Local search with weighting schemes for the CG:SHOP 2022 competition

Florian Fontan
Independent Researcher, France

Pascal Lafourcade
Université Clermont-Auvergne, CNRS, Mines de Saint-Étienne, LIMOS, 63000 Clermont-Ferrand, France

Luc Libralesso
Atoptima, 16 Place Sainte Eulalie, 33000 Bordeaux, France

Benjamin Momège
Independent Researcher, France

Abstract

This paper describes the heuristics used by the `LASAOFOOFUBESTINNRRALLDECA`[1] team for the CG:SHOP 2022 challenge. We introduce a new greedy algorithm that exploits information about the challenge instances, and hybridize two classical local-search schemes with weighting schemes. We found 211/225 best-known solutions. Hence, with the algorithms presented in this article, our team was able to reach the 3rd place of the challenge, among 40 participating teams.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases heuristics, vertex coloring, digital geometry

Category CG challenge

Supplementary Material `https://github.com/librallu/dogs-color`

Funding *Pascal Lafourcade*: This work is supported by the French ANR PRC grant MobiS5 (ANR-18-CE39-0019), DECRYPT (ANR-18-CE39-0007), and SEVERITAS (ANR-20-CE39-0005).

1 Introduction

The Fourth Geometric Optimization Challenge proposed in SoCG 2022 is about finding Minimum Plane Partitions. Given a geometric graph with vertices represented by points in the plane, and edges by straight-line connections between vertices, we aim to partition edges into as few subsets of disjoint lines, and to minimize the partition size.

Solving an instance of this challenge is equivalent to solving an instance of the classical vertex coloring problem (VCP). Indeed, we can define a graph $G' = (V', E')$ where V' is the set of lines, and there is an edge in E' between two intersecting lines. Finding a valid vertex coloring of G' with k colors is equivalent to finding a coloring of the original problem with k colors. We refer the reader to the competition for more details about the vertex coloring problem transformation [TODO].

In our approach for the SoCG challenge, we made the choice to focus only on solving the equivalent vertex coloring problem instances instead of directly solving the competition instances. We show that "classical" graph coloring algorithms are competitive, even with ad-hoc approaches for the SoCG challenge.

[1] At the beginning of the competition, we could not find some team name. Thus, we used a simple permutation of the first team members names letters, hence the funny name.

We combined some classical algorithms found in the vertex coloring literature (two local-search neighborhoods presented below) with a dynamic weighting scheme present in many methods for solving other graphs problems (and closely related to the conflict optimizer used in the previous CG:SHOP competition [2]). Furthermore, we show that this new algorithm is simple to implement, yet efficient as it enabled us to reach the 3rd place in the CG:SHOP 2022 competition.

The main idea of our approach is to optimize some solutions with some local search algorithms. We use two techniques to find some solutions: a segment orientation greedy ad-hoc algorithm, and DSATUR, one of the most common greedy algorithms in the vertex coloring literature. Then, using these initial solutions, we apply two optimization techniques: a conflict minimization approach with a weighting scheme: Conflict Weighting Local Search (CWLS), a partial coloring approach with a weighting scheme: Partial Weighting Local Search (PWLS).

In Section 2, we present some mainstream algorithmic components to solve the vertex coloring problem. Section 3 present the greedy algorithms we implemented (namely the saturation degree and orientation-based greedy algorithms). Sections 4 and 5 present the local-search and weighting schemes we used for the competition (namely the conflict minimization, and vertices non-colored minimization). Finally, we compare our approaches in Section 6.

2 Related work

Multiple resolution methods were developed these last 40 years:

A first category consists of greedy algorithms. These algorithms are used to find good quality initial solutions in a short amount of time. We present two of most considered greedy approaches in Section 3.

A second category of algorithms consists of exact methods using branch-and-bound algorithms. These algorithms are complete search methods, thus are able to find an optimal solution and prove that they are indeed optimal. Such methods extend the DSATUR heuristic by allowing it to backtrack [11, 3]. Another category of exact methods (branch-and-cut-and-price) decompose the vertex coloring problem into an iterative resolution of two sub-problems [10, 6, 4]. The higher level that maintains a set of valid colors (defined by an independent set). It aims to cover all the vertices with a minimum-size set of colors, thus solving a set-covering problem (often described as the "master problem"). A lower level (solving a sub-problem) aiming to find a new valid coloring that would be promising, thus solving a maximum weight independent set problem (often described as the "pricing problem"). Such methods are usually able to find the optimal coloring for graphs with a few hundred vertices. However, the CG:SHOP 2022 competition instances involve at least a few thousands vertices, thus they appear to be not suited for this competition.

Finally, a third category of algorithms consists of local search algorithms. These methods start by an initial solution (often found by some greedy algorithm), remove some color, thus making the solution infeasible but with fewer colors, then apply some perturbations, aiming to make it feasible again. We present this approach in more details in Sections 4 and 5.

3 Finding Initial Solutions

Applying some local search scheme requires some initial solutions. We present in this section two greedy algorithms that enable us to find promising initial solutions quickly.

We use two approaches to find initial solutions. The best one is used for our other algorithms.

1. **DSATUR.** DSATUR is one of the most common greedy algorithms in the vertex coloring literature. It was introduced in 1979 [1]. It selects the vertex that has the most colors in its neighborhood, and assigns it to the first non-conflicting color until no vertex is left uncolored. Ties are broken by selecting the vertex with maximum degree. This algorithm does not take into account the specificities of the CG:SHOP 2022 competition.
2. **Segment Oriented Greedy.** We introduce a greedy algorithm that exploits the segment orientations. This algorithm is inspired from the *Recursive Largest First* algorithm, introduced in 1979 [8]. Recursive Largest first colors the graph one color at a time. The algorithm searches for an independent set of maximum size, assigns these vertices the same color, removes the newly colored vertices, and repeats until all vertices are assigned a color. The algorithm we propose (Segment Oriented Greedy) takes advantage of the fact that most of the challenge instances consist of long segments. Thus, if segments are almost parallel, it is likely that they do not intersect (thus forming an independent set). This greedy algorithm first sorts the segments by orientation, ranging from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$. For each segment in this order, we try to color it using the first available color. If we fail, we introduce a new color. This algorithm is efficient, produces interesting initial solutions and takes into account the specificities of the competition.

4 Conflict Weighting Local Search (CWLS)

One classical approach for the vertex coloring involves allowing solutions with conflicting vertices (two adjacent vertices with the same color). It was introduced in 1987 [7] and called TABUCOL (more details in Section 4). It starts with an initial solution, removes a color (usually the one with the least number of vertices), and assigns uncolored vertices with a new color among the remaining ones. This is likely to lead to some conflicts (*i.e.* two adjacent vertices sharing a same color). The local search scheme selects a conflicting vertex, and tries to swap its color, choosing the new coloring that minimizes the number of conflicts. If it reaches a state with no conflict, we obtain a new best-known solution. The process is repeated until the stopping criterion is met. Originally, TABUCOL also contains some tabu mechanism to avoid cycling through some states. In this article, we refer to this local-search neighborhood as conflict-minimization. TABUCOL and its conflict-minimization neighborhood, obtained at the time excellent performance. It was later embedded in a large variety of algorithms, including many state-of-the-art algorithms (to the best of our knowledge). For instance MACOL [9] that uses TABUCOL as a mutation operator, and the GPX crossover to combine solutions.

Learning scheme: While the original TABUCOL algorithm includes this "tabu-list" mechanism to avoid cycling, it is not always sufficient, and requires some hyperparameter tuning in order to obtain a good performance on a large variety of instances. To overcome this issue, we developed a weighting scheme inspired from recent work in graph problems (namely Row Weighting Local Search for the set covering problem [5]). The idea is to penalize each conflict the algorithm introduces, so frequently occurring conflicts have a larger penalty, thus less likely to be selected in the future. This allows the algorithm to naturally diversify the solutions it obtains. Similarly to the Row Weighting Local Search algorithm, we use a simple weighting scheme that initializes all the weights to 1 (thus one weight per edge). Each time a conflict is introduced, its weight is increased by 1. The algorithm minimizes the total weight instead of the total number of conflicts.

5 Partial Weighting Local Search (PWLS)

Another local search algorithm solving the vertex coloring problem was introduced in 2008 (PARTIALCOL). This algorithm proposes a new local search scheme that allows partial coloring (thus allowing uncolored vertices). The goal is to minimize the number of uncolored vertices. Similarly to TABUCOL, PARTIALCOL starts with an initial solution, removes one color (unassigning its vertices), and performs local search iterations until no vertex is left uncolored. When coloring a vertex, the adjacent conflicting vertices are uncolored. Then, the algorithm repeats the process until all vertices are colored, or the stopping criterion is met.

Learning scheme: We also introduce some weighting scheme for the PARTIALCOL local-search neighborhood. We assign a weight for each vertex, and the objective of the local search is to minimize the total weight of uncolored vertices. Each time a vertex becomes colored, its weight is increased by 1.

6 Results and Conclusions

All algorithms were executed on an Intel(R) Xeon(R) CPU E5-2687W v4 @ 3.00 GHz CPU, with 30 MB cache, and 768 GB RAM for 24 hours for each instance. To speed up the experiments, we ran 24 parallel runs on our machine. All algorithms have been implemented in the *rust* programming language and are publically available on a GitHub repository[2].

In order to compare the efficiency of our algorithms, we use the ARPD (Average Relative Percentage Deviation) as a comparison metric, which is defined as follows:

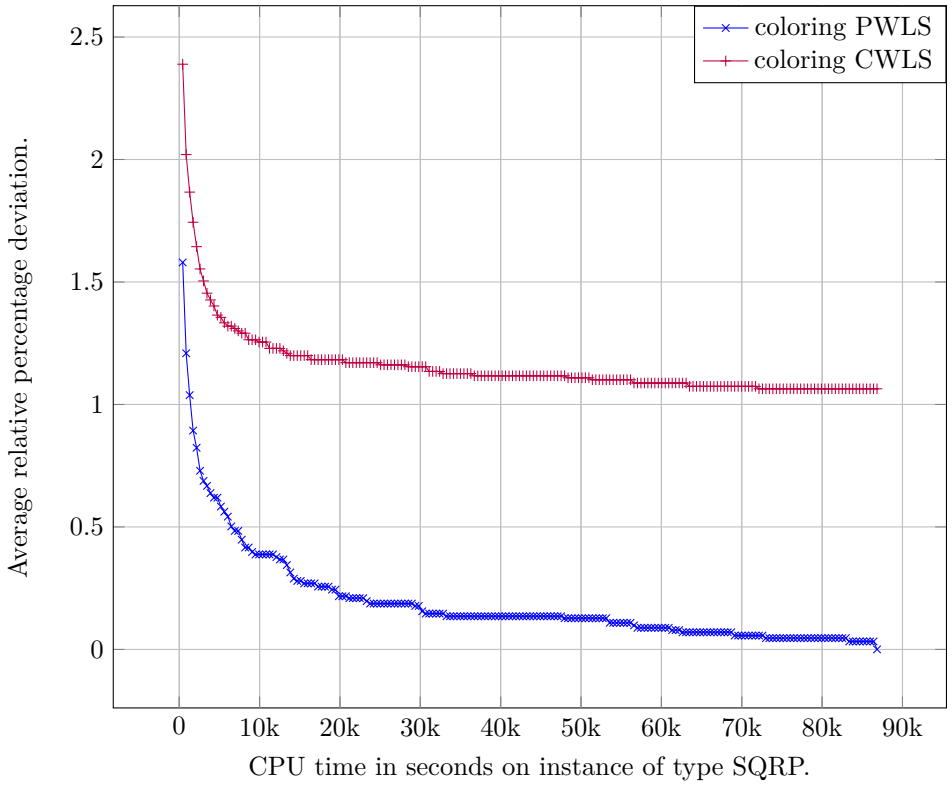
$$ARPD_{Ia} = \sum_{i \in I} \frac{M_{ai} - M_i^*}{M_i^*} \cdot \frac{100}{|I|}$$

where I is a set of instances with similar characteristics (in the same instance class), M_{ai} corresponds to the objective obtained by algorithm a on instance i. And M_i^* the reference solution objective for the instance i. The reference value is the best solution found by our team during the competition. The ARPD describes the performance of a given algorithm on a given instance class. This aggregation allows us to observe the overall behavior of some algorithm on some instance class. An ARPD close to 0 indicates that the algorithm is close to the best solutions we found during the competition (thus, the lower, the better).

Table 1 compares all the algorithms we implemented on each class of instances. Greedy algorithms report large ARPDs, thus being far from the best-known solutions. The orientation greedy generally outperforms the DSATUR algorithm as it handles instances with long segments, where the orientation is a good conflict predictor (thus all instances but the VISP-like). DSATUR outperforms the orientation greedy on the visp-like instances (where the segment localization is more important than the orientation). Local search weighting schemes allow getting closer to the best-known solutions. It is interesting to note that CWLS and PWLS are complementary, as CWLS seems to be more efficient than PWLS on RVISP/VISP classes of instances, and PWLS being more efficient on the others. Both local search methods are able to reach almost optimal solutions (less than 2% deviation to the best-known solution in average).

Figure 1 compares the solution quality over time of CWLS and PWLS on the SQRP and VISP instances. Both algorithms seem to be complementary depending on the instance class.

[2] `https://github.com/librallu/dogs-color`



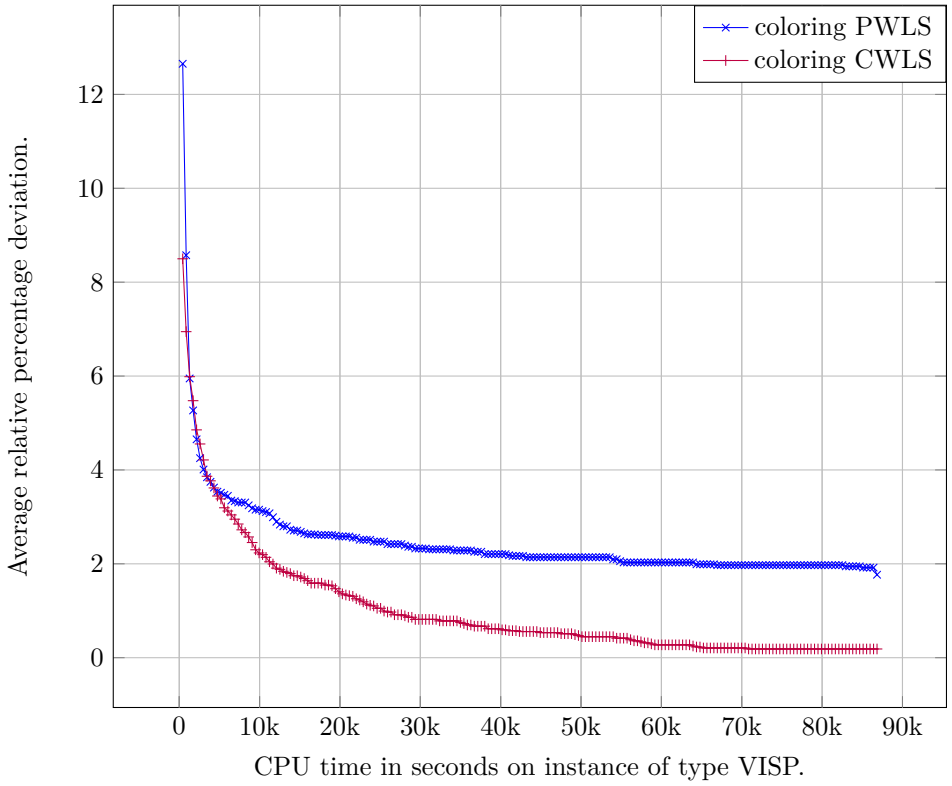


Figure 1 Comparison of the performance of CWLS and PWLS over time.

Instance class	CWLS	PWLS	DSATUR	orientation greedy
reecn	1.04	**0.0**	24.31	16.41
rsqrp	0.51	0.11	24.03	5.89
rsqrpecn	0.0	0.0	24.86	18.08
rvisp	**0.06**	0.99	25.06	26.94
rvispecn	0.0	0.14	30.9	49.93
sqrp	1.06	**0.0**	31.22	4.9
sqrpecn	0.88	**0.03**	22.74	14.11
visp	**0.19**	1.77	29.23	27.6
vispecn	0.49	**0.08**	34.75	54.66

Table 1 Average Relative Percentage Deviation comparison for all instance classes over 24h runs. Bold values indicate the algorithm that is statistically significantly better than the other one using a *Wilcoxon signed-rank test* with $\alpha = 0.05$.

From this competition, we study that adding a weighting scheme significantly improves the algorithm performance, making the algorithm more robust. This weighting allows a simple adaptive diversification mechanism. We studied two local-search neighborhoods (conflict-minimization and partial-coloring-minimization). For both of them, we presented some weighting scheme: Conflict Weighting Local Search (CWLS), and Partial Weighting Local Search (PWLS) CWLS and PWLS are both performant, and complementary depending on the instance class.

In the future, we plan to study the integration of such weighting schemes within memetic algorithms. Indeed, memetic algorithms are known to obtain excellent performance on classical vertex coloring instances.

References

1. Daniel Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.
2. Loïc Crombez, Guilherme D. da Fonseca, Yan Gerard, Aldo Gonzalez-Lorenzo, Pascal Lafourcade, and Luc Libralesso. Shadoks Approach to Low-Makespan Coordinated Motion Planning. In *37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189, pages 63:1–63:9, 2021. doi:10.4230/LIPIcs.SoCG.2021.63.
3. Fabio Furini, Virginie Gabrel, and Ian-Christopher Ternier. An improved dsatur-based branch-and-bound algorithm for the vertex coloring problem. *Networks*, 69(1):124–141, 2017.
4. Fabio Furini and Enrico Malaguti. Exact weighted vertex coloring via branch-and-price. *Discrete Optimization*, 9(2):130–136, 2012.
5. Chao Gao, Xin Yao, Thomas Weise, and Jinlong Li. An efficient local search heuristic with row weighting for the unicost set covering problem. *European Journal of Operational Research*, 246(3):750–761, 2015.
6. Stefano Gualandi and Federico Malucelli. Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS Journal on Computing*, 24(1):81–100, 2012.
7. Alain Hertz and Dominique de Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351, 1987.
8. Frank Thomson Leighton. A graph coloring algorithm for large scheduling problems. *Journal of research of the national bureau of standards*, 84(6):489, 1979.
9. Zhipeng Lü and Jin-Kao Hao. A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203(1):241–250, 2010.

189 **10** Anuj Mehrotra and Michael A Trick. A column generation approach for graph coloring.
190 *INFORMS Journal on Computing*, 8(4):344–354, 1996.
191 **11** Pablo San Segundo. A new dsatur-based algorithm for exact vertex coloring. *Computers &*
192 *Operations Research*, 39(7):1724–1733, 2012.