# An Analysis of Solution Properties of the Graph Coloring Problem

*Jean-Philippe Hamiez*[*] *Jin-Kao Hao*[†]

[*] LGI2P, École des Mines d'Alès, EERIE
Parc Scientifique Georges Besse – 30035 Nîmes Cedex 01 (France)
Email: hamiez@site-eerie.ema.fr

[†] LERIA, Université d'Angers
2, Bd. Lavoisier – 49045 Angers Cedex 01 (France)
Email: Jin-Kao.Hao@univ-angers.fr

## 1 Introduction

This paper presents an analysis of solutions of the *Graph Coloring Problem* (GCP). Given an undirected graph $G(V, E)$ with a vertex set $V$ and an edge set $E$, the goal of GCP is to find a color assignment to every vertex in $V$ such that any pair of adjacent (or connected) vertices receive different colors, and the total number of colors required for the feasible color assignment be minimized; the smallest color size corresponds to the *chromatic number* $\chi(G)$ of graph $G$. Many practical problems, such as timetable construction [19] or frequency assignment [11], can be mapped into a GCP[1]. Graph coloring is also a classic constraint satisfaction problem with applications to many problems in Artificial Intelligence.

GCP in an arbitrary graph is a well-known *NP-complete* problem [12] and only small problem instances can be solved exactly within a reasonable amount of time in the general case [7]. It is also hard even to approximate the chromatic number of a graph. In [20], it is proved that for some $\epsilon > 0$, approximating the chromatic number within a factor of $n^{\epsilon}$ is NP-hard. Indeed, one of the best known approximation algorithm [14] provides an extremely poor performance guarantee[2] of $O(n(\log\log n)^2/(\log n)^3)$ for a graph with $n$ vertices.

The reported heuristics to try to solve the graph coloring problem range from greedy constructive methods, such as DSATUR [1] and the Recursive Largest First algorithm [19], to sophisticated hybrid strategies like HCA [6, 10] and those proposed in [21]. The last ones are among the most efficient approaches for GCP. We may also mention local search meta-heuristics, e.g., simulated annealing [17] or tabu search [15, 8, 5], and neural network attempts [16], ... Reference [9] gives new lower bounds for some well-studied DIMACS graphs [18].

Although most authors in the graph coloring community try to explain why their approach gives good results on some graphs and poor ones on others (mainly by analyzing the behavior of the algorithm according to different options and settings), few studies are available concerning analysis of properties of GCP solutions. Moreover, to our knowledge, no algorithm exploiting such properties exists for the GCP. In our opinion, one interesting study could be to look for *topological* properties of GCP solutions. In this paper, we are concerned by studying a particular property that may be called *frozen same sets* (FS2 for short) which are sets of vertices that are always in the same color class of a solution.

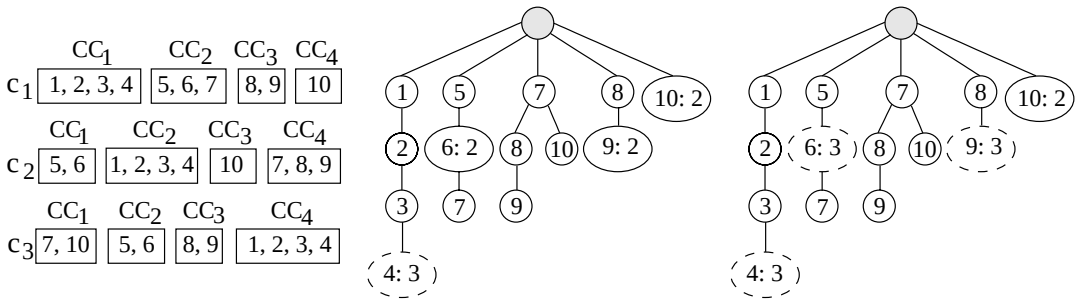
[1] See [3], for instance, for an extensive list of other references on GCP applications.
[2] The performance guarantee is the maximum ratio, taken over all inputs, of the color size over the chromatic number.

More formally, a set of vertices $\{v_1, \ldots, v_m\}$ is a FS2 if $\forall c \in \mathbf{C}$, $\mathrm{col}_c[v_1] = \cdots = \mathrm{col}_c[v_m]$, with $\mathbf{C}$ the set of all legal $k$-colorings[3] of a graph $G$ and $\mathrm{col}_c[u]$ the color of vertex $u$ in solution $c$. One recent study we found about topological analysis of GCP solutions is due to J. Culberson and I.P. Gent [4]. They define *frozen same pairs* as pairs of vertices that are always in the same color class, formally, $(u, v)$ is a frozen same pair if $\forall c \in \mathbf{C}$, $\mathrm{col}_c[u] = \mathrm{col}_c[v]$. Note that this definition is a particular case of frozen same sets. They report results for 4-coloring random graphs, 3-coloring random graphs and 3-coloring triangle-free graphs using a backtracking program. One of their most interesting conclusions is that, at phase transition[4], the coloring hardness relies on the large size of *critical subgraphs*[5], which are not easily checkable to confirm uncolorability quickly; the size of critical subgraphs evolving with the number of vertices. Furthermore, existence of *critical edges*[6], that are difficult to check explicitly, makes the search of solutions harder, "suggesting that hardness at phase transitions is an algorithm independent property".

Topological analysis may help explain the behaviour of some algorithms on a particular graph. Such an analyse can also be used to develop new algorithms relying on solutions properties. The goal of our study is to present such an analysis for GCP solutions by means of frozen same sets. The paper begins by describing the analysis schemes (section 2). Experimental results on most of well-studied DIMACS graphs [18] are presented in section 3, followed by some concluding remarks.

## 2 Analysis schemes

A diversity measure on all the **configurations** of a population is often used in population-based algorithms as a decision criterion to stop the search: when diversity evolves no more or is sufficiently low, the process stops. Frozen same sets can be used with non-deterministic algorithms running on a single configuration to measure diversity of **solutions** found within multiple executions. We experimented this approach with a two-level analysis which is practically implemented using intersection of sets and structurally represented by binary trees. This section presents the methodology we used to implement the search for FS2.




(a) Three sample solutions (b) Illustration of surface analysis (c) Illustration of depth analysis

Figure 1: Illustration of global and depth analysis

In a first global scheme, the *surface analysis*, we only look for *global FS2* such that each set contains all the vertices of a color class. In other words, the set $s_1$ of all the vertices of the color class $\mathrm{CC}_1$ from solution $\mathrm{c}_1$ is a global FS2 if $\forall\ \mathrm{c}_p \in \mathbf{C}, p > 1, \exists\ \mathrm{CC}_i \in \mathrm{c}_p / s_i = s_1$. Figure 1(b) gives a representation as a tree (nodes are vertices) of a surface analysis on three sample solutions (figure 1(a)), each one being

[3] $k$-coloring is the decision problem associated with the optimization GCP checking whether it exists a coloring with $k$ colors or not.

[4] Phase transition separates overconstrained (probability of finding a solution near 0) from underconstrained instances (probability near 1), see, e.g., [2].

[5] A critical subgraph is a subgraph which is uncolorable, but which becomes colorable if any edge is removed from it.

[6] The critical set of edges is the set of edges that occur in every critical subgraph.

composed of four color classes (numbers map vertices). Let *final nodes* be the particular nodes which contain additional information, e.g. *6: 2*; the first number is a vertex and the second shows the number of times the set composed of vertices which belong to the path from root to a final node mapped entirely a color class within all solutions. For instance, *6: 2* means "the color class $\{5, 6\}$ appeared two times". Thus, global FS2 can directly be retrieved by following paths from root to all final nodes which have an occurrence number equal to the number of solutions studied. There is only one global FS2 in our illustration: $\{1, 2, 3, 4\}$.

The previous analysis scheme is sometimes enough to identify frozen same sets; we will see in the section of results that many DIMACS benchmark graphs do contain such sets. But, one main disadvantage of this **global** analysis is that it is absolutely blind of *frozen same subsets* (FS3 for short), which can be defined as subsets of vertices of a color class in all solutions. Formally, $\{v_1, \ldots, v_m\}$ is a FS3 if $\forall$ $c_p \in \mathbf{C}, \exists$ $CC_i \in c_p / \{v_1, \ldots, v_m\} \subseteq s_i$. For instance, in figure 1(b), {5, 6} is a FS3 since it appears in $CC_2$ in $c_1$, $CC_1$ in $c_2$, and $CC_2$ in $c_3$, but the surface analysis cannot retrieve it. Figure 1(c) gives an illustration of this *depth analysis*. Here again, frozen same subsets can directly be extracted by following paths from root to all final nodes which have an occurrence number equal to the number of solutions studied. Here, there are three FS3: {1, 2, 3, 4}, {5, 6}, and {8, 9}.

# 3 Computational results

To generate solutions to be studied, we use a generic tabu search based algorithm[7] designed for various coloring problems.

## 3.1 Generic Tabu Search

*Generic Tabu Search* (GTS for short) is a general algorithm to solve the GCP and also the T-coloring problem which are special cases of set T-coloring. We just recall here the main components of GTS (see [5] for full description of the method):

- *Initial configuration.* GTS uses a *DSATUR-based* greedy algorithm [1] to generate initial configurations;
- *Configuration re-generation.* The re-generation aims at producing a ($k$-1)-coloring from a $k$-coloring. The nodes in $CC_k$ are given a new color from $[1..k-1]$ minimizing the conflicts over the graph;
- *Searching for proper coloring.* Beginning from a conflicting re-generated configuration, GTS iteratively makes *1-moves*, changing the color of a vertex to another one, until achieving a proper coloring. Each time a solution is found with $k$ colors, a new configuration is re-generated from this solution with $k-1$ colors and the search process starts again. The algorithm stops when an optimal or a best known coloring is obtained or when a maximum number of moves have been carried out without finding a solution.

## 3.2 Results

In this section, we give, for most of the DIMACS benchmark graphs [18], the results of *surface analysis* and *depth analysis* (section 2). We specify for each graph studied if the set of solutions found by GTSD, which stands for our implementation of GTS[8], contain *frozen same sets*, or *frozen same subsets* if there is no FS2.

[7] See [13] for details about the tabu search metaheuristic.

[8] GTSD is coded in C (CC compiler with –O5 option) and executed on a Sun Ultra 1 (256 RAM, 143 MHz).

Columns 1-5 in Table 1 show for each graph studied, respectively, its name, number of vertices and edges, its chromatic number (or its best known lower bound when unknown), and the best coloring found by GTSD after 10 millions moves. The last column shows the results of the analysis, in percentage of solutions which contain at least one FS2 or one FS3. In other words, entries with a "100%" pattern in the analysis field mean that FS2 (or FS3) has been found, otherwise (entries $< 100\%$), we give the number of solutions (in percentage) which contain a FS2 (or FS3). Analyses were performed over five to ten solutions generated by GTSD.

| Graph name | $\|V\|$ | $\|E\|$ | $\chi$ | $k$ | Analysis (%) |
|---|---|---|---|---|---|
| school1 | 385 | 19095 | 14 | 14 | 100 (FS2) |
| school1_nsh | 352 | 14612 | 14 | 14 | 50 (FS2) |
| dsjr500.1 | 500 | 3555 | 12 | 12 | 20 (FS2) |
| dsjr500.1c | 500 | 121275 | 84 | 85 | 30 (FS2) |
| r125.1 | 125 | 209 | 5 | 5 | 20 (FS2) |
| r125.1c | 125 | 7501 | 46 | 46 | 100 (FS2) |
| r125.5 | 125 | 3838 | 36 | 36 | 80 (FS3) |
| r250.1 | 250 | 867 | 8 | 8 | 50 (FS2) |
| r250.1c | 250 | 30227 | 64 | 64 | 100 (FS2) |
| r250.5 | 250 | 14849 | 65 | 66 | 100 (FS2) |
| r1000.1 | 1000 | 14378 | 20 | 20 | 30 (FS2) |
| r1000.1c | 1000 | 485090 | $\geq 90$ | 98 | 100 (FS2) |
| le450_15a | 450 | 8168 | 15 | 15 | 0 (FS2) |
| le450_15b | 450 | 8169 | 15 | 15 | 0 (FS2) |
| le450_15c | 450 | 16680 | 15 | 15 | 100 (FS2) |
| le450_15d | 450 | 16750 | 15 | 15 | 40 (FS2) |
| mulsol.i.1 | 197 | 3925 | 49 | 49 | 100 (FS2) |
| flat300_20_0 | 300 | 21375 | 20 | 20 | 100 (FS2) |
| flat300_26_0 | 300 | 21633 | 26 | 26 | 100 (FS2) |
| flat300_28_0 | 300 | 21695 | 28 | 32 | 100 (FS3) |
| flat1000_50_0 | 1000 | 245000 | 50 | 50 | 100 (FS2) |
| flat1000_60_0 | 1000 | 245830 | 60 | 60 | 100 (FS2) |
| dsjc125.5 | 125 | 3891 | $\geq 10$ | 17 | 100 (FS2) |
| dsjc250.5 | 250 | 15668 | $\geq 11$ | 28 | 100 (FS3) |
| dsjc500.5 | 500 | 62624 | $\geq 13$ | 50 | 50 (FS3) |

Table 1: Some results of analysis on DIMACS benchmark graphs using GTSD

From Table 1 we can make several remarks. First, most of the studied graphs have frozen same sets. So, FS2 may certainly rely on topological properties of the graphs. Recall that this result has already been observed on random graphs by J. Culberson and I.P. Gent in [4]. Graph analysis on solutions generated with another search algorithm may give more insight to this observation. For a few graphs, when no FS2 is found, or with a low percentage, frozen same subsets are sometimes identified. Nevertheless, ten graphs have less than 60% of FS2 (or FS3), meaning that solutions are sufficiently different. Especially, the le450_15a and le450_15b graphs seem to have a large number of different solutions since we do not find frozen same set. For flat graphs, except flat300_28_0 since GTSD finds no optimal solution, the analysis reveals that all solutions were identical. This can be due to some structural properties of these graphs and we believe that other flat graphs, optimally colored, may reached the same property. Furthermore, one may wonder if such graphs have only one solution (equivalent solutions par permutation of colors are excluded). More studies on graphs structures could help us understanding why these graphs contain frozen same sets and subsets.

Table 2 gives results of GTSA, a modified GTSD which uses the results of the analysis of solutions with $k+\epsilon$ colors generated by GTSD (within 10 millions moves) to build the initial configuration, with $k$ colors, as follow: first, FS2 (or FS3) identified in five to ten solutions with $k+\epsilon$ colors are sorted in

decreasing order of appearance percentage; then, color classes are filled one by one with these sorted sets, beginning with the highest representative set and ending when all the $k$ color classes are initialized or when no more representative set is available; finally, free vertices are given a color in $[1..k]$ such that it minimizes the conflicts over the graph. For GTSA, times and moves entries include the generation of solutions with $k + \epsilon$ colors, the analysis process, and the search for a proper $k$-coloring (within 10 millions moves). All reported results are averaged over five to ten runs.

| Graph name | $\chi$ | GTSD | | | GTSA | | | |
|---|---|---|---|---|---|---|---|---|
| | | $k$ | Time (sec.) | Moves | $k$ | Time (sec.) | Moves | $k+\epsilon$ |
| dsjr500.1c | 84 | 85 | 278 | 88972 | 85 | 13 | 10643 | 87 and 86 |
| dsjr500.5 | 122 | 127 | 3 | 376 | *124* | 4009 | 13930549 | 128 |
| r125.5 | 36 | 36 | 2 | 25736 | 36 | $< 1$ | 406 | 37 |
| r250.1c | 64 | 64 | $< 1$ | 252 | 64 | $< 1$ | 54 | 65 |
| le450_15a | 15 | 15 | 13 | 84848 | 15 | 2 | 15228 | 17 |
| le450_15c | 15 | 15 | 71 | 412719 | 15 | 6 | 31460 | 16 |
| le450_15d | 15 | 15 | 14 | 75269 | 15 | 6 | 29786 | 16 |
| dsjc125.5 | $\geq 10$ | 17 | 15 | 242968 | 17 | $< 1$ | 9081 | 18 |
| dsjc500.5 | $\geq 13$ | 50 | 5585 | 16532498 | 50 | 282 | 1003149 | 52 |

Table 2: Using solution analysis to improve GTS

The most important remark regarding results in Table 2 is that solutions analysis speeds up the search for a solution. Indeed, all reported results of GTSA are better than those of GTSD. For some graphs, the benefit factor (considering the number of moves) is more than ten! GTSA is even able to find much better coloring for the dsjr500.5 graph. For the le450_15c and le450_15d graphs, GTSA finds a proper and optimal coloring immediately in building the initial configuration, so, in reading the results of the analysis. This last remark suggests that solutions with $k + \epsilon$ colors share common informations with solutions at $k$ colors.

# 4 Conclusion

In this paper, we have studied two properties of solutions for the graph coloring problem, which we called *frozen same sets* and *frozen same subsets*. We have introduced an implementation technique based on a tree structure. This technique allows us to avoid the permutation problem of coloring solutions. Analyses have been reported on most of the well-studied DIMACS graphs. We observed, according to the percentage of FS2 in a set of solutions, that a large number of these graphs contain frozen same sets, suggesting that their existence relies more on eventual topological properties of the graphs than on the algorithm used to solve the GCP. The analysis reveals that some graphs have multiple different solutions (i.e., entropy near 1) while others share common coloring informations. Especially, the flat graphs seem to lead to identical solutions (entropy equals 0).

Solution analysis has also been used to improve a *generic tabu search* [5] designed for various graph coloring problems. The main improvement has been observed on the speed: we found solutions in less number of moves. Indeed, some solutions are then generated more than ten times quicker. A better quality (i.e., a lower color size) result have also been achieved for one graph. This underlines that the initial configuration of a local search algorithm may certainly be of great influence on its results.

More generally, this study has shown that a better knowledge on topological properties of instances can reduce the algorithm effort to find solutions of a combinatorial optimisation problem and we believe that such an approach may be useful for other constrained problems.

## References

[1] D. Brélaz. New methods to color the vertices of a graph. *Commun. ACM*, 22(4):251–256, 1979.

[2] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In J. Mylopoulos and R. Reiter, editors, *Proc. of IJCAI-91*, pages 331–337, Morgan Kaufmann, 1991.

[3] J. Culberson. Bibliography on graph coloring. Available on the world wide web at *http://liinwww.ira.uka.de/bibliography/Theory/graph.coloring.html*, 2000.

[4] J. Culberson and I.P. Gent. Well out of reach: Why hard problems are hard. Research Report APES-13-1999.

[5] R. Dorne and J.K. Hao. Tabu search for graph coloring, T-colorings and set T-colorings. In S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors, *Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization*, chapter 6, pages 77–92, Kluwer Academic Publishers, 1998.

[6] R. Dorne and J.K. Hao. A new genetic local search algorithm for graph coloring. *Lecture Notes in Computer Science* 1498:745–754, Springer, 1998.

[7] N. Dubois and D. de Werra. EPCOT: an efficient procedure for coloring optimally with tabu search. *Computers Math. Appl.*, 25(10/11):35–45, 1993.

[8] J.A. Ferland and C. Fleurent. Object-Oriented Implementation of Heuristic Search Methods for Graph Coloring, Maximum Clique, and Satisfiability. In [18], pages 619–652, 1996.

[9] N. Funabiki and T. Higashino. A Minimal-State Processing Search Algorithm for Graph Colorings Problems. *IEICE Trans. Fundamentals*, volume E83–A(7):1420–1430, 2000.

[10] P. Galinier and J.K. Hao. Hybrid evolutionary algorithms for graph coloring. *J. Combin. Optim.*, 3(4):379–397, 1999.

[11] A. Gamst. Some lower bounds for a class of frequency assignment problems. *IEEE Trans. Veh. Tech.*, 35(1):8–14, 1986.

[12] M.R. Garey and D.S. Johnson. Computers and intractability: A guide to theory of NP-completness. W.H. Freeman and Company, New York, 1979.

[13] F. Glover and M. Laguna. Tabu Search. Kluwer Academic Publishers, 1997.

[14] M.M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Inform. Process. Lett.*, 45:19–23, 1993.

[15] A. Hertz and D. de Werra. Using Tabu Search Techniques for Graph Coloring. *Computing*, 39:345–351, 1987.

[16] A. Jagota. An adaptive, multiple restarts neural network algorithm for graph coloring. *European J. Oper. Res.*, 93:257–270, 1996.

[17] D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation. II. Graph coloring and number partitioning. *Oper. Res.*, 39(3):378–406, 1991.

[18] D.S. Johnson and M.A. Trick., editors. *Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, 1993*. Volume 26 of *DIMACS Series in Discr. Math. and Theoretical Comput. Sci.*, American Math. Soc., 1996.

[19] F.T. Leighton. A graph coloring algorithm for large scheduling problems. *J. Res. Nat. Bur. Stand.*, 84:489–506, 1979.

[20] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Proc. 25th Annual ACM Symp. Theory of Comput.*, pages 286–293, 1993.

[21] C.A. Morgenstern. Distributed coloration neighborhood search. In [18], pages 335–357, 1996.