E$ a mutation probability

$$p_e = 1 - e^{-\nu_e t_e}, \tag{1}$$

where ν_e is the mutation rate and t_e is the time elapsed along the edge e. (The analytical form of (1) derives from a continuous-time Markov process of mutation along the edge. See e.g. [48].) The *Jukes-Cantor process* is defined as follows:

- Associate to the root a sequence $\mathbf{s}_r = (s_{r,1}, \ldots, s_{r,k}) \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^k$ of length k where each site $s_{r,i}$ is uniform in $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$.
- Let $U = \{r\}$.
- Repeat until $U = \emptyset$:
  - Pick a $u \in U$.
  - Let u^- be the parent of u.
  - Associate a sequence $\mathbf{s}_u \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^k$ to u as follows: $\mathbf{s}_u$ is obtained from $\mathbf{s}_{u^-}$ by mutating each site in $\mathbf{s}_{u^-}$ independently with probability $p_{(u^-,u)}$; when a mutation occurs at a site i, replace $s_{u,i}$ with a uniformly chosen state in $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$.
  - Remove u from U and add the children (if any) of u to U.

Let T^{-r} be the tree T where the root is suppressed, i.e., where the two edges adjacent to the root are combined into a single edge. We let $\mathcal{L}[T,(p_e)_e,k]$ be the distribution of the sequences at the *leaves* $\mathbf{s}_1,\ldots,\mathbf{s}_n$ under the Jukes-Cantor process. We define the **single-locus phylogeny estimation problem** as follows:

Given sequences at the leaves $(\mathbf{s}_1,\ldots,\mathbf{s}_n)\sim\mathcal{L}[T,(p_e)_e,k]$, recover the (leaf-labelled) unrooted tree T^{-r}.

(One may also be interested in estimating the p_es, but we focus on the tree. The root is in general not identifiable.) This problem has a long history in evolutionary biology. A large number of estimation techniques have been developed. See e.g. [24]. For a survey of the learning perspective on this problem, see e.g. [40]. On the theoretical side, much is known about the sequence length—or, in other words, the number of samples—required for a perfect reconstruction with high probability, including both information-theoretic lower bounds [49, 34, 35, 39] and matching algorithmic upper bounds [22, 11, 12, 44]. More general models of molecular evolution have also been considered in this context; see e.g. [23, 9, 37, 13, 3].

Nowadays, it is common for biologists to have access to *multiple* genes—or even full genomes. This abundance of data, which on the surface may seem like a blessing, in fact comes with significant new challenges. See e.g. [18, 42] for surveys. One important issue is that different genes may have incompatible evolutionary histories—represented by incongruent gene trees. In other words, if one were to solve the phylogeny estimation problem *separately* for several genes, one may in fact obtain *different* trees. Such incongruence can be explained in some cases by estimation error, but it can also result from deeper biological processes such as horizontal gene transfer, gene duplications and losses, and incomplete lineage sorting [33]. The latter phenomenon, which will be explained in Section 2, is the focus of this paper.

Accounting for this type of complication necessitates a *two-level hierarchical model* for the input data. Let $S=(V,E,L,r)$ be a rooted leaf-labelled binary *species tree*, i.e., a tree representing the actual succession of past divergences for a group of organisms. To each gene j shared by all species under consideration, we associate a *gene tree* $T_j=(V_j,E_j,L)$, mutation probabilities $(p^j_e)_{e\in E_j}$, and sequence length k_j. The triple $(T_j,(p^j_e)_{e\in E_j},k_j)$ is picked at random according to a given distribution $\mathcal{G}[S,(\nu_e,t_e)_{e\in E}]$ which depends on the *unknown* species tree, mutation parameters ν_e and inter-speciation times t_e. It is standard to assume that the gene trees are conditionally independent given the species tree. In the context of incomplete lineage sorting, the distribution of the gene trees, $\mathcal{G}$, is given by the so-called *multispecies coalescent*, which is a canonical model for combining speciation history and population genetic effects [43]. The detailed description of the model is deferred to Section 2, as it is not needed for a high-level overview of our results. For the readers not familiar with population genetics, it is useful to think of T_j as a noisy version of S (which, in particular, may result in T_j having a different (leaf-labelled) topology than S).

Our two-level model of sequence data is then as follows. Given a species tree S, parameters $(\nu_e,t_e)_{e\in E}$ and a number of genes m:

1. **[First level: gene trees]** Pick m independent gene trees and parameters
$$(T_j,(p^j_e)_{e\in E_j},k_j)\sim\mathcal{G}[S,(\nu_e,t_e)_{e\in E}],\qquad j=1,\ldots,m.$$
2. **[Second level: leaf sequences]** For each gene $j=1,\ldots,m$, generate sequence data at the leaves L according to the (single-locus) Jukes-Cantor process, as described above,
$$(\mathbf{s}^j_1,\ldots,\mathbf{s}^j_n)\sim\mathcal{L}[T_j,(p^j_e)_e,k_j],\qquad j=1,\ldots,m,$$
independently of the other genes.

We define the **multi-locus phylogeny estimation problem** as follows:

> Given sequences at the leaves $(\mathbf{s}_1^j, \ldots, \mathbf{s}_n^j)$, $j = 1, \ldots, m$, generated by the process above, recover the (leaf-labelled) unrooted species tree S^{-r}.

In the context of incomplete lineage sorting, this problem is the focus of very active research in statistical phylogenetics [31, 17, 2, 42]. In particular, there is a number of theoretical results, including [15, 16, 14, 38, 32, 1, 45, 10, 46, 47]. However, many of these results concern the statistical properties (identifiability, consistency, convergence rate) of species tree estimators *that (unrealistically) assume perfect knowledge of the T_js.* We only have a very incomplete picture of the properties of estimators that are based on sequence data, i.e., that do *not* require the knowledge of the T_js. (See below for an overview of prior results.)

Here we consider the data requirement of such estimators based on the sequences. To simplify, we assume that all genes have the same length, i.e., that $k_j = k$ for all $j = 1, \ldots, m$ for some k. (Because our goal is to derive a lower bound, such simplification is largely immaterial.) Our results apply to an important class of methods known as *distance-based methods*, which we briefly describe now. In the single-locus phylogeny estimation problem, a natural way to infer T^{-r} is to use the fraction of substitutions between each pair, i.e., letting $\|\cdot\|_1$ denote the ℓ_1-distance,

$$\theta(\mathbf{s}_a, \mathbf{s}_b) := \|\mathbf{s}_a - \mathbf{s}_b\|_1, \qquad \forall a, b \in [n]. \tag{2}$$

We refer to reconstruction methods relying solely on the $\theta(\mathbf{s}_a, \mathbf{s}_b)$s as distance-based methods. Assume for instance that $\nu_e = \nu$ for all e, i.e., the so-called molecular clock hypothesis. Then it is easily seen that single-linkage clustering (e.g., [25]) applied to the *distance matrix* $(\theta(\mathbf{s}_a, \mathbf{s}_b))_{a,b\in[n]}$ converges to T^{-r} as $k \to +\infty$. (In this special case, the root can be recovered as well.) In fact, T can be reconstructed perfectly as long as, for each a, b, $\frac{1}{k}\theta(\mathbf{s}_a, \mathbf{s}_b)$ is close enough to its expectation (e.g. [48])

$$\theta_{ab} := \frac{3}{4}(1 - e^{-d_{ab}}) \quad \text{with} \quad d_{ab} := \sum_{e \in P(a,b)} \nu_e t_e,$$

where $P(a, b)$ is the edge set on the unique path between a and b in T. Here "close enough" means $O(f)$ where $f := \min_e \nu_e t_e$. This observation can been extended to general ν_es. See e.g. [22] for explicit bounds on the sequence length required for perfect reconstruction with high probability.

Finally, to study distance-based methods in the *multi-locus* case, we restrict ourselves to the following **multi-locus distance estimation problem**:

> Given an accuracy $\varepsilon > 0$ and distance matrices $\theta(\mathbf{s}_a^j, \mathbf{s}_b^j)_{a,b\in[n]}$, $j = 1, \ldots, m$, estimate d_{ab} as defined above within ε for all a, b.

Observe that, once the d_{ab}s are estimated within sufficient accuracy, i.e., within $O(f)$, the species tree can be reconstructed using the techniques referred to in the single-locus case.

1.2 Our results

How is all this related to the sparse signal detection problem? Our main goal here is to provide a lower bound on the amount of data required for perfect reconstruction, in terms of m (the number of genes) and k (the sequence length). Consider the three possible (rooted, leaf-labelled) species trees with three leaves, as depicted in Figure 1, where we let the time to the most recent divergence be $1 - f$ (from today) and the time to the earlier divergence

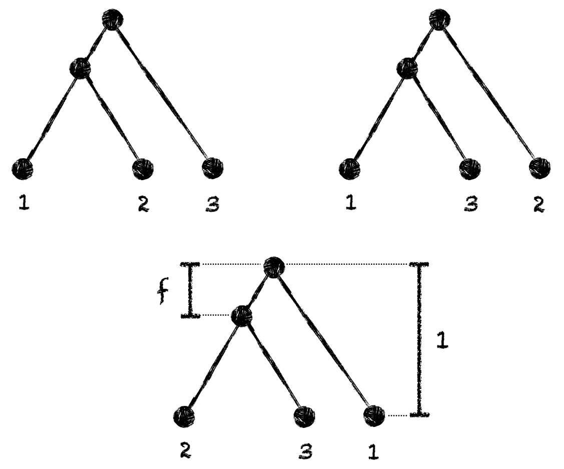


Figure 1 Three species trees.

be 1. In order for a distance-based method to distinguish between these three possibilities, i,e., to determine which pair is closest, we need to estimate the d_{ab}s within $O(f)$ accuracy. Put differently, within the multi-locus distance estimation problem, it suffices to establish a lower bound on the data required to distinguish between a two-leaf species tree S with $d_{12} = 2$ and a two-leaf species tree S^+ with $d_{12} = 2 - 2f$, where in both cases $\nu_e = 1$ for all e. We are interested in the limit $f \to 0$.

Let $\mathbb{P}_0$ and $\mathbb{Q}$ be the distributions of $\theta(\mathbf{s}_1^1, \mathbf{s}_2^1)$ for a single gene under S and S^+ respectively, where for ease of notation the dependence on k is implicit. For m genes, we denote the corresponding distributions by $\mathbb{P}_0^{\otimes m}$ and $\mathbb{Q}^{\otimes m}$. To connect the problem to sparse signal detection we observe below that, under the multispecies coalescent, $\mathbb{Q}$ is in fact a *mixture* of $\mathbb{P}_0$ and a sparse signal $\mathbb{P}_1$, i.e.,

$$\mathbb{Q} = (1 - \sigma_f)\,\mathbb{P}_0 + \sigma_f\,\mathbb{P}_1, \tag{3}$$

where $\sigma_f = O(f)$ as $f \to 0$.

When testing between $\mathbb{P}_0^{\otimes m}$ and $\mathbb{Q}^{\otimes m}$, the optimal sum of Type-I (false positive) and Type-II (false negative) errors is given by (e.g. [8])

$$\inf_A \{\mathbb{P}_0^{\otimes m}(A) + \mathbb{Q}^{\otimes m}(A^c)\} = 1 - \|\mathbb{P}_0^{\otimes m} - \mathbb{Q}^{\otimes m}\|_{\mathrm{TV}}, \tag{4}$$

where $\|\cdot\|_{\mathrm{TV}}$ denotes the total variation distance. Because $\sigma_f = O(f)$, for any k, in order to distinguish between $\mathbb{P}_0$ and $\mathbb{Q}$ one requires that, at the very least, $m = \Omega(f^{-1})$. Otherwise the probability of observing a sample originating from $\mathbb{P}_1$ under $\mathbb{Q}$ is bounded away from 1. In [38] it was shown that, provided that $k = \Omega(f^{-2}\log f^{-1})$, $m = \Omega(f^{-1})$ suffices. At the other end of the spectrum, when $k = O(1)$, a lower bound for the single-locus problem obtained by [49] implies that $m = \Omega(f^{-2})$ is needed. An algorithm achieving this bound under the multispecies coalescent was recently given in [10].

We settle the full spectrum between these two regimes. Our results apply when $k = f^{-2+2\kappa}$ and $m = f^{-1-\mu}$ where $0 < \kappa, \mu < 1$ as $f \to 0$.

▶ **Theorem 1** (Lower bound). *For any $\delta > 0$, there is a $c > 0$ such that*

$$\|\mathbb{P}_0^{\otimes m} - \mathbb{Q}^{\otimes m}\|_{\mathrm{TV}} \le \delta,$$

whenever

$$m \le c\frac{1}{f^2\sqrt{k}}.$$

Notice that the lower bound on m interpolates between the two extremal regimes discussed above. As k increases, a more accurate estimate of the gene trees can be obtained and one expects that the number of genes required for perfect reconstruction should indeed decrease. The form of that dependence is far from clear however. We in fact prove that our analysis is tight.

▶ **Theorem 2** (Matching upper bound)**.** *For any $\delta > 0$, there is a $c' > 0$ such that*

$$\|\mathbb{P}_0^{\otimes m} - \mathbb{Q}^{\otimes m}\|_{\mathrm{TV}} \ge 1 - \delta,$$

whenever

$$m \ge c'\frac{1}{f^2\sqrt{k}}.$$

Moreover, there is an efficient test to distinguish between $\mathbb{P}_0^{\otimes m}$ and $\mathbb{Q}^{\otimes m}$ in that case.

Our proof of the upper bound actually gives an efficient reconstruction algorithm under the molecular clock hypothesis. We expect that the insights obtained from proving Theorem 1 and 2 will lead to more accurate practical methods as well in the general case.

1.3 Proof sketch

Let Z be an exponential random variable with mean 1. We first show that, under $\mathbb{P}_0$ (respectively $\mathbb{Q}$), $\theta(\mathbf{s}_1^1, \mathbf{s}_2^1)$ is binomial with k trials and success probability $\frac{3}{4}\left(1 - e^{-2(\zeta+Z)}\right)$, where $\zeta = 1$ (respectively $\zeta = 1-f$). Equation (3) then follows from the memoryless property of the exponential, where σ_f is the probability that $Z \le f$.

A recent result of [6] gives a formula for the detection boundary of the sparse signal detection problem for general $\mathbb{P}_0$, $\mathbb{P}_1$. However, applying this formula here is non-trivial. Instead we bound directly the total variation distance between $\mathbb{P}_0^{\otimes m}$ and $\mathbb{Q}^{\otimes m}$. Similarly to the approach used in [6], we work instead with the Hellinger distance $H^2(\mathbb{P}_0^{\otimes m}, \mathbb{Q}^{\otimes m})$ which tensorizes as follows (see e.g. [8])

$$\frac{1}{2}H^2(\mathbb{P}_0^{\otimes m}, \mathbb{Q}^{\otimes m}) = 1 - \left(1 - \frac{1}{2}H^2(\mathbb{P}_0, \mathbb{Q})\right)^m, \tag{5}$$

and further satisfies

$$\|\mathbb{P}_0^{\otimes m} - \mathbb{Q}^{\otimes m}\|_{\mathrm{TV}}^2 \le H^2(\mathbb{P}_0^{\otimes m}, \mathbb{Q}^{\otimes m})\left[1 - \frac{1}{4}H^2(\mathbb{P}_0^{\otimes m}, \mathbb{Q}^{\otimes m})\right]. \tag{6}$$

All the work is in proving that, as $f \to 0$,

$$H^2(\mathbb{P}_0, \mathbb{Q}) = O\left(f^2\sqrt{k}\right).$$

More details are given in Section 3.1.

The proof of Theorem 2 on the other hand involves the construction of a statistical test that distinguishes between $\mathbb{P}_0^{\otimes m}$ and $\mathbb{Q}^{\otimes m}$. In the regime $k = O(1)$, an optimal test (up to constants) compares the means of the samples [10]. In the regime $k = \omega(f^{-2})$, an optimal test (up to constants) compares the minima of the samples [38]. A natural way to interpolate between these two tests is to consider an appropriate quantile. We show that the $1/\sqrt{k}$-quantile leads to the optimal choice.

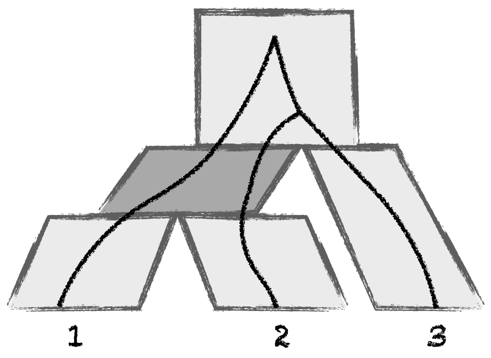


Figure 2 An incomplete lineage sorting event. Although 1 and 2 are more closely related in the species tree (fat tree), 2 and 3 are more closely related in the gene tree (thin tree). This incongruence is caused by the failure of the lineages originating from 1 and 2 to coalesce within the shaded branch.

1.4 Organization.

The gene tree generating model is defined in Section 2. The proofs of the main theorems are omitted from this extended abstract. These can be found at the Arxiv version of the paper [36].

2 Further definitions

In this section, we give more details on the model.

2.1 A little coalescent theory

As we mentioned in the previous section, our gene tree distribution model $\mathcal{G}[S, (\nu_e, t_e)_{e\in E}]$ is the *multispecies coalescent* [43]. We first explain the model in the two-species case. Let 1 and 2 be two species and consider a common gene j. One can trace *back in time* the lineages of gene j from an individual in 1 and from an individual in 2 until the first *common* ancestor. The latter event is called a *coalescence*. Here, because the two lineages originate from different species, coalescence occurs in an ancestral population. Let τ be the time of the divergence between 1 and 2 (back in time). Then, under the multispecies coalescent, the *coalescence time* is $\tau + Z$ where Z is an exponential random variable whose mean depends on the effective population size of the ancestral population. Here we scale time so that the mean is 1. (See e.g. [21] for an introduction to coalescent theory.)

We immediately get for the two-level model of sequence data:

▶ **Lemma 3** (Distance distribution). *Let S be a two-leaf species tree with $d_{12} = 2\tau$ and $\nu_e = 1$ for all e and let $\theta(\mathbf{s}_1^1, \mathbf{s}_2^1)$ be as in (2) for some k. Then the distibution of $\theta(\mathbf{s}_1^1, \mathbf{s}_2^1)$ is binomial with k trials and success probability $\frac{3}{4}\left(1 - e^{-2(\tau+Z)}\right)$.*

The memoryless property of the exponential gives:

▶ **Lemma 4** (Mixture). *Let S be a two-leaf species tree with $d_{12} = 2$ and let S^+ be a two-leaf species tree with $d_{12} = 2 - 2f$, where in both cases $\nu_e = 1$ for all e. Let $\mathbb{P}_0$ and $\mathbb{Q}$ be the distributions of $\theta(\mathbf{s}_1^1, \mathbf{s}_2^1)$ for a single gene under S and S^+ respectively. Then, there is $\mathbb{P}_1$*

such that,

$$\mathbb{Q} = (1 - \sigma_f)\, \mathbb{P}_0 + \sigma_f\, \mathbb{P}_1,$$

where $\sigma_f = O(f)$, as $f \to 0$. More specifically, $\mathbb{P}_1$ is obtained by conditioning $\mathbb{Q}$ on the event that Z is $\leq f$ and σ_f is the probability of that event.

More generally (this paragraph may be skipped as it will not play a role below), consider a species tree $S = (V, E; L, r)$ with n leaves. Each gene $j = 1, \ldots, m$ has a genealogical history represented by its gene tree T_j distributed according to the following process: looking backwards in time, on each branch of the species tree, the coalescence of any two lineages is exponentially distributed with rate 1, independently from all other pairs; whenever two branches merge in the species tree, we also merge the lineages of the corresponding populations, that is, the coalescence proceeds on the *union* of the lineages. More specifically, the probability density of a realization of this model for m independent genes is

$$\prod_{j=1}^{m} \prod_{e \in E} \exp\left(-\binom{O_j^e}{2}\left[\sigma_j^{e,O_j^e+1} - \sigma_j^{e,O_j^e}\right]\right) \prod_{\ell=1}^{I_j^e - O_j^e} \exp\left(-\binom{\ell}{2}\left[\sigma_j^{e,\ell} - \sigma_j^{e,\ell-1}\right]\right),$$

where, for gene j and branch e, I_j^e is the number of lineages entering e, O_j^e is the number of lineages exiting e, and $\sigma_j^{e,\ell}$ is the ℓ^{th} coalescence time in e; for convenience, we let $\sigma_j^{e,0}$ and $\sigma_j^{e,I_j^e - O_j^e+1}$ be respectively the divergence times of e and of its parent population. The resulting trees T_js may have topologies that differ from that of the species tree S. This may occur as a result of an incomplete lineage sorting event, i.e., the failure of two lineages to coalesce in a population. See Figure 2 for an illustration.

2.2 A more abstract setting

Before discussing the proofs, we re-set the problem in a more generic setting that will make the computations more transparent. We consider two distributions $\mathbb{P}_0$ and $\mathbb{P}_1$ for a random variable θ taking values in $\{0, \ldots, k\}$ for some k. We assume that the distribution of θ takes the form

$$\mathbb{P}_0[\theta = \ell] = \binom{k}{\ell} \mathbb{E}_0[X^\ell (1 - X)^{k-\ell}],$$

where $\mathbb{E}_0$ is the expectation operator corresponding to $\mathbb{P}_0$, and X is some random variable admitting a density over $[0, 1]$. The distribution is similarly defined under $\mathbb{P}_1$. We make the following assumptions, which are satisfied in the setting of the previous section:

A1. Under $\mathbb{P}_0$ and $\mathbb{P}_1$, X admits a density whose support is (p_0, p^0) under $\mathbb{P}_0$ and $(p_0 - \phi_f, p_0)$ under $\mathbb{P}_1$, where $0 < p_0 < p^0 < 1$ (independent of f) and $\phi_f = O(f)$. (In the setting of Lemma 4, $p_0 = \frac{3}{4}(1 - e^{-2})$, $p_0 - \phi_f = \frac{3}{4}(1 - e^{-(2-2f)})$, and $p^0 = 3/4$.)

A2. Under $\mathbb{P}_0$, the density of X (on its support) is in $[\rho, \rho^{-1}]$ for some $\rho > 0$ (independent of f) away from p^0, that is, below some $p_0 < \bar{p} < p^0$. (In the setting of Lemma 4, under $\mathbb{P}_0$ the density of X on (p_0, p^0) is $\frac{4e^{1/2}}{3}(1 - 4x/3)^{-3/4}$.)

As before, we let

$$\mathbb{Q} = (1 - \sigma_f)\, \mathbb{P}_0 + \sigma_f\, \mathbb{P}_1,$$

for some $\sigma_f = O(f)$.

3 Main steps of the proof

We give a few more details on the proofs.

3.1 Lower bound

We briefly sketch the main steps of the proof of the lower bound. In the abstract setting of Section 2, the Hellinger distance can be written as

$$\begin{aligned} H^2(\mathbb{P}_0,\mathbb{Q}) &= \sum_{j=0}^{k}\left[\sqrt{\mathbb{Q}[\theta=j]}-\sqrt{\mathbb{P}_0[\theta=j]}\right]^2 \\ &= \sum_{j=0}^{k}\left[\sqrt{1+\sigma_f\left(\frac{\mathbb{P}_1[\theta=j]}{\mathbb{P}_0[\theta=j]}-1\right)}-1\right]^2\mathbb{P}_0[\theta=j] \\ &= \sum_{j=0}^{k}\left[\sqrt{1+\sigma_f\left(\frac{\mathbb{E}_1[X^j(1-X)^{k-j}]}{\mathbb{E}_0[X^j(1-X)^{k-j}]}-1\right)}-1\right]^2\mathbb{P}_0[\theta=j] \qquad (7)\end{aligned}$$

We prove the following proposition, which implies Theorem 1.

▶ **Proposition 5.** *Assume that $k=f^{-2+2\kappa}$ where $0<\kappa<1$ and that Assumptions A1 and A2 hold. As $f\to 0$,*

$$H^2(\mathbb{P}_0,\mathbb{Q})=O\left(f^2\sqrt{k}\right).$$

From (7), in order to bound the Hellinger distance, we need to control the ratio $\frac{\mathbb{E}_1[X^j(1-X)^{k-j}]}{\mathbb{E}_0[X^j(1-X)^{k-j}]}$ and the probability $\mathbb{P}_0[\theta=j]$. Because the standard deviation of θ/k is $O(1/\sqrt{k})$ and $f\sqrt{k}=o(1)$, the dominant term in the sum (7) turns out to come from X being within $O(1/\sqrt{k})$ of p_0 under $\mathbb{E}_0$ (an event of probability $O(1/\sqrt{k})$) and θ/k being within $O(1/\sqrt{k})$ of p_0 as well (in which case the ratio $\frac{\mathbb{E}_1[X^j(1-X)^{k-j}]}{\mathbb{E}_0[X^j(1-X)^{k-j}]}$ is of order $O(1)$). The contribution of the dominant term is then indeed of order $O(f^2\sqrt{k})$. The full details are somewhat delicate and appear in the Arxiv version of the paper [36].

3.2 Upper bound

To prove the upper bound, we use (4) and construct an explicit test A. Let W be the number of genes such that $\theta/k\le p_0$. Let $w=\mathbb{P}_0[\theta/k\le p_0]$ and $w'=\mathbb{Q}[\theta/k\le p_0]$. Then $W\sim\mathrm{Bin}(m,w)$ under $\mathbb{P}_0$ and $W\sim\mathrm{Bin}(m,w')$ under $\mathbb{Q}$. Let

$$w^*=mw+\frac{m}{2}(w'-w)=mw'-\frac{m}{2}(w'-w),$$

and consider the event

$$A=\{W\ge w^*\}.$$

We show in the Arxiv version of the paper [36] that $\mathbb{P}_0^{\otimes m}[A]\le\frac{\delta}{2}$, and $\mathbb{Q}^{\otimes m}[A^c]\le\frac{\delta}{2}$ when c' is large enough.

Acknowledgments. We thank Gautam Dasarathy and Rob Nowak for helpful discussions.

References

1 Elizabeth S. Allman, James H. Degnan, and John A. Rhodes. Identifying the rooted species tree from the distribution of unrooted gene trees under the coalescent. *Journal of Mathematical Biology*, 62(6):833–862, 2011.

2 Christian N.K. Anderson, Liang Liu, Dennis Pearl, and Scott V. Edwards. Tangled trees: The challenge of inferring species trees from coalescent and noncoalescent genes. In Maria Anisimova, editor, *Evolutionary Genomics*, volume 856 of *Methods in Molecular Biology*, pages 3–28. Humana Press, 2012.

3 Alexandr Andoni, Constantinos Daskalakis, Avinatan Hassidim, and Sébastien Roch. Global alignment of molecular sequences via ancestral state reconstruction (extended abstract). In *ICS*, pages 358–369, 2010.

4 Anand Bhaskar and Yun S. Song. Descartes' rule of signs and the identifiability of population demographic models from genomic variation data. *Ann. Statist.*, 42(6):2469–2493, 2014.

5 T. Tony Cai, X. Jessie Jeng, and Jiashun Jin. Optimal detection of heterogeneous and heteroscedastic mixtures. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 73(5):629–662, 2011.

6 T.T. Cai and Yihong Wu. Optimal detection of sparse mixtures against a given null distribution. *Information Theory, IEEE Transactions on*, 60(4):2217–2232, April 2014.

7 L. Cayon, J. Jin, and A. Treaster. Higher criticism statistic: detecting and identifying non-gaussianity in the wmap first-year data. *Monthly Notices of the Royal Astronomical Society*, 362(3):826–832, 2005.

8 T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley Series in Telecommunications. John Wiley & Sons Inc., New York, 1991. A Wiley-Interscience Publication.

9 M. Cryan, L. A. Goldberg, and P. W. Goldberg. Evolutionary trees can be learned in polynomial time. *SIAM J. Comput.*, 31(2):375–397, 2002. short version, Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS 98), pages 436-445, 1998.

10 Gautam Dasarathy, Robert D. Nowak, and Sébastien Roch. New sample complexity bounds for phylogenetic inference from multiple loci. In *2014 IEEE International Symposium on Information Theory, Honolulu, HI, USA, June 29 – July 4, 2014*, pages 2037–2041, 2014.

11 Constantinos Daskalakis, Elchanan Mossel, and Sébastien Roch. Evolutionary trees and the ising model on the bethe lattice: a proof of steel's conjecture. *Probability Theory and Related Fields*, 149:149–189, 2011. 10.1007/s00440-009-0246-2.

12 Constantinos Daskalakis, Elchanan Mossel, and Sébastien Roch. Phylogenies without branch bounds: Contracting the short, pruning the deep. *SIAM J. Discrete Math.*, 25(2):872–893, 2011.

13 Constantinos Daskalakis and Sébastien Roch. Alignment-free phylogenetic reconstruction. In *RECOMB*, pages 123–137, 2010.

14 Michael DeGiorgio and James H Degnan. Fast and consistent estimation of species trees using supermatrix rooted triples. *Molecular Biology and Evolution*, 27(3):552–69, March 2010.

15 J. H. Degnan and N. A. Rosenberg. Discordance of species trees with their most likely gene trees. *PLoS Genetics*, 2(5), May 2006.

16 James H. Degnan, Michael DeGiorgio, David Bryant, and Noah A. Rosenberg. Properties of consensus methods for inferring species trees from gene trees. *Systematic Biology*, 58(1):35–54, 2009.

17 James H. Degnan and Noah A. Rosenberg. Gene tree discordance, phylogenetic inference and the multispecies coalescent. *Trends in Ecology and Evolution*, 24(6):332–340, 2009.

18 Frederic Delsuc, Henner Brinkmann, and Herve Philippe. Phylogenomics and the reconstruction of the tree of life. *Nat Rev Genet*, 6(5):361–375, 05 2005.

19 R. L. Dobrusin. A statistical problem arising in the theory of detection of signals in the presence of noise in a multi-channel system and leading to stable distribution laws. *Theory of Probability & Its Applications*, 3(2):161–173, 1958.

20 David Donoho and Jiashun Jin. Higher criticism for detecting sparse heterogeneous mixtures. *Ann. Statist.*, 32(3):962–994, 06 2004.

21 Richard Durrett. *Probability models for DNA sequence evolution.* Probability and its Applications (New York). Springer, New York, second edition, 2008.

22 P. L. Erdös, M. A. Steel;, L. A. Székely, and T. A. Warnow. A few logs suffice to build (almost) all trees (part 1). *Random Struct. Algor.*, 14(2):153–184, 1999.

23 P. L. Erdös, M. A. Steel;, L. A. Székely, and T. A. Warnow. A few logs suffice to build (almost) all trees (part 2). *Theor. Comput. Sci.*, 221:77–118, 1999.

24 J. Felsenstein. *Inferring Phylogenies.* Sinauer, New York, New York, 2004.

25 Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning.* Springer Series in Statistics. Springer, New York, second edition, 2009. Data mining, inference, and prediction.

26 Yu. I. Ingster. Some problems of hypothesis testing leading to infinitely divisible distributions. *Math. Methods Statist.*, 6(1):47–69, 1997.

27 X. Jessie Jeng, T. Tony Cai, and Hongzhe Li. Optimal sparse segment identification with application in copy number variation analysis. *J. Amer. Statist. Assoc.*, 105(491):1156–1166, 2010.

28 T. H. Jukes and C. Cantor. Mammalian protein metabolism. In H. N. Munro, editor, *Evolution of protein molecules*, pages 21–132. Academic Press, 1969.

29 Junhyong Kim, Elchanan Mossel, Miklos Z. Racz, and Nathan Ross. Can one hear the shape of a population history? *Theoretical Population Biology*, 100(0):26–38, 2015.

30 Martin Kulldorff, Richard Heffernan, Jessica Hartman, Renato Assuncao, and Farzad Mostashari. A space time permutation scan statistic for disease outbreak detection. *PLoS Med*, 2(3):e59, 02 2005.

31 Liang Liu, Lili Yu, Laura Kubatko, Dennis K. Pearl, and Scott V. Edwards. Coalescent methods for estimating phylogenetic trees. *Molecular Phylogenetics and Evolution*, 53(1):320–328, 2009.

32 Liang Liu, Lili Yu, and Dennis K. Pearl. Maximum tree: a consistent estimator of the species tree. *Journal of Mathematical Biology*, 60(1):95–106, 2010.

33 Wayne P. Maddison. Gene trees in species trees. *Systematic Biology*, 46(3):523–536, 1997.

34 E. Mossel. On the impossibility of reconstructing ancestral data and phylogenies. *J. Comput. Biol.*, 10(5):669–678, 2003.

35 E. Mossel. Phase transitions in phylogeny. *Trans. Amer. Math. Soc.*, 356(6):2379–2404, 2004.

36 E. Mossel and S. Roch. Distance-based species tree estimation: information-theoretic trade-off between number of loci and sequence length under the coalescent. ArXiv e-print 1504.05289, 2015.

37 Elchanan Mossel and Sébastien Roch. Learning nonsingular phylogenies and hidden Markov models. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 366–375, New York, 2005. ACM.

38 Elchanan Mossel and Sébastien Roch. Incomplete lineage sorting: Consistent phylogeny estimation from multiple loci. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 7(1):166–171, 2010.

39 Elchanan Mossel, Sébastien Roch, and Allan Sly. On the inference of large phylogenies with long branches: How long is too long? *Bulletin of Mathematical Biology*, 73:1627–1644, 2011. 10.1007/s11538-010-9584-6.

40 Raphaël Mourad, Christine Sinoquet, Nevin Lianwen Zhang, Tengfei Liu, and Philippe Leray. A survey on latent tree models and applications. *J. Artif. Intell. Res. (JAIR)*, 47:157–203, 2013.

41 Simon Myers, Charles Fefferman, and Nick Patterson. Can one learn history from the allelic spectrum? *Theoretical Population Biology*, 73(3):342–348, 2008.

42 Luay Nakhleh. Computational approaches to species phylogeny inference and gene tree reconciliation. *Trends in ecology & evolution*, 28(12):10.1016/j.tree.2013.09.004, 12 2013.

43 Bruce Rannala and Ziheng Yang. Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. *Genetics*, 164(4):1645–1656, 2003.

44 Sebastien Roch. Toward extracting all phylogenetic information from matrices of evolutionary distances. *Science*, 327(5971):1376–1379, 2010.

45 Sebastien Roch. An analytical comparison of multilocus methods under the multispecies coalescent: The three-taxon case. In *Pacific Symposium in Biocomputing 2013*, pages 297–306, 2013.

46 Sebastien Roch and Mike Steel. Likelihood-based tree reconstruction on a concatenation of alignments can be positively misleading. *Theoretical Population Biology*, 2015. To appear.

47 Sebastien Roch and Tandy Warnow. On the robustness to gene tree estimation error (or lack thereof) of coalescent-based species tree methods. *Systematic Biology*, 2015. In press.

48 C. Semple and M. Steel. *Phylogenetics*, volume 22 of *Mathematics and its Applications series*. Oxford University Press, 2003.

49 M. A. Steel and L. A. Székely. Inverting random functions. II. Explicit bounds for discrete maximum likelihood estimation, with applications. *SIAM J. Discrete Math.*, 15(4):562–575 (electronic), 2002.

Deterministically Factoring Sparse Polynomials into Multilinear Factors and Sums of Univariate Polynomials[*]

Ilya Volkovich

Department of EECS, CSE Division, University of Michigan, Ann Arbor, MI, USA
`ilyavol@umich.edu`

Abstract

We present the first efficient deterministic algorithm for factoring sparse polynomials that split into multilinear factors and sums of univariate polynomials. Our result makes partial progress towards the resolution of the classical question posed by von zur Gathen and Kaltofen in [6] to devise an efficient deterministic algorithm for factoring (general) sparse polynomials. We achieve our goal by introducing *essential factorization schemes* which can be thought of as a relaxation of the regular factorization notion.

1998 ACM Subject Classification F.2.1 Numerical Algorithms and Problems

Keywords and phrases Derandomization, Multivariate Polynomial Factorization, Sparse polynomials

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2015.943

1 Introduction

In this paper we study the problem of factorization of sparse polynomials.

1.1 Multivariate Polynomial Factorization

One of the fundamental problems in algebraic complexity is the problem of polynomial factorization: given a polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ over a field $\mathbb{F}$, find its irreducible factors. Other than being natural, the problem has many applications such as list decoding [29, 9] and derandomization [10]. A large amount of research has been devoted to finding efficient algorithms for this problem (see e.g. [5]) and numerous *randomized* algorithms were designed [6, 12, 15, 5, 13, 4]. However, the question of whether there exist *deterministic* algorithms for this problem remains an interesting open question (see [5, 17]).

1.2 Sparse Polynomials

Let $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a n-variate polynomial over the field $\mathbb{F}$. We denote by $\|P\|$ the *sparsity* of P. That is, the number of non-zero monomials in P. Suppose that the individual degree of each variable x_i is bounded by d, then the above number can reach $(d+1)^n$. Our case of interest is when $\|P\| \ll (d+1)^n$. Indeed, in various applications [30, 6, 1, 7, 25, 21] the desired regime is when $\|P\| = \mathrm{poly}(n, d)$. Such polynomials are refereed to as *sparse* polynomials. More generally, we call a polynomial P *s-sparse* if $\|P\| \leq s$. Otherwise, we say that P is *s-dense*.

[*] Research partially supported by NSF grant CCF-1161233.

Coming up with an efficient deterministic factorization algorithm for sparse polynomials (given as a list of monomials) is a classical open question posed by von zur Gathen and Kaltofen in [6]. An inherent difficulty in tackling the problem lies within the fact that a factor of a sparse polynomial need not be sparse. The following example demonstrates that a blow-up in the sparsity of a factor can be super-polynomial over any field. A similar example appears as Example 5.1 in [6].

▶ **Example 1.** Let $n \geq 1$. Consider the polynomial $f(\bar{x}) = \prod\limits_{i \in [n]} (x_i^n - 1)$ which can be written as a product of $g(\bar{x}) = \prod\limits_{i \in [n]} (1 + x_i + \ldots + x_i^{n-1})$ and $h(\bar{x}) = \prod\limits_{i \in [n]} (x_i - 1)$.
Observe that $\|f\| = \|h\| = 2^n$ while $\|g\| = n^n$, resulting in a quasi-polynomial blow-up.

Consequently, just writing down the irreducible factors as lists of monomials can take super-polynomial time [1]. In fact, the randomized algorithm of [6] assumes that the upper bound on the sparsity of the factors is known. In light of this difficulty, a simpler problem was posed in that same paper: Given $m + 1$ sparse polynomials $f_1, f_2, \ldots f_m, g$ test if $g = f_1 \cdot f_2 \cdot \ldots \cdot f_m$. This problem is referred to as "testing sparse factorization".

Over the last three decades this question has seen only a very partial progress. For the testing version of the problem, Saha et al. [20] presented an efficient deterministic algorithm for the special case when the sparse polynomials are sums of univariate polynomials. (P is a *sum of univariate polynomial* or a *sum of univariates*, for short, if it can be written as a sum of univariate polynomials. That is, $P = \sum_{i=1}^{n} T_i(x_i)$.) Shpilka & Volkovich [25] gave efficient deterministic factorization algorithms for multilinear sparse polynomials (see Lemma 8 for more details). In this work, we make another step towards the resolution of the problem. We consider the model of sparse polynomials that split into multilinear factors or "multilinearly-split" for short. Formally, we say that a polynomial P is *multilinearly-split* if it can be written as a product of multilinear polynomials.

Clearly, this model extends the one considered by Shpilka & Volkovich. Moreover, it can be seen as a multivariate version of algebraically closed fields in the following sense. The only irreducible univariate polynomials over algebraically closed fields are linear polynomials (i.e. $\alpha x + \beta$) since every univariate polynomial splits into linear factors. However, this is not the case in the multivariate setting. For example, the polynomial $P(x, y) = x^2 + y$ is irreducible over any field. In our model, the above phenomenon does not occur as every polynomial splits into multilinear factors. In addition, our model evades the aforementioned inherent difficulty since a multilinear factor of a sparse polynomial is itself a sparse polynomial (see Lemma 23 for more details). Below is our main result:

▶ **Theorem 2** (Main). *There exists a deterministic algorithm that given an s-sparse multilinearly-split polynomial $F \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree d outputs its irreducible multilinear factors. The running time of the algorithm is* $\text{poly}(n, d, s, p, \ell)$ *when $\mathbb{F} = \mathbb{F}_{p^\ell}$ and* $\text{poly}(n, d, s, b)$ *when $\mathbb{F} = \mathbb{Q}$ and b is the bit complexity of the coefficients in F.*

Our next results extend the ones in [20].

▶ **Theorem 3.** *Let $F \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial of degree d that splits into sums of univariates. There exists a deterministic algorithm that given an oracle access to F outputs its irreducible factors. The running time of the algorithm is* $\text{poly}(n, d, p, \ell)$ *when $\mathbb{F} = \mathbb{F}_{p^\ell}$ and* $\text{poly}(n, d, b)$ *when $\mathbb{F} = \mathbb{Q}$ and b is the bit complexity of the coefficients in F.*

[1] Although g is not irreducible, this issue can be resolved using standard techniques. For example, by considering the product $f + yh = (g + y)h$ for a new variable y.

Combining the result with the efficient algorithm of [20] for testing divisibility by a sum of univariates, we obtain the following:

▶ **Theorem 4.** *There exists a deterministic algorithm that given an s-sparse polynomial to $F \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree d outputs its irreducible factors that are sums of univariates (if any). The running time of the algorithm is* $\text{poly}(n, d, s, p, \ell)$ *when* $\mathbb{F} = \mathbb{F}_{p^\ell}$ *and* $\text{poly}(n, d, s, b)$ *when* $\mathbb{F} = \mathbb{Q}$ *and b is the bit complexity of the coefficients in F.*

Note that the running times of our algorithms are essentially optimal, as we are invoking the state-of-the-art deterministic factoring algorithm for a constant number of variables. We note that even for the univariate case the best known deterministic factoring algorithm has a polynomial dependence on the characteristic p. While in the randomized setting, the dependence is polynomial in $\log p$. A lot of effort was invested in trying to derandomize the factorization algorithm when the characteristic of $\mathbb{F}$ is large (see e.g. [23, 5, 3, 17]).

1.3 Techniques

Let $\mathcal{C}$ be a class of polynomials and let $\text{fact}(\mathcal{C})$ denote the class of factors of $P \in \mathcal{C}$. Suppose we want to map n-variate polynomials from $\mathcal{C}$ to (other) polynomials with, potentially, a smaller number of variables in a way that two distinct (composite) polynomials $P, Q \in \mathcal{C}$ remain distinct under the map. In particular, this implies that each irreducible polynomial in $\text{fact}(\mathcal{C})$ must be mapped into a non-constant polynomial. Moreover, a pair of non-similar, irreducible polynomials must be mapped into a pair of non-similar polynomials. And, ideally, an irreducible polynomial should remain irreducible. Those goals are achieved in [6, 11, 12, 15] and other works by considering a projection to a random low-dimensional space (i.e a line or a plane). The purpose of the last two requirements is to ensure that factors from different images could not be combined together. In other words, there is only one way to interpret a product of images under the map. As preserving irreducibility deterministically is still an open question, we introduce a relaxation of the original requirements with the hope that it would be easier to fulfill. We call it an *essential factorization scheme*.

Consider a polynomial map $\{\mathrm{H}\}_n = \mathbb{F}^{t(n)} \to \mathbb{F}^n$ with $t \ll n$ such that for every irreducible $P \in \text{fact}(\mathcal{C})$ the composition $P(\mathrm{H})$ might be reducible, yet results in a polynomial that contains an irreducible "essential" factor $\Psi_{\mathrm{H}}(P)$ which describes P uniquely. In addition, $\Psi_{\mathrm{H}}(P)$ cannot be a factor of any $Q(\mathrm{H})$ when $P \neq Q \in \text{fact}(\mathcal{C})$. Given that property of H, for any $F \in \mathcal{C}$ the polynomial $F(\mathrm{H})$ describes F uniquely. Consequently, for any $F, R \in \mathcal{C}$ we get that $F \equiv R \iff F(\mathrm{H}) = R(\mathrm{H})$. We formalize this notion in Definition 17.

Observe that this reasoning can be extended to handle products of polynomials for $\mathcal{C}$. That is, $\prod_{i=1}^{k} F_1$. Consequently, if we could establish an essential factorization scheme for sparse polynomials, we would solve the sparse factorization testing problem.

Unfortunately, we are not there yet for the entire set of sparse polynomials. In this paper, we make a step towards this goal by establishing an essential factorization scheme for multilinear sparse polynomials. In fact, our scheme has an additional property: given a factor, we can efficiently decide whether or not it is an essential factor of some polynomial P. Moreover, we can efficiently compute P from its essential factor $\Psi_{\mathrm{H}}(P)$. Consequently, in order to compute the irreducible factors of multilinearly-split polynomial F, we first compute the irreducible factor of $F(\mathrm{H})$ and then recover the "original" factors of F. We note that since $F(\mathrm{H})$ is t-variate polynomial with $t \ll n$, we can carry out the factorization phase deterministically by a brute-force derandomization of the best randomized algorithm while still being efficient. Formally, see Lemma 24

We show that our essential factorization scheme works for some other classes of multilinear polynomials as well. Our construction can be seen as another link in the line of works [14, 25, 19] that connect polynomial factoring and polynomial identity testing.

1.4 Organization

We start by some basic definitions and notation in Section 2. In Section 3, we formally introduce essential factorization schemes and demonstrate their properties. In that same section, we also construct such a scheme for classes of multilinear polynomials. In Section 4, we present our factoring algorithm for sparse multilinearly-split polynomials, thus proving our main theorem. Finally, in Section 5 we present an algorithm for factoring polynomials that split into sums of univariate (Theorem 3) and for finding sums of univariate factors of sparse polynomials (Theorem 4). We conclude the paper with some remarks in Section 6.

2 Preliminaries

Let $\mathbb{F}$ denote a field, finite or otherwise, and let $\overline{\mathbb{F}}$ denote its algebraic closure. We assume that elements of $\mathbb{F}$ are represented in binary using some standard encoding.

2.1 Polynomials

A polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ *depends* on a variable x_i if there are two inputs $\bar{\alpha}, \bar{\beta} \in \overline{\mathbb{F}}^n$ differing only in the i^{th} coordinate for which $P(\bar{\alpha}) \neq P(\bar{\beta})$. We denote by $\text{var}(P)$ the set of variables that P depends on. We say that P and Q are *similar* and denote by it $P \sim Q$ if $P = \alpha Q$ for some $\alpha \neq 0 \in \mathbb{F}$.

For a polynomial $P(x_1, \ldots, x_n)$, a variable x_i and a field element α, we denote with $P|_{x_i=\alpha}$ the polynomial resulting from substituting α to x_i. Similarly given a subset $I \subseteq [n]$ and an assignment $\bar{a} \in \mathbb{F}^n$, we define $P|_{x_I=\bar{a}_I}$ to be the polynomial resulting from substituting a_i to x_i for every $i \in I$.

▶ **Definition 5** (Leading Coefficient). Let $x_i \in \text{var}(f)$. We can write: $P = \sum_{j=0}^{d} P_j \cdot x_i^j$ such that $\forall j, x_i \notin \text{var}(P_j)$ and $P_d \not\equiv 0$. The *leading coefficient* of P w.r.t to x_i is defined as $\text{lc}_{x_i}(P) \triangleq P_d$. The *individual degree* of x_i in P is defined as $\deg_{x_i}(P) \triangleq d$.

It easy to see that for every $P, Q \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $i \in [n]$ we have that: $\text{lc}_{x_i}(P \cdot Q) = \text{lc}_{x_i}(P) \cdot \text{lc}_{x_i}(Q)$.

▶ **Definition 6.** For $P, Q \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $\ell \in [n]$ let $D_\ell(P, Q)$ be the polynomial defined as follows:

$$D_\ell(P,Q)(\bar{x}) \triangleq \left| \begin{pmatrix} P & P|_{x_\ell=0} \\ Q & Q|_{x_\ell=0} \end{pmatrix} \right| (\bar{x}) = (P \cdot Q|_{x_\ell=0} - P|_{x_\ell=0} \cdot Q)(\bar{x}).$$

Note that D_ℓ is a bilinear transformation. The following lemma from [21] gives a useful property of D_ℓ that is easy to verify.

▶ **Lemma 7** ([21]). *Let $P, Q \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be irreducible multilinear polynomials and let $\ell \in \text{var}(P)$. Then $D_\ell(Q, P) \equiv 0$ iff $P \mid Q$.*

The next corollary from [25] shows that a multilinear sparse polynomial can be factored efficiently. Moreover, all its factors are sparse.

▶ **Lemma 8** (Corollary from [25]). *Given a multilinear polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, there is a* $\text{poly}(n, \|P\|)$ *time deterministic algorithm that outputs the* irreducible *factors, $h_1, \ldots, h_k$ of P. Furthermore, $\|h_1\| \cdot \|h_2\| \cdot \ldots \cdot \|h_k\| = \|P\|$.*

2.2 Commutator

The Commutator was originally defined in [25] where it was used to devised efficient factorization algorithms for classes of multilinear polynomials. Later, it was also used in reconstruction algorithms [8, 26] for arithmetic formulae. The following definitions are taken from [25].

▶ **Definition 9.** Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial. We say that f is (x_i, x_j)-*decomposable* if f can be written as $f = g \cdot h$ for polynomials g and h such that $i \in \mathrm{var}(g) \setminus \mathrm{var}(h)$ and $j \in \mathrm{var}(h) \setminus \mathrm{var}(g)$.

▶ **Definition 10** (Commutator)**.** Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multilinear polynomial and let $i, j \in [n]$. We define the *commutator* between x_i and x_j as $\Delta_{ij} f \overset{\Delta}{=} f|_{x_i=1,x_j=1} \cdot f|_{x_i=0,x_j=0} - f|_{x_i=1,x_j=0} \cdot f|_{x_i=0,x_j=1}$.

The crucial property of the commutator is given by the lemma below.

▶ **Lemma 11** ([25])**.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multilinear polynomial and let $i, j \in \mathrm{var}(f)$. Then f is (x_i, x_j)-decomposable if and only if $\Delta_{ij} f \equiv 0$.*

The following observation connects between Δ_{ij} and D_i.

▶ **Observation 12.** $\Delta_{ij}(P) = D_i \left(P|_{x_j=1}, P|_{x_j=0}\right)$.

2.3 Maps and Generators for Classes of Polynomials

In this section, we formally define the notion of generators and hitting sets for polynomials as well as describe a few of their useful properties. For a further discussion see [24, 28, 16].

A map $\mathcal{G} = (\mathcal{G}^1, \ldots, \mathcal{G}^n) : \mathbb{F}^q \to \mathbb{F}^n$ is a *generator* for the polynomial class $\mathcal{C}$ if for every non-zero n-variate polynomial $P \in \mathcal{C}$, it holds that $P(\mathcal{G}) \not\equiv 0$. The image of the map $\mathcal{G}$ is denoted as $\mathrm{Im}\,(\mathcal{G}) = \mathcal{G}(\overline{\mathbb{F}}^q)$. Ideally, q should be very small compared to n. A set $\mathcal{H} \subseteq \mathbb{F}^n$ is a *hitting set* for a polynomial class $\mathcal{C}$, if for every non-zero polynomial $P \in \mathcal{C}$, there exists $\bar{a} \in \mathcal{H}$, such that $P(\bar{a}) \neq 0$. A generator can also be viewed as a map containing a hitting set for $\mathcal{C}$ in its image. That is, for every non-zero $P \in \mathcal{C}$, there exists $\bar{a} \in \mathrm{Im}\,(\mathcal{G})$ such that $P(\bar{a}) \neq 0$. In identity testing, generators and hitting sets play the same role. Given a generator one can easily construct a hitting set by evaluating the generator on a large enough set of points. Conversely in [24], an efficient method of constructing a generator from a hitting set was given.

▶ **Lemma 13** ([24])**.** *Let $|\mathbb{F}| > n$. Given a set $\mathcal{H} \subseteq \mathbb{F}^n$, there is an algorithm that runs in time* $\mathrm{poly}(|\mathcal{H}|, n, \log|\mathbb{F}|)$ *and constructs a map $\mathcal{G}(\bar{w}) : \mathbb{F}^t \to \mathbb{F}^n$ such that $\mathcal{G}(\bar{0}) = \bar{0}$, $\mathcal{H} \subseteq \mathrm{Im}\,(\mathcal{G})$ with $t \overset{\Delta}{=} \lceil \log_n |\mathcal{H}| \rceil$ and the individual degrees of $\mathcal{G}^i$ are bounded by $n-1$. Moreover, for each $\bar{a} \in \mathcal{H}$, its preimage, $\bar{\beta} \in \mathbb{F}^q$ s.t. $\bar{a} = \mathcal{G}(\beta)$, can be computed in time* $\mathrm{poly}(|\mathcal{H}|, n)$.

2.4 SV-Generator

The $G_{n,k}$ generator was defined in [24] where it was shown that for certain values of k the map $G_{n,k}$ is generator for read-once polynomials. In [16] this was generalized to multilinear read-k polynomials[2]. We will use the $G_{n,k}$ in our construction.

[2] A read-k polynomial is a polynomial computable by a formula where each variable appears at most k times.

▶ **Definition 14** (SV-Generator [24]). Let $a_1, \dots, a_n$ denote n distinct elements from a field $\mathbb{F}$ and for $i \in [n]$ let $L_i(x) \doteq \prod_{j \neq i} \frac{x-a_j}{a_i-a_j}$ denote the corresponding Lagrange interpolant. For every $k \in \mathbb{N}$, define

$$G_{n,k}(y_1, \dots, y_k, z_1, \dots, z_k) \doteq \left(\sum_{j=1}^{k} L_1(y_j)z_j, \sum_{j=1}^{k} L_2(y_j)z_j, \dots, \sum_{j=1}^{k} L_n(y_j)z_j \right).$$

Let $(G_{n,k})_i$ denote the i^{th} component of $G_{n,k}$; we refer to a_i as the *Lagrange constant* associated with this i^{th} component.

For intuition, it is helpful to view the action of $G_{n,1}(y_1, z_1)$ on a random element of $\mathbb{F}^2$ as selecting a random variable (via the value of y_1) and a random value for that variable (via the value of z_1). This is not completely accurate because for values outside the Lagrange constants the generator does not uniquely select a component. Since the SV-generator is a polynomial map, it is natural to define the sum of two copies of the generator by their component-wise sum and to furthermore view $G_{n,k}$ as the sum of k independent choices of variables and values. For this reason, we take the convention that for two generators $\mathcal{G}_1$ and $\mathcal{G}_2$ with the same output length that $\mathcal{G}_1 + \mathcal{G}_2$ is the generator obtained by adding a sample from $\mathcal{G}_1$ to an independent sample from $\mathcal{G}_2$, and where the seed variables are implicitly relabelled so as to be disjoint. With this convention in mind, the SV-generator has a number of useful properties that follow immediately from its definition.

▶ **Proposition 15** ([24, 16]). *Let k, k' be positive integers.*

1. $G_{n,k}(\bar{y}, \bar{0}) \equiv \bar{0}$.
2. $G_{n,k}(y_1, \dots, y_k, z_1, \dots, z_k)|_{y_k=a_i} = G_{n,k-1}(y_1, \dots, y_{k-1}, z_1, \dots, z_{k-1}) + z_k \cdot \bar{e}_i$, *where $\bar{e}_i$ is the 0-1-vector with a single 1 in position i and a_i the i^{th} Lagrange constant.*
3. $G_{n,k}(y_1, \dots, y_k, z_1, \dots, z_k) + G_{n,k'}(y_{k+1}, \dots, y_{k+k'}, z_{k+1}, \dots, z_{k+k'})$
$= G_{n,k+k'}(y_1, \dots, y_{k+k'}, z_1, \dots, z_{k+k'})$

The first item states that zero is in the image of the SV-generator. The second item shows how to make a single output component (and no others) depend on a particular z_j. The final item shows that sums of independent copies of the SV-generator are equivalent to a single copy of the SV-generator with the appropriate parameter k. The above properties give rise to the following operator.

▶ **Definition 16** (Reviving). Let $P \in \mathbb{F}[x_1, x_2, \dots, x_n]$ be a polynomial and $\mathcal{G}_n(\bar{w}) \triangleq (\mathcal{G}_n^1(\bar{w}), \dots, \mathcal{G}_n^n(\bar{w}))$ be a polynomial map. Let $k \leq n$. Consider $\mathrm{H}_n \triangleq \mathcal{G}_n(\bar{w}) + G_{n,k}(\bar{y}, \bar{z})$. Let $i \in [n]$. We call the operation:

$$\mathcal{R}_{\{x_i\}}(P(\mathrm{H}_n)) \triangleq P(\mathrm{H}_n)|_{y_k=a_i, z_k=x_i-\mathcal{G}^i, z_1=z_2=\dots=z_{k-1}=0}$$

a *revival of* x_i. By Proposition 15, the result of such a revival equals:

$$P(\mathcal{G}_n^1(\bar{w}), \dots, \mathcal{G}_n^{i-1}(\bar{w}), x_i, \mathcal{G}_n^{i+1}(\bar{w}), \dots, \mathcal{G}_n^n(\bar{w}))$$

which can be seen as lifting the polynomial map substituted into x_i. Similarly, we can extend the definition $\mathcal{R}_I$ to any subset $I \subseteq [n]$ of size $|I| \leq k$ as reviving all the variables in I.

3 Essential Factorization Scheme

In this section, we formally define the notions of essential factors and essential factorization schemes. For a class of polynomials $\mathcal{C}$ we denote by $\mathrm{fact}(\mathcal{C}) = \{P \mid \exists g \neq 0, P \cdot g \in \mathcal{C}\}$ the class of factors of $\mathcal{C}$.

▶ **Definition 17** (Essential Factorization Scheme)**.** Let $\mathcal{C}$ be a class of polynomials over a field $\mathbb{F}$. We say that a polynomial map $\{\mathrm{H}\}_n = \mathbb{F}^{t(n)} \to \mathbb{F}^n$ is an *essential factorization scheme* for $\mathcal{C}$ if there exists (another) map $\Psi_{\mathrm{H}} : \mathbb{F}[x_1, x_2, \ldots, x_n] \to \mathbb{F}[w_1, w_2, \ldots, w_{t(n)}]$ such that given two irreducible polynomials $P, Q \in \mathrm{fact}(\mathcal{C})$:

1. $\Psi_{\mathrm{H}}(P)$ is a non-constant, irreducible factor of $P(\mathrm{H})$, called the *essential* factor of P.
2. $\Psi_{\mathrm{H}}(P) \mid Q(\mathrm{H})$ iff $P \sim Q$.

Let us discuss the definition. Let $P \in \mathrm{fact}(\mathcal{C})$ be an irreducible factor of some $F \in \mathcal{C}$. The intuition is that the essential factor of P, i.e. $\Psi_{\mathrm{H}}(P)$, should contain "all the essential" information about P and, in addition it cannot appear as a factor of any other polynomial $Q \in \mathrm{fact}(\mathcal{C})$. In particular, it follows from the definition that $\Psi_{\mathrm{H}}(P) \sim \Psi_{\mathrm{H}}(Q)$ iff $P \sim Q$. The next lemma shows that our definition is sufficient in achieving our original goal. That is, ensuring that two distinct (composite) polynomials $F, R \in \mathcal{C}$ remain distinct under the mapping.

▶ **Lemma 18** (Uniqueness from essential factorization)**.** *Let $F, R \in \mathcal{C}$ be two polynomials (not necessarily irreducible) and let* H *be as in the above definition. Then $F \equiv R$ iff $F(\mathrm{H}) = R(\mathrm{H})$.*

Proof. The proof is by induction on $\deg(F) + \deg(R)$. The base case is when both F and R are constant polynomials and the claim clearly follows. Now suppose wlog that F is non-constant. By the properties of H, $F(\mathrm{H})$ is also non-constant and since $F(\mathrm{H}) = R(\mathrm{H})$, R must be non-constant as well. Let $F = P_1 \cdot \ldots \cdot P_k$ and $R = Q_1 \cdot \ldots \cdot Q_\ell$ denote F's and R's factorization into irreducible factors (possibly with repetitions), respectfully where $P_i, Q_j \in \mathrm{fact}(\mathcal{C})$. We have that:

$$P_1(\mathrm{H}) \cdot \ldots \cdot P_k(\mathrm{H}) = F(\mathrm{H}) = R(\mathrm{H}) = Q_1(\mathrm{H}) \cdot \ldots \cdot Q_\ell(\mathrm{H}).$$

By definition, $\Psi_{\mathrm{H}}(P_1) \mid P_1(\mathrm{H})$. Therefore, by uniqueness of factorization, there exist $j \in [\ell]$ such that $\Psi_{\mathrm{H}}(P_1) \mid Q_j(\mathrm{H})$, since $\Psi_{\mathrm{H}}(P_1)$ is an irreducible polynomial. By Property 2, there exists $\alpha \neq 0 \in \mathbb{F}$ such that $P_1 = \alpha Q_j$. Now, consider: $F' \triangleq \frac{F}{P_1}$ and $R' \triangleq \frac{R}{\alpha Q_j}$. It follows that $F'(\mathrm{H}) = R'(\mathrm{H})$ when $\deg(F') + \deg(R') < \deg(F) + \deg(R)$. By the induction hypothesis $F' \equiv R'$ and thus $F = P' \cdot P_1 \equiv Q' \cdot \alpha Q_j = R$. ◀

3.1 Essential Factorization Schemes for Multilinear Polynomials

In this section, we show how to construct essential factorization schemes for classes of multilinear polynomials that admit efficient identity testing algorithms. In fact, if we want to apply our results for a class $\mathcal{C}$, we require algorithms for a somewhat larger class.

Let $\mathcal{C}$ be a class of multilinear polynomials over the field $\mathbb{F}$. From Lemma 8, it follows that $\mathrm{fact}(\mathcal{C}) = \mathcal{C}$. $\mathcal{G}_n(\bar{w}) \triangleq \left(\mathcal{G}_n^1(\bar{w}), \ldots, \mathcal{G}_n^n(\bar{w})\right)$ be a generator for polynomials of the form $D_i(P, Q)$ where $P, Q \in \mathcal{C}$ are irreducible, n-variate polynomials and $i \in [n]$. We show that the map $\mathrm{H}_n \triangleq \mathcal{G}_n(\bar{w}) + G_{n,2}(y_1, y_2, z_1, z_2)$ is an essential factorization scheme for $\mathcal{C}$. As was mentioned earlier, this construction demonstrates another connection between polynomial factorization and polynomial identity testing. We begin by specifying Ψ_{H}. To that end, we require the following definition:

▶ **Definition 19** (Variable-Essential Factor)**.** Let $P(\mathrm{H}_n) = f_1 \cdot f_2 \cdots f_m$ be the unique factorization of $P(\mathrm{H}_n)$ into irreducible factors. We define the *variable-essential factor* of P, as f_e such that for each $x_i \in \mathrm{var}(P)$, we have that $x_i \in \mathrm{var}(\mathcal{R}_{\{x_i\}}(f_e))$. To avoid ambiguity, we take the monic[3] f_e.

[3] The coefficient of the largest monomial according to the lexicographic order in 1.

Given the above, we define $\Psi_{\mathrm{H}}(P)$ to be the variable-essential factor of P. Observe that applying $\mathcal{R}_{\{x_i\}}$ on $P(\mathrm{H}_n)$ results in applying $\mathcal{R}_{\{x_i\}}$ on each factor f_ℓ. Therefore, since P is a multilinear polynomial there can be at most one factor f_e that depends on x_i, when revived. Consequently, there can be at most one factor f_e with the required property. However, this still does not guarantee an existence of such a variable-essential factor. We will now show that in our case such a factor always exists.

▶ **Lemma 20.** *Let $P \in \mathrm{fact}(\mathcal{C}) = \mathcal{C}$ be an irreducible polynomial. Then $\Psi_{\mathrm{H}}(P)$ is well-defined.*

Proof. As previously, let $P(\mathrm{H}_n) = f_1 \cdot f_2 \cdots f_m$ be the unique factorization of $P(\mathrm{H}_n)$ into irreducible factors. First, we claim that for each $x_i \in \mathrm{var}(P)$ there exists $k_i \in [m]$ such that $x_i \in \mathrm{var}(\mathcal{R}_{\{x_i\}}(f_{k_i}))$. By definition $\mathcal{R}_{\{x_i\}}(P(\mathrm{H}_n)) = P_i(\mathcal{G}_n)\cdot x_i + P_0(\mathcal{G}_n)$ when $P = P_i x_i + P_0$. Since $P_i = D_i(P, 1)$ we get that the map $\mathcal{G}_n$ hits P_i which implies that $P(\mathrm{H}_n)$ depends on x_i, and the claim follows. To finish the proof, we need to show that $k_i = k_j$ for all $x_j, x_i \in \mathrm{var}(P)$.

Assume for a contradiction and wlog that $k_1 = 1, k_2 = 2$. As P is an irreducible polynomial, $\Delta_{12}(P) \not\equiv 0$ by Lemma 11. By Observation 12, the map $\mathcal{G}_n$ hits $\Delta_{12}(P)$. In other words, there exists $\bar{\beta} \in \mathrm{Im}\,(\mathcal{G}_n)$ such that $\Delta_{12}(P)(\bar{\beta}) \neq 0$ and $P(x_1, x_2, \beta_3, \ldots, \beta_n)$ depends of x_i and x_j. Let $\bar{\gamma} \in \mathcal{G}^{-1}(\bar{\beta})$. Consider

$$\tilde{P} \triangleq \mathcal{R}_{\{x_1,x_2\}}(P(\mathrm{H}_n))|_{\bar{w}=\bar{\gamma}} = P(x_1, x_2, \mathcal{G}_n^3(\bar{\gamma}), \ldots, \mathcal{G}_n^n(\bar{\gamma})) = P(x_1, x_2, \beta_3, \ldots, \beta_n).$$

By the choice of $\bar{\beta}$, the LHS depends on both x_i and x_j. On the other hand,

$$P(x_1, x_2, \beta_3, \ldots, \beta_n) = f_1(x_1, x_2, \beta_3, \ldots, \beta_n) \cdot f_2(x_1, x_2, \beta_3, \ldots, \beta_n) \cdots f_m(x_1, x_2, \beta_3, \ldots, \beta_n)$$

so $x_i \in \mathrm{var}(f_i)$ for $i = 1, 2$ and by Lemma 11 $\Delta_{12}(P)(\bar{\beta}) = 0$, thus reaching a contradiction. ◀

As was established, Ψ_{H} is well-defined and satisfies Property 1 of Definition 17. Note that given a list of purported factors it is easy to identify the variable-essential ones by reviving one variable at a time and testing dependence. Since $P(\mathrm{H})$ is a $t(n)$-variate polynomial of polynomial degree and typically $t(n) \ll n$, testing dependence can be carried out efficiently by a computing the monomial expansion of $P(\mathrm{H})$. In particular, in light of this uniqueness it must be the case that $\Psi_{\mathrm{H}}(P) \mid Q(\mathrm{H}) \implies \Psi_{\mathrm{H}}(P) \sim \Psi_{\mathrm{H}}(Q)$. Therefore, in order to show that Ψ_{H} satisfies Property 2 it is sufficient to show $\Psi_{\mathrm{H}}(P) \sim \Psi_{\mathrm{H}}(Q) \implies P \sim Q$. The intuition is that $\Psi_{\mathrm{H}}(P)$ should contain all the information about P since $\Psi_{\mathrm{H}}(P)$ encapsulates in itself the information on each single variable of P.

▶ **Lemma 21.** *Let $P, Q \in \mathcal{C}$ be two irreducible polynomials. Then $\Psi_{\mathrm{H}}(P) \sim \Psi_{\mathrm{H}}(Q)$ iff $P \sim Q$.*

Proof. The first direction is trivial. For the other direction note that since $\Psi_{\mathrm{H}}(P)$ and $\Psi_{\mathrm{H}}(Q)$ are both normalized we actually have that $f \triangleq \Psi_{\mathrm{H}}(P) = \Psi_{\mathrm{H}}(Q)$. In other words, $P(\mathrm{H}_n) = f \cdot P'$ and $Q(\mathrm{H}_n) = f \cdot Q'$. For $x_i \in \mathrm{var}(P)$, we can write: $P = P_i x_i + P_0$, $Q = Q_i x_i + Q_0$. Consider the revival of x_i in both $P(\mathrm{H}_n)$ and $Q(\mathrm{H}_n)$. By the definition of the variable-essential factors, $x_i \in \mathrm{var}(\mathcal{R}_{\{x_i\}}(f))$. Therefore:

$$\mathcal{R}_{\{x_i\}}(P(\mathrm{H}_n)) = (P_i(\mathcal{G}_n) \cdot x_i + P_0(\mathcal{G}_n)) = (\hat{f}_i x_i + \hat{f}_0) \cdot \mathcal{R}_{\{x_i\}}(P')$$
$$\mathcal{R}_{\{x_i\}}(Q(\mathrm{H}_n)) = (Q_i(\mathcal{G}_n) \cdot x_i + Q_0(\mathcal{G}_n)) = (\hat{f}_i x_i + \hat{f}_0) \cdot \mathcal{R}_{\{x_i\}}(Q')$$

where $\mathcal{R}_{\{x_i\}}(f) = \hat{f}_i x_i + \hat{f}_0$. By setting $x_i = 0$ we obtain:

$$P_i(\mathcal{G}_n) = \hat{f}_i \cdot \mathcal{R}_{\{x_i\}}(P')\ ,\ P_0(\mathcal{G}_n) = \hat{f}_0 \cdot \mathcal{R}_{\{x_i\}}(P')$$
$$Q_i(\mathcal{G}_n) = \hat{f}_i \cdot \mathcal{R}_{\{x_i\}}(Q')\ ,\ Q_0(\mathcal{G}_n) = \hat{f}_0 \cdot \mathcal{R}_{\{x_i\}}(Q').$$

And hence: $D_i(P,Q)(\mathcal{G}_n) = P_i(\mathcal{G}_n) \cdot Q_0(\mathcal{G}_n) - Q_i(\mathcal{G}_n) \cdot P_0(\mathcal{G}_n) \equiv 0$. Since $\mathcal{G}_n$ hits $D_i(P,Q)$ we know that $D_i(P,Q) \equiv 0$ to begin with. As P, Q are both irreducible, $P \sim Q$ by Lemma 7. ◀

The following theorem summarizes this section.

▶ **Theorem 22.** *Let $\mathcal{C}$ be a class of multilinear polynomials over the field $\mathbb{F}$ and let $\mathcal{G}_n(\bar{w})$ be a generator for the polynomials of the form $D_i(P,Q)$ where $P, Q \in \mathcal{C}$ are irreducible, n-variate polynomials and $i \in [n]$. Then $\mathrm{H}_n \triangleq \mathcal{G}_n(\bar{w}) + G_{n,2}(y_1, y_2, z_1, z_2)$ is an essential factorization scheme for $\mathcal{C}$. And in particular, for all $F, R \in \mathcal{C}$: $F \equiv R \iff F(\mathrm{H}_n) = R(\mathrm{H}_n)$.*

4 Factoring Sparse Multilinearly-Split Polynomials

In this section we prove our main result - Theorem 2. First, we give the outline of the proof. We say that a set $\mathcal{H}$ is an *interpolating set* for a class $\mathcal{C}$ if for every $P \in \mathcal{C}$ the evaluations $P|_{\mathcal{H}}$ determine P uniquely. In particular, an interpolating set can serve as a hitting set since $P \equiv 0 \iff P|_{\mathcal{H}} \equiv 0$.

Let $\mathcal{H}$ be the interpolating set for sparse polynomials given by Lemma 26. Our plan is to evaluate each essential factor separately on $\mathcal{H}$ and then apply the reconstruction algorithm of Lemma 26 to recover the original factors. However, there are couple of obstacles that stand in our way. First of all, how do we get access to every essential factor separately? To overcome this obstacle, we use $\mathcal{H}$ in conjunction with Theorem 22. Observe that $\mathcal{H}$ hits polynomials of the form $D_i(P,Q)$ where P and Q are sparse. Therefore, it satisfies the conditions of Theorem 22 (invoking Lemma 13). We then invoke Lemma 24 to factor our polynomial. As the new number of variables is small, this step can be carried out efficiently. This leads us to a second obstacle: we only obtain evaluations of the essential factors rather than the original factors.

Although by definition the essential factors contain "enough" information, this information might still be insufficient for the reconstruction algorithm since in order to reconstruct a sparse polynomial P the algorithm requires the values of P on $\mathcal{H}$ while we only have the values of a factor of P at hand. For the second obstacle, we make our reconstruction algorithm more "resilient" to information loss by extedning it to handle rational functions (Lemma 25).

We now move to the formal proof. To this end, we require the following results. The first result states that a multilinear factor of sparse polynomial is itself a sparse polynomial. Example 1 demonstrates that this is not the case for general sparse polynomials.

▶ **Lemma 23** ([8]). *Let $0 \not\equiv P, Q \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials such that P is multilinear. Then $P \mid Q \implies \|P\| \leq \|Q\|$.*

The next result which is implicit in many factorization algorithms, exhibits an efficient factorization algorithm for certain regime of parameters. In particular, for polynomials with constantly-many variables and a polynomial degree. We note that this the state-of-the-art algorithm for this regime of parameters.

▶ **Lemma 24** (Implicit [5, 12]). *There exists a deterministic algorithm that given a t-variate, degree d polynomial P over $\mathbb{F}$ outputs its irreducible factors. The running time of the algorithm is $(d,p,\ell)^{\mathcal{O}(t)}$ when $\mathbb{F} = \mathbb{F}_{p^\ell}$ and $(d,b)^{\mathcal{O}(t)}$ when $\mathbb{F} = \mathbb{Q}$ and b is the bit complexity of the coefficients in P.*

The following result converts a reconstruction algorithm for sparse polynomials into a reconstruction algorithm for sparse rational functions, introducing only a polynomial overhead.

▶ **Lemma 25** ([2]). *Let A be a deterministic algorithm that can reconstruct a s-sparse polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree d in time $T(n, s, d, |\mathcal{H}|)$ given the evaluations $P|_{\mathcal{H}}$. Let $R, Q \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be two coprime s-sparse polynomials of degree d and $\bar{\sigma} \in \mathbb{F}^n$ such that $Q(\bar{\sigma}) \neq 0$. Finally, let $V \subseteq \mathbb{F}$ be a subset of size $2d$. Then there exists a deterministic algorithm B that given the evaluations $(R/Q)|_{V \cdot \mathcal{H} + \bar{\sigma}}$ [4] outputs R', Q' such that $R' = cR$ and $Q' = cQ$ for some $c \neq 0 \in \mathbb{F}$ in time $\mathrm{poly}(|\mathcal{H}|, n, d, T(n, s, d, |\mathcal{H}|))$ and uses the algorithm A as an oracle. If $Q(\bar{\sigma}) = 0$ the algorithm fails.*

We conclude the list with an efficient reconstruction algorithm for sparse polynomials.

▶ **Lemma 26** ([18]). *Let $n, s, d > 1$. There exists a deterministic algorithm that in time $\mathrm{poly}(n, s, d)$ outputs an interpolating set $\mathcal{H}$ such that given the evaluations $P|_{\mathcal{H}}$ of a s-sparse polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree d in time $\mathrm{poly}(n, s, d)$ the algorithm can reconstruct P.*

We are ready to proceed with the proof of our main theorem (Theorem 2). Our algorithm combines the above results. The description of the algorithm is given in Algorithm 1.

```
Input: $s$-sparse, multilinearly-split polynomial $F \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree $d$
Output: A list $P_1, \ldots, P_k$ of the irreducible factors of $F$. That is, $F = P_1 \cdot \ldots \cdot P_k$.

1  Choose a subset $\{1\} \in V \subseteq \mathbb{F}$ of size $2d$ ;
2  Invoke the algorithm in Lemma 26 with $n, 2s^2, d = 2n$ to obtain an interpolating set $\mathcal{H}$;
3  Apply Lemma 13 on $\mathcal{H}' = V \cdot \mathcal{H}$ to obtain the map $\mathcal{G}$ /* note that $\mathcal{H} \subseteq \mathcal{H}' \subseteq \mathrm{Im}(\mathcal{G})$ */
4  Set $\mathrm{H}_n \overset{\Delta}{=} \mathcal{G}(\bar{u}) + \mathcal{G}(\bar{w}) + G_{n,2}(\bar{y}, \bar{z})$ ;
5  Use Lemma 24 to Factor $F(\mathrm{H}_n)$. Let $S$ be the set of the irreducible factors ;
6  Initialize $E \leftarrow \emptyset$ /* The set of all the essential factors */
7  foreach $f \in S, i \in [n]$ do
8      if $x_i \in \mathrm{var}(\mathcal{R}_{\{x_i\}}(f))$ /* Check by looking at the monomial expansion */
9      then
10         $E \leftarrow E \cup \{(\mathcal{R}_{\{x_i\}}(f), i)\}$ /* Move to the next $f \in S$ */
   /* Reconstruct the original factors */
11 foreach $(\hat{f}, i) \in E, \bar{\beta} \in \mathcal{G}^{-1}(\mathcal{H})$ do
12     Set: $\hat{f}_0(\bar{u}, \bar{w}) \overset{\Delta}{=} \hat{f}|_{x_i=0}, \hat{f}_i(\bar{u}, \bar{w}) \overset{\Delta}{=} \hat{f}|_{x_i=1} - \hat{f}_0$ ;
13     Apply Lemma 25 jointly with the reconstruction algorithm from Lemma 26 on
       $\hat{f}_0(\bar{u}, \bar{\beta}) / \hat{f}_i(\bar{u}, \bar{\beta})$ to obtain $R', Q'$. ;
14     On a success, output $P = Q' \cdot x_i + R'$ ;
```
Algorithm 1. Factoring algorithm for sparse multilinearly-split polynomials.

Proof of Theorem 2. We analyze Algorithm 1. For the running time we get $(n, s, d, p, \ell)^{\mathcal{O}(t)}$ when $\mathbb{F} = \mathbb{F}_{p^\ell}$ and $(n, s, d, b)^{\mathcal{O}(t)}$ when $\mathbb{F} = \mathbb{Q}$. By Lemmas 13 and 26 $t = \mathcal{O}(\log_n |\mathcal{H}|) = \mathcal{O}(\log_n (nsd))$. Therefore, if all the parameters are $\mathrm{poly}(n)$ we get the claimed running time.

[4] $V \cdot \mathcal{H} + \bar{\sigma} \overset{\Delta}{=} \{\alpha \cdot \bar{a} + \bar{\sigma} \mid \alpha \in V, \bar{a} \in \mathcal{H}\}$

We now move to the correctness. Let $F = P_1 \cdot P_2 \cdot \ldots \cdot P_m$ be F's factorization into irreducible factors (possibly with repetitions). By Lemma 23, each P_j above is s-sparse. First, observe that $\mathcal{H}$ (and consequently $\mathcal{H}'$) is a hitting set for $D_i(P, Q)$ where $P, Q \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ are s-sparse multilinear polynomials. By Lemma 13, the map $\mathcal{G}(\bar{u})$ hits those polynomials. As $\mathcal{G}(\bar{0}) = \bar{0}$, the same holds true for $\mathcal{G}(\bar{u}) + \mathcal{G}(\bar{w})$ as well. By Theorem 22, H_n is an essential factorization scheme for s-sparse multilinear polynomials. Therefore, by the properties of Definition 17 each $\Psi_{\mathrm{H}}(P_j)$ is a non-constant factor of $F(\mathrm{H})$ and $\Psi_{\mathrm{H}}(P_j) \sim \Psi_{\mathrm{H}}(P_k)$ iff $P_j \sim P_k$. Therefore, we can access all $\Psi_{\mathrm{H}}(P_j)$-s by factoring $F(\mathrm{H})$ and reviving one variable at a time to distinguish essential factors from the non-essential ones. Now, let $f = \Psi_{\mathrm{H}}(P_j)$ and $P_j(\mathrm{H}_n) = f \cdot P'_j$. By repeating the reasoning in the proof of Lemma 21 we get:

$$[P_j]_i(\mathcal{G}(\bar{u}) + \mathcal{G}(\bar{w})) \cdot x_i + [P_j]_0(\mathcal{G}(\bar{u}) + \mathcal{G}(\bar{w})) = \mathcal{R}_{\{x_i\}}(P_j(\mathrm{H}_n)) =$$
$$\hat{f}(\bar{u}, \bar{w}) \cdot \mathcal{R}_{\{x_i\}}(P'_j) = (\hat{f}_i(\bar{u}, \bar{w})x_i + \hat{f}_0(\bar{u}, \bar{w})) \cdot \mathcal{R}_{\{x_i\}}(P'_j)$$

and hence

$$[P_j]_i(\mathcal{G}(\bar{u}) + \mathcal{G}(\bar{w})) = \hat{f}_i(\bar{u}, \bar{w}) \cdot \mathcal{R}_{\{x_i\}}(P'_j)$$
$$[P_j]_0(\mathcal{G}(\bar{u}) + \mathcal{G}(\bar{w})) = \hat{f}_0(\bar{u}, \bar{w}) \cdot \mathcal{R}_{\{x_i\}}(P'_j).$$

when $P_j = [P_j]_i \cdot x_i + [P_j]_0$. Since $[P_j]_i$ is a non-zero s-sparse polynomial, there exists $\bar{\sigma} \in \mathcal{H}$ such that $[P_j]_i(\sigma) \neq 0$. By Lemma 13 we can efficiently iterate over $\mathcal{H}$ to find $\bar{\beta} \in \mathcal{G}^{-1}(\sigma)$. Finally observe that

$$\hat{f}_0(\bar{u}, \bar{\beta}) / \hat{f}_i(\bar{u}, \bar{\beta}) = [P_j]_0(\mathcal{G}(\bar{u}) + \bar{\sigma}) \;/\; [P_j]_i(\mathcal{G}(\bar{u}) + \bar{\sigma}).$$

Therefore given access to $\hat{f}_0(\bar{u}, \bar{\beta}) / \hat{f}_i(\bar{u}, \bar{\beta})$ the algorithm can query the polynomial $[P_j]_0 / [P_j]_i$ on every point of the forms $V \cdot \mathcal{H} + \sigma$ as required by Lemma 25. Consequently, we can apply Lemma 25 jointly with the reconstruction algorithm from Lemma 26 to obtain $R' = c[P_j]_0, Q' = c[P_j]_i$ resulting in $P = Q' \cdot x_i + R' = c[P_j]_i \cdot x_i + c[P_j]_0 = cP_j$ and we are done. ◀

5 Factoring Products of Sums of Univariates

In this section we prove Theorems 3 and 4. As was mentioned earlier, P is a sum of univariates if it is of the form $P = \sum_{i=1}^{n} T_i(x_i)$ Models related to these polynomial were previously studied in the literature [22, 20, 27]. We begin with a simple observation.

▶ **Observation 27.** *Let $I \subseteq [n]$ be a set of size $|I| \leq k \leq n$. Then*

$$\mathcal{R}_I \left(P(G_{n,k}) \right) = \sum_{i \in I} T_i(x_i) + \sum_{j \notin I} T_j(0).$$

Next, we require two results from [20].

▶ **Lemma 28** ([20]). *There exists a deterministic algorithm that given an s-sparse polynomial to $F \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and a sum of univariatess $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ both of degree d and $e \geq 0$ checks if $P^e \mid F$. The running time of the algorithm is $\mathrm{poly}(n, d, e, s, \log p, \ell)$ when $\mathbb{F} = \mathbb{F}_{p^\ell}$ and $\mathrm{poly}(n, d, e, s, b)$ when $\mathbb{F} = \mathbb{Q}$ and b is the bit complexity of the coefficients in F.*

▶ **Lemma 29** ([20]). *Let $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial that is a sum of univariates with $|\mathrm{var}(P)| \geq 3$. Then either P is irreducible, or P is a p-th power of some polynomial where $p = \mathrm{char}(\mathbb{F})$.*

▶ **Corollary 30.** *Let $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be an irreducible polynomial that is a sum of univariates and let $x_j, x_k \in \mathrm{var}(P)$ (not necessarily distinct). Then there exists a set $\{x_j, x_k\} \subseteq I \subseteq [n]$ of size $|I| \leq 3$ such that $P|_{\bar{x}_{[n]\setminus I} = \bar{0}_I}$ is irreducible as well.*

Proof. If $|\mathrm{var}(P)| \leq 3$, the claim clearly holds. Let $P = \sum\limits_{i=1}^{n} T_i(x_i)$ and suppose that $|\mathrm{var}(P)| \geq 4$. Since P is irreducible, there exists $\ell \in [n]$ such that $T_\ell(x_\ell)$ is not a perfect p-th power. Consider the set $I = \{x_j, x_k, x_\ell\}$. If $|I| < 3$, add arbitrary elements from $\mathrm{var}(P)$ to I so that $|I| = 3$. By definition, $P|_{\bar{x}_{[n]\setminus I}}$ is an trivariate polynomial which is not a perfect p-th power, due to $T_\ell(x_\ell)$, and thus irreducible by Lemma 29. ◀

We show that the map $\mathrm{H}_n \triangleq G_{n,3}(\bar{y}, \bar{z})$ is an essential factorization scheme for sum of univariates. Similarly to Section 3.1, we define $\Psi_{\mathrm{H}}(P)$ as the (monic) variable-essential factor of $P(\mathrm{H}_n)$. We now show that it satisfies Definition 17.

▶ **Lemma 31.** *Let P be an irreducible polynomial which can be expressed as sums of univariates. Then $\Psi_{\mathrm{H}}(P)$ is well-defined.*

Proof. Let $P(\mathrm{H}_n) = f_1 \cdot f_2 \cdots f_m$ be the unique factorization of $P(\mathrm{H}_n)$ into irreducible factors. Let $x_i \in \mathrm{var}(P)$. By Observation 27, $\mathcal{R}_{\{x_i\}}(P(\mathrm{H}_n)) = T_i(x_i) + \sum_{j \neq i} T_j(0)$. Therefore, at least one of f_k-s depends on x_i. Now, assume for a contradiction and wlog that $x_j \in \mathrm{var}(\mathcal{R}_{\{x_j\}}(f_1))$ and $x_k \in \mathrm{var}(\mathcal{R}_{\{x_k\}}(f_2))$. As $x_j, x_k \in \mathrm{var}(P)$, let I be the set guaranteed by Corollary 30. By Observation 27 we have that:

$$P|_{\bar{x}_{[n]\setminus I} = \bar{0}_I} = \mathcal{R}_I\left(P(\mathrm{H}_n)\right) = \mathcal{R}_I(f_1) \cdot \mathcal{R}_I(f_2) \cdots \mathcal{R}_I(f_m)$$

in contradiction to the irreducibility of $P|_{\bar{x}_{[n]\setminus I} = \bar{0}_I}$. ◀

We are ready to proceed with the proof of Theorem 3. The description of the algorithm is given in Algorithm 2.

Proof of Theorem 3. We analyze Algorithm 2. The claim regarding the running time is clear. Let $F = P_1 \cdot P_2 \cdot \ldots \cdot P_m$ be F's factorization into irreducible factors (possibly with repetitions). Observe that the algorithm finds all the variable-essential factors f of each such P. That is, $f = \Psi_{\mathrm{H}}(P)$ and $P(\mathrm{H}_n) = f \cdot P'$. We claim that for each P_j the algorithm outputs $\alpha_j \cdot P_j$ for some $0 \neq \alpha_j \in \mathbb{F}$. Therefore, the correct constant is found in Line 9. By definition, $\mathcal{R}_{\{x_i\}}(P') \in \mathbb{F}$ and hence $\mathcal{R}_{\{x_i\}}(f) \sim \mathcal{R}_{\{x_i\}}(P(\mathrm{H}_n))$. We consider three cases:

1. $V = \{x_i\}$ for some $i \in [n]$. Then the algorithm outputs $\frac{\mathcal{R}_{\{x_i\}}(f)}{\mathrm{lc}_{x_i}\left(\mathcal{R}_{\{x_i\}}\right)} \sim \mathcal{R}_{\{x_i\}}(P(\mathrm{H}_n)) = P$ (as the first term is 0).
2. $V = \{x_i, x_j\}$ for some $i \neq j \in [n]$. Then the algorithm outputs $\frac{\mathcal{R}_{\{x_i,x_j\}}(f)}{\mathrm{lc}_{x_i}\left(\mathcal{R}_{\{x_i,x_j\}}\right)} \sim \mathcal{R}_{\{x_i\}}(P(\mathrm{H}_n)) = P$ (as the second term is 0).
3. $|V| \geq 3$. By Lemma 29 P must be of the form $P = \sum_{m=1}^{n} T_m(x_m)$. We get that:

$$\mathcal{R}_{\{x_i,x_j\}}(f) = \frac{\mathcal{R}_{\{x_i,x_j\}}(f)}{\mathcal{R}_{\{x_i,x_j\}}(P')} = \frac{\mathcal{R}_{\{x_i,x_j\}}(f)}{\mathcal{R}_{\{x_i,x_j\}}(P')} = \frac{T_i(x_i) + T_j(x_j) + \sum_{m \neq i,j} T_m(0)}{\mathcal{R}_{\{x_i,x_j\}}(P')}$$

   Therefore

$$\mathrm{lc}_{x_i}\left(\mathcal{R}_{\{x_i,x_j\}}(f)\right) = \frac{\mathrm{lc}_{x_i}(T_i)}{\mathcal{R}_{\{x_i,x_j\}}(P')}$$

```
Input: A polynomial $F \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree $d$ which splits into sums of
       univariate polynomials.
Output: A list $P_1, \ldots, P_k$ of the irreducible factors of $F$. That is, $F = P_1 \cdot \ldots \cdot P_k$.

1 Set $H_n \triangleq G_{n,3}(\bar{y}, \bar{z})$ ;
2 Use Lemma 24 to Factor $F(H_n)$. Let $S$ be the set of the irreducible factors ;
3 foreach $f \in S$ do
4     Compute $V = \{ i \mid x_i \in \mathrm{var}(\mathcal{R}_{\{x_i\}}(f)) \}$ /* By brufeforce monomial expansion
        */
5     if $|V| = 0$ then continue to the next $f \in S$;
6     Pick $i \in V$ ;
7     Output
          $P = \sum_{j \in V \setminus \{i\}} \frac{\mathcal{R}_{\{x_i,x_j\}}(f)}{\mathrm{lc}_{x_i}(\mathcal{R}_{\{x_i,x_j\}})} - (|V| - 2) \cdot \frac{\mathcal{R}_{\{x_i\}}(f)}{\mathrm{lc}_{x_i}(\mathcal{R}_{\{x_i\}})}$
8 Compute $\alpha \in \mathbb{F}$ such that $F(H_n) = \alpha \cdot P_1(H_n) \cdot \ldots \cdot P_k(H_n)$ /* via 6-variate
    polynomial interpolation                                                                  */
9 Set $P_1 \leftarrow \alpha \cdot P_1$.
```

Algorithm 2. Factoring algorithm for polynomials that split into sums of univariate polynomials

and hence

$$\frac{\mathcal{R}_{\{x_i,x_j\}}(f)}{\mathrm{lc}_{x_i}\left(\mathcal{R}_{\{x_i,x_j\}}\right)} = \frac{T_i(x_i) + T_j(x_j) + \sum_{m \neq i,j} T_m(0)}{\mathrm{lc}_{x_i}(T_i)}.$$

Similarly,

$$\frac{\mathcal{R}_{\{x_i\}}(f)}{\mathrm{lc}_{x_i}\left(\mathcal{R}_{\{x_i\}}\right)} = \frac{T_i(x_i) + \sum_{\ell \neq i} T_\ell(0)}{\mathrm{lc}_{x_i}(T_i)}.$$

Consequently, the algorithm outputs $\frac{\sum_{m=1}^{n} T_m(x_m)}{\mathrm{lc}_{x_i}(T_i)} \sim P$.

◀

We now move to the proof of Theorem 4. The naive approach is to Apply 2 to an arbitrary sparse polynomial and then use Lemma 28 to get rid of the spurious factors. However, it might be the case that $F(G_{n,3}) \equiv 0$ although $F \not\equiv 0$. We solve this problem by considering F along the line $F(G_{n,3} + t \cdot \bar{a})$ when $\bar{a} \in \mathbb{F}^n$ is such that $F(\bar{a}) \neq 0$. We then set $t = 0$. Formally, the description of the algorithm is given in Algorithm 3.

Proof of Theorem 4. We analyze Algorithm 3. The analysis is similar to the analysis of Algorithm 2 barring two observations. First, observe that $F(H_n) \not\equiv 0$ since $F(H_n)|_{z_1=z_2=z_3=0,t=1} = F(\bar{a}) \neq 0$. Second, let P be a sum of univariates such that $P \mid F$. Then $\Psi_H(P) \in L$. ◀

6 Conclusions and Remarks

In this paper we give the first efficient deterministic factorization algorithm for sparse polynomials that split into multilinear factors and sums of univariate polynomials. The

```
Input: An $s$-sparse polynomial $F \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree $d$
Output: A list $P_1, \ldots, P_k$ of factors of $F$ such that $P_j$ is a sum of univariates.

1 Find $\bar{a} \in \mathbb{F}^n$ such that $F(\bar{a}) \neq 0$ ;
2 Set $\mathrm{H}_n \triangleq G_{n,3}(\bar{y}, \bar{z}) + t \cdot \bar{a}$ ;
3 Use Lemma 24 to Factor $F(\mathrm{H}_n)$. Let $S$ be the set of the irreducible factors ;
4 Compute $S' \triangleq \{f|_{t=0} \mid f \in S\}$ ;
5 Use Lemma 24 to the polynomials in $S'$. Let $S''$ be the result.;
6 Invoke Algorithm 2 with $S''$ instead of $S$. Let $L$ be the result. ;
7 foreach $P \in L$ do
8     Find the largest $e \leq d$ such that $P^e \mid F$ using Lemma 28. ;
9     if $e > 0$ then Output $P$ $e$ times.;
```

Algorithm 3. Computing sums of univariates factors of sparse polynomials.

key ingredient in the algorithm is the Essential Factorization Schemes. We hope that these schemes could be applied to handle richer classes of sparse polynomials.

A natural question to ask is whether it would possible to extend the algorithm to compute multilinear factors of an arbitrary sparse polynomial. Another open question is to improve the dependence on the characteristic from polynomial to polylogarithmic.

On a final note, Example 5.1 in [6] is followed by a question (quote): "Can the output size for the factoring problem be actually more than quasi-polynomial in the input size?" Our next example provides a positive answer to this question over fields with super-polylogarithmic characteristics.

▶ **Example 32.** Let $p = 2k-1$ be an odd prime, $\mathbb{F} = \mathbb{F}_{p^\ell}$ and $n, \ell \geq 1$. Consider the polynomial $f(\bar{x}) = (x_1+x_2+\ldots+x_n)^{p+1}$ which can be written as a square of $g(\bar{x}) = (x_1+x_2+\ldots+x_n)^k$. Observe that $f(\bar{x}) = (x_1^p + x_2^p + \ldots + x_n^p) \cdot (x_1 + x_2 + \ldots + x_n)$ and therefore $\|f\| \leq n^2$. On the other hand, $\|g\| = \binom{n+p/2-1}{p/2} = \Omega\left((\frac{n+p}{p})^p + (\frac{n+p}{n})^n\right)$.

Acknowledgments. The author would like to thank the anonymous referees for their comments.

References

1 M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynominal interpolation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 301–309, 1988.

2 A. M. Cuyt and W. Lee. Sparse interpolation of multivariate rational functions. *Theor. Comput. Sci.*, 412(16):1445–1456, 2011.

3 S. Gao, E. Kaltofen, and A. G. B. Lauder. Deterministic distinct-degree factorization of polynomials over finite fields. *J. Symb. Comput.*, 38(6):1461–1470, 2004.

4 J. von zur Gathen. Who was who in polynomial factorization:. In *ISSAC*, page 2, 2006.

5 J. von zur Gathen and J. Gerhard. *Modern computer algebra.* Cambridge University Press, 1999.

6 J. von zur Gathen and E. Kaltofen. Factoring sparse multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, 1985.

7 E. Grigorescu, K. Jung, and R. Rubinfeld. A local decision test for sparse polynomials. *Inf. Process. Lett.*, 110(20):898–901, 2010.

8 A. Gupta, N. Kayal, and S. V. Lokam. Reconstruction of depth-4 multilinear circuits with top fanin 2. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 625–642, 2012. Full version at http://eccc.hpi-web.de/report/2011/153.

9 V. Guruswami and M. Sudan. Improved decoding of reed-solomon codes and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

10 V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

11 E. Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM J. on computing*, 14(2):469–489, 1985.

12 E. Kaltofen. Factorization of polynomials given by straight-line programs. In S. Micali, editor, *Randomness in Computation*, volume 5 of *Advances in Computing Research*, pages 375–412. JAI Press Inc., Greenwhich, Connecticut, 1989.

13 E. Kaltofen. Polynomial factorization: a success story. In *ISSAC*, pages 3–4, 2003.

14 E. Kaltofen and P. Koiran. On the complexity of factoring bivariate supersparse (lacunary) polynomials. In *ISSAC*, pages 208–215, 2005.

15 E. Kaltofen and B. M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. of Symbolic Computation*, 9(3):301–320, 1990.

16 Z. S. Karnin, P. Mukhopadhyay, A. Shpilka, and I. Volkovich. Deterministic identity testing of depth 4 multilinear circuits with bounded top fan-in. *SIAM J. on Computing*, 42(6):2114–2131, 2013.

17 N. Kayal. *Derandomizing some number-theoretic and algebraic algorithms*. PhD thesis, Indian Institute of Technology, Kanpur, India, 2007.

18 A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–223, 2001.

19 S. Kopparty, S. Saraf, and A. Shpilka. Equivalence of polynomial identity testing and deterministic multivariate polynomial factorization. In *Proceedings of the 29th Annual IEEE Conference on Computational Complexity (CCC)*, pages 169–180, 2014.

20 C. Saha, R. Saptharishi, and N. Saxena. A case of depth-3 identity testing, sparse factorization and duality. *Computational Complexity*, 22(1):39–69, 2013.

21 S. Saraf and I. Volkovich. Blackbox identity testing for depth-4 multilinear circuits. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 421–430, 2011. Full version at http://eccc.hpi-web.de/report/2011/046.

22 N. Saxena. Diagonal circuit identity testing and lower bounds. In *Automata, Languages and Programming, 35th International Colloquium*, pages 60–71, 2008. Full version at http://eccc.hpi-web.de/eccc-reports/2007/TR07-124/index.html.

23 V. Shoup. A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic. In *ISSAC*, pages 14–21, 1991.

24 A. Shpilka and I. Volkovich. Improved polynomial identity testing for read-once formulas. In *APPROX-RANDOM*, pages 700–713, 2009. Full version at http://eccc.hpi-web.de/report/2010/011.

25 A. Shpilka and I. Volkovich. On the relation between polynomial identity testing and finding variable disjoint factors. In *Automata, Languages and Programming, 37th International Colloquium (ICALP)*, pages 408–419, 2010. Full version at http://eccc.hpi-web.de/report/2010/036.

26 A. Shpilka and I. Volkovich. On reconstruction and testing of read-once formulas. *Theory of Computing*, 10:465–514, 2014.

27 A. Shpilka and I. Volkovich. Read-once polynomial identity testing. *Computational Complexity*, 2015.

28 A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

29 M. Sudan. Decoding of reed solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

30 R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 216–226, 1979.