An Analysis of Solution Properties of the Graph Coloring Problem

Jean-Philippe Hamiez (jean-philippe.hamiez@ema.fr)
LGI2P, École des Mines d'Alès (site EERIE)
Parc Scientifique Georges Besse
F-30035 Nîmes Cedex 01, France

Jin-Kao Hao (jin-kao.hao@univ-angers.fr)
LERIA, Université d'Angers (U.F.R. des Sciences)
2, Bd. Lavoisier
F-49045 Angers Cedex 01, France

Abstract. This paper is interested in analyzing solution properties of the Graph Coloring Problem (GCP). For this purpose, we introduce a property based on the notion of *representative sets* which are sets of vertices that are always colored the same in a set of solutions. Experimental results on well-studied DIMACS graphs show that many of them contain such sets and give interesting information about the diversity of the solutions. We also show how such an analysis may be used to improve a tabu search algorithm.

Keywords: graph coloring, solution analysis, representative sets, tabu search

Abbreviations: GCP – Graph coloring problem; RS – Representative set; CRS – Complete representative set; PRS – Partial representative set; FREQ – Frequency; SA_STRUCT – Surface analysis structure; DA_STRUCT – Depth analysis structure; GTS – Generic tabu search; TL – Tabu list; GTSD – GTS with DSATUR-based initialization; GTSA – GTS with initialization based on analysis

1. Introduction

Given an undirected graph $G = (V, E)$ with a vertex set V and an edge set E, the goal of the graph coloring problem (GCP for short) 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 (*assignment approach*). The smallest color size corresponds to the *chromatic number* $\chi(G)$ of graph G. The GCP can also be defined more formally as follow: partition V into a minimal number of k subsets $V_1, \ldots, V_k$ such that $\forall u \in V_i (u \in V, 1 \leq i \leq k)$, $\nexists v \in V_i, v \in V / (u, v) \in E$ (*partition approach*).

Many practical problems, such as timetable construction (Leighton, 1979) or frequency assignment (Gamst, 1986), can be mapped into a GCP. Graph coloring is also a classic constraint satisfaction problem (Tsang, 1993) with applications to many problems in Artificial Intelligence.

GCP in an arbitrary graph is a well-known *NP-complete* problem (Garey and Johnson, 1979) and only small problem instances can be solved exactly within a reasonable amount of time in the general case (Dubois and de Werra, 1993). It is also hard even to approximate the chromatic number of a graph. In Lund and Yannakakis (1993), it is proved that for some $\epsilon > 0$, approximating the chromatic number within a factor of $|V|^\epsilon$ is NP-hard. Indeed, one of the best known approximation algorithm (Halldórsson, 1993) provides an extremely poor performance guarantee[1] of $O(|V|(\log\log|V|)^2/(\log|V|)^3)$.

The reported heuristics for the solution of the graph coloring problem range from greedy constructive methods, such as DSATUR (Brélaz, 1979) and the Recursive Largest First algorithm (Leighton, 1979), to sophisticated hybrid strategies like HCA (Dorne and Hao, 1998(b); Galinier and Hao, 1999) and those proposed by Morgenstern (1996) or Funabiki and Higashino (2000). The last ones are among the most efficient approaches for GCP. We may also mention local search meta-heuristics, e.g., simulated annealing (Johnson et al., 1991) or tabu search (Hertz and de Werra, 1987; Ferland and Fleurent, 1996(b); Dorne and Hao, 1998(a)), pure genetic algorithms (Davis, 1991), neural network attempts (Jagota, 1996), and scatter search (Hamiez and Hao, 2002). One can find more methods in Johnson and Trick (1996).

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.

Analysis of solution properties may help explain the behaviour of some algorithms on a particular graph. Such an analysis can also be used to develop new algorithms relying on such properties. In this paper, we are concerned with studying a particular property that may be called *representative sets* (RS for short). Informally, a RS for a graph is composed of a set of vertices shared by its solutions. More formally, given a set C_k of legal k-colorings (solutions with exactly k colors) of a graph G, $\{v_1, \ldots, v_m\}(m > 1), v_j \in V(1 \leq j \leq |V|)$, is a RS if $\forall c_i \in C_k(1 \leq i \leq |C_k|), col_i(v_1) = \cdots = col_i(v_m)$, with $col_i(v_j)$ being the color of vertex v_j in solution c_i.

[1] The performance guarantee is the maximum ratio, taken over all inputs, of the color size over the chromatic number.

A RS may coincide exactly with a complete color class in which case the RS is called a *complete representative set* (CRS). It may also correspond simply to a subset of color classes in which case the RS is called *partial representative set* (PRS). The formal definitions of CRS and PRS are given later in Section 3. Like the notion of "backbone" for the satisfiability problem (Monasson et al., 1999), the RS reveals an invariant of solutions for the graph coloring problem.

The paper begins by recalling some existing analysis schemes (Sect. 2). Then, we describe the analysis schemes we propose (Sect. 3). Experimental results on a set of well-studied DIMACS graphs are presented in Sect. 4. A simple algorithm using analysis information is then introduced (Sect. 5), showing a better performance. Some concluding remarks and future studies are discussed in the last section.

2. Some existing analysis schemes

Before reporting any previous studies done in the field of graph coloring analysis, and so the usefulness of such research works, let us first recall a few notions and definitions used later on.

- A *critical subgraph* is a subgraph which is uncolorable with a given number k of colors but which becomes k-colorable if any edge is removed from it.
- The set of *critical edges* is the set of edges that occur in every critical subgraph.
- A $G_{n,p}$ *random graph* denotes a graph with n vertices, where p is the probability that there exists an edge between any pair of vertices.
- *Phase transition* separates over-constrained problems (probability of finding a solution near 0) from under-constrained instances (probability near 1). See, e.g., Cheeseman et al. (1991).

One recent study we found about topological analysis of GCP solutions is due to Culberson and Gent (1999). They define *frozen same pairs* as pairs of vertices that are always in the same color class, formally, $(u, v) \in V \times V, u \neq v$, is a frozen same pair if $\forall c_i \in C_k, col_i(u) = col_i(v)$ where $col_i(u)$ is the color of vertex u in solution c_i. Note that this definition is a particular case of representative sets introduced in this paper. They reported 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, the coloring hardness relies on the large size of *critical subgraphs*, 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*, that are difficult to check explicitly, makes the search of solutions harder, "suggesting that hardness at phase transitions is an algorithm independent property". A similar study can be found in Galinier (1999).

Hertz et al. (1994) developed the first topological analysis for the k-coloring problem. They investigated in particular the number and distribution of *local optima* of the k-coloring problem. All local optima of small random graphs ($|V| \leq 20$) were enumerated, up to a permutation of the colors, using a branch-and-bound procedure. The authors observed that the percentage of local optima which are global ones increases with the value of k. Then, the probability of reaching a local optimum which is a global one is extremely small for $k < \chi(G)$ (over-constrained problem, no valid k-coloring). Furthermore, when k is increasing from $\chi(G)$, "most local optima which are not global have a small number of conflicts". Finally, using additional statistical measures, they also explain the performance of a tabu algorithm (Glover and Laguna, 1997) according to the evolution of k around $\chi(G)$. If k is a few units above $\chi(G)$, tabu search seems effective since the landscape contains few *valleys* (or *plateaus*). It produces poor results when $k \leq \chi(G)$ since a small proportion of local optima are global ones.

Yokoo (1997) also analyzed the search space landscape of the 3-coloring problem. Analyses were performed using a descent algorithm[2] (with restart) over small random instances. He reported results by varying the edge density ($|E|/|V|$): probability of satisfiability, number of solutions, number of local minima, ... The main objective was to clarify the cause of a paradoxical phenomenon. For incomplete algorithms, problems are easier beyond the phase transition region (few solutions) than problems in the phase transition region (although there are more solutions). One of the main results showed that, while increasing the edge density, the number of local minima decreases due to plateaus of small size. Thus, more paths lead to solutions.

The performance of several hybrid algorithms developed for the GCP are examined by Ferland and Fleurent (1996(a)). The authors reported comparative results for $G_{n,p}$ random graphs and Leighton graphs (Leighton, 1979). They mainly study the effect of three string-based crossovers (*1-point*, *2-point* and *uniform* (Davis, 1991)) on the

[2] The algorithm moves iteratively to a better neighboring configuration, i.e., with fewer conflicts, until a local minimum is reached.

quality (number of conflicts) of solutions. For a 450-node Leighton graph, they also give an interesting analysis, from a topological point of view, of the effect of these crossovers on the entropy of a population. Recall that the entropy measure evaluates the diversity of a population[3]. Hence, it can be used to monitor the convergence of a population or to provide information on the behavior of population-based algorithms using different options and settings. They note that "the uniform crossover operator converges more slowly, but ultimately gives better results". In other words, the uniform crossover seems to insure a certain level of diversity.

Walsh (1999) studied graphs with a *small world* topology, i.e., graphs in which vertices are highly clustered yet the path length between them is small (Watts and Strogatz, 1998). He demonstrated that register allocation graphs have a small world topology and observed similar results with other DIMACS benchmark graphs. He also used a DSATUR-based backtracking algorithm (Brélaz, 1979) to color graphs with a small world topology generated according to the model proposed by Watts and Strogatz (1998)[4]. Results showed that the cost of solving such graphs has a heavy-tailed distribution. He suspected then that "problems with a small world topology can be difficult to color since local decisions quickly propagate globally". Finally, to combat this heavy-tailed distribution, he tried the strategy of randomization and restart (Gomes et al., 1998). As in other studies, this technique appeared particularly effective to eliminate these heavy tails. See Walsh (2001), for extended results on these graphs. Analysis are also reported for other non-uniform graphs generated using alternative models, namely, *ultrametric* graphs (Hogg, 1996) and *power law* graphs (Barabási and Albert, 1999).

3. Extended analysis schemes

In this section, we present two analysis schemes based on the notion of representative sets. The first scheme is designed to extract complete representative sets (CRS), i.e., complete color classes shared by a set of given k-colorings. The second one aims at extracting partially shared representative sets (PRS) of color classes. These two schemes may be considered as two applications of the same idea at two different levels. To implement these schemes, we use extensively binary tree techniques.

[3] See Welsh (1988), e.g., for discussions about interesting properties of a measure based on entropy.

[4] Starting from a regular graph, randomly rewire each edge with probability p. As p increases from 0, the graph develops a small world topology.

3.1. Extracting complete representative sets: surface analysis

3.1.1. *Surface analysis: definition and example*

Recall that the vertices of a representative set (RS) $\{v_1, \ldots, v_m\}(m > 1), v_j \in V(1 \leq j \leq |V|)$ are always colored the same in a set of solutions, i.e., $\forall c_i \in C_k(1 \leq i \leq |C_k|), col_i(v_1) = \cdots = col_i(v_m)$, with C_k a set of legal k-colorings of a graph G and $col_i(v_j)$, the color of vertex v_j in solution c_i.

To be a complete representative set (CRS), the set $\{v_1, \ldots, v_m\}$ must coincide exactly with a color class for each of the given k-colorings in C_k. In other words, the color class $V_j^i \subseteq V(1 \leq j \leq k, 1 \leq i \leq |C_k|)$ from solution c_i is a complete RS if $\forall c_p \in C_k, p \neq i, \exists V_q^p \in c_p(V_q^p \subseteq V)/V_q^p = V_j^i$.

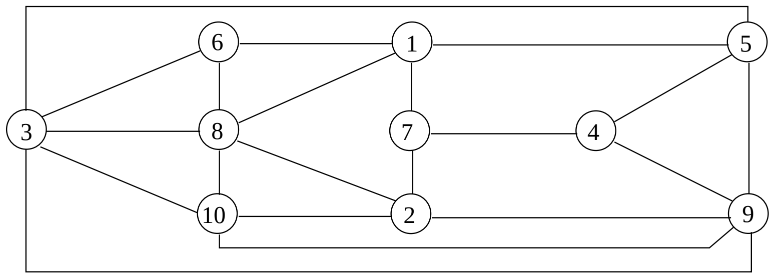

Figure 1. A sample input graph.

To explain the underlining idea, we will use the example shown in Fig. 2 (left) where the sample set $C_k(k = 4)$ comprises three solutions c_1, c_2 and $c_3(|C_k| = 3)$ for the sample input graph of 10 vertices (numbered from 1 to 10) shown in Fig. 1. In this example, one complete color class {1,2,3,4} appears in all the three solutions. Thus we have only one CRS.

In what follows, we use this example to explain in detail the technique used to identify and extract such a CRS from a given set of k-colorings.

3.1.2. *Surface analysis: a binary tree based technique*

The core technique for extracting CRS is a representation of the given solutions as a binary tree. In such a tree, each node corresponds to a vertex in the given graph. Fig. 2 (right) shows such a representation for the solutions c_1, c_2 and c_3.

¿From a logical point of view, two steps are necessary to carry out a surface analysis: building the binary tree and then extracting the CRS from the tree. From a practical point of view, these two steps are merged such that CRS are successively identified and extracted while building the tree. Let us now see how this happens.

Let *root vertices* be the nodes that can be reached from the root node of the tree (included) following right edges ({1, 5, 7, 8, 10} in our sample). In such a structure, a path exists from a *root vertex* v_r to all nodes v_i reached following left edges, if v_r and all v_i belong to the same color class of a solution. For instance, the path $7 \to 8 \to 9$ exists since vertices 7, 8 and 9 are in the same color class (V_4) in solution c_2.

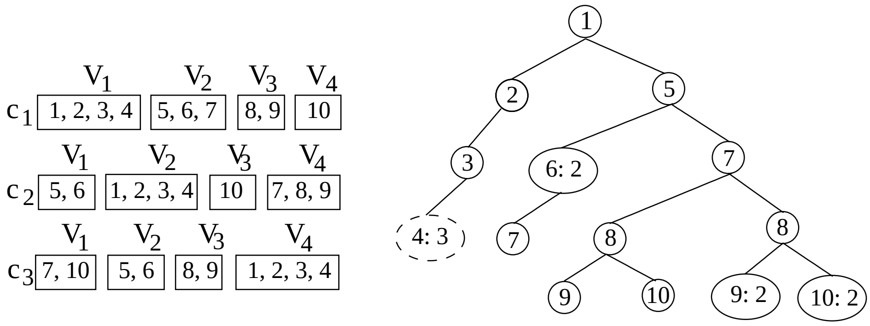

Figure 2. Illustration of the surface analysis.

When building the tree[5], right edges are added (and, so, root vertices) to include all the color classes of C_k that are not already in the structure. They are also used to represent color classes issued from the same root vertex. This is the case in Fig. 2 for the path $7(\to 8) \to 10$ since the root vertex 7 appears in $V_4 = \{7, 8, 9\}$ of c_2 **and** in $V_1 = \{7, 10\}$ of c_3. If one adds a fourth solution containing, for instance, the color class $\{7, 9\}$, then node 8 will still be the left node of 7, 9 will be inserted as a right node of 8 and 10 will become the right node of 9 (instead of 8).

Let *representative nodes* be the particular nodes which contain additional information, e.g. *6: 2* in the tree of Fig. 2. The first number is a vertex of the studied graph and the second (let us call it *FREQ*, for "frequency") shows the number of times the set composed of vertices which belong to a path from a root vertex to a representative node covers entirely a color class within all studied solutions. For instance, *6: 2* means "the color class $\{5, 6\}$ appeared two times (in c_2 and c_3) as a complete color class over the studied solutions $\{c_1, c_2, c_3\}$". Thus, CRS can directly be retrieved by following paths from the illustration corresponding to the path $1 \to 2 \to 3 \to 4$ (tree node with a dashed line style).

Algorithm 1 gives an outline of the procedure we used to build the surface analysis structure (denoted by SA_STRUCT). $first(V_j^i)$ refers to the first vertex of V_j^i. "min_FREQ" is an integer parameter ($1 \leq min\_FREQ \leq |C_k|$) used to restrict the output to CRS which have a minimal frequency. For instance, it has been set to 1 to generate the structure illustrated in Fig. 2 (right drawing). Notice that min_FREQ influences the size of SA_STRUCT.

Finally, let us say that the order the solutions are examined has no incidence on the identification of CRS. Indeed, the core treatment of Algorithm 1 consists in comparing each color class from any solution with color classes of the other solutions (line 3), the order of these comparisons being irrelevant. For instance, Fig. 2 (right) has been generated following the order permutation $\pi_1 = \{c_1, c_2, c_3\}$, but $\pi_2 = \{c_2, c_3, c_1\}$ will give the same result.

Algorithm 1. Building the surface analysis structure.

begin
$SA\_STRUCT \leftarrow \emptyset$
for $i = 1$ **to** $|C_k|$ **by** 1 **do**
[1] **for** $j = 1$ **to** k **by** 1 **do**
[2] **if** $V_j^i \notin SA\_STRUCT(V_j^i \in c_i)$ **then**
$counter \leftarrow 1$ /* counts the frequency of V_j^i in C_k */
[3] **for** $i' = i + 1$ **to** $|C_k|$ **by** 1 **do**
if $V_j^i = V^{i'}_{col_{i'}(first(V_j^i))}$ **then**
$counter \leftarrow counter + 1$
if $counter \geq min\_FREQ$ **then**
[4] $SA\_STRUCT \leftarrow SA\_STRUCT \cup V_j^i$
$FREQ_{V_j^i} \leftarrow counter$
end

One must pay attention to the structures storing the representative sets and the solutions since all the color classes from all the solutions are first checked for existence in SA_STRUCT (line 2). If a color class V_j^i is already in the tree, it is just ignored since it has already been considered. Insertion (line 4) may also be easy and fast. We implemented the SA_STRUCT and solutions storage by means of binary trees because they are well convenient for searching and inserting, but balanced trees (*B-Trees*) may also be of great interest.

3.1.3. *Time and space complexity*
Algorithm 1 has an $O(|V|^3)$ worst time complexity, weighted by $|C_k|^2$:

- If $k = 1$ (one color class) then *SA_STRUCT* only contains one CRS (which is V) and $FREQ_V = |C_k|$. This leads to a simplified algorithm running in $O(1)$. Remark that this case corresponds to null graphs ($|E| = 0$).

- If $k = |V|$ then *SA_STRUCT* contains exactly $|V|$ CRS, each one being composed of a single vertex with frequency $|C_k|$. Here again the algorithm is reduced and runs in $O(1)$. In this case, the input graph $G = (V, E)$ is complete ($|E| = |V|(|V| - 1)/2$) if $k = \chi(G)$.

- The worst case is the most commonly encountered and occurs when $1 < k < |V|$. Lines 1–3 require here a maximum of $|V|$ elementary operations each, leading to an overall time complexity of $O(|V|^3)$. Note that insertions (line 4) can easily be achieved in $O(1)$ by storing the possible right place when searching (line 2).

Recall that Algorithm 1 aims at counting the frequency of each color class in C_k. So, assuming that all color classes occur exactly once and *min_FREQ* $= 1$ (worst case), *SA_STRUCT* contains exactly $k|C_k|$ items. This leads to an $O(|V|)$ worst space complexity, up to a factor of $|C_k|$.

3.2. Extracting partial representative sets: depth analysis

3.2.1. *Definition and example*
The previous analysis scheme allows us to identify complete representative sets; we will see later that many DIMACS benchmark graphs do contain such sets. However, this analysis scheme is interested only in complete color classes and is blind of subsets of color classes. Indeed, if there is no entire color class shared by all the studied solutions, surface analysis cannot give useful information.

For instance, in Fig. 2 (left), the set of vertices $\{5, 6\}$ is shared by the given colorings. However, it represents only a *subset* of the color class V_2 of the solution c_1 though it corresponds to a complete color class in c_2 (V_1) and in c_3 (V_2). Surface analysis will miss such a partial representative set (PRS).

This section presents the *depth analysis* which is designed to extract PRS. This scheme allows us to discover a finer and deeper information compared to surface analysis.

Before giving the description of the technique used for depth analysis, let us define formally the notion of partial representative set.

Given a set C_k of legal k-colorings of a graph G, the set of vertices $\{v_1, \ldots, v_m\}, v_i \in V(1 < m \leq |V|)$, is a PRS if:

1. $\exists(V_a^q, V_b^{q'})/\{v_1, \ldots, v_m\} = V_a^q \cap V_b^{q'}(V_a^q \in c_q, a \in [1..k], c_q \in C_k, V_b^{q'} \in c_{q'}, b \in [1..k], c_{q'} \in C_k, q \neq q')$ and
2. $\forall c_p \in C_k(1 \leq p \leq |C_k|), \exists V_j^p \in c_p(j \in [1..k])/\{v_1, \ldots, v_m\} \subseteq V_j^p$.

3.2.2. *Depth analysis: the technique*

The technique used to carry out a depth analysis is quite similar to that of a surface analysis. The main difference is that partial representative sets are not required to cover an entire color class; they can be **subsets** of color classes. Here, the tree structure also stores **intersections** of color classes. In this case, the "frequency" counts the number of **inclusions** of any intersection into color classes.

Figure 3 gives an illustration of this *depth analysis*, with respect to the graph of Fig. 1. Like for complete representative sets, partial representative sets can directly be extracted by following paths from root vertices to all representative nodes which have a frequency equal to the number of solutions studied. For our example, there are three PRS: {1, 2, 3, 4}, {5, 6}, and {8, 9}.

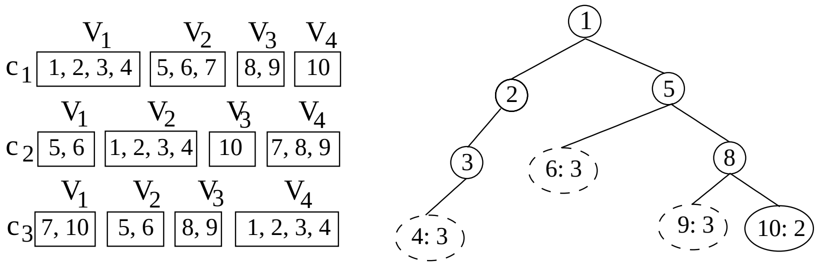

Figure 3. Illustration of the depth analysis.

Algorithm 2 gives an outline of the procedure we used to build the tree structure of the depth analysis (denoted by DA_STRUCT). Some practical details are voluntarily omitted for a greater readability. Let us say simply that special data structures are necessary to allow fast searching (line 1) and inserting (line 3) operations. Here again, the order the solutions are examined has no influence on the identification of PRS since all intersections of color classes from any couple of solutions are checked for inclusion into the classes of the other solutions.

Algorithm 2. Building the depth analysis structure.

begin
$DA\_STRUCT \leftarrow \emptyset$
for $i = 1$ **to** $|C_k| - 1$ **by** 1 **do**
  for $j = 1$ **to** k **by** 1 **do**
    for $i' = i + 1$ **to** $|C_k|$ **by** 1 **do**
      for $j' = 1$ **to** k **by** 1 **do**
[1]   **if** $V_j^i \cap V_{j'}^{i'} \notin DA\_STRUCT(V_j^i \in c_i, V_{j'}^{i'} \in c_{i'})$ **then**
          $counter \leftarrow 1$ /* counts the frequency of $V_j^i \cap V_{j'}^{i'}$ in C_k */
[2]   **for** $i'' = i + 1$ **to** $|C_k|$ **by** 1 **do**
            if $(V_j^i \cap V_{j'}^{i'}) \subseteq V^{i''}_{col_{i''}(first(V_j^i \cap V_{j'}^{i'}))}$ **then**
              $counter \leftarrow counter + 1$
          if $counter \geq min\_FREQ (2 \leq min\_FREQ \leq |C_k|)$ **then**
[3] $DA\_STRUCT \leftarrow DA\_STRUCT \cup (V_j^i \cap V_{j'}^{i'})$
            $FREQ_{V_j^i \cap V_{j'}^{i'}} \leftarrow counter$
end

3.2.3. *Time and space complexity*
Algorithm 2 has an $O(|V|^4)$ worst time complexity, weighted by $|C_k|^3$:

- If $k = 1$ or $k = |V|$, each PRS is a CRS with frequency $|C_k|$. So, this naturally leads to a simplified procedure running in $O(1)$.

- for $1 < k < |V|$, the additional cost of Algorithm 2, compared to the time complexity of Algorithm 1 (Sect. 3.1.3), comes from line 2 (which requires $|C_k||V|$ elementary operations at most), leading to a time complexity $O(|V|^4)$.

The space complexity of DA_STRUCT only relies on the number of times the test in line 1 is true (assuming also that $V_j^i \cap V_{j'}^{i'} \neq \emptyset$). So, suppose that it is always true and $min\_FREQ = 2$ (worst case).

¿From a theoretical point of view, this case generates a maximum of $\#PRS$ different sets to be included in the depth analysis structure, with:

$$\#PRS = \frac{k|C_k|(k|C_k| - 1)}{2} - |C_k| \frac{k(k-1)}{2} \tag{1}$$

$\#PRS$ can be retrieved by building a complete graph $G' = (V', E')$ with $|V'| = k|C_k|$ vertices, each vertex $v_j^i \in V'$ representing the color class V_j^i in c_i. Each edge $(v_j^i, v_{j'}^{i'}) \in E'$ means that $V_j^i \cap V_{j'}^{i'} \notin DA\_$

STRUCT (and $V_j^i \cap V_{j'}^{i'} \neq \emptyset$). At this stage $|E'| = k|C_k|(k|C_k| - 1)/2$ (first part of Eq. 1).

We must remove all $(v_j^i, v_{j'}^i)$ edges from E' since $V_j^i \cap V_{j'}^i = \emptyset$ (V_j^i and $V_{j'}^i$ belong to the same solution c_i): there are $k|C_k|(k-1)/2$ (second part of the equation).

Eq. 1 leads to an $O(|V|^2)$ worst space complexity, up to a factor of $|C_k|^2$.

4. Computational results

4.1. Generic tabu search

To generate solutions to be studied, we use the generic tabu search (GTS) algorithm from Dorne and Hao (1998(a)). GTS is a general algorithm designed to solve several coloring problems (graph coloring, T-coloring and set T-coloring). We recall here the main components of GTS and the general procedure (see Algorithm 3).

Initial configuration GTS uses a *DSATUR-based* greedy algorithm (Brélaz, 1979) to generate initial configurations.

Configuration re-generation The re-generation aims at producing a (k-1)-coloring from a k-coloring. The nodes in the last color class k are given a new color from $[1..k-1]$ in such a way that the number of conflicting nodes[6] over the graph is minimized.

Searching for proper coloring Beginning from a re-generated conflicting configuration, GTS iteratively makes best *1-moves*, changing the color of a conflicting node to another one, until achieving a proper coloring. "Best moves" are those which minimize the difference between the cost (the number of conflicting nodes) of the configuration before the move is made and the cost of the configuration after the move is performed[7]. A tabu move leading to a configuration better than the best configuration found so far is always accepted (*aspiration criterion*).

The tabu tenure l is dynamically computed using the following formula:

$$l = \alpha \times f(c) + random(g) \tag{2}$$

[6] A node $u \in V_i$ is said to be conflicting if $\exists v \in V_i / (u, v) \in E$.
[7] If there are multiple best 1-moves, one is chosen randomly.

where $f(c)$ stands for the number of conflicting edges in configuration c. $random(g)$ is a function which returns an integer value uniformly chosen in $[1..g]$. α weights the number of conflicting edges. A move m can be characterized by a triplet (u, V_{old}, V_{new}), $u \in V$, V_{old} and V_{new} being, respectively, the previous and the new colors of u. So, when a move m is performed, assigning u to the color class V_{old} is forbidden for the next l iterations by introducing the (u, V_{old}) couple in the tabu list TL.

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 over again. The algorithm stops when an optimal or a k-coloring (k fixed) is obtained or when a maximum number of moves have been carried out without finding a solution.

Algorithm 3. Generic tabu search.
begin
$TL \leftarrow \emptyset$ /* Initialize the tabu list TL to empty */
Generate the initial configuration c using DSATUR
Re-generate c with one color less
while not *Stop condition* **do**
  $c^* \leftarrow c$
  while $f(c^*) > 0$ **and not** *Stop condition* **do**
    Update c by performing a best 1-move $m(u, V_{old}, V_{new})$
    $TL \leftarrow TL \cup (u, V_{old})$
    if $f(c) < f(c^*)$ **then**
      $c^* \leftarrow c$
  if $f(c^*) = 0$ **then**
    Re-generate c with one color less
end

4.2. Analysis results

In this section we give the results of *surface analysis* (Sect. 3.1) and *depth analysis* (Sect. 3.2) for a set of the DIMACS benchmark graphs[8]. To generate solutions, we use the GTSD procedure which is our implementation of the above GTS algorithm[9]. GTSD is initiated with a *DSATUR* algorithm. For each graph, we indicate if the set of solutions found by GTSD contains *complete representative sets* or *partial representative sets*.

[8] Available via anonymous FTP from *FTP://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/*.

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

Settings The α and g parameters used for computing the dynamic tabu tenure l (Eq. 2) were empirically determined, respectively 2 and 10 at most for all graphs. A maximum of 10 million moves were allowed for the search process.

Table I. Some results of analysis on DIMACS benchmark graphs using GTSD.

Graph				Analysis			
Name	$\|V\|$	$\|E\|$	χ	k	FREQ. (%)	#RS	$\overline{\|\text{RS}\|}$
school1	385	19095	14	14	100 (CRS)	1	29
school1_nsh	352	14612	14	14	50 (CRS)	3	23.3
dsjr500.1	500	3555	12	12	20 (CRS)	1	4
dsjr500.1c	500	121275	84	85	30 (CRS)	17	6
r125.1	125	209	5	5	20 (CRS)	3	5
r125.1c	125	7501	46	46	100 (CRS)	37	2.8
r125.5	125	3838	36	36	90 (CRS)	1	4
r250.1	250	867	8	8	50 (CRS)	1	3
r250.1c	250	30227	64	64	100 (CRS)	31	3.9
r250.5	250	14849	65	66	100 (CRS)	14	3.6
r1000.1	1000	14378	20	20	30 (CRS)	1	3
r1000.1c	1000	485090	≥ 90	98	100 (CRS)	18	10.4
le450_15a	450	8168	15	15	0 (CRS)	0	0
le450_15b	450	8169	15	15	0 (CRS)	0	0
le450_15c	450	16680	15	15	100 (CRS)	1	30
le450_15d	450	16750	15	15	40 (CRS)	5	30
mulsol.i.1	197	3925	49	49	100 (CRS)	24	1.5
flat300_20_0	300	21375	20	20	100 (CRS)	20	15
flat300_26_0	300	21633	26	26	100 (CRS)	26	11.5
flat300_28_0	300	21695	28	32	100 (PRS)	1	3
flat1000_50_0	1000	245000	50	50	100 (CRS)	50	20
flat1000_60_0	1000	245830	60	60	100 (CRS)	60	16.7
dsjc125.5	125	3891	≥ 10	17	100 (CRS)	1	8
dsjc250.5	250	15668	≥ 11	28	100 (PRS)	1	2
dsjc500.5	500	62624	≥ 13	50	50 (PRS)	2	2

Columns 1–4 in Table I show for each graph, its name, the number of nodes and edges and its chromatic number (or its best known lower bound) respectively. The last four columns show the results of the analysis: best coloring found by GTSD after 10 million moves, percentage of solutions containing at least one RS, number of RS and mean RS

size. Entries with a "100 %" pattern in the "FREQ. (%)" field indicate that a RS has been found. If no RS is found in *all* the studied solutions (entries $<$ 100 %), we give the number of solutions (in percentage) that contain a RS. Analyses were performed over five to ten solutions generated by GTSD.

¿From Table I we can make several remarks. First, 12 out of the 25 studied graphs have complete representative sets. Note that most of them have a significant number of RS (compared to k) with an average total size (#RS $\times$ $\overline{|\text{RS}|}$) going from almost 20 % to 100 % of $|V|$, although these graphs belong to quite different families.

Second, for a few graphs, when no CRS is found, or with a low percentage, partial representative sets are sometimes identified. Nevertheless, ten graphs have less than 60 % of RS, meaning that solutions are quite different for these graphs. Especially, the le450_15a and le450_15b graphs have no color class in common.

Third, for flat graphs, except flat300_28_0 for which GTSD finds no optimal solution, the analysis reveals that all solutions were identical. This may be due to some structural properties of these graphs and we believe that other flat graphs, optimally colored, may have the same property[10]. It is possible that only one solution (equivalent solutions by permutation of colors are excluded) exists for each of these graphs.

5. Boosting the performance of GTSD by exploiting the analysis results

We show below a simple way of boosting the performance of the above GTSD algorithm by exploiting the results of solution analysis (Fig. 4 illustrates the idea).

Our main motivations are twofold. First, it is always easier to color a graph with $k + \epsilon$ colors (integer $\epsilon > 0$) than with k colors[11]. In other words, GTSD needs a less number of moves to find $(k + \epsilon)$-colorings than to reach solutions with k colors. Secondly, we believe that solutions with $k+\epsilon$ colors share common characteristics (representative sets) with k-colorings.

[10] The designer of these graphs had inadvertently left out a *feature* whose omission made these graphs significantly easier to solve than he had previously imagined. He also stated that "they could be optimally colored by a simple greedy algorithm using a *particular natural ordering*". See Jagota (1996).

[11] There are more alternative to color any vertex since ϵ additional colors are available.

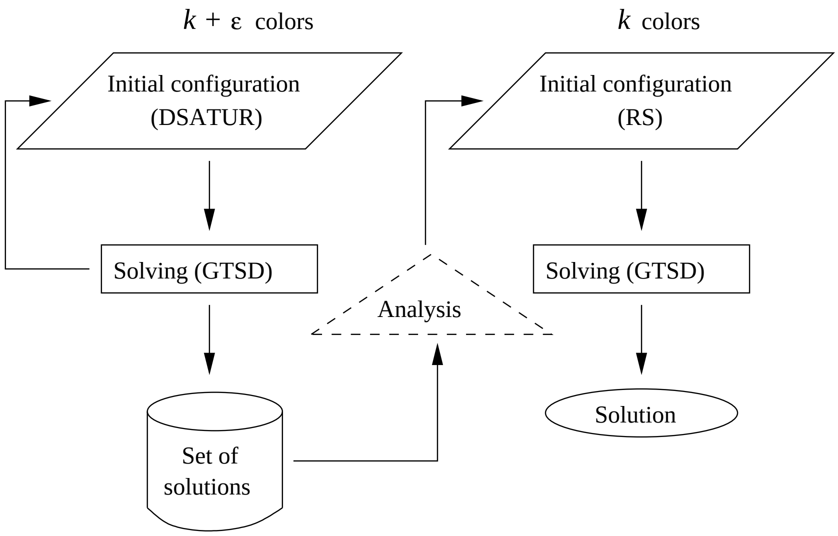

Figure 4. Overall view of the GTSA global process.

5.1. Using analysis results to build an initial configuration

Recall that the initial configuration of GTSD is built using the DSATUR greedy algorithm. Now, we modify the way the GTSD procedure is initiated as follows.

For a given graph, we first generate a set of five to ten solutions with $k + \epsilon$ colors (integer $\epsilon > 0$). These solutions are analyzed to extract RS. Then the extracted RS are sorted in decreasing order of appearance percentage (frequency). Finally, a configuration with only k colors is constructed as follows. Color classes are filled one by one with these sorted RS, beginning with the representative set most frequently encountered and ending when all the vertices are included in the k color classes or when no more representative set is available. In the latter case, free vertices are given a color in $[1..k]$ such that the conflicts over the graph are minimized.

For the presentation purpose, we call GTSA (A for Analysis) the overall procedure using this special initialization and including the search for a valid k-coloring.

5.2. Computational results of GTSA

Table II gives results of GTSA. Times (in seconds) and the number of moves include the generation of solutions with $k+\epsilon$ colors, the analysis process, the above initialization step and the search for a proper k-coloring (within 10 million moves). All reported results are averaged over five to ten runs. They were obtained using the same α and g parameters (Eq. 2) for GTSA and GTSD, namely 2 and 10 at most (respectively) for all graphs.

Table II. Using solution analysis to improve GTS.

Graph	χ	GTSD			GTSA			
		k	Time	Moves	k	Time	Moves	$k+\epsilon$
dsjr500.1c	84	85	278	88972	85	13	10643	87
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	≥ 10	17	15	242968	17	< 1	9081	18
dsjc500.5	≥ 13	50	5585	16532498	50	282	1003149	52

¿From Table II, we notice that solution analysis remarkably speeds up the search. Indeed, all reported results of GTSA are better than those of GTSD in terms of resolution speed. For some graphs, the resolution times needed to find a solution is divided by a factor of ten. GTSA is even able to find much better colorings for the dsjr500.5 graph in terms of solution quality (using 124 colors instead of 127). Finally, for the le450_15c and le450_15d graphs, an optimal coloring with 15 colors is found directly by constructing the initial configuration using the results of solution analysis with 16 colors. This last remark suggests that these graphs have representative sets not only for a set of solutions with k colors but also for solutions with different number of colors.

6. Conclusions and discussions

In this paper, we have introduced the notion of *representative sets* to characterize an intrinsic property of solutions for the graph coloring problem. We have also presented two practical schemes allowing for the extraction of *complete representative sets* and *partial representative sets* from a set of given solutions. Thanks to binary tree techniques, we avoid the permutation problem of coloring solutions.

Analyses have been carried out on a set of well-studied DIMACS graphs. We observed that a large number of these graphs contain representative sets, sometimes quite numerous with consequent total size.

One may suspect some links between the existence, number and size, of representative sets and special topological structure of the graphs. However, more evidence is needed to confirm or refute such a hypothesis. The analysis also reveals that some graphs have quite different solutions while others share common coloring information.

We have also used the analysis result to improve a tabu search algorithm. This is achieved by building a special initial configuration with k-colors from solutions with $k + \epsilon$ (integer $\epsilon > 0$) colors. We observed that this simply technique greatly speeds up the initial search algorithm. There are certainly other possibilities to integrate such analysis results in a search algorithm.

Representative sets may also be useful in the context of population-based coloring algorithms to measure the diversity of the configurations of a population, leading to a new stopping criterion for this kind of algorithm. Similarly, representative sets can be used with non-deterministic coloring algorithms running on a single configuration to measure diversity of solutions found within multiple executions.

To carry out the analysis proposed in this paper, one needs a (large) set of solutions. These solutions may be difficult to obtain when the problem instance is hard to solve. One possibility to get around this problem would be to relax the requirement for legal solutions. In this case, the analysis may be based on improper colorings (with a few conflicts). Such colorings can be obtained more easily and more quickly by a tabu (or any other search) algorithm with a limited number of iterations or a faster (say greedy) algorithm. Of course, the results produced in such a way will be less accurate. Nevertheless such a treatment constitutes a fast approximate analysis and may still give useful information about the solutions.

Acknowledgements

We would like to thank gratefully the reviewers of the paper for their questions and comments, which help to improve the quality of the paper. This work was partially supported by the Sino-French Joint Laboratory in Computer Science, Control and Applied Mathematics (LIAMA) and the Sino-French Advanced Research Programme (PRA).

References

Barabási, A.L. and Albert, R.: Emergence of Scaling in Random Networks, *Science* **286**, pp. 509–512, 1999.

Brélaz, D.: New Methods to Color the Vertices of a Graph, *Communications of the Association for Computing Machinery* **22(4)**, pp. 251–256, 1979.

Cheeseman, P., Kanefsky, B. and Taylor, W.M.: Where the Really Hard Problems Are, In Mylopoulos, J. and Reiter, R., editors, *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, Sidney, Australia, pp. 331–337. Morgan Kaufmann, 1991.

Culberson, J. and Gent, I.P.: Well Out of Reach: Why Hard Problems Are Hard, Research Report APES-13-1999, *http://www.dcs.st-and.ac.uk/~apes/reports/apes-13-1999.ps.gz*, June 1999.

Davis, L.: *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

Dorne, R. and Hao, J.K.: Tabu Search for Graph Coloring, T-Colorings and Set T-Colorings, In Voss, S., Martello, S., Osman, I.H. and Roucairol, C., editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pp. 77–92. Kluwer Academic Publishers, 1998(a).

Dorne, R. and Hao, J.K.: A New Genetic Local Search Algorithm for Graph Coloring, *Lecture Notes in Computer Science* **1498**, pp. 745–754. Springer Verlag, 1998(b).

Dubois, N. and de Werra, D.: EPCOT: An Efficient Procedure for Coloring Optimally with Tabu Search, *Computers and Mathematics with Applications* **25(10/11)**, pp. 35–45, 1993.

Ferland, J.A. and Fleurent, C.: Genetic and Hybrid Algorithms for Graph Coloring, *Annals of Operations Research* **63**, pp. 437–461, 1996(a).

Ferland, J.A. and Fleurent, C.: Object-Oriented Implementation of Heuristic Search Methods for Graph Coloring, Maximum Clique, and Satisfiability, In Johnson and Trick (1996), pp. 619–652, 1996(b).

Funabiki, N. and Higashino, T.: A Minimal-State Processing Search Algorithm for Graph Colorings Problems, *IEICE Transactions on Fundamentals* **E83–A(7)**, pp. 1420–1430, July 2000.

Galinier, P.: *Étude des Métaheuristiques pour la Résolution du Problème de Satisfaction de Contraintes et de la Coloration de Graphes*, PhD thesis (Computer science), Université Montpellier 2, France, January 1999.

Galinier, P. and Hao, J.K.: Hybrid Evolutionary Algorithms for Graph Coloring, *Journal of Combinatorial Optimization* **3(4)**, pp. 379–397, 1999.

Gamst, A.: Some Lower Bounds for a Class of Frequency Assignment Problems, *IEEE Transactions of Vehicular Technology* **35(1)**, pp. 8–14, 1986.

Garey, M.R. and Johnson, D.S.: *Computers and Intractability: A Guide to Theory of NP-Completness*. W.H. Freeman and Company, New York, 1979.

Glover, F. and Laguna, M.: *Tabu Search*. Kluwer Academic Publishers, 1997.

Gomes, C., Selman, B. and Kautz, H.: Boosting Combinatorial Search Through Randomization, *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison (WI), USA, pp. 431–437, 1998.

Halldórsson, M.M.: A Still Better Performance Guarantee for Approximate Graph Coloring, *Information Processing Letters* **45(1)**, pp. 19–23, 1993.

Hamiez, J.P. and Hao, J.K.: Scatter Search for Graph Coloring, *Lecture Notes in Computer Science* **2310**, pp. 168–179. Springer Verlag, 2002.

Hertz, A., Jaumard, B. and Poggi de Aragao, M.: Local Optima Topology for the k-Coloring Problem, *Discrete Applied Mathematics* **49**, pp. 257–280, 1994.

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

Hogg, T.: Refining the Phase Transition in Combinatorial Search, *Artificial Intelligence* **81(1-2)**, pp. 127–154, 1996.

Jagota, A.: An Adaptive, Multiple Restarts Neural Network Algorithm for Graph Coloring, *European Journal of Operational Research* **93**, pp. 257–270, 1996.

Johnson, D.S., Aragon, C.R., McGeoch, L.A. and Schevon, C.: Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning, *Operations Research* **39(3)** pp. 378–406, 1991.

Johnson, D.S. and Trick, M.A., editors: *2nd DIMACS Implementation Challenge: Cliques, Coloring and Satisfiability*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science **26**. American Mathematical Society, 1996.

Leighton, F.T.: A Graph Coloring Algorithm for Large Scheduling Problems, *Journal of Research of the National Bureau of Standards* **84**, pp. 489–506, 1979.

Lund, C. and Yannakakis, M.: On the Hardness of Approximating Minimization Problems, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing* pp. 286–293, 1993.

Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B. and Troyansky, L.: Determining Computational Complexity from Characteristic 'Phase Transitions', *Nature* **400(8)**, pp. 133–137, 1999.

Morgenstern, C.A.: Distributed Coloration Neighborhood Search, In Johnson and Trick (1996), pp. 335–357, 1996.

Tsang, E.P.K.: *Foundations of Constraint Satisfaction*. Academic Press, London and San Diego, 1993.

Walsh, T.: Search in a Small World, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)* **2**, pp. 1172–1177, Stockholm, Sweden, July 31–August 6 1999. Morgan Kaufmann Publishers.

Walsh, T.: Search on High Degree Graphs, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)* **1**, pp. 266–271, Seattle (Washington), USA, August 4th–10th 2001.

Watts, D.J. and Strogatz, S.H.: Collective Dynamics of 'Small-World' Networks. *Nature* **393**, pp. 440–442, 1998.

Welsh, D.: *Codes and Cryptography*. Oxford University Press, New York, 1988.

Yokoo, M.: Why Adding More Constraints Makes a Problem Easier for Hill-Climbing Algorithms: Analyzing Landscapes of CSPs, *Proceedings of the Third International Conference on Principles and Practice of Constraint Programming (CP-97)*, pp. 356–370, 1997.