Surveying Cooperative Approaches for the Graph Coloring Problem

Gabriel Perez, Corey Roberts, and Javier Valenzuela

Department of Electrical and Computer Engineering
University of Texas at Austin,
Austin, TX 78712
{gabriel_ohseas_perez, croberts22, javier.valenzuela}@utexas.edu

Abstract. The Graph Coloring Problem is an NP-Complete problem with a rich history of approximation algorithms and heuristics. In this paper, we explore a modification to cooperative local search approaches for the graph coloring problem.

1 Introduction

The Graph Coloring Problem and its variants are long existing problems with a very low barrier to understanding. Despite that, the class of problems are incredibly difficult problem to solve exactly. Applications of the class of problems include identifying whether a circuit board has a short circuit, solving Sudoku for any size grid, and problems of scheduling exams for university lecturers [Lew15].

In this paper, we explore the idea of algorithms with various hyper-parameters cooperating to determine whether we get improved lower bounds for the Graph Coloring Problem approximations.

2 Definitions

2.1 Graph & Graph Properties

A graph, G, is defined as $G = (V, E)$ a set of n vertices V and m edges E. For this class of problems, we have a function $c : V \mapsto \mathbb{N}$. This function is called the

coloring function. The coloring is *legal* if $\forall(u,v) \in E \implies c(u) \neq c(v)$. If the graph can be colored by using k colors, then the graph is called $k-colorable$. The minimum k that can be used to color the graph G is called the *chromatic number* of the graph G denoted χ. Another important term is neighboring coloring, which involves changing the color of a single vertex.

2.2 Central Algorithm

The central algorithm at the heart of these heuristics is to start at $|V|$ colorings, see if a feasible one exists, and then proceed downward. This approach was used for all of the algorithms discussed in this paper. Appendix B describes additional starting coloring schemes considered for the algorithms, but ultimately not used.. Another key idea put into practice is the *portfolio* approach, in which algorithms are seeded with a combination of hyper-parameters across the different threads and consequently reduce the current coloring size faster.

2.3 Sharing Information

Throughout this paper, there are references to the term sharing information. It is important to discuss what that precisely means. In this context, it represents the ability for algorithm threads to share what the lowest k determined so far is. In addition, they are capable of sharing information about what moves they have made. This idea takes the form of a statistic matrix, as suggested by Li [Li17]. It was implemented for PartialCol algorithm and Tabucol algorithm. It was not performed for the Metropolis algorithm. AntCol implements its own information sharing scheme via the pheromone trails.

3 Existing Graph Coloring Problem Approaches

In this section of the paper, we will reference some existing techniques and heuristics for graph coloring as they are vital to understanding the hypothesis.

3.1 Tabucol & PartialCol

Tabucol is an algorithm proposed by Hertz and de Werra [HW87]. The implementation here is the modified version by Galinier and Hao [**TabuCol**] and discussed in the *Guide to Graph Coloring* textbook [Lew15]. The algorithm fundamentally revolves around moves, i.e., shifting a coloring to a neighboring coloring by only *moving* the vertex causing the most conflicts to another color class. The algorithm attempts to prevent cyclic issues by making certain moves *taboo*, hence the name. The rate at which moves become accessible again is dependent upon algorithm parameters. Another algorithm that was used in this paper is called PartialCol. PartialCol itself is very similar to the tabu approaches, but organizes vertices that cannot be colored without conflict.

3.2 Metropolis Algorithm

The metropolis algorithm simple algorithm in which we select vertices at random, recolor them, analyze the impact of the coloring change using a cost (or potential) function H, which denotes how many forbidden edges exist with the current coloring. Color transitioning moves are approved or rejected the with a probability based on the cost of the move. This algorithm utilizes a temperature coefficient utilize to adapt the likelihood of bad moves to be accepted, in the effort of avoiding solutions to dwell on local minimums.

3.3 Antcol

Antcol is an algorithm that was proposed by Thompson and Dowsland [Dow07]. It is an algorithm that utilizes both global and local search operators inspired by how ants determine optimal paths between their colonies and food sources. Ants tend to wander around randomly until a food source is found, where they then leave a pheromone trail for other ants to pick up on. This is key to Antcol: the algorithm attempts to make a handful of individual solutions (i.e."ants") made in a non-deterministic manner. Then, each individual solution contributes to a global "trail" solution that subsequent "ants" then use as better heuristics. If features identified in each solution lead to better solutions, subsequent ants will continue to use those features using a matrix analogous to an ant's pheromone trail. Over time, inadequate solutions will disappear with an evaporation rate, giving stronger features more prominence.

4 Cooperative Extensions to the Graph Coloring Problem

This paper is comprised of several experiments:

- Assessing whether one sees improved lower bounds on χ by running concurrent metropolis algorithms sharing information.
- Assessing the performance improvements of of PartialCol and Tabucol sharing lower bounds and partial colorings in identifying lower χ.
- Assessing cooperative Tabucol approaches with other algorithms - that is to say extending the results of Li's use of the statistic matrix. [Li17] The statistic matrix in this scenario is just a means to feature moves that are made less frequently.

– Assessing Antcol and tuning hyperparameters in both non-cooperative and cooperative modes. Notably, tuning the evaporation rate and number of ants used to approximate a solution.

The source code is available in this project. It was written using a standard Java console application with Gradle for managing dependencies. The most important of those dependencies is JGraphT, an open source library for handling graphs. The graph format that was used is the popular DIMACS format. Graphs used in this project are listed in the appendix in Table 1.

4.1 Cooperative Metropolis Algorithm

The first experiment was assessing whether results for the Metropolis algorithm could be improved using multiple threads running different parameters and sharing information about what the lowest k currently available is. Results from this experiment are captured in the appendix in Figure 1. Noticeable difference in the minimum coloring achievable can be observed. Still, this approach did not yield any significant advances over using different algorithms such as Tabucol and PartialCol, whose results are discussed below.

4.2 Cooperative Tabucol Algorithm

The results in Figure 2 were intended to provide discussion with reference towards [Li17]. It was found that the portfolio approach decreased the error on our optimal coloring determinations. However, the statistic matrix approach did not yield significant benefits (and appeared, in fact, detrimental). Possible explanations are listed in the PartialCol section.

4.3 Cooperative PartialCol Algorithm

The third experiment performed was analyzing the use of cooperative approaches and the statistic matrix for PartialCol. The results echoed those from Tabucol in

that the statistic matrix did not provide any inherent benefit over the standard cooperative approaches or the multithreaded simple portfolio approach. The experiments performed showed a hinderance in performance instead. See the average error in Figure 3. One possibility is that the implementation by Li had a different synchronization method. The implementation for this project was synchronized in access, but not in terms of iterations. The approach presented simply utilizes a portfolio approach to minimize the odds getting stuck in local minima, as the cooperative approach did not provide too much overhead.

4.4 Antcol

The final experiment performed was assessing Antcol and its hyperparameters: notably, if tuning the number of ants or the evaporation rate of the pheromone matrix would increase or decrease the optimal approximation. The average error for the hyperparameters that were included in [Lew15] was approximately 0.18. This implementation included an evaporation rate of 0.75. When adjusting this to be higher or lower, we can see based on the graph in Figure 4 that lower values of evaporation tended to cater towards lower errors. Specifically, performance for smaller known-colorings proved to reach the optimal coloring. When we experimented using a shared dataset across multiple instances of Antcol, we noticed that the error increased in size. This is likely due to the somewhat random, sequential nature of how Antcol works. Because each individual contribution ends up contributing to the global matrix for each instance, it is possible that a solution set may differ than that of another. While the difference in average error is small, it's worth noting that both increasing the number of ants and lowering evaporation rates improved optimality at an expense in computational effort.

5 Conclusion

In this report we examined several different cooperative approaches to various local search algorithms. In addition, we leveraged the novel approach of Li and extended it to the PartialCol algorithm. For the evaluated graphs, no noticeable advantages were identified, and distinct disadvantages in terms of average error were observed instead. For all the algorithms, however, the portfolio approach seemed to provide improvements. Several other approaches were evaluated (for instance, a simple evolutionary algorithm), but it was found that they did not performed well enough to be viable).

6 Appendix A. Graph Coloring Algorithms Evaluation Results

The following list is the collection of graphs used to evaluate the algorithms described in this paper. "Optimal is Approximate" denotes whether we know a lower bound for said graph.

Optimal is Approximate	Optimal Coloring	Graph Name	Vertices	Edges
F	5	dsjc125.1	125	736
F	17	dsjc125.5	125	3891
F	44	dsjc125.9	125	6961
F	8	dsjc250.1	250	3218
T	28	dsjc250.5	250	15668
F	72	dsjc250.9	250	27897
F	26	flat300_26_0	300	21633
F	28	flat300_28_0	300	21695
F	10	jean	80	254
F	15	le450_15b	450	8169

Optimal is Approximate	Optimal Coloring	Graph Name	Vertices	Edges
F	15	le450_15d	450	16750
F	5	le450_5a	450	5714
F	5	le450_5b	450	5734
F	20	r1000.1	1000	14378
F	5	r125.1	125	209
F	36	r125.5	125	3838
F	8	r250.1	250	867
F	64	r250.1c	250	30227
T	28	r250.5	250	14849
F	14	school1_nsh	352	14612
F	14	school1	385	19095

Table 1: Collection of graphs used in the project.

Note that multithreaded experiments were run with 8 threads. The notation $a[x, y]$ means that parameter a was chosen with arbitrary values between x and y. In lieu of α and β, a and b were used.

Figures 1 through 4 show accuracy performance for the different algorithms evaluated.

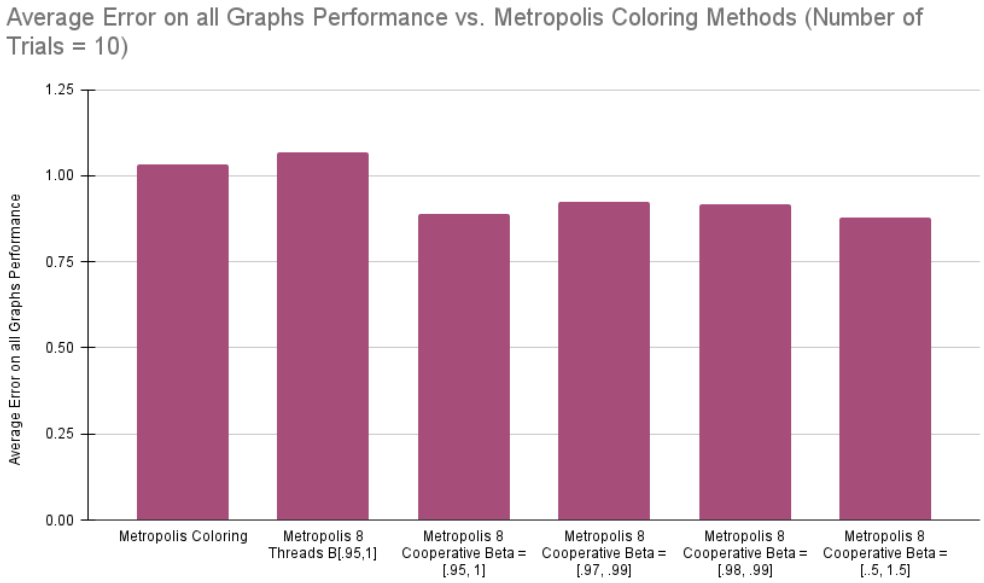

Fig. 1. Metropolis Results

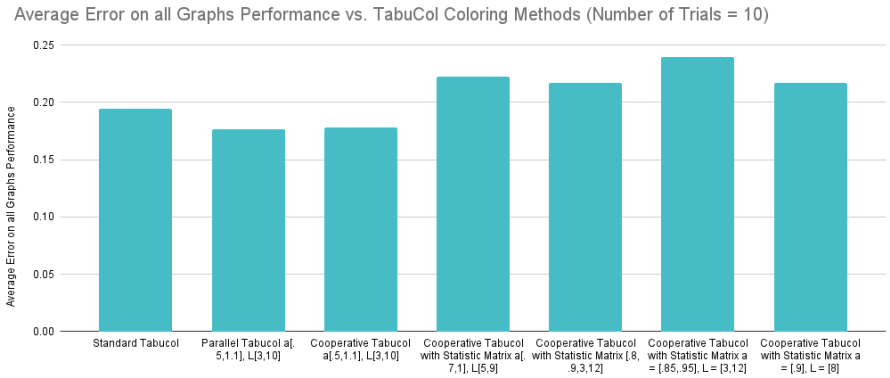

Fig. 2. Tabucol Results

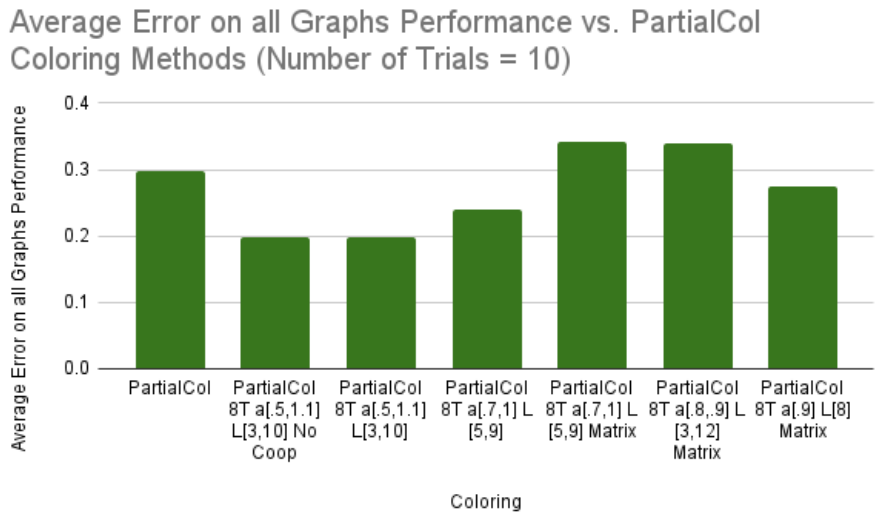

Fig. 3. PartialCol Results

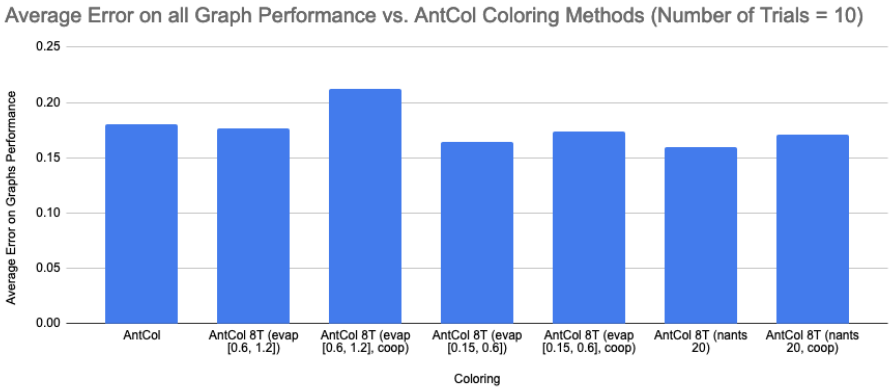

Fig. 4. AntCol Results

7 Appendix B. Coloring Initialization Comparison

[h] While reviewing literature for the experiments presented in this paper, it was observed that certain authors would use different initial values for their coloring. As a side experiment, a brief comparison was made between a few of the initialization options presented. The initialization schemes compared were denominated simple-ordered, random, and unassigned-neighbor strategies. The simple-ordered strategy is the general case of the initialization presented in the Central Algorithm section. Vertex are assigned a coloring in an orderly fashion from 1 to k, and colors are cycled over until all vertex have a color. The random strategy is self-explanatory, each vertex gets assigned a color from 1 to k randomly with a uniform distribution. The last strategy, unassigned-neighbor, tries to exploit lightly connected graphs, by initializing all vertices to an *uncolored* state. This strategy first assigns a root vertex to color 1 and then traverses the graph, coloring each uncolored vertex found with color 1 should it not share an edge with a vertex of that color. Once the traversal is completed, a vertex from the uncolored vertex set is colored with color 2, and a new traversal begins, comparing neighbors to new color as the metric for coloring vertices. This process is repeated for all k colors. The scenario that some vertices are left uncolored after the strategy is completed is high, particularly on heavily connected graphs, and the remaining uncolored vertices will be assigned randomly.

For the comparison experiment, it was decided to use the Metropolis algorithm due to the simplicity of the implementation of the benchmark for a proof-of-concept test. Results are shown in 5.

While results vary slightly, it was concluded that no one seeding strategy yielded significantly better results than the rest, so a single strategy was used for the evaluation of algorithms in the main portion of the project.

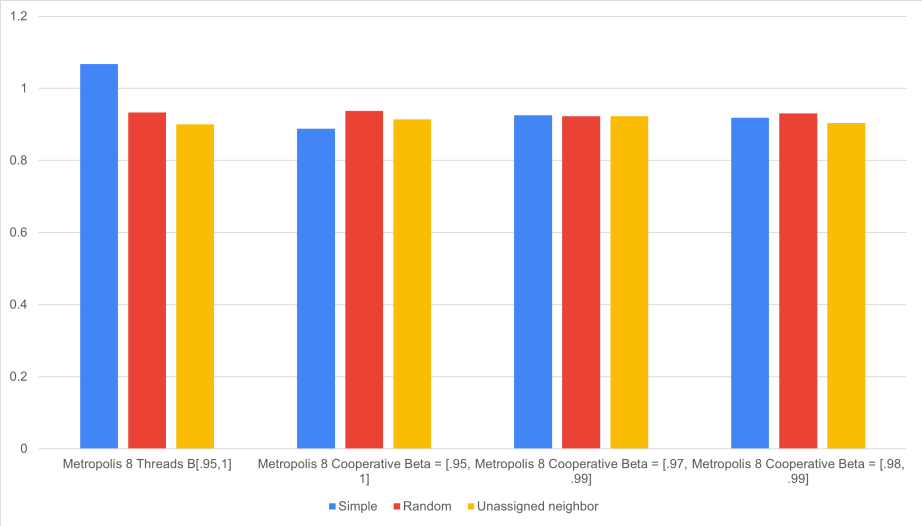

Fig. 5. Average error for different graph initialization strategies

References

[Dow07] Dowsland Kathryn; Thompson, J. "n improved ant colony optimisation heuristic for graph colouring". en. In: (2007). DOI: 10.1016/j.dam.2007.03.025. URL: https://www.researchgate.net/publication/222313866_An_improved_ant_colony_optimisation_heuristic_for_graph_colouring.

[HW87] Hertz, A. and Werra, D. "Werra, D.: Using Tabu Search Techniques for Graph Coloring. Computing 39, 345-351". In: *Computing* 39 (Dec. 1987). DOI: 10.1007/BF02239976.

[Lew15] Lewis, R. R. *A Guide to Graph Colouring: Algorithms and Applications*. 1st. Springer Publishing Company, Incorporated, 2015. ISBN: 3319257285.

[Li17] Li, G. "Solving the Graph Coloring Problem with Cooperative Local Search". en. In: (2017). DOI: 10.5445/IR/1000083192. URL: https://publikationen.bibliothek.kit.edu/1000083192.