A new DSATUR-based algorithm for exact vertex coloring

Pablo San Segundo*

Centro de Automática y Robótica (CAR), José Abascal 2, 28006, Madrid, Spain

ARTICLE INFO

Keywords:
Color
Graph
Exact
DSATUR

ABSTRACT

This paper describes a new exact algorithm PASS for the vertex coloring problem based on the well known DSATUR algorithm. At each step DSATUR maximizes saturation degree to select a new candidate vertex to color, breaking ties by maximum degree w.r.t. uncolored vertices. Later Sewell introduced a new tiebreaking strategy, which evaluated available colors for each vertex explicitly. PASS differs from Sewell in that it restricts its application to a particular set of vertices. Overall performance is improved when the new strategy is applied selectively instead of at every step. The paper also reports systematic experiments over 1500 random graphs and a subset of the DIMACS color benchmark.

1. Introduction

For a given undirected graph $G=(V,E)$ the *vertex coloring problem* (VCP) requires to assign a color to every vertex so that adjacent vertices are all colored differently and the total number of colors employed is minimized. The origins of the vertex coloring problem (VCP) can be traced back to a letter written to Sir Willian Hamilton by the renowned mathematician Augustus de Morgan in 1852, where the famous *Four Color Theorem* has its roots.

VCP is a well known and deeply studied NP-hard problem in graph theory [1], which has attracted the attention of many researchers for its implications in computational theory. Moreover, many real world applications in a wide variety of engineering fields have also been found, such as register allocation [2], scheduling [3–5], air traffic [6], etc., which suggest the importance of efficient algorithms in practice.

Let n and m be the cardinalities of the vertex set V and edge set E, respectively. $\deg(v)$ is the degree of a given vertex v, i.e. the number of adjacent vertices (alias neighbors). $\Delta(G)$ denotes the degree of the graph, the maximum degree of any of its vertices. A vertex v is said to be *dominated* by vertex w if the neighbor set of w is a superset of the neighbor set of v.

Subsets of V where all vertices are pairwise non-adjacent are called *independent sets*. Any k *coloring* of G partitions V in disjoint k independent sets named *color classes*. A *clique* in G is any subgraph with all its vertices pairwise adjacent. $\omega(G)$ denotes the *maximum clique* in G, the clique with the largest possible size. The *chromatic number* of a graph $\chi(G)$ is the minimum number of colors required to color G, i.e. the size of its optimum coloring.

A *partial coloring* of a graph $\Pi(U)$, $U \subset V$, is and admissible assignment of colors to vertices in U. Given a partial coloring Π, $c(\Pi)$ is the total number of colors used in the partition and $c(v \in \Pi)$ denotes the color label assigned to v. $\rho(v)$ refers to the *chromatic degree* of a vertex (alias its *saturation degree*), the number of adjacent vertices which have been assigned different colors.

Compared with other related problems such as the *maximum clique* or the *maximum independent set*, VCP is considerably more challenging; for example, it is possible to compute maximum clique exactly in graphs with some hundreds of vertices whereas fast exact coloring of a random graph with 80 vertices and 0.5 edge density already requires efficient algorithms and a powerful CPU. In fact, exact methods for VCP usually suffer from the same problem: it is easy to find a *good* admissible coloring $c(G)$, but proving that no $c(G)-1$ admissible coloring exists becomes very difficult.

The hardness of exact VCP computation has given rise to many efficient heuristics and metaheuristics for the domain. Initial approaches favored greedy constructive algorithms, which color each vertex exactly once given a heuristic rule to compute the next color label based on the current partial coloring. Fast but usually sensible to some initial criteria (e.g. vertex ordering) the most important examples are the simple greedy sequential (SEQ) and the maximum chromatic degree heuristic DSATUR [7].

Constructive algorithms, which produce color classes sequentially have also attracted attention, such as the *recursive largest first* (RLF [5]) algorithm or MAXIS [9], an improved variant of [8], which is known to perform well over difficult random graphs. Worth mentioning is also the *iterated greedy algorithm* IG [9], which iterates over suboptimal colorings by reordering a subset of color classes following certain criteria at each step.

Classical metaheuristics applied to VCP start from an initial incorrect color partition and gradually attempt to remove conflicts at each step. Many evolutionary algorithms are based on

* Tel.: +34 913 363 061; fax: +34 913 363 010.
E-mail address: pablo.sansegundo@upm.es

tabu search, *simulated annealing* or some form of hybridation. In view of the exponential character of exact coloring, real life problems usually resort to these methods.

The first proposed algorithm, TABUCOL [10], employed tabu search. In [11] a simulated annealing algorithm was proposed, which compared different neighborhoods; later the *Impasse Class Neighborhood* (ICN) described in [12] became for a long time the most effective approach to VCP. Algorithm MIPS-CLR [13] combines the Tabu Search technique with the ICN to produce a very effective algorithm. Hybrid algorithms integrating local search and diversification via crossover, as in [14] or [15], showed that diversification was able to improve the performance of local search. Of the more recent algorithms we point out [16], in a similar vein as MIPS-CLR, and MMT [17]. The latter is based on ICN and a crossover operator, which is an adaptation of the *Greedy Partitioning Crossover* proposed in [15].

The inherent difficulty of VCP makes exact algorithms rather scarce compared with approximate coloring methods. We give a brief overview of the leading algorithms.

Integer programming models for VCP have recently attracted much attention because of the improvement of general purpose ILP solvers. The first known ILP model for VCP is the so called descriptive model (VCP-DES), which for exact *k-coloring* can be defined as

$$\begin{array}{ll} (I) & x_{ik}+x_{jk}\le 1 \quad \forall (i,j)\in E, \forall k \\ (II) & \sum\limits_{k} x_k = 1 \quad \forall i \quad x_{i,k}\in\{0,1\} \end{array} \tag{1}$$

VCP-DES uses $n\cdot k$ Boolean variables, which represent all possible color assignments to vertices. Constraints of type I model that no two adjacent vertices can have the same color and constraints of type II do not allow different color labels in any vertex.

This model has been used in literature, mainly for its simplicity. Recently, Méndez-Díaz and Zabala [19,20] showed a way to exploit it in a *branch and cut* scheme by refining VCP-DES with further inequalities to remove symmetries – possibly the most relevant drawback of such models for any ILP solver – and thus obtain a stronger continuous relaxation.

A *set cover* ILP model (VCP-SC) was first employed successfully using column generation by Mehrotra and Trick [21] in a *branch and price* algorithm. VCP-SC can be formalized as follows:

$$\begin{array}{l} \text{Minimize} \quad \sum\limits_{s} x_s \\ \text{s.t.} \\ \sum\limits_{s\in\{s:i\in S\}} x_s \ge 1 \quad \forall i\in V \\ x_s\in\{0,1\} \end{array} \tag{2}$$

where the basic Boolean variables x_s represent (maximal) independent sets. A value of x_s to 1 indicates that the set is given a label (color) in the VCP-SC solution; in case a 0 is assigned to x_s that (color) set will not correspond to the minimum coloring. Constraints establish that all vertices have to be covered by at least one independent set.

Branch and price algorithms, which exploit VCP-SC are currently the best way to attack exact VCP for large structured graphs and a number of improvements have recently appeared, such as [22–24].

Worth mentioning is an additional ILP *set packing* model for VCP, which has recently been exploited in [25], performing similarly to VCP-SC in the reported tests. To end this overview we refer the reader to [18] for a detailed survey on both approximate and exact methods.

Color heuristics embedded in a complete enumeration scheme lead also to exact algorithms. Of these, branch and bound DSATUR [7], later improved by Sewell [26] is undoubtedly the best known and has been a standard *de facto* for many years (see Section 2 for a more detailed analysis of these algorithms).

DSATUR is a simple and efficient algorithm, which has been known to perform surprisingly well over random graphs. Outperformed by recent branch and price algorithms on the more difficult graphs it is still frequently employed both *stand-alone* or as part of more sophisticated algorithms to quickly compute initial upper bounds or to 'wrap up' on small subproblems (c.f. many of the algorithms described in [12]). This makes research on DSATUR-based solvers, in the opinion of the authors, still very important in practice.

This paper describes a new DSATUR-based algorithm. It is structured from here in 5 parts. Section 2 reviews existing related algorithms. Section 3 describes the new algorithm while Section 4 provides a case study. Section 5 reports experiments and finally Section 6 is the conclusions section.

2. DSATUR review

The general outline of sequential vertex coloring algorithms is to color each vertex in turn until there are no more left. The simplest form is the greedy coloring heuristic known as SEQ. Let V be the set of vertices to be colored and let $K=\{v_1,\ldots,v_n\}$ be any strict ordering of V. SEQ (V, K) can then be described as follows:

SEQ(V,K)
for $v := v_1$ to v_n
assign the smallest possible color to v
endfor

Different definitions of K give rise to different heuristics (e.g. [27,28] are examples of a fixed strategy). DSATUR computes K dynamically at each step choosing the vertex with maximum saturation degree w.r.t. the current partial coloring. In case of ties, the vertex with maximum degree over the set of (as yet) uncolored vertices is selected. A further tie is broken lexicographically.

An implicit enumeration of vertices combined with any of the vertex selection rules used in GREEDY produce exact color algorithms. A direct implementation, however, would lead to many spurious subproblems being analyzed. Brown [29] was the first to reduce complexity by enumerating only *tight* colorations in the graph.

Definition. A coloring $c(G)$of a given graph G is *tight* given a strict ordering of its vertices $v_1,\ldots,v_n$ if

$$\begin{array}{ll} c(v_{i+1})\le \text{colors}(i)+1 & \forall\, i=1,\ldots,n-1 \\ \text{colors}(i)=\max_s c(v_s) & 1<s\le i, \quad c(v_1)=1 \end{array} \tag{3}$$

Brélaz [7] used the Brown enumeration to implement exact DSATUR. Throughout the text, terms in capital letters will refer to both the heuristic rule and the exact algorithm. Subscript 'h' will be added when an explicit reference to the rule is required.

DSATUR recursively enumerates all tight partial colorings which, in turn, become new subproblems. For any partial k-coloring at a given step, a vertex is chosen using DSATUR_h and assigned either one of the k colors already used, or color $k+1$ if none are available. This gives a maximum branching factor of $k+1$ (colors) at any point in the search tree.

Leaf nodes are admissible colorings. When a leaf node is reached, the size of the coloring is compared with the current minimal coloring. If the latter is minored, the champion is unseated and the leaf node determines a new upper bound (UB) for the chromatic number. UB is used at any step to prune the search tree when the

size of the current partial coloring k is greater or equal to UB. This makes DSATUR a branch and bound algorithm.

Applying DSATUR_{h} produces an initial partial coloring, which corresponds to a clique. The size of this initial clique is used implicitly as lower bound (LB) during search, determining an end condition when $UB=LB$. Note that once LB has been fixed it does not change during the whole procedure whereas UB gradually decreases. This is an important disadvantage compared with exact ILP methods, where LB also rises dynamically by solving the continuous relaxation problem.

A number of improvements to the original DSATUR algorithm were suggested by Sewell [26]. Apart from an initial reordering of vertices, Sewell proposed a new tie breaking strategy and reported a reduction in the number of subproblems searched under some restrictions. We will refer to this new tiebreak as the *Sewell rule*. It can be described informally as follows:

SEWELL RULE

Select from the set of vertices tied at maximum saturation degree the one with the maximum number of common available colors in the neighborhood of uncolored vertices.

Compared with DSATUR_{h}, this heuristic rule is more informed since it explicitly takes into account the structure of the graph together with the number of available colors when branching. Note that DSATUR_{h} only estimates future color availability using degree and runs in $O(n^2)$ whereas SEWELL_{h} runs in $O(n^3)$ in the worst case.

Apart from the additional overhead, SEWELL_{h} is known to obtain, on average, worse admissible colorings than DSATUR_{h} at the early stages of the search. To overcome this problem, Sewell further proposed to compute *a priori* upper and lower bounds, which are also fed into the search algorithm. Reported experiments with a very tight UB seemed to outperform DSATUR in a benchmark of random graphs and a small set of real-world graphs. In spite of this, the SEWELL rule has not met many followers in recent years.

3. A new tiebreaking strategy

This paper proposes a new tiebreaking strategy for DSATUR, which is much less expensive to compute than SEWELL but has a similar pruning effect. It will be referred to as the PASS rule (in reference to the corresponding author's family name).

Let $G=(V,E)$ be a given graph to be colored of size n. Let Π be the l partial k-coloring ($c(\Pi)=k, |\Pi|=l$) obtained by DSATUR at the current step and let $U=\{v_{l+1},v_{l+2},\ldots,v_n\}$ be the set of remaining uncolored vertices. For any pair of adjacent vertices in U we define function $same : V^2 \rightarrow \text{naturals}$; such that

$$same(v_1,v_2)=\begin{cases} 0 & (v_1,v_2)\notin E(U) \\ |F(v_1)\cap F(v_2)| & (v_1,v_2)\in E(U) \end{cases} \quad (4)$$

where $F(v_i)$ is the set of admissible color labels for vertex v_i.

DSATUR initially applies DSATUR_{h} to select vertices in U with maximum saturation degree. Let $T\subseteq U$ be the set of such candidate vertices:

$$T=\max_i(\rho_{\Pi}(v_i)) \quad v_i\in U \quad (5)$$

SEWELL_{h} will then select a vertex $v_s\in T$, which minimizes the number of available colors in the set $U\backslash\{v_s\}$ in child subproblems. In practice, this is implemented by maximizing the number of common available colors in the uncolored neighborhood:

$$v_{sel}=\max_{v_s\in T}\left(\sum_{\substack{v\in U \\ v\neq v_s}} same(v_s,v)\right) \quad (6)$$

The new PASS rule introduces a further variation: it selects the new candidate from T in a similar manner as Sewell but *restricts its application to candidate vertices in T only*:

$$v_{sel}=\max_{v_s\in T}\left(\sum_{\substack{v\in T \\ v\neq v_s}} same(v_s,v)\right) \quad (7)$$

Further ties are broken lexicographically in all cases.

SEWELL_{h} minimizes the number of subproblems by systematically reducing available colors in subproblems at deeper levels of the search tree. By restricting this idea to the set of tied vertices in T, PASS employs a different strategy. It aims at *reducing color domains of vertices, which are already known to have the least number of available colors*, and so therefore more likely to require a new color in deeper levels of the search. Note that, at each step, all candidate vertices in T are, by definition, those with maximum saturation degree (and therefore have the least number of free colors). This fact justifies the PASS strategy.

However, consistent application of PASS does not improve the overall performance because in the initial steps of the search there are many vertices in T (w.r.t. to U) and the desired effect is not obtained. It is much better to selectively apply PASS in steps where the number of available colors of candidate vertices in T is below a certain parameter μ, which is computed dynamically. PASS performs better when μ is evaluated at each step in the following way:

$$\mu=c(\Pi)-\rho_{\Pi}(v\in T) \quad \mu\geq 0 \quad (8)$$

The first term in (8), $c(\Pi)$, is the number of assigned colors in the current partial coloring. The second term, the saturation degree (related to Π) of every candidate vertex, represents the number of colors in $c(\Pi)$ not available for every vertex in T (note that the particular colors can differ for each vertex). Therefore the difference between both terms is a direct measure of color availability. When PASS is applied in steps where candidate vertices have low values of μ, it is more likely that vertices in T in future subproblems will require a new color. A step where $\mu=0$ indicates that every vertex in T requires an extra color $k=c(|\Pi|)+1$ and so the part of the search tree, which hangs from the current node is very likely to be pruned.

The current implementation of PASS uses DSATUR_{h} tiebreak rule in all steps where μ exceeds a given threshold TH fixed at the beginning of the search (typically TH=3; Section 5.1 explains the choice). In steps where $\mu\leq \text{TH}$ the PASS rule is applied. We note that μ is the same for all candidate vertices and so has to be computed just once per step, an important practical advantage of (8). For TH=0, PASS is equivalent to DSATUR. Pseudocode for PASS is given in full.

$\text{PASS}(U,\prod,C_{\max},TH)$

U is the current set of uncolored vertices; Π is the current partial coloring; C_{max} is the size of minimal coloring; TH is a fixed threshold.

Initial values: $G=(V,E)$; $U:=V$; $\Pi:=\phi$; $C_{\max}:=0$; $TH:=3$
Returns: $C_{max}:=\chi(G)$
Step 1. Select vertex $v\in U$ with maximum saturation degree. Let T be the set of tied vertices. If $\mu=c(\Pi)-\rho(v\in T)\leq \text{TH}$ break ties using PASS_{h} (7) else break ties using DSATUR_{h} (5).
Step 2. if $(U=\phi)$ then let $C_{\max}:=c(\Pi)$ and return
Step 3. Select c(v) with the lowest possible color used in Π. If none are available assign $c(v):=c(\Pi)+1$

Step 4. if $c(v) \geq C_{max}$ return.
Step 5. PASS($\Pi+$(v,c(v)), U$-${v}, C_{max}, TH)

4. A case study

Graph G depicted in Fig. 1(A) has been chosen to compare the performance of DSATUR, SEWELL, and new PASS.

Notation $v_F^{(k)}$ will be used to describe the search trees. It summarizes two facts: (1) vertex v has been assigned color k ($(v,k)\in\Pi$ in the current step) and (2) further tight enumerations with $c(v)\in F$ are still possible. Subscripts or superscripts may be missing when unnecessary in the context. F is the set of remaining available colors for v (as in (4)) and each color constitutes a branching point in the search. Note that $k\notin F$ always holds.

$P=\{v_{1_{F_1}}^{(c_1)}, v_{2_{F_2}}^{(c_2)}, \ldots, v_{d_{F_d}}^{(c_m)}, \ldots, v_{n-1_{F_{n-1}}}, v_{n_{F_n}}\}$ refers to the current path, the vertex ordering reflecting the choices taken by the heuristic at each branching point. Every vertex $v_i\in P$ with $F_i\neq\phi$ is a subproblem at the ith level in the tree. For a given P, $c(\Pi)=m$ and $|\Pi|=d$ hold. When $d=n$ a leaf node has been reached and $c(\Pi=G)$ is an admissible coloring for VCP.

The behavior of the three algorithms is compared after the first minimal coloring has been established by DSATUR (i.e. the initial branch of the search tree is assumed the same in all three cases). This allows a cleaner comparison while not losing generality since SEWELL is known to produce, on average, a worse initial coloring than DSATUR as explained in [26].

The first minimal coloring obtained by DSATUR is a 5-coloring ($C_{\max}=5$). While traversing the first branch, DSATUR finds a 3-clique formed by vertices $\{1_\phi^{(1)}, 3_\phi^{(2)}, 6_\phi^{(3)}\}$. The remaining uncolored vertices all tie in saturation degree ($\rho=2$) so vertex 2 is chosen lexicographically to generated the child subproblem and assigned available color 1. The procedure continues in a similar way until all vertices are colored (at leaf node L) having traversed path $P_L=\{1_{\{\phi\}}^{(1)}, 3_{\{\phi\}}^{(2)}, 6_{\{\phi\}}^{(3)}, 2_{\{4\}}^{(1)}, 8_{\{\phi\}}^{(4)}, 4_{\{5\}}^{(2)}, 5_{\{5\}}^{(4)}, 7_{\{5\}}^{(3)}, 9_{\{\phi\}}^{(5)}\}$. Since $C_{\max}=5$, it is possible to remove this color from the corresponding F sets so that P_L is simplified to $\{1_{\{\phi\}}^{(1)}, 3_{\{\phi\}}^{(2)}, 6_{\{\phi\}}^{(3)}, 2_{\{4\}}^{(1)}, 8_{\{\phi\}}^{(4)}, 4_{\{\phi\}}^{(2)}, 5_{\{\phi\}}^{(4)}, 7_{\{\phi\}}^{(3)}, 9_{\{\phi\}}^{(5)}\}$ with only one branching point at vertex 2. All three algorithms backtrack to this vertex, label it with color 4 and update F sets and saturation degrees in the remaining uncolored vertices accordingly so that at the new step $P_{L+1}=\{1_{\{\phi\}}^{(1)}, 3_{\{\phi\}}^{(2)}, 6_{\{\phi\}}^{(3)}, 2_{\{\phi\}}^{(4)}, 8_{\{1\}}, 4_{\{2\}}, 5_{\{2,4\}}, 7_{\{3\}}, 9_{\{3\}}\}$ with remaining uncolored vertices $U=\{4.5.7.8.9\}$. Induced subgraph $G'=G(U,E(U))$ is depicted in Fig. 1(B). Saturation degrees for each vertex in U are $\rho(4)=3$, $\rho(5)=2$, $\rho(7)=3$, $\rho(8)=3$, $\rho(9)=3$ with maximum saturation degree set $T=\{4,7,8,9\}$ (see Fig. 1(C)).

The next step turns out critical for the performance of the different algorithms. To establish a clear comparison we denote by H_{DSATUR}, H_{SEWELL} and H_{PASS} the set of values obtained by the corresponding rules over vertices in T.

DSATUR uses maximum degree in G' to break ties. At present $\Delta G'=\deg_{G'}(4,5,9)=3$, $\deg_{G'}(7)=2$ and $\deg_{G'}(8)=1$ (see Fig. 1(B)) so $H_{DSATUR}=\{3,2,1,3\}$ and the first vertex in T (vertex 4) is chosen with color assignment $c(4)=2$.

In the case of SEWELL and PASS, ties are broken using Eqs. (6) and (7), respectively. SEWELL rule for vertex 4 considers common available colors in adjacent vertices in G', (e.g. $H_{SEWELL}(4)=1$ since vertex 4 shares just color 2 with neighbor vertex 5), which results in $H_{SEWELL}=\{1,1,0,1\}$ and the same lexicographical choice as DSATUR.

PASS applies the same criteria as SEWELL, but evaluates only neighbor vertices with maximum chromatic degree (the T set). In particular, vertex 4 shares no available color now (vertex 5 does not belong to T) so $H_{PASS}=\{0,1,0,1\}$and this time vertex 7 is selected with color label 3.

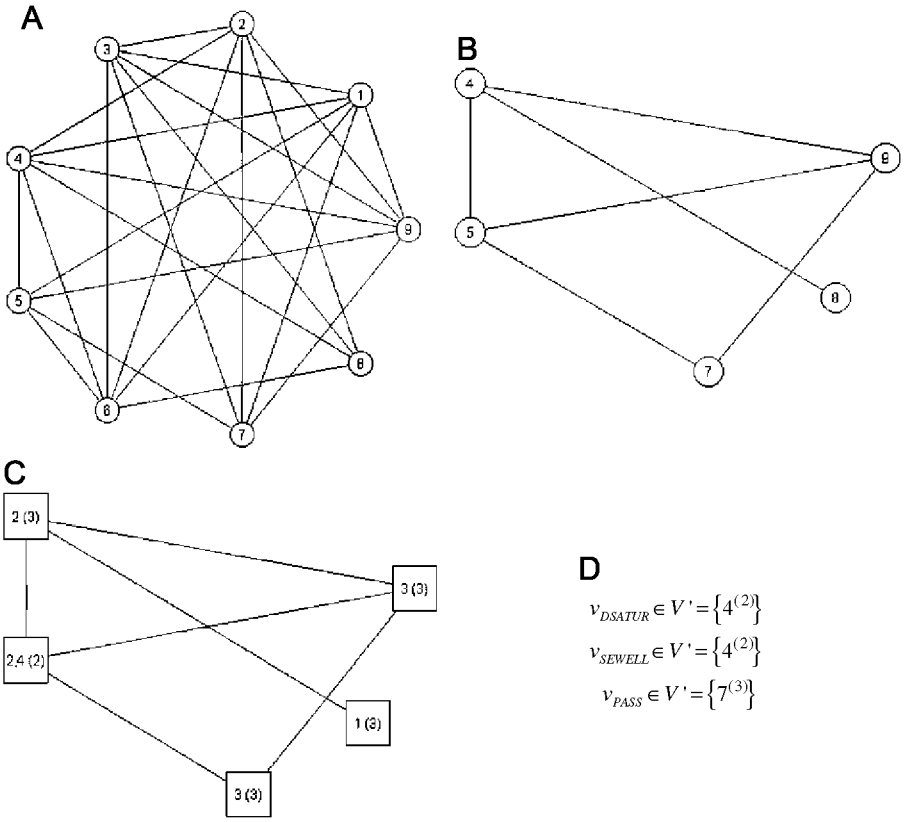

Fig. 1. A color example. (A) A simple graph. (B) Induced graph $G'=G(\{4,5,7,8,9\})$ by uncolored vertices after first backtrack (C) Available colors and saturation degree (in brackets) for each vertex in G' (see B). (D) Candidate vertices selected from by DSATUR, SWELL and new PASS. In brackets color assignments for each vertex.

It is clear that, on average, PASS will be less expensive to compute than SEWELL ($|T| \leq |U|$), but the key factor is that it has captured structure better. SEWELL selects vertex 4 on the basis of removing color 2 option from vertex $5_{\{2,4\}}$ in child subproblems, but $\rho(5)$ is not maximal and color 4 will still be available to continue the search. PASS picks vertex 7 to remove color 3 from the singleton F set in $9_{\{3\}}$ thus producing a cut in the new subproblem.

This is confirmed in the next step. In the child subproblem generated by SEWELL a 4 coloring is still possible and all uncolored vertices $U=\{5_{\{4\}},7_{\{3\}},8_{\{1\}},9_{\{3\}}\}$ have non-empty F sets, so it cannot be pruned.

Fig. 1(D) shows the different choices made by all three algorithms at step G'. As usual, color assignments appear as superscripts in brackets. The total number of steps needed to complete the search for DSATUR, SEWELL and PASS was 14, 13 and 12, respectively.

5. Experiments

This section reports a number of experiments on random $G_{n,p}$ graphs and a subset of the DIMACS color benchmark.

$G_{n,p}$ graphs are random graphs generated by including each possible edge independently with probability p for a given a list of n vertices. The density of such graph is approximately p and has been used extensively to test algorithms for VCP as they are hard known to be difficult to color. For the report we have generated a benchmark of 1500 random graph instances of varying sizes and densities (RB). Specifically RB includes 50 $G_{n,p}$ graphs for sizes $n=\{60,70,75\}$ and densities ranging from 0.1 up to 0.9 at 0.1 intervals. 50 instances of $G_{80,p}$ with $p=\{0.1,0.2,0.3\}$ complete the collection.

The DIMACS benchmark is currently the standard set for experimenting algorithms for VCP and other NP-hard problems (c.f. [30]; instances are publicly available at ftp://dimacs.rutgers.edu/pub/challenge/graph/). The benchmark set includes a number of artificial graphs (queen, latin_square, Insertion, etc.), random graphs (DSJC, standard $G_{n,p}$ graphs), geometric random graphs (DSJR and r) and graphs derived from real word problems amongst others (c.f. http://mat.gsia.cmu.edu/COLOR03/ for a description of each family). The subset used in the report comprises 102 graphs (mainly the subset used in [22]) so as to establish a comparison with leading branch and price algorithm MMT-BP.

Some instances from the DIMACS benchmark can be preprocessed in order to reduce their size. This is done by removing vertices while preserving chromatic degree by applying recursively the following two rules: 1. Any dominated vertex can be removed from the graph. 2. Any vertex v such that $\deg(v) < LB$ (where LB refers to, as usual, a lower bound on $\chi(G)$) can be removed from the graph (c.f. [20,25]). Whenever possible, graphs have been preprocessed prior to the start of the search. We note that this simplification attempt is useless on the random RB benchmark.

Section 5.1 analyses the basic properties of $PASS_h$ w.r.t. $DSATUR_h$ and $SEWELL_h$ while Section 5.2 compares PASS with existing algorithms in literature.

5.1. Analysis of the new heuristic rule

The new algorithm employs $PASS_h$ rule for tie breaks when parameter μ is below a certain threshold TH and DSATUR rule otherwise. Important issues are how to determine empirically TH, the ratio of steps where $PASS_h$ is applied (w.r.t. $DSATUR_h$) and how well it captures structure (w.r.t. $SEWELL_h$).

Table 1 reports the percentage of steps where $PASS_h$ was fired instead of $DSATUR_h$ when running PASS over our benchmark set RB using thresholds TH for parameter μ ranging from 1 to 5. Values of TH greater than three have almost no effect in PASS and have an additional overhead so TH has been fixed to 3 in the rest of experiments.

Table 1
Percentage of steps where $PASS_h$ was fired for varying thresholds of parameter μ using the RB benchmark. Table entries are averaged for each row accordingly (500 graphs for $n=60$, 70 and 75; 150 for $n=80$).

n	$\mu \leq 1$	$\mu \leq 2$	$\mu \leq 3$	$\mu \leq 4$	$\mu \leq 5$
60	63.4	76.0	76.1	76.1	76.1
70	68.1	81.5	81.7	81.7	81.7
75	69.9	84.1	84.3	84.3	84.3
80	77.3	87.7	87.7	87.7	87.7

Table 2
Number of steps taken by PASS and SEWELL using a subset of the RB benchmark. Each entry is averaged over 50 instances.

n	p	PASS	SEWELL	n	p	PASS	SEWELL
70	0.1	110	200	75	0.1	433	501
70	0.2	992	2056	75	0.2	7370	9223
70	0.3	126,827	134,562	75	0.3	136,937	208,925
70	0.4	925,802	1,090,343	75	0.4	4,058,103	4,134,748
70	0.5	2,861,863	3,513,657	75	0.5	20,408,727	19,473,294
70	0.6	4,345,015	4,610,283	75	0.6	41,117,085	30,571,612
70	0.7	7,421,229	7,128,713	75	0.7	44,881,962	32,458,948
70	0.8	1,439,024	1,795,241	75	0.8	10,621,766	10,420,784
70	0.9	57,024	113,820	75	0.9	1,261,113	1,386,572

Intuitively, TH captures color availability in maximum saturated vertices. Table 1 shows that the sizes of available color sets during search in RB graphs are no greater than 3, and that the percentage of steps where $\mu \leq$ TH, for a fixed TH, varies more or less linearly with size.

Table 2 presents results about informedness of $PASS_h$ w.r.t. $SEWELL_h$. PASS is only significantly less informed for the harder problems of size 75 (notably $p=0.6$, 0.7) but is even more informed than SEWELL in the simpler ones. Since the overhead introduced by SEWELL is linear w.r.t. PASS, this validates the new heuristic.

5.2. Comparison with existing algorithms in literature

In this section we compare results obtained by PASS with leading ILP and DSATUR-based algorithms.

In general, exact ILP algorithms for VCP should be able to reduce the gap between LB and UB better than any DSATUR variant since they are able to produce tighter LBs by solving the continuous relaxation problem at each step of the branching scheme. In DSATUR-based algorithms, the lower bound computed at the beginning of the search or determined by the initial clique stays fixed in the remainder of the search.

In practice, however, reducing the LB–UB gap is important specifically if it helps to prove optimality. When it is not possible to conclude optimality within a certain time limit, LBs have an undeniable theoretical interest but do not help to produce admissible colorings (they might help to construct maximal cliques). For this reason the comparison between PASS and ILP-based algorithms has been based on either achieving optimality or tighter UBs; reported data on improving LBs, as in [20] or [24] has not been considered in this report.

Our main reference algorithm is MMT-BP [22], a leading column generation algorithm enhanced by an efficient metaheuristic procedure MMT described in [17]. MMT-BP uses MMT during initialization to improve the performance of the branching

scheme both by obtaining an initial feasible solution (an UB) as well as a starting pool of columns for the VCP-SC set cover model described in (2). We note that DSATUR is one of the algorithms used in MMT.

Results for the branch and cut approach employed in BC-Col by Méndez-Díaz and Zabala [19], are also reported. During initialization, BC-Col computes a maximal clique (LB) with a greedy algorithm and then runs DSATUR for 5 s to obtain an initial UB.

As reference DSATUR [7] we have selected the Michael Trick implementation available at http://mat.gsia.cmu.edu/COLOR/solvers/trick.c, which is commonly used for comparison purposes. We have also implemented an efficient SEWELL variant based on PASS. All three algorithms (DSATUR, SEWELL and PASS) have the same initialization stage: an exact leading maximum clique algorithm BB-MCP described in [31] is run for 5 s to obtain an initial LB (although in the majority of cases it finds an optimal clique in the first second). Vertices in this initial clique are then placed first and the remaining vertices are ordered by non-increasing degree prior to the start of the search.

Tests for the DSATUR variants were run on an I7-CPU 920@2.67 GHz with 6 GB of RAM (Windows O.S., Visual Studio–O_2). To allow an approximate (but meaningful) comparison on time results reported elsewhere, benchmark program *dfmax* is publicly available at URL ftp://dimacs.rutgers.edu/pub/dsj/clique/dfmax.c. Computing times obtained on different machines are calibrated w.r.t. the performance obtained on this program over a benchmark set of random instances. We are aware of the limitations of this (typical) procedure, but there is simply no other option when it is not possible to run all implementations in the same machine.

User times for *dfmax* in our machine were 0.031, 0.234, 1.419 and 5.336 s for r200.5, r300.5, r400.5 and r500.5, respectively.

Neither MMT-BP nor BC-Col was publicly available. Color expert Enrico Malaguti complied with tests for MMT-BP over our random benchmark set RB. The machine he used was a PIV@2.4 GHz with 2 GB RAM (Windows O. S.). Reported machine user times were 2.0 and 7.0 s for r400.5 and r500.5, respectively, and less than a second for r300.5 and r200.5. The averaged user time ratio for r400.5 and r500.5 between both machines is 1.332 in favor of our computer.

The original tests for BC-Col where run on a Sun ULTRA workstation @ 140 MHz with 288 MB of RAM, which took 24 s to compute r500.5, as reported in [22]. No other comparison results are available and the authors did not supply further information on our request. Based on r500.5, the user time ratio is 4.497 in favor of our computer. In [22] a similar comparison was established; we note, in both cases, the approximate nature of the ratio.

Table 3 reports tests over our random benchmark RB for the $G_{n,p}$ graph family. MMT-BP was run with a time limit of 1800 s; in the other three algorithms the limit was fixed at 1200 s (the calibrated time limit 1800/1.332=1351 s has been rounded in favor of MMT-BP). Times for MMT-BP shown in Table 3 (and reported by Malaguti specifically for this research) have been calibrated using the 1.332 ratio.

For each row entry in the table, 50 graphs were computed and results averaged (including entries for columns $\chi(G)$,$\omega(G)$). Column *Fail* is the number of graphs where optimality was not proved inside the time limit. Maximum clique column $\omega(G)$ is the LB of all DSATUR variants (as found by BB-MCP) and has been

Table 3
Performance over random benchmark RB. Times are averaged over 50 instances for each row and measured in seconds. Column *Fail* is the number of cases when the algorithm was unable to determine an optimum coloring. Time limit was fixed at 1800 s for MMT-BP and 1200 s for the rest to account for the different machines.

n	p	$\chi(G)$	**MMT-BP** [22]				$\omega(G)$	**DSATUR** [7]			**SEWELL** [26]			**PASS**		
			LB	**UB**	**Time**	**Fail**		**UB**	**Time**	**Fail**	**UB**	**Time**	**Fail**	**UB**	**Time**	**Fail**
60	0.1	4.0	4.0	0.1	0.2	0	3.4	4.0	0.0	0	4.0	0.0	0	4.0	0.0	0
60	0.2	5.5	5.1	134.0	191.4	**3**	4.5	5.5	0.0	0	5.5	0.0	0	5.5	0.0	0
60	0.3	7.0	7.0	30.8	44.0	**1**	5.5	7.0	0.0	0	7.0	0.0	0	7.0	0.0	0
60	0.4	8.9	8.3	183.0	261.5	**2**	6.5	8.9	0.1	0	8.9	0.1	0	8.9	0.1	0
60	0.5	10.7	10.1	95.2	135.9	**1**	8	10.7	0.4	0	10.7	0.5	0	10.7	0.2	0
60	0.6	12.9	12.5	22.3	31.9	0	9.7	12.9	0.6	0	12.9	0.6	0	12.9	0.3	0
60	0.7	15.6	15.3	8.7	12.4	0	12	15.6	0.5	0	15.6	0.6	0	15.6	0.2	0
60	0.8	19.1	19.0	2.1	3.0	0	16	19.1	0.3	0	19.1	0.3	0	19.1	0.1	0
60	0.9	25.7	25.7	0.2	0.2	0	24	25.7	0.0	0	25.7	0.0	0	25.7	0.0	0
70	0.1	4.0	4.0	0.2	0.3	0	3.6	4.0	0.0	0	4.0	0.0	0	4.0	0.0	0
70	0.2	6.0	5.8	32.8	46.8	0	4.6	6.0	0.0	0	6.0	0.0	0	6.0	0.0	0
70	0.3	7.8	7.1	605.1	864.5	**20**	5.6	7.8	0.3	0	7.8	0.5	0	7.8	0.2	0
70	0.4	9.7	9.1	470.0	671.4	**14**	7	9.7	2.8	0	9.7	4.4	0	9.7	1.7	0
70	0.5	11.8	11.2	437.2	624.6	**13**	8.3	11.8	10.8	0	11.8	15.2	0	11.8	5.5	0
70	0.6	14.1	13.6	156.4	223.4	**4**	10	14.1	16.3	0	14.1	20.9	0	14.1	8.4	0
70	0.7	17.2	16.9	57.4	82.0	0	13	17.2	30.0	0	17.2	33.4	0	17.2	14.3	0
70	0.8	21.4	21.2	12.9	18.5	0	17	21.4	6.1	0	21.4	8.1	0	21.4	2.6	0
70	0.9	28.5	28.5	0.4	0.6	0	26	28.5	0.2	0	28.5	0.4	0	28.5	0.1	0
75	0.1	4.0	4.0	16.1	23.0	0	3.6	4.0	0.0	0	4.0	0.0	0	4.0	0.0	0
75	0.2	6.0	6.0	36.1	51.6	**1**	4.6	6.0	0.0	0	6.0	0.0	0	6.0	0.0	0
75	0.3	8.0	7.5	181.6	259.4	**12**	5.7	8.0	0.4	0	8.0	0.8	0	8.0	0.2	0
75	0.4	10.0	9.5	357.5	510.7	**11**	7	10.0	12.8	0	10.0	17.7	0	10.0	6.8	0
75	0.5	12.1	11.8	163.4	233.5	**4**	8.6	12.1	67.6	0	12.1	90.5	0	12.1	35.4	0
75	0.6	14.9	14.3	487.5	696.5	**13**	10	14.9	153.4	0	14.9	153.4	**1**	14.9	72.2	0
75	0.7	18.0	17.6	131.2	187.5	**2**	13	18.0	141.7	**1**	18.0	166.4	**1**	18.0	76.6	0
75	0.8	22.4	22.1	32.6	46.6	0	18	22.4	42.4	0	22.4	51.5	0	22.4	17.2	0
75	0.9	1.0	29.9	21.0	29.9	0	26	1.0	4.2	0	1.0	5.2	0	1.0	1.6	0
80	0.1	4.3	4.0	334.7	478.1	**10**	3.8	4.3	0.0	0	4.3	0.0	0	4.3	0.0	0
80	0.2	6.3	6.0	387.3	553.2	**15**	4.8	6.3	0.2	0	6.3	0.3	0	6.3	0.1	0
80	0.3	8.2	7.9	324.0	462.8	**11**	5.8	8.2	11.3	0	8.2	17.8	0	8.2	7.3	0

Table 4
Comparison between PASS and other exact algorithms over a subset of the DIMACS color benchmark. Bold face time entries for MMT-BP and PASS indicate that an optimum coloring was found by one of them but not the other. Cursive time entries indicate that optimality was proved at least 5 times faster w.r.t. each other. Reported times for BC-Col and MMT-BP are taken directly from the indicated sources and have not been calibrated. We estimate the 4.49 and 1.33 ratios in favor of our computer for BC-Col and MMT-BP, respectively. User times are measured in seconds.

name	n	m	χ	BC-Col [19]			MMT-BP[22]			LBDS	DSATUR [7]		SEWELL [26]		PASS	
				LB	UB	Time	LB	UB	Time		UB	Time	UB	Time	UB	Time
queen8_8	64	728	9	9	9	3.0	9	9	3.6	8	9	8.3	9	3.4	9	3.0
queen8_12	96	1368	12	12	12	init	12	12	0.2	12	12	0.0	12	0.0	12	0.0
queen9_9	81	2112	10	9	11	tout	10	10	36.6	9	10	tout	10	tout	10	466.0
queen10_10	100	2940	11	10	12	tout	11	11	**686.9**	10	12	tout	12	tout	12	tout
queen11_11	121	3960	11	11	14	tout	11	11	**1865.7**	11	13	tout	13	tout	13	tout
queen12_12	144	5192	12	12	15	tout	12	13	tout	12	14	tout	14	tout	14	tout
queen13_13	169	6656	13	13	16	tout	13	14	tout	13	15	tout	16	tout	15	tout
queen14_14	196	8372	14	14	17	tout	14	15	tout	14	17	tout	17	tout	16	tout
queen15_15	225	10,360	15	15	18	tout	15	16	tout	15	18	tout	18	tout	18	tout
queen16_16	256	12,640	16	16	20	tout	16	17	tout	16	19	tout	19	tout	19	tout
qg.order30	900	26,100	30	30	30	init	30	30	0.2	30	30	0.6	30	0.1	30	*0.0*
qg.order40	1600	62,400	40	40	42	tout	40	40	2.9	40	41	tout	40	0.5	40	*0.2*
qg.order60	3600	212,400	60	60	63	tout	60	60	**3.8**	60	63	tout	60	66.1	61	tout
myciel6	95	755	7	5	7	tout	4	7	tout	2	7	tout	7	tout	7	tout
myciel7	191	2360	8	5	8	tout	5	8	tout	2	8	tout	8	tout	8	tout
miles250	128	387	8	8	8	init	8	8	5.0	8	8	0.0	8	0.0	8	*0.0*
miles500	128	1170	20	20	20	init	20	20	3.7	20	20	0.0	20	0.0	20	*0.0*
miles750	128	2113	31	31	31	init	31	31	0.2	31	31	0.0	31	0.0	31	*0.0*
miles1000	128	3216	42	42	42	0.2	42	42	0.2	42	42	0.0	42	0.0	42	*0.0*
miles1500	128	5198	73	73	73	0.1	73	73	0.1	73	73	0.0	73	0.0	73	*0.0*
anna	138	493	11	11	11	init	11	11	3.6	11	11	0.0	11	0.0	11	*0.0*
huck	74	301	11	11	11	init	11	11	0.2	11	11	0.0	11	0.0	11	*0.0*
jean	80	254	10	10	10	init	10	10	0.2	10	10	0.0	10	0.0	10	*0.0*
david	87	406	11	11	11	init	11	11	0.2	11	11	0.0	11	0.0	11	*0.0*
fpsol2.i.1	496	11,654	65	65	65	0.6	65	65	10.6	65	65	0.0	65	0.0	65	*0.0*
fpsol2.i.2	451	8691	30	30	30	1.2	30	30	11.2	30	30	0.0	30	0.0	30	*0.0*
fpsol2.i.3	425	8688	30	30	30	1.1	30	30	10.0	30	30	0.0	30	0.0	30	*0.0*
inithx.i.1	864	18,707	54	54	54	init	54	54	21.0	54	54	0.0	54	0.0	54	*0.0*
inithx.i.2	645	13,979	31	31	31	init	31	31	9.2	31	31	0.0	31	0.0	31	*0.0*
inithx.i.3	621	13,969	31	31	31	init	31	31	9.9	31	31	0.0	31	0.0	31	*0.0*
mug88_1	88	146	4	3	4	11.0	4	4	9.6	3	4	77.1	4	120.0	4	324.0
mug88_25	88	146	4	3	4	184.0	4	4	10.6	3	4	295.0	4	215.0	4	191.0
mug100_1	100	166	4	3	4	60.0	4	4	14.4	3	4	786.0	4	tout	4	tout
mug100_25	100	166	4	3	4	60.0	4	4	12.0	3	4	978.0	4	tout	4	tout
mulsol.i.1	197	3925	49	49	49	init	49	49	0.2	49	49	0.0	49	0.0	49	*0.0*
mulsol.i.2	188	3885	31	31	31	init	31	31	4.7	31	31	0.0	31	0.0	31	*0.0*
mulsol.i.3	184	3916	31	31	31	init	31	31	0.2	31	31	0.0	31	0.0	31	*0.0*
mulsol.i.4	185	3946	31	31	31	init	31	31	0.2	31	31	0.0	31	0.0	31	*0.0*
mulsol.i.5	186	3973	31	31	31	init	31	31	6.0	31	31	0.0	31	0.0	31	*0.0*
school1	385	19,095	14	14	14	init	14	14	0.4	14	14	0.0	14	0.1	14	0.1
school1_nsh	352	14,612	14	14	14	init	14	14	17.0	14	14	0.6	14	0.5	14	*0.4*
le450_5c	450	9803	5	5	5	init	5	5	0.1	5	5	0.0	5	73.6	5	*0.0*
le450_5d	450	9757	5	5	10	tout	5	5	*0.2*	5	5	1060.0	11	tout	5	98.1
le450_15a	450	8168	15	15	17	tout	15	15	**0.4**	15	17	tout	18	tout	16	tout
le450_15b	450	8169	15	15	17	tout	15	15	**0.2**	15	16	tout	16	tout	16	tout
le450_15c	450	16,680	15	15	24	tout	15	15	**3.1**	15	24	tout	22	tout	22	tout
le450_15d	450	16,750	15	15	23	tout	15	15	**3.8**	15	23	tout	23	tout	23	tout
le450_25a	450	8260	25	25	25	init	25	25	0.1	25	25	0.0	25	0.0	25	*0.0*
le450_25b	450	8263	25	25	25	init	25	25	0.1	25	25	0.0	25	0.0	25	*0.0*
1-FullIns_4	93	593	5	5	5	0.1	4	5	tout	3	5	tout	5	tout	5	0.2
1-FullIns_5	282	3247	6	4	6	tout	4	6	tout	3	6	tout	6	tout	6	tout
2-FullIns_3	52	201	5	5	5	0.1	5	5	2.9	4	5	84.6	5	79.0	5	*0.0*
2-FullIns_4	212	1621	6	5	6	tout	5	6	tout	4	6	tout	6	tout	6	tout
2-FullIns_5	852	12,201	7	5	7	tout	5	7	tout	4	7	tout	7	tout	7	tout
3-FullIns_3	80	346	6	6	6	0.1	6	6	**2.9**	5	6	tout	6	tout	6	tout
3-FullIns_4	405	3524	7	6	7	tout	6	7	tout	5	7	tout	7	tout	7	tout
3-FullIns_5	2030	33,571	8	6	8	tout	5	8	tout	5	8	tout	8	tout	8	tout
4-FullIns_3	114	541	7	7	7	3.0	7	7	**3.4**	6	7	tout	7	tout	7	tout
4-FullIns_4	690	6650	8	7	8	tout	7	8	tout	6	8	tout	8	tout	8	tout
4-FullIns_5	4146	77,305	?	6	9	tout	6	9	tout	6	9	tout	9	tout	9	tout
5-FullIns_3	154	792	8	8	8	2.0	8	8	**4.6**	7	8	tout	8	tout	8	tout
5-FullIns_4	1085	11,395	?	8	9	tout	8	9	tout	7	9	tout	9	tout	9	tout
1-Insertions_4	67	232	5	5	5	2.0	3	5	tout	2	5	tout	5	tout	5	**240.0**

Table 4 (*continued*)

name	n	m	χ	BC-Col [19]			MMT-BP[22]			LBDS	DSATUR [7]		SEWELL [26]		PASS	
				LB	UB	Time	LB	UB	Time		UB	Time	UB	Time	UB	Time
1-Insertions_5	202	1227	?	4	6	tout	3	6	tout	2	6	tout	6	tout	6	tout
1-Insertions_6	607	6337	?	4	7	tout	3	7	tout	2	7	tout	7	tout	7	tout
2-Insertions_4	149	541	4	4	5	tout	3	5	tout	2	5	tout	5	tout	5	tout
2-Insertions_5	597	3936	?	3	6	tout	2	6	tout	2	6	tout	6	tout	6	tout
3-Insertions_3	56	110	4	4	4	1.0	3	4	tout	2	4	2.6	4	0.3	4	0.6
3-Insertions_4	281	1046	?	3	5	tout	3	5	tout	2	5	tout	5	tout	5	tout
3-Insertions_5	1406	9695	?	3	6	tout	2	6	tout	2	6	tout	6	tout	6	tout
4-Insertions_3	79	156	4	3	4	tout	3	4	tout	2	4	tout	4	96.9	4	**351.0**
4-Insertions_4	475	1795	?	3	5	tout	2	5	tout	2	5	tout	5	tout	5	tout
ash331GPIA	662	4185	4	4	4	51.0	4	4	45.9	3	5	tout	4	0.1	4	*0.0*
ash608GPIA	1216	7844	4	4	4	692.0	3	4	tout	3	5	tout	5	tout	4	**0.1**
ash958GPIA	1916	12,506	4	4	5	tout	3	4	tout	3	5	tout	5	tout	4	**0.4**
abb313GPIA	1557	53,356	?	8	10	tout	7	9	tout	8	10	tout	10	tout	10	tout
will199GPIA	701	6772	7			tout	7	7	**80.7**	6	7	tout	7	tout	7	tout
DSJC125.1	125	736	5	5	5	0.9	5	5	142.0	4	5	0.0	5	0.0	5	*0.0*
DSJC125.5	125	3891	17	17	13	tout	16	17	tout	10	19	tout	19	tout	19	tout
DSJC125.9	125	6961	44	42	47	tout	43	44	tout	34	47	tout	47	tout	46	tout
DSJC250.1	250	3218	?	5	9	tout	5	8	tout	4	9	tout	9	tout	9	tout
DSJC250.5	250	15,668	?	13	36	tout	15	28	tout	12	35	tout	36	tout	34	tout
DSJC250.9	250	27,897	?	48	88	tout	71	72	tout	39	86	tout	88	tout	82	tout
DSJC500.1	500	12,458	?	5	15	tout	4	12	tout	5	15	tout	15	tout	14	tout
DSJC500.5	500	62,624	?	13	63	tout	11	48	tout	13	62	tout	61	tout	62	tout
DSJC1000.1	1000	49,629	?	6	26	tout	5	20	tout	6	25	tout	25	tout	25	tout
DSJC1000.5	1000	249,826	?	15	116	tout	13	92	tout	14	110	tout	110	tout	110	tout
DSJC1000.9	1000	449,449	?	65	301	tout	51	226	tout	57	300	tout	300	tout	300	tout
DSJR500.1	500	3555	12	12	12	init	12	12	35.3	12	12	0.0	12	0.0	12	0.0
DSJR500.1c	500	121,275	85	78	88	tout	85	85	**288.5**	80	87	tout	88	tout	85	tout
DSJR500.5	500	58,862	122	119	130	tout	122	122	**342.2**	120	130	tout	130	tout	130	tout
latin_sq._10	900	307,350	?	90	129	tout	90	108	tout	90	130	tout	130	tout	130	tout
games120	120	638	9	9	9	init	9	9	0.2	9	9	0.0	9	0.0	9	0.0
zeroin.i.1	211	4100	49	49	49	init	49	49	4.4	49	49	0.0	49	0.0	49	*0.0*
zeroin.i.2	211	3541	30	30	30	init	30	30	4.5	30	30	0.0	30	0.0	30	*0.0*
zeroin.i.3	206	3540	30	30	30	init	30	30	3.6	30	30	0.0	30	0.0	30	*0.0*
wap01a	2368	110,871	?	41	46	tout	40	43	tout	41	48	tout	47	tout	46	tout
wap02a	2464	111,742	?	40	45	tout	40	42	tout	40	49	tout	46	tout	46	tout
wap05a	905	43,081	50	50	51	tout	50	50	293.2	50	50	0.0	50	0.0	50	*0.0*
wap06a	947	43,571	40	40	44	tout	40	40	**175.0**	40	49	tout	48	tout	47	tout
wap07a	1809	103,368	?	40	46	tout	40	42	tout	40	47	tout	45	tout	44	tout
wap08a	1870	104,176	?	40	47	tout	40	42	tout	40	45	tout	45	tout	45	tout

placed immediately to the left of them for clarity. *Time* entries average user times in seconds.

In this scenario, PASS clearly outperforms the other algorithms, proving optimality faster in the majority of cases (and never performing worse). The ratio of improvement in time w.r.t. DSATUR and SEWELL rises in moderately difficult instances to nearly triple (e.g. (60,0.7) or (70, 0.8)) and decreases with the more difficult ones to around double (e.g. (75, 0.5)). This can be explained by the fact that there are fewer vertices tied at maximum saturation degree in the former case so $PASS_h$ is more effective. On the easier less dense instances, we report similar performances in all three algorithms up to a tenth of a second. Out of the 1500 instances in RB, SEWELL failed in two (rows (75, 0.6), (75, 0.7)) and DSATUR in one ((75, 0.6)). PASS was the only algorithm to finish the full set inside the limit.

Interestingly, leading branch and price MMT-BP was totally outperformed in RB, even by the older DSATUR variants, which points at a structural weakness of the set cover VCP-SC model (2). Specifically, it performed worse on the *a priori* easier less dense graphs (e.g. (70, 0.3), (80, 0.1–0.3)). This fact has not been mentioned explicitly in existing literature to our knowledge, probably because a systematic test on random graphs has been somewhat lacking in recent branch and price reports, (i.e.[18,23,24]) and constitutes, in the opinion of the author, an additional contribution of this research.

Further evidence of a possible structural weakness was confirmed by tests carried out by VCP expert Stefano Gualandi (complying to our request) over a subset of RB fed to the branch and price algorithm described in [23]. Reported times did not essentially improve MMT-BP, so they have not been included in the report. Of interest is the fact that in the original column generation algorithm LPCOLOR [21], the same trend appeared but in a smaller scale. The reduced computing power at the time could only handle a small subset of random graphs.

Column generation techniques applied to VCP require to solve, at each step, a *maximum weighted stable set* slave problem (MWSS), which is NP-hard in the general case (c.f. [21] or [18] for a detailed explanation) so effective procedures to solve MWSS become critical in overall performance. As density decreases, the slave problem at each step of the branching scheme is also harder. This would explain why MMT-BP fails to compute optimality for instances of size 80 in 36 cases out of 150, whereas DSATUR based algorithms solve the whole subset without difficulty.

Table 4 reports comparisons between the four previous algorithms and branch-and-cut BC-Col. Most column headers have the same interpretation as in Table 3. LB_{DS} entries (column 10) refer to the lower bound obtained during the initial stage of all three DSATUR variants which, in the majority of cases, is a maximum clique. To emphasize the comparison between PASS and MMT-BP, bold face time entries for both algorithms indicate that an

optimum was found by one and not the other. Cursive time entries indicate that optimality was proved at least five times faster w.r.t. each other.

Time limits for BC-Col and MMT-BP were fixed at 7200 and 2400 seconds respectively, so 1500 s was chosen as time limit for all DSATUR variants to account for the difference in machine times. We note that time limits based on available *dfmax* data should be approximately 1600 and 1800 s, respectively, so again the rounding in the tests is in favor of the previous existing algorithms.

BC-Col and MMT-BP entries in the table were taken directly from the original sources ([19,22] respectively) and have not been calibrated in Table 4. This seemed preferable to correcting times based on the available weak evidence, and was the same policy adopted in [22] w.r.t. BC-Col user times. The difference in machines is, however, explicitly mentioned when needed in the comparative analysis.

With the exception of *qg.order60*, random graph *DJSC500.5*, *school1* and the *mug* family, PASS is clearly superior to the other two DSATUR variants, finding faster and/or better UBs in many cases. This validates the new tie-breaking strategy for large structured DIMACS graphs.

Also PASS has improved performance of DSATUR-based algorithms w.r.t. BC-Col compared to reported results in [22]: there are many instances where the UB found by PASS is tighter (e.g. 5 cases in the *queen* family, 4 in the *le450*, *ash958GPIA*, 8 in the *DSJC* random family, etc.). BC-Col, on the other hand, finds better bounds in 5 cases.

Column generation based MMT-BP outperforms PASS in large DIMACS graphs. There are 14 cases where MMT-BP finds an optimum coloring whereas PASS cannot prove optimality. Moreover in the *queen*, *le450*, *DSJC* and *wap* families MMT-BP also captures structure better and tends to find better bounds.

On the other hand, comparing with the reported results in [22] the difference between UBs is now tighter on average. Furthermore, PASS finds an optimum coloring in 6 cases where MMT-BP exceeds the time limit. We also note that when PASS is able to prove optimality, it is usually faster than MMT-BP, since the latter is much more complex and requires hard instances for column generation overhead to pay off. Specifically, it takes PASS more than 5 times less than its counterpart (machine differences accounted for) to compute chromatic degree in more than 30 cases. In the opinion of the author, this makes it a very useful tool in practice (notice that DSATUR is also used both in BC-Col and MMT-BP during initialization, so PASS could be used to improve this stage in both cases).

6. Conclusions

This paper describes a new exact coloring algorithm (PASS) based on the well known DSATUR algorithm of Brélaz and a later improvement proposed by Sewell. PASS introduces a new tiebreaking strategy, which can be computed much faster than Sewell because it is restricted to a subset of vertices. Reported results show that it is no less informed on average and improves overall performance when selectively applied in steps where the number of available colors for maximal saturated vertices is below a given threshold.

Today DSATUR-based algorithms are still very much employed in practice because of their simplicity and efficiency. In many cases they are also applied at some stage in metaheuristics or in the more complex exact algorithms. Moreover, reported results reveal that PASS clearly outperforms a leading column generation algorithm in a benchmark of 1500 random graphs. In the opinion of the authors the above reasons make PASS a very useful tool for exact VCP in real-life applications.

Acknowledgments

This work is funded by the Spanish Ministry of Science and Technology (ARABOT: DPI 2010-21247-C02-01) and supervised by CACSA whose kindness we gratefully acknowledge. We also want to thank Enrico Malaguti and Stefano Gualandi for running two leading branch and price solvers on our random benchmark RB.

References

[1] Garey. M, Johnson D. Computers and intractability: a guide to the theory of NP completeness. San Francisco, California: Freeman; 1979.
[2] Chow Fred C, John L. Hennessy. The priority-based coloring approach to register allocation. ACM Trans. Program. Lang. Syst. 1990;12(4):501–36.
[3] Zufferey N, Amstutz P, Giaccari. P. Graph coloring approaches for a satellite range scheduling problem. Journal of Scheduling 2008;11(4):263–77.
[4] Gamache M, Hertz A, Ouellet. JO. A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. Computers & Operations Research 2007;34(8):2384–95.
[5] Leighton. FT, Graph A. Coloring algorithm for large scheduling problems. Journal of Research of the National Bureau of Standards 1979;84(6):489–506.
[6] Barnier N, Brisset Graph P. Coloring for air traffic flow management. In: Proceedings of the CPAIOR'02 fourth international workshop on integration of AI and OR techniques in constraint programming for combinatorial optimisation problems. Le Croisic, France; 2002. p. 133–47.
[7] Brélaz D. New methods to color the vertices of a graph. Communications of the ACM 1979;22(4):251–6.
[8] Bollobás B., Thomason A. Random graphs of small order. Random graphs '83. Annals of discrete mathematics, vol. 28, Section 6. North-Holland Publishing Co.; 1985. p. 47–97.
[9] Culberson J. Iterated greedy graph coloring and the difficulty landscape. Technical Report 92-07. Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada; 1992.
[10] Hertz A, de Werra D. Using tabu search techniques for graph coloring. Computing 1987;39(4):345–51.
[11] Johnson DD, Aragon CR, McGeoch LA, Schevon C. Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. Operations Research 1991;39:378–406.
[12] Morgenstern CA. Distributed coloration neighborhood search. In: Johnson DS, Trick MA, editors. Cliques, coloring, and satisfiability: 2nd DIMACS. Implementation Challenge, 1993. DIMACS series in discrete mathematics and theoretical computer science. American Mathematical Society; 1996. p. 335–58.
[13] Funabiki N, Higashino T. A minimal-state processing search algorithm for graph coloring problems. IEICE Transactions on Fundamentals 2000;E83-A:1420–30.
[14] Fleurent C, Ferland J. Object-oriented implementation of heuristics search methods for graph coloring, maximum clique, and satisfiability. DIMACS series in discrete mathematics and theoretical computer science. American Mathematical Society, vol. 26; 1996. p. 619–52.
[15] Galinier P, Hao JK. Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization 1999;3:379–97.
[16] Blöchliger I, Zufferey. N. A reactive tabu search using partial solutions for the graph coloring problem. Computers & Operations Research 2008;35:960–75.
[17] Malaguti E, Monaci M, Toth. P, Metaheuristic A. Approach for the vertex coloring problem. Informs Journal of Computing 2008;20(2):302–16.
[18] Malaguti E, Toth. P. A survey on vertex coloring problems. International Transactions in Operational Research 2010;17(1):1–34.
[19] Mendez-Diaz I, Zabala. P. A branch-and-cut algorithm for graph coloring. Discrete Applied Mathematics 2006;154(5):826–47.
[20] Méndez-Díaz I, Zabala P, Cutting Plane A. Algorithm for graph coloring. Discrete Applied Mathematics 2008;156:159–79.
[21] Mehrotra A, Trick. MA. A column generation approach for graph coloring. INFORMS Journal on Computing 1996;8(4):344–54.
[22] Malaguti E, Monaci M, Toth P. An exact approach for the Vertex Coloring Problem. Discrete Optimization 2011;8(2):174–90.
[23] Gualandi S, Malucelli F. Exact solution of graph coloring problems via constraint programming and column generation. Technical report. Optimization Online; 2010.
[24] Held S, Sewell EC, Cook W. Safe lower bounds for graph coloring. Accepted for IPCO; 2011.
[25] Hansen P, Labbé M, Schindl. D. Set covering and packing formulations of graph coloring: algorithms and first polyhedral results. Discrete Optimization 2009;6:135–47.
[26] Sewell E. An improved algorithm for exact graph coloring. In: Trick MA, Johnson DS, editors. Cliques, coloring, and satisfiability. Proceedings of the second DIMACS implementation challenge, vol. 26. American Mathematical Society; 1996. p. 359–73.
[27] Matula. DW, Marble. G, Issacson. JD. Graph coloring algorithms. In: Read RC, editor. Graph theory and computing. New York: Academic Press; 1972.

[28] Welsh DJA, Powell. MB. An upper bound for the chromatic number of a graph and its application to timetabling problems. The Computer Journal 1967;10(1):85–6.
[29] Brown JR. Chromatic scheduling and the chromatic number problem. Management Science 1972;19(4):456–63 Application Series, Part 1.
[30] Johnson DS, Trick MA., editors. Cliques, coloring, and satisfiability: 2nd DIMACS implementation challenge, 1993. DIMACS series in discrete mathematics and theoretical computer science. American Mathematical Society; 1996.
[31] San Segundo P, Rodriguez-Losada D, Jimenez A. An exact bit-parallel algorithm for the maximum clique problem. Computers & Operations Research 2011;38(2):571–81.