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

$$\sum_{i \in S : e \in \text{span}(i)} d_i \leq c_e \text{ for each edge } e \in E.$$

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. 265–283

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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].

$$\begin{aligned} \text{maximize} \quad & \sum_i w_i \cdot x_i && \textbf{(Nat-LP)} \\ \text{s.t.} \quad & \sum_{i: e \in \text{span}(i)} d_i \cdot x_i \leq c_e \quad \forall\, e \in E \\ & 0 \leq x_i \leq 1 \quad \forall\, 1 \leq i \leq n \end{aligned}$$

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 $1 \leq i \leq n$ 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 $d_i \leq B$ for all tasks i and $B \leq c_e$ for all edges e (the *no-bottleneck* property) then the integrality gap is at most 48 [11]. Second, if $\delta > 0$ is such that $d_i \leq (1 - \delta) \cdot c_e$ 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 $\sum_{i \in S} x_i \leq 1$ 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 $\sum_{i \in S} x_i \leq \text{rank}(S)$ 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, $\text{span}(i)$ 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 $\text{rank}(S)$ 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.

$$\begin{aligned}
\text{maximize} \quad & \sum_i w_i \cdot x_i && \textbf{(Rank-LP)}\\
\text{s.t.} \quad & \sum_{i:e\in \text{span}(i)} d_i \cdot x_i \le c_e \quad \forall\, e \in E\\
& \sum_{i\in S} x_i \le \text{rank}(S) \quad \forall \text{ intersecting sets of tasks } S && (3)\\
& 0 \le x_i \le 1 \quad \forall\, 1 \le i \le n
\end{aligned}$$

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. $x_i = 1 \implies \sum_{e \in \text{span}(i)} d_i \cdot y_e \leq 2$
3. $y_e > 0 \implies \sum_{i: e \in \text{span}(i)} d_i \cdot x_i \geq \frac{c_e}{2}$
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.

```
Initialize S, D^u and D^s to ∅.
for each task i in increasing order of demand d_i do
    if S ∪ {i} is feasible then add i to S
    else
        Let B(i) be the edges whose capacities are violated by S ∪ {i}.
        if some e ∈ B(i) is undersaturated by S then add i to D^u
        else add i to D^s
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

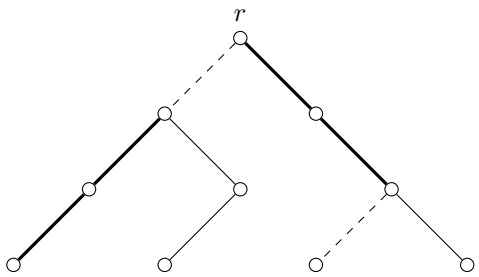

Figure 1 The tree T' is partitioned into six paths. Two are drawn with bold edges, two with thin edges, and two with dashed edges. Note that the root has degree 2, but the path it lies on is broken into two paths.

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 $\sum_{e \in \text{span}(i)} d_i y_e \geq 1$.*

Proof. The statement holds for each i of the form $i(e)$ for some $e \in F$ by Lemma 13 (and noting $P(e,r) \subseteq \text{span}(i)$). 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 $\sum_{e' \in \text{span}(i)} d_i y_{e'} \geq \sum_{e' \in P(e,r)} d_i y_{e'} \geq \sum_{e' \in P(e,r)} d_{i(e)} y_{e'} = 1$ 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 $e \in \text{span}(i)$ lies on some $P(a,r)$ path for some endpoint a of i, then $\sum_{e \in \text{span}(i)} d_i y_e \leq 2$.

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 $k = h$.

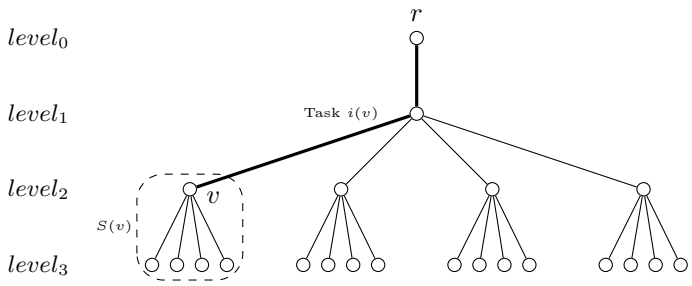

Figure 2 Bad instance T^3. The span of task $i(v)$ is drawn with bold lines. The outlined group of nodes are the starting points of tasks in $S(v)$.

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 $\text{span}(i)$ 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. $x_i = 1 \implies \sum_{e \in \text{span}(i)} d_i \cdot y_e \leq k$
3. $y_e > 0 \implies \sum_{i : e \in \text{span}(i)} d_i \cdot x_i \geq \frac{c_e}{2}$
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.

$$\begin{aligned} 0 \leq z^T Y z &= z_0 Y_{0,0} z_0 + 2\sum_i z_0 z_i Y_{0,i} + \sum_{i,j} z_i z_j Y_{i,j} \\ &= 1 - 2\sum_{i \in S} x_i + \sum_{i,j \in S} Y_{i,j} \\ &= 1 - 2\sum_{i \in S} x_i + \sum_{i \in S} x_i \\ &= 1 - \sum_{i \in S} x_i. \end{aligned}$$

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 $\text{diag}(Y) = (1, x)$. ◀

In fact, this proof does not require the "level 1" protection matrices for $x \in \text{LS}_+^2(\mathcal{P})$ 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* $\max\{w^T \cdot x : x \in \text{LS}_+^2(\mathcal{P})\}$ *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.