Conflict-Based Local Search for Minimum Partition into Plane Subgraphs

Jack Spalding-Jamieson
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Brandon Zhang
Vancouver, Canada

Da Wei Zheng
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Abstract

This paper examines the approach taken by team gitastrophe in the CG:SHOP 2022 challenge. The challenge was to partition the edges of a geometric graph, with vertices represented by points in the plane and edges as straight lines, into the minimum number of planar subgraphs. We used a simple variation of a conflict optimizer strategy used by team Shadoks in the previous year's CG:SHOP to rank second in the challenge.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases local search, planar graph, graph colouring, geometric graph, conflict optimizer

Digital Object Identifier 10.4230/LIPIcs.SoCG.2022.72

Category CG Challenge

Supplementary Material
Software (Source Code): `https://github.com/jacketsj/cgshop2022-gitastrophe`
archived at `swh:1:dir:0e86e287cc9a882064e46283cb35cbd64b0df4e8`

1 Introduction

Given a graph $G = (V, E)$ and an assignment $f : V \to \mathbb{Z}^2$ inducing a straight-line drawing in $\mathbb{R}^2$ with integer vertex coordinates, the *minimum partition into plane subgraphs* problem asks for a partition of the edges E into a minimal number of sets $E_1, E_2, \ldots, E_k$ such that for each subgraph $G_i = (V, E_i)$, f induces a planar straight-line drawing. That is, no pair of edges from the same subset intersect, except possibly at their common endpoint. This was the problem posed in the 2022 Computational Geometry Challenge (CG:SHOP 2022). For more detail about the challenge, we refer readers to the summary paper [5].

Reduction to vertex-colouring

Solving the minimum partition into plane subgraphs problem for $G = (V, E)$ is equivalent to solving the well-studied minimum vertex-colouring problem for the intersection *conflict graph* G' with $V(G') = E(G)$ and $E(G')$ equal to the set of intersections in the provided straight-line drawing. We did not explicitly use the geometric properties of the instances and instead solved the aforementioned vertex colouring problem.

Henceforth, we will only refer to the intersection conflict graph G' induced by the instance. Vertices will refer to the vertices $V(G')$, and edges will refer to the edges $E(G')$. Our goal is to partition the vertices using a minimum set of colour classes $\mathcal{C} = \{C_i\}$, where no two vertices in the same colour class C_i are incident to a common edge.

Existing literature

There are many existing practical heuristic algorithms [11, 10, 13, 14, 1] to the vertex-colouring problem. Many of these algorithms used DIMACS benchmark [9] graphs to evaluate their results. In subsection 3.3 we compare the results of our methods for these instances. Most of the benchmark instances had comparatively few edges (on the order of thousands or millions); the largest intersection graphs considered in the CG:SHOP challenge had over 1.5 billion edges.

We found a variation of the *conflict optimizer* strategy employed by team Shadoks for CG:SHOP 2021 [4] to be effective. We describe this strategy in Section 2. Using this strategy, we, team `gitastrophe`, placed second overall, and first among all junior teams. This result was surprising to us, as our methods were relatively simple, relying exclusively on the naive reduction to vertex-colouring. The first- and third-place teams also make use of similar techniques [3] [6], although the fourth place team uses a very different SAT-based approach [12].

2 Methods

2.1 Solution initialization

We used the traditional greedy algorithm of Welsh and Powell [15] to obtain initial solutions: order the vertices in decreasing order of degree, and assign each vertex the minimum-label colour not used by its neighbours. We attempted to use different orderings for the greedy algorithm, such as sorting by the slope of the line segment associated with each vertex, and we also tried numerous other strategies. Ultimately, we found that after running our solution optimizer for approximately the same amount of time, all initializations resulted in equal number of colours.

2.2 Solution optimization: conflict search

Our most successful method for improvement of the solutions was inspired by the conflict optimization approach used by the Shadoks team for CG:SHOP 2021 [4]. At a high-level, our algorithm will iteratively attempt to eliminate a selected colour class. The details are as follows:

1. Pick a random colour class C to be eliminated. Uncolour all vertices in C and add all vertices in that colour class to a conflict set S. We maintain only a valid vertex-colouring for the set $V(G') - S$. Once S is empty, we will have produced a valid vertex colouring of G' which uses one fewer colour.
2. Pick and remove a random element v from S. For each colour class, we compute the *conflict score* with v. The conflict score of a colour class C_i is

$$\sum_{\substack{u \in C_i \\ (u,v) \in E(G')}} 1 + q(u)^2 \tag{1}$$

   where $q(u)$ is the number of times that u has been removed from the conflict set S in previous iterations of this step.
3. Pick the colour class C_i with the lowest conflict score. Uncolour all vertices in C_i which are adjacent to v and add those vertices to S. Insert v into C_i.
4. Repeat steps 2 and 3 until the set S is empty.

There is no guarantee that this algorithm terminates. In practice, we restart the procedure when any value of $q(u)$ surpasses a fixed threshold.

The primary differences between our approach to conflict optimization and those of the first and third place teams are the choice of an exponent of 2 in Step 2, and the behaviour when $q(u)$ surpasses its fixed threshold.

Modifications to the conflict optimizer

Taking inspiration from memetic algorithms, which alternate between an intensification and a diversification stage, we continually switched between a phase where we used the above conflict score, and one where we minimized only the number of conflicts (i.e. we replaced the conflict score of (1) with $\sum_{u\in C_i,(u,v)\in E(G')} 1$). Each phase lasted for 10^5 iterations. Adding the conflict-minimization phase gave minor improvements to some of the challenge instances.

2.3 Failed approach: memetic algorithms

Although many of the leading approaches to vertex colouring are memetic, our attempts at implementing them performed poorly. These memetic algorithms take a long time to run on the standard DIMACS instances [9], and did not scale well to the much larger intersection graphs in the challenge.

We implemented the memetic algorithms Evo-Div [11] and HEAD [10], but neither of these approaches were able to improve on the scores obtained by the conflict optimizer. Both of these algorithms use TABUCOL [8], a tabu search algorithm, as their local search component, so we tried to replace it with the conflict optimizer. However, this proved to be ineffective. This may be attributed to a critical difference between TABUCOL and the conflict optimizer: the conflict optimizer does not expressly minimize the number of conflicting edges in the colouring, and only hopes to eventually resolve all conflicting vertices.

3 Results

3.1 Implementation

The conflict optimizer frequently looked up edges in the intersection graph. To speed this process up, we precomputed the adjacency matrix of the graph and stored it in memory for fast access. Our C++ implementation is available on Github.

3.2 Challenge computing environment

To perform our computations during the challenge, we mainly used a 32-core server with two Xeon E5-2698 v3s. We spent about 2 days of CPU time per instance to obtain our best solutions. Table 1 shows the scores of our greedy initialization, scores after running the conflict optimizer for 10 minutes, 1 hour, and 24 hours, and the best result we obtained in the challenge. Our algorithm obtains good results on many instances after a short period of time; it comes close to matching the best solutions we obtained in the challenge within 24 hours (and surpasses some, as there is randomness in the algorithm).

Table 1 Results of our algorithm on a subset of the challenge instances after fixed amounts of optimization time. Note that on instances `visp31334` and `reecn51526` we obtained better results after 24 hours than our final results from the challenge.

Instance	Greedy	10m	1h	24h	Final
`rvisp5013`	71	50	49	49	49
`rsqrpecn8051`	284	177	176	176	176
`sqrp10642`	186	124	124	124	124
`rsqrp14364`	225	137	137	137	137
`reecn16388`	210	152	152	151	151
`vispecn19370`	285	199	196	194	194
`sqrpecn23715`	657	436	425	423	423
`visp26405`	119	83	83	82	81
`sqrp28863`	316	209	192	191	191
`visp31334`	132	83	83	83	82
`vispecn35198`	379	262	246	242	243
`visp38574`	193	143	136	135	134
`sqrp41955`	362	236	214	204	204
`sqrpecn45700`	802	503	471	465	465
`visp48558`	230	159	147	144	144
`reecn51526`	456	334	317	311	312
`visp55158`	182	130	123	122	122
`vispecn58391`	609	440	394	370	369
`visp62685`	174	132	120	119	117
`vispecn65831`	711	522	473	442	440
`sqrpecn69904`	1152	740	693	651	650
`sqrp72075`	483	342	312	272	271

3.3 Comparison on DIMACS dataset

We ran our algorithm on the difficult DIMACS instances [9] to gauge our algorithm's performance on non-geometric graphs.

Table 2 shows our results after running our algorithm for 10 minutes, compared with some of the state of the art colouring algorithms HEAD [10] and QACOL [13, 14].

Surprisingly, the conflict optimizer works extremely poorly on random graphs, but is fast and appears to perform well on geometric graphs, matching the best-known results [7]. Interestingly, these geometric graphs are not intersection graphs as in the Challenge, but are generated based on a distance threshold.

Applying Cheeger's inequality [2], we note the intersection graphs resulting from the challenge instances have noticeably lower edge conductance than random graphs, and we believe this plays a part in the performance of the conflict optimizer.

4 Conclusion

The conflict optimizer approach was very effective for the large geometric intersection graphs for the CG:SHOP 2022 challenge. Further investigation is needed into the reason the conflict optimizer approach was effective.

Table 2 Comparison of our method with state-of-the-art graph colouring algorithms. The conflict optimizer underperforms except on the geometric graphs `r`X`.`Y and `dsjr`X`.`Y.

Instance	Colours	HEAD [10]	QACOL [13, 14]
`dsjc250.5`	29	28	28
`dsjc500.1`	13	12	12
`dsjc500.5`	52	47	48
`dsjc500.9`	130	126	126
`dsjc1000.1`	21	20	20
`dsjc1000.5`	93	82	82
`dsjc1000.9`	235	222	222
`r250.5`	65	65	65
`r1000.1c`	98	98	98
`r1000.5`	234	245	238
`dsjr500.1c`	85	85	85
`dsjr500.5`	122	-	122
`le450_25c`	26	25	25
`le450_25d`	26	25	25
`flat300_28_0`	33	31	31
`flat1000_50_0`	91	50	-
`flat1000_60_0`	93	60	-
`flat1000_76_0`	92	81	81
`C2000.5`	173	146	145
`C4000.5`	317	266	259

References

1. Daniel Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.
2. Jeff Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. *Problems in analysis*, 625(195-199):110, 1970.
3. Loïc Crombez, Guilherme D. da Fonseca, Yan Gerard, and Aldo Gonzalez-Lorenzo. Shadoks approach to minimum partition into plane subgraphs. In *Symposium on Computational Geometry (SoCG)*, pages 71:1–71:8, 2022.
4. Loïc Crombez, Guilherme D da Fonseca, Yan Gerard, Aldo Gonzalez-Lorenzo, Pascal Lafourcade, and Luc Libralesso. Shadoks approach to low-makespan coordinated motion planning (cg challenge). In *37th International Symposium on Computational Geometry (SoCG 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.
5. Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Stefan Schirra. Minimum partition into plane subgraphs: The CG: SHOP Challenge 2022. *CoRR*, abs/2203.07444, 2022. `arXiv:2203.07444`.
6. Florian Fontan, Pascal Lafourcade, Luc Libralesso, and Benjamin Momège. Local search with weighting schemes for the CG:SHOP 2022 competition. In *Symposium on Computational Geometry (SoCG)*, pages 73:1–73:6, 2022.
7. Olivier Goudet, Cyril Grelier, and Jin-Kao Hao. A deep learning guided memetic framework for graph coloring problems, 2021. `arXiv:2109.05948`.
8. Alain Hertz and Dominique de Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351, 1987.
9. David S Johnson and Michael A Trick. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993*, volume 26. American Mathematical Soc., 1996.

10 Laurent Moalic and Alexandre Gondran. Variations on memetic algorithms for graph coloring problems. *Journal of Heuristics*, 24(1):1–24, 2018.

11 Daniel Cosmin Porumbel, Jin-Kao Hao, and Pascale Kuntz. An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. *Computers & Operations Research*, 37(10):1822–1832, 2010.

12 André Schidler. SAT-based local search for plane subgraph partitions. In *Symposium on Computational Geometry (SoCG)*, pages 74:1–74:8, 2022.

13 Olawale Titiloye and Alan Crispin. Quantum annealing of the graph coloring problem. *Discret. Optim.*, 8:376–384, 2011.

14 Olawale Titiloye and Alan Crispin. Parameter tuning patterns for random graph coloring with quantum annealing. *PloS one*, 7(11):e50060, 2012.

15 D. J. A. Welsh and M. B. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, January 1967.