An improved DSATUR-based Branch and Bound for the Vertex Coloring Problem

Fabio Furini, Virginie Gabrel, Ian-Christopher Ternier

PSL, Université Paris Dauphine, CNRS, LAMSADE UMR 7243
75775 Paris Cedex 16, France.
fabio.furini@dauphine.fr
virginie.gabrel@dauphine.fr
ianchristopher.ternier@dauphine.fr

Abstract

Given an undirected graph, the Vertex Coloring Problem (VCP) consists of assigning a color to each vertex of the graph in such a way that two adjacent vertices do not share the same color and the total number of colors is minimized. DSATUR-based Branch and Bound (DSATUR) is an effective exact algorithm for the VCP. One of its main drawback is that a lower bound is computed only once and it is never updated. We introduce a reduced graph which allows the computation of lower bounds at nodes of the branching tree. We compare the effectiveness of different classical VCP bounds, plus a new lower bound based on the 1-to-1 mapping between VCPs and Stable Set Problems. Our new DSATUR outperforms the state of the art for random VCP instances with high density, significantly increasing the size of instances solved to proven optimality. Similar results can be achieved for a subset of high density DIMACS instances.

keywords: Graph Coloring, DSATUR, Branch and Bound.

1. Introduction

Given an undirected graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges, a *coloring* C of G is a partition of V into k non empty *stable sets*: $C = \{V_1, \dots, V_k\}$, where all vertices belonging to V_i are colored with the same color i ($i = 1, \dots, k$). The *chromatic number* of G, denoted by $\chi(G)$, is the minimum number of stable sets (or equivalently colors) in a coloring of G and the *Vertex Coloring Problem* (VCP) is the problem of determining the chromatic number of the graph G. The VCP is one of the classical NP-hard problems (see Garey and Johnson [5]) in graph theory with application in many areas including: scheduling, timetabling, register allocation, frequency assignment, communication networks and many others (see [4, 7, 10, 12, 22, 17, 23, 31]). We adress the interested reader to Malaguti and Toth [25] for a complete survey on the topic. A preliminary version of this manuscript appeared in Furini et al. [39].

The VCP has received a large amount of attention in the last decades and many articles investigated an exact implicit enumeration algorithm called *DSATUR-based Branch and Bound* (DSATUR), first introduced by Brélaz [3] and then improved by Sewell [9] and San Segundo [27]. It is a Branch-and-Bound algorithm where at each node of the branching tree the children nodes are created by assigning feasible colors to a non-colored vertex; thus at each node of the branching tree, we have a partial coloring of G and at each leaf we have a coloring of G. Formally, a *partial coloring* $\tilde{C}$ of G is a partition of a subset of vertices $\tilde{V} \subset V$ into $\tilde{k}$ stable sets or colors ($\tilde{C} = \{\tilde{V}_1, \dots, \tilde{V}_{\tilde{k}}\}$), while the remaining vertices $V \setminus \tilde{V}$ are non-colored. Many rules have been proposed in the literature to determine the sequence of vertices to be colored (see Section 2 for further details on DSATUR). It is worth mentioning that DSATUR has also been successfully applied to other variants of VCPs, see for example Méndez-Diaz et al. [38].

In all the DSATUR versions proposed in the literature, a lower bound is computed once at the root node of the algorithm as a heuristic maximal clique and it is never updated. A second trivial lower bound also used in the literature is the number of colors $\tilde{k}$ of a partial coloring $\tilde{C}$. The principal idea of this manuscript consists of updating and improving the quality of the lower bound during the branching scheme. In order to do that, we introduce a *Reduced Graph* associated to a partial coloring which allows to update the lower bounds. We implement and compare the classical lower bounds for VCP, i.e, the clique number, a bound based on the stability number, the fractional chromatic number and the Hoffman bound. Since all these bounds turn out to be useful only in reducing the number of nodes of DSATUR but not the computing time, we investigate a new bound based on a 1-to-1 mapping between VCPs and Stable Sets Problems. Thanks to this new bound we manage to reduce both the number of nodes and the computing time for random VCP instances with high density and for a subset of high density DIMACS instances.

For random graphs DSATUR outperforms other exact algorithms, see San Segundo [27]. For DIMACS VCP instances instead, Branch-and-Price algorithms based on the Integer Linear Programming (ILP) formulation of Mehrotra and Trick [8] guarantees the best performances. Many articles study ways of improving this class of exact algorithms. Malaguti et al. [28] focus on finding the most efficient way to solve the pricing subproblem, proposing a tabu-search metaheuristic which speeds up the computational convergence. Gualandi and Malucelli [29] propose instead to solve the pricing subproblem using constraint programming techniques. Cook et al. [30] also work on this formulation tackling numerical difficulties in the context of column generation, deriving a way of computing numerically safe bounds. Finally, Morrison et al. [33] work on new branching rules that preserve the graph structure at each node of the branching tree.

The remainder of the paper is organized as follows. In Section 2, we recall DSATUR and present different vertex selection rules. In Section 3, we present and computationally compare the VCP lower bounds. In Section 4, we introduce the Reduced Graph used to compute VCP lower bounds starting from a partial coloring. In Section 5, we discuss extensive computational results and depict further possible lines of research on the topic.

2. State of the art: DSATUR-based Branch and Bound

In this section, we recall the DSATUR-based Branch and Bound called for brevity DSATUR in the following. We base our review on the notation offered by San Segundo [27]. The algorithm is based on DSATUR$_h$ (see Brélaz [3]) which is a greedy heuristic algorithm where each vertex $u \in V$ is iteratively colored with a feasible color. Given a partial coloring $\tilde{C}$ and a vertex $u \in V$, the *saturation degree* DSAT$(u, \tilde{C})$ corresponds to the number of different colors in its neighbourhood $N(u)$. At each iteration of DSATUR$_h$, the vertex with the highest DSAT value is colored until a feasible heuristic coloring of the entire graph G is obtained, the number of colors is a valid upper bound for $\chi(G)$.

An exact branch and bound algorithm can be derived from DSATUR$_h$. Given a partial coloring and an uncolored vertex $u \in V$, instead of fixing its color in a greedy way, a branching tree is created by coloring u with all the feasible colors already used in the partial coloring plus a new one. At each node of this branching tree, we are given a partial coloring $\tilde{C}$ with $\tilde{k}$ colors, an upper bound (UB) and a lower bound (LB) on $\chi(G)$. Trivially, $\tilde{k}$ can be used as a lower bound for $\chi_{\tilde{C}}(G)$, i.e, the chromatic number of G partially colored by $\tilde{C}$. A lower bound for $\chi(G)$ can be obtained executing DSATUR$_h$, since the first colored vertices with different colors necessarily form a clique in G. Both bounds are weak and the maximal heuristic clique found by DSATUR$_h$ is typically never updated during the execution of DSATUR.

In Algorithm 1 and Algorithm 2, we give the pseudo code of DSATUR. Precisely, Algorithm 1 receives in input the graph G to be colored and the lower/upper bounds computed via DSATUR$_h$ and it produces in output the optimal coloring C^* of value $\chi(G)$. In Algorithm 2, the mechanism to create the children nodes is described, i.e, after an uncolored vertex is selected and if $\max\{\tilde{k}, LB\} < UB$, up to $\tilde{k}+1$ children nodes are created by coloring the selected vertex with all the feasible colors in $\tilde{C}$ plus a new one. In case all vertices are colored and $\tilde{k} < UB$, the best incumbent solution value and the best solution are updated respectively.

Algorithm 1: DSATUR

Data: $G = (V, E)$: graph to color
Result: optimal coloring C^* of value $\chi(G)$
$LB, UB \leftarrow$ DSATUR$_h$;
DSATUR($\emptyset$);
return C^*

Algorithm 2: DSATUR($\tilde{C}$)

if *all the vertices are colored* **then**
    if $\tilde{k} < UB$ **then**
        $C^* \leftarrow \tilde{C}, UB \leftarrow \tilde{k}$;
    end
else
    if $\max\{\tilde{k}, LB\} < UB$ **then**
        select a non-colored vertex v;
        for *every feasible color* $i \in \tilde{C}$ *plus a new one* **do**
            $\widehat{C} \leftarrow \tilde{C}$, add v in $\widehat{V}_i$;
            DSATUR($\widehat{C}$);
        end
    end
end

The basic Vertex Selection Rule (VSR), proposed in Brélaz [3], consists of coloring the vertex with the maximum DSAT value, thus it minimizes the number of children nodes. During the execution of DSATUR, it often happens that many different vertices share the same maximum DSAT value, i.e., creating possible ties. Rules to break ties have been introduced in the literature:

(i) In Brélaz [3], ties are broken considering the maximum degree or, in case of further ties, the lexicographical order is used. The complexity of this rule is $O(n^2)$.

(ii) In Sewell [9] instead, ties are broken considering the maximum number of common available colors in the neighborhood of uncolored vertices. The complexity of this rule is $O(n^3)$.

(iii) In San Segundo [27], the Sewell rule is extended considering only uncolored vertices that are also candidates in the tie. In the worst case, the complexity is the same as Sewell's rule but it is faster on average.

In our implementation of DSATUR we follow the VSR proposed in San Segundo [27], this VSR has computationally proven to produce the smallest branching tree and accordingly the best computing time (see Section 5 for further details on our implementation of DSATUR).

3. Lower bounds for the Vertex Coloring Problem

In this section we review the classical lower bounds for the VCP. In addition we present a new bound based on a 1-to-1 mapping between VCPs and Stable Set Problems. For each bound we discuss the framework used to compute it and its computational complexity. The focus of this manuscript is to explore the idea of using these lower bounds to speed up the convergence of DSATUR. Accordingly not only the strength of these bounds is important but also the computing time necessary to obtain them. Thus, we conclude this section with an extensive computational comparison with a special attention on their potential impact on the performances of DSATUR.

3.1 Lower bounds review

Clique number $\omega(G)$. Recalling that a *clique* is a subset of fully connected vertices, the *clique number* $\omega(G)$ is the maximal size of a clique of G. The following holds:

$$\chi(G) \geq \omega(G) \tag{1}$$

This lower bound comes from the fact that in any clique all vertices should have different colors. Trivially any heuristically found clique of size $\omega^h(G)$ also provides a valid lower bound for the VCP ($\chi(G) \geq \omega^h(G)$). In our computational tests, $\omega^h(G)$ corresponds to the value of the heuristic clique produced by DSATUR$_h$. Computing the clique number is NP-hard (see Garey and Johnson [5]) but in practice very effective exact solvers are available in the literature. We address the interested reader to Wu and Hao [36] for a recent survey on the topic. In our computational tests, we decided to use one of the most efficient clique solvers, i.e., the combinatorial Branch and Bound and Dynamic Programming based exact algorithm named Cliquer (see [11]).

Lower bound based on the stability number $\chi_\alpha(G)$. Recalling that a *stable set* is a subset of fully disconnected vertices, the *stability number* $\alpha(G)$ is the maximal size of a stable set of G. The following holds:

$$\chi(G) \geq \chi_\alpha(G) = \left\lceil \frac{n}{\alpha(G)} \right\rceil \tag{2}$$

Since a coloring is a partition into stable sets, the best we can hope for is having all stable sets of maximal size (see Schrijver [34] for further details). Computing $\alpha(G)$ is an NP-hard problem (see Garey and Johnson [5]), but since $\alpha(G)$ is equivalent to the clique number $\omega(\bar{G})$ of the *complement graph* $\bar{G} = (V, \bar{E}$[1]$)$, we use Cliquer to efficiently obtain it.

Fractional Coloring number $\chi_f(G)$. Following the notation proposed in Schrijver [34], the Fractional Coloring number $\chi_f(G)$ is the minimum value of $\lambda_1 + \cdots + \lambda_k$ with $\lambda_1, \ldots, \lambda_k \in \mathbb{R}_+$ such that there exist stable sets $S_1, \ldots, S_k$ with

$$\lambda_1 A^{S_1} + \cdots + \lambda_k A^{S_k} = 1.$$

Where for any stable set S, A^S denotes the incidence vector of S in $\mathbb{R}^{|V|}$; that is for any $v \in G$:

[1] $\bar{E} = \{(i,j) : i,j \in V, i \neq j, (i,j) \notin E\}$

$$A^S(v) := \begin{cases} 1 & \text{if } v \in S, \\ 0 & \text{otherwise.} \end{cases}$$

The Fractional Coloring number corresponds to the linear programming relaxation of the VCP formulation proposed by Mehrotra and Trick ([8]) and it is NP-hard to compute (see Grötschel et al. [6]). The following holds:

$$\chi(G) \geq \chi_f^*(G) = \lceil \chi_f(G) \rceil \tag{3}$$

It is well known that $\chi_f(G)$ provides strong VCP bounds, but it requires Column Generation (CG) techniques to be computed. All state-of-the-art Branch-and-Price algorithms rely on this bound since during the branching scheme the stable sets previously generated in the branching nodes speed up the update of this lower bound (see [28] for further details). Unfortunately, this warm start trick cannot be directly translated into our framework when we update the lower bound in DSATUR due to the nature of the Reduced Graph which changes structures in function of the partial colorings (see Section 4 for further details). Accordingly, we do not go for efficiency in terms of computing time and we implement a basic CG approach directly using CPLEX to solve the Restricted Master Problem and to solve the subproblems (we refer the interested reader to [13] for further details on CG). Nevertheless this bound is kept in our analysis to evaluate the quality of the other bounds.

Hoffman number $\chi_H(G)$. Hoffman proves that the following is a lower bound for $\chi(G)$ (see Hoffman [1]):

$$\chi(G) \geq \chi_H^*(G) = \lceil \chi_H(G) \rceil = 1 - \frac{\epsilon_{max}(H)}{\epsilon_{min}(H)} \tag{4}$$

where H is the adjacency matrix of G while ϵ_{max} and ϵ_{min} are the largest and the smallest eigenvalues of H respectively. The eigenvalues can be computed in a polynomial time using the C++ LAPACK library.

3.2 A new lower bound

Lower bound based on an auxiliary graph $\chi_{G_A}(G)$. Cornaz and Jost [19] and Palubeckis [21] prove a 1-to-1 correspondence between colorings in G and stable sets in an *auxiliary graph* G_A. The following Theorem holds:

Theorem 1 (Cornaz and Jost [19]) *For any graph G and any acyclic orientation of its complementary graph, there is a one-to-one correspondence between the set of all colorings of G and the set of all stable sets of G_A. Moreover, for any coloring $\{V_1, \ldots, V_k\}$ and its corresponding stable set $\tilde{S}$ in G_A, we have: $|\tilde{S}| + k = |V|$. In particular:*

$$\alpha(G_A) + \chi(G) = |V|.$$

To build the auxiliary graph G_A, it is necessary to define an acyclic orientation $\overrightarrow{G}$ of $\bar{G}$. Then G_A corresponds to the line-graph[2] $L(\overrightarrow{G})$ after the removal of all edges between pairs of arcs which are simplicial[3]

[2]The line-graph $L(\overrightarrow{G})$ of $\overrightarrow{G}$ is defined as follows: each arc of $\overrightarrow{G}$ corresponds to a vertex of $L(\overrightarrow{G})$ and two vertices are linked by an edge in the $L(\overrightarrow{G})$ if they correspond to two adjacent arcs in $\overrightarrow{G}$.

[3]A pair of arcs $\{a, b\}$ of $\overrightarrow{G}$ is called a simplicial pair if $a = (u, v)$, $b = (u, w)$, and (v, w) or (w, v) is an arc of $\overrightarrow{G}$, for three distinct vertices u, v, w

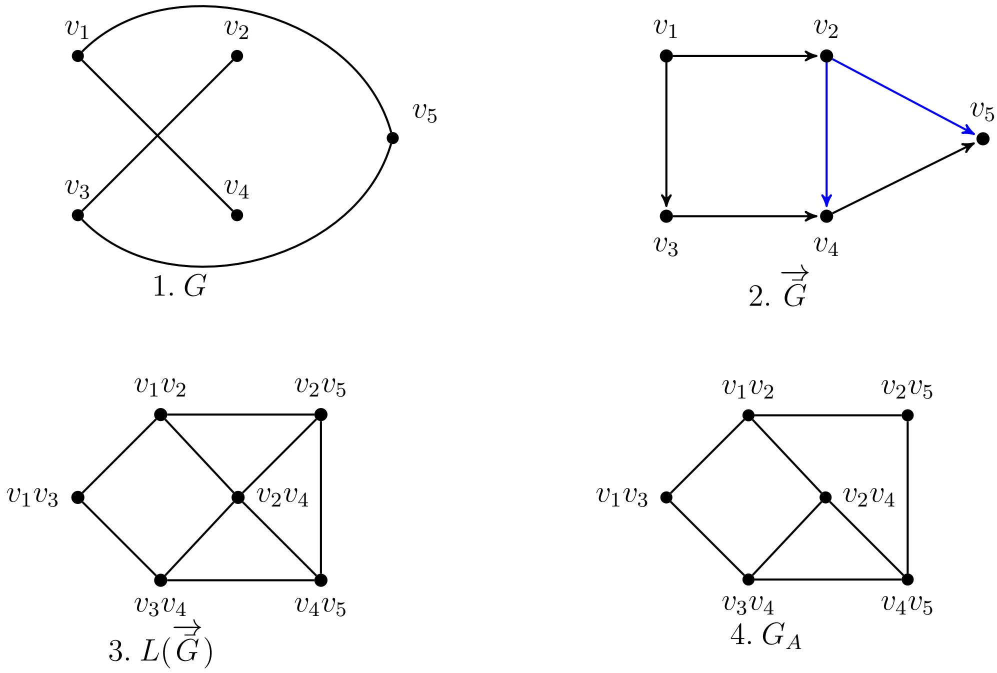


Figure 1: Transformation from a graph G to G_A

in $\overrightarrow{G}$. Precisely, given a simplicial pair of arcs $a = (v_i, v_j)$ and $b = (v_i, v_k)$ the corresponding edge (a, b) is removed from $L(\overrightarrow{G})$.

We now illustrate the construction of G_A using the example of Figure 1. The original graph consists of 5 nodes and 4 edges (part 1 of Figure 1). Then the acyclic orientation $\overrightarrow{G}$ is depicted in part 2 of Figure 1 where $(v_i, v_j) \in \overrightarrow{E}$ if $(v_i, v_j) \in \bar{E}$ and $i < j$. The next step consists of creating the line-graph $L(\overrightarrow{G})$ as depicted in part 3 of Figure 1. Finally the Auxiliary Graph G_A is given in part 4 of figure 1. Only one simplicial pair (in blue) is present in $\overrightarrow{G}$, i.e., (v_2, v_5) and (v_2, v_4), and accordingly the corresponding edge has been removed from $L(\overrightarrow{G})$. From Figure 1, it is clear that any vertex belonging to a stable set in G_A allows to reduce of one unity the upper bound $|V|$ on $\chi(G)$. In other words, if a vertex $(v_iv_j) \in G_A$ belongs to a stable set, it means that vertex v_j can be colored with the same color of v_i, i.e. "saving" in this manner a color. Finally for any simplicial pair of arcs (v_i, v_j) and (v_i, v_k) in $\overrightarrow{G}$, removing the arc (v_iv_j, v_iv_k) in G_A reflects the fact that once v_k has been colored in the same way as v_i then also v_j can take the same color (and vice-versa).

Any upper bound $\bar{\alpha}(G_A)$ of the stability number $\alpha(G_A)$ gives us a valid lower bound for $\chi(G)$ denoted $\chi_{G_A}(G)$. The following holds:

$$\chi(G) \geq \chi_{G_A}(G) = |V| - \lceil \bar{\alpha}(G_A) \rceil. \tag{5}$$

Many upper bounds are present in the literature for the stability number $\alpha(G_A)$. After extensive preliminary tests, we decide to exploit an upper bound based on the *edge formulation* for the Maximal Stable Set problem (MSSP). The edge formulation is an ILP where $\alpha(G_A) = \max x$ over x in $STAB(G_A)$, that is, the set of vectors of $\mathbb{R}^{V_{G_A}}$ satisfying

	ω	χ_{G_A}	ω^h	χ_α	χ^*_f	χ^*_H
ω		0.00	99.25	69.40	0.00	100.00
χ_{G_A}	82.84		100.00	100.00	0.75	100.00
ω^h	0.00	0.00		2.24	0.00	2.99
χ_α	4.48	0.00	97.76		0.00	85.82
χ^*_f	88.81	35.82	100.00	100.00		100.00
χ^*_H	0.00	0.00	92.54	1.49	0.00	

Table 1: Instance percentage with better lower bound value

	ω	χ_{G_A}	ω^h	χ_α	χ^*_f	χ^*_H
ω		17.16	0.75	26.12	11.19	0.00
χ_{G_A}	-		0.00	0.00	63.43	0.00
ω^h	-	-		0.00	0.00	4.48
χ_α	-	-	-		0.00	12.69
χ^*_f	-	-	-	-		0.00
χ^*_H	-	-	-	-	-	

Table 2: Instance percentage with equal lower bound value (ties)

	ω	χ_{G_A}	ω^h	χ_α	χ^*_f	χ^*_H
ω		99.25	0.00	49.25	100.00	71.64
χ_{G_A}	0.75		0.00	0.00	94.03	0.00
ω^h	100.00	100.00		74.63	100.00	100.00
χ_α	50.00	100.00	25.37		100.00	71.64
χ^*_f	0.00	5.97	0.00	0.00		0.00
χ^*_H	28.36	100.00	0.00	28.36	100.00	

Table 3: Instance percentage with faster computing time

	ω	χ_{G_A}	ω^h	χ_α	χ^*_f	χ^*_H
ω		0.00	0.00	34.33	0.00	71.64
χ_{G_A}	0.75		0.00	0.00	0.75	0.00
ω^h	0.00	0.00		2.24	0.00	2.99
χ_α	2.99	0.00	25.37		0.00	70.90
χ^*_f	0.00	0.75	0.00	0.00		0.00
χ^*_H	0.00	0.00	0.00	1.49	0.00	

Table 4: Instance percentage with faster computing time and better lower bound value

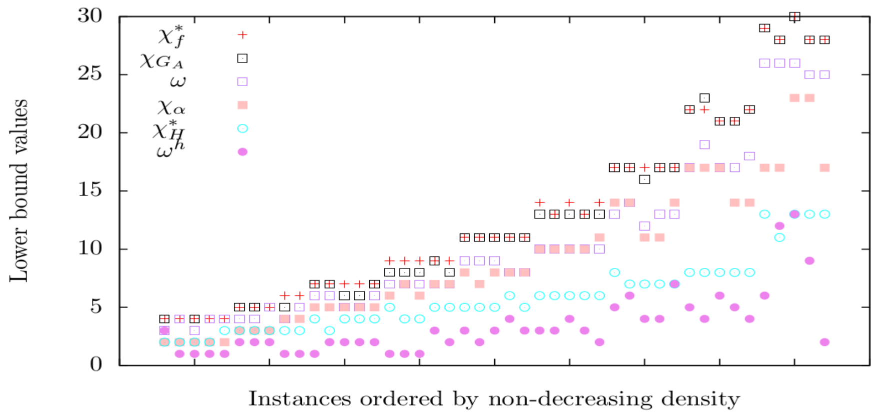
$$x_u + x_v \leq 1 \qquad uv \in E_A \tag{6}$$
$$x_v \in \{0,1\} \qquad v \in V_A. \tag{7}$$

Inequalities (6) are called the edge inequalities and any (not necessarily integer) solution x of (6)-(7) is called a fractional stable set. Many families of valid inequalities can be separated to improve the quality of the continuous relaxation of the edge formulation. We decide instead to use the generic valid inequalities generated by CPLEX (version 12.6) at the root node. In this way we obtain what we have called $\chi_{G_A}(G)$. Thanks to extensive computational experiments, we identify that the most effective families of inequalities are the *Clique Cuts*, the *Zero-half Cuts* and the *Gomory Fractional Cuts*. We exploit in this manner the strength of CPLEX in computing quickly strong bounds that can be successfully exploited to speed up the converge of DSATUR (see Section 5 for further details).

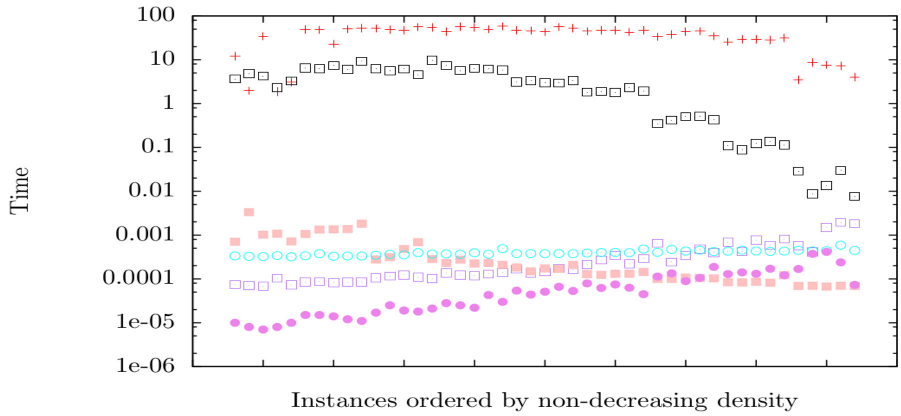
3.3 Comparison between lower bounds

To test the lower bounds, we use the random instances introduced in San Segundo [27] with 70, 75 and 80 vertices and density (denoted d) varying from 0.1 to 0.9. Since not all the densities are present for the instances of 80 vertices, we complete this test-bed generating the missing instances using the same procedure used in [27]. For each density value and vertex number, we select 5 different instances, building in this manner a test-bed of 120 instances.

Tables 1,2,3 and 4 compare each bound against each other in terms of values and computing times. The entries in the table correspond to the percentage of instances respecting a certain criteria as follows. For Table 1 and 2 we report the percentage of instances where the "row" lower bound value is strictly larger (Table 1) or equal (Table 2) than the "column" lower bound value. For Table 3, we report instead



(a)



(b)

Figure 2: Lower bound comparison for random instances with different densities and $n = 70$

the percentage of intances where the "row" lower bound computing time is smaller than the "colum" lower bound computing time. For example, in Table 3, we can see that bound $\chi_{G_A}(G)$ is faster than bound $\chi_f^*(G)$ in 94.03% of the instances. In Table 4, we report the percentage of instances where the "row" lower bound computing time is smaller than the "colum" lower bound computing time and the "row" lower bound value is larger than the "colum" lower bound value.

The two subfigures of Figure 2 graphically compare the lower bound values and the computing time for instances of 70 vertices and density varying from 0.1 to 0.9. Figure 2a presents the lower bound values on the vertical axis and the density in horizontal one (5 instances with same density). Figure 2b presents the lower bound computing times on the vertical axis and the density on the horizontal one. Each point represents a particular instance. No bound fully dominates all the others in terms of computing time and value. We can see in Figure 2b that the best lower bound values are provided by $\chi_{G_A}(G)$ and $\chi_f^*(G)$ which are also the most time consuming ones. Among the "fast" but weaker lower bounds, $\omega(G)$ tends to dominate $\chi_\alpha(G)$ and $\chi_H^*(G)$ in terms of lower bound values. The lower bound $\omega^h(G)$ is the fastest but of very poor quality.

As far as the strongest bounds are concerned, $\chi_{G_A}(G)$ is equal to $\chi_f^*(G)$ in around 63% of the instances and, in the remaining cases, the difference between the values does not exceed 1. According to the construction of graph G_A, the more dense the graph is the faster the bound is computed, since the number of nodes $|V_A|$ of G_A is equal to the number of edges of $\bar{G}$. While it gets faster, it preserves its quality and, accordingly, its pruning potential once included in DSATUR. This fact makes $\chi_{G_A}(G)$ the best lower bound for the new DSATUR algorithm. Thanks to extensive computational results, we notice that the use of all the other lower bounds does not speed up DSATUR (see the Appendix for further details).

We finally report in Table 5 the lower bounds for a subset of DIMACS instances (`ftp://dimacs.rutgers.edu/pub/challenge/graph/`) where we have been able to compute the lower bound $\chi_{G_A}(G)$ within a time limit of 3600 seconds. The results are similar to the ones obtained for random instances. From the table, we can see that the overall quality of $\chi_\alpha(G)$, $\chi^*_H(G)$ and $\omega^h(G)$ is very poor while $\chi_{G_A}(G)$ provides the best lower bound for many instances. For dense instances, $\chi_{G_A}(G)$ dominates $\chi^*_f(G)$ in terms of computing time.

4. Reduced graph and the improved DSATUR-based Branch and Bound

In order to make lower bounds dependent on a partial coloring $\tilde{C}$ obtained during the execution of DSATUR, we introduce a new graph. The *Reduced Graph* $G^{\tilde{C}} = (V^{\tilde{C}}, E^{\tilde{C}})$ is composed of the subgraph of G induced by the non-colored vertices plus $\tilde{k}$ vertices, one for each color. Each new vertex $\tilde{v}_i$, representing color i, is connected to all the uncolored neighbours of the vertices of V_i and to all the others $\tilde{k}-1$ new vertices. Thus, the subgraph of $G^{\tilde{C}}$ induced by the $\tilde{k}$ new vertices is a clique. The reduced graph becomes smaller increasing the number of colored vertices thus also the lower bounds become easier to compute. An example using a partially colored graph of 6 vertices is given in Figure 3, where two colors (1 and 2) are used and three vertices are uncolored. The Reduced Graph has 5 vertices, two representing the classes of colors plus the three original uncolored vertices.

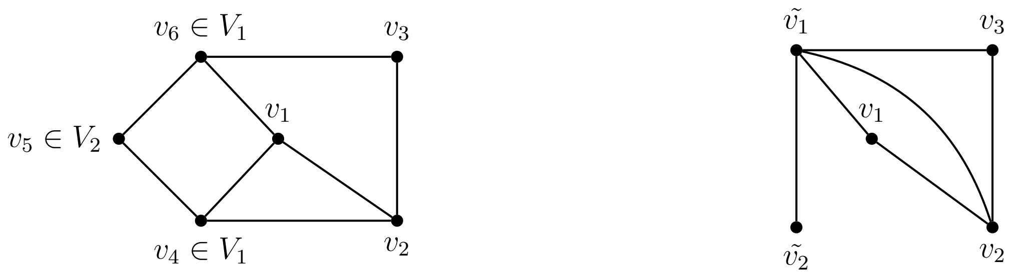


Figure 3: A partially colored graph G and the Reduced Graph $G^{\tilde{C}}$

Recalling that we denote by $\chi_{\tilde{C}}(G)$ the chromatic number of G partially colored by $\tilde{C}$, the following holds:

Lemma 1 $\chi_{\tilde{C}}(G) = \chi_{\tilde{C}}(G^{\tilde{C}})$

Proof.

Any feasible coloring $\hat{C} = \{\hat{C}_1, \ldots, \hat{C}_k\}$ in G containing the coloring $\tilde{C}$ (inducing $k \geq \tilde{k}$) can be mapped into an unique feasible coloring $\{\hat{C}'_1, \ldots, \hat{C}'_k\}$ in $G^{\tilde{C}}$ using the same number of colors k. For each color i, $i = 1, \ldots, k$, two cases arise: if $\hat{C}_i \subseteq V \setminus \tilde{C}$, then $\hat{C}'_i \leftarrow \hat{C}_i$ (since the subgraphs induced by $\hat{C}_i$ are equivalent in G and $G^{\tilde{C}}$), otherwise it exists a unique stable set j such that $\tilde{C}_j \subseteq \hat{C}_i$, then $\hat{C}'_i \leftarrow \{\tilde{v}_j\} \cup \{\hat{C}_i \setminus \tilde{C}_j\}$ with $\tilde{v}_j$ representing $\tilde{C}_j$ in $G^{\tilde{C}}$. By construction, $\hat{C}'_i$ is a stable set $G^{\tilde{C}}$. Any feasible coloring $\{\hat{C}'_1, \ldots, \hat{C}'_k\}$ in $G^{\tilde{C}}$ can be mapped into an unique feasible coloring $\{\hat{C}_1, \ldots, \hat{C}_k\}$ containing the coloring $\tilde{C}$ in G. For each color i, $i = 1, \ldots, k$, two cases arise: if $\hat{C}'_i \subseteq V \setminus \tilde{C}$, then $\hat{C}_i \leftarrow \hat{C}'_i$, otherwise an unique vertex $\tilde{v}_j$ representing $\tilde{C}_j$ belongs to $\hat{C}'_i$ (let us recall that the $\tilde{k}$ vertices representing $\tilde{C}$ in $G^{\tilde{C}}$

Instance			Value						Time					
name	n	d	χ_f^*	χ_{G_A}	ω	χ_α	χ_H^*	ω^h	χ_f^*	χ_{G_A}	ω	χ_α	χ_H^*	ω^h
queen5_5	25	0.53	**5**	**5**	**5**	**5**	4	4	0.01	0.01	**0**	**0**	**0**	**0**
queen6_6	36	0.46	**7**	**7**	6	6	5	4	0.37	0.3	**0**	**0**	**0**	**0**
queen7_7	49	0.40	**7**	**7**	**7**	**7**	5	4	0.06	0.24	**0**	**0**	**0**	**0**
queen8_8	64	0.36	**9**	8	8	8	6	4	3.8	1.75	**0**	**0**	**0**	**0**
queen8_12	96	0.30	**12**	**12**	**12**	**12**	8	2	0.95	3.67	**0**	**0**	**0**	**0**
queen9_9	81	0.33	**9**	**9**	**9**	**9**	7	4	12.07	7.1	**0**	**0**	**0**	**0**
queen10_10	100	0.30	**10**	**10**	**10**	**10**	8	4	21.71	28.36	**0**	**0**	**0**	**0**
queen11_11	121	0.27	**11**	**11**	**11**	**11**	9	4	22.78	41.51	**0**	**0**	**0**	**0**
queen12_12	144	0.25	**12**	**12**	**12**	**12**	10	4	30.02	241.34	**0**	**0**	0.01	**0**
myciel3	11	0.36	3	**4**	2	2	2	1	0.02	**0**	**0**	**0**	**0**	**0**
myciel4	23	0.28	**4**	**4**	2	2	2	1	0.44	0.11	**0**	**0**	**0**	**0**
myciel5	47	0.22	**4**	**4**	2	2	1	1	11.91	0.88	**0**	**0**	**0**	**0**
myciel6	95	0.17	**4**	**4**	2	2	2	1	248.67	18.89	**0**	**0**	**0**	**0**
miles250	128	0.05	**8**	**8**	**8**	2	4	3	0.06	4.89	**0**	**0**	**0**	**0**
miles500	128	0.14	**20**	19	**20**	7	5	12	0.15	5.91	**0**	**0**	**0**	**0**
miles750	128	0.26	**31**	**31**	**31**	10	6	11	0.24	2.62	**0**	**0**	**0**	**0**
miles1000	128	0.40	**42**	**42**	**42**	16	7	29	0.52	1.38	**0**	**0**	**0**	0.01
miles1500	128	0.64	**73**	71	**73**	25	8	51	0.71	0.46	0.01	**0**	**0**	0.01
anna	138	0.05	**11**	**11**	**11**	1	3	6	0.05	5.6	**0**	**0**	**0**	**0**
huck	74	0.11	**11**	**11**	**11**	2	1	4	0.03	0.3	**0**	**0**	**0**	**0**
jean	80	0.08	**10**	**10**	**10**	2	3	3	0.04	0.63	**0**	**0**	**0**	**0**
david	87	0.11	**11**	**11**	**11**	2	3	7	0.05	0.8	**0**	**0**	**0**	**0**
games120	120	0.09	**9**	**9**	**9**	5	3	1	0.08	5.64	**0**	0.04	**0**	**0**
mug88_1	88	0.04	**4**	3	3	3	2	2	3.89	2.01	**0**	0.01	**0**	**0**
mug88_25	88	0.04	**4**	3	3	3	1	1	3.35	1.78	**0**	**0**	**0**	**0**
mug100_1	100	0.03	**4**	3	3	3	2	1	5.56	1.88	**0**	10.29	**0**	**0**
mug100_25	100	0.03	**4**	3	3	3	2	2	6.05	1.62	**0**	23.3	**0**	**0**
mulsol.i.1	197	0.20	**49**	**49**	**49**	1	4	2	1.32	12.55	**0**	**0**	**0**	**0**
mulsol.i.2	188	0.22	**31**	**31**	**31**	2	2	2	0.89	8.28	**0**	0.01	**0**	**0**
mulsol.i.3	184	0.23	**31**	**31**	**31**	2	2	2	0.83	6.72	**0**	**0**	**0**	**0**
mulsol.i.4	185	0.23	**31**	**31**	**31**	2	2	2	0.82	6.69	**0**	**0**	**0**	**0**
mulsol.i.5	186	0.23	**31**	**31**	**31**	2	2	2	0.78	7.75	**0**	**0**	**0**	**0**
1-FullIns_3	30	0.23	**4**	3	3	2	1	3	0.14	0.09	**0**	**0**	**0**	**0**
1-FullIns_4	93	0.14	**4**	**4**	3	2	2	3	76.63	22.4	**0**	**0**	**0**	**0**
2-FullIns_3	52	0.15	**5**	4	4	2	1	4	0.21	0.44	**0**	**0**	**0**	**0**
3-FullIns_3	80	0.11	**6**	5	5	2	2	5	0.44	2.82	**0**	**0**	**0**	**0**
4-FullIns_3	114	0.08	**7**	6	6	2	2	6	0.58	14.23	**0**	**0**	**0**	**0**
5-FullIns_3	154	0.07	**8**	7	7	2	2	7	0.49	41.47	**0**	**0**	**0**	**0**
1-Insertions_4	67	0.10	**3**	**3**	2	2	2	1	49.8	4.51	**0**	**0**	**0**	**0**
2-Insertions_3	37	0.11	**3**	**3**	2	2	2	1	2.16	0.25	**0**	**0**	**0**	**0**
2-Insertions_4	149	0.05	**3**	**3**	2	2	1	1	1259.95	475.2	**0**	**0**	**0**	**0**
3-Insertions_3	56	0.07	**3**	**3**	2	2	2	1	10.03	1.98	**0**	**0**	**0**	**0**
4-Insertions_3	79	0.05	**3**	**3**	2	2	2	1	32.89	8.26	**0**	**0**	**0**	**0**
DSJC125.1	125	0.09	**5**	4	4	3	3	1	969.62	2302.61	**0**	10.88	**0**	**0**
DSJC125.9	125	0.90	**43**	**43**	34	31	16	10	266.61	0.16	9.27	**0**	0.01	**0**
DSJC250.9	250	0.90	-	**71**	-	50	23	7	tl	6.71	tl	**0**	0.01	0.03
DSJR500.1c	500	0.97	-	**85**	-	38	25	2	tl	2.11	tl	**0.01**	0.09	**0.01**
r125.1c	125	0.97	**46**	**46**	**46**	17	16	7	15.18	0.01	**0**	0.01	**0**	**0**
r125.1	125	0.03	**5**	**5**	**5**	2	3	3	0.22	3.21	**0**	**0**	**0**	**0**
r125.5	125	0.50	**36**	**36**	**36**	25	7	27	1.31	2.68	**0**	**0**	**0**	**0**
r250.1c	250	0.97	**64**	**64**	**64**	-	22	9	488.09	0.03	772.31	tl	**0.01**	**0.01**
r250.1	250	0.03	**8**	**8**	**8**	-	4	2	0.48	131.78	**0**	tl	0.48	**0**
r250.5	250	0.48	**65**	**65**	**65**	41	8	43	19.31	89	0.73	**0**	0.01	0.02
r1000.1c	1000	0.97	-	**95**	-	41	25	8	tl	325.53	tl	**0.01**	0.35	0.45
zeroin.i.1	211	0.19	**49**	**49**	**49**	1	3	11	0.95	18.12	**0**	**0**	**0**	0.01
zeroin.i.2	211	0.16	**30**	**30**	**30**	1	2	3	0.56	15.69	**0**	**0**	**0**	**0**
zeroin.i.3	206	0.17	**30**	**30**	**30**	1	2	3	0.58	14.35	**0**	**0**	0.01	**0**

Table 5: Comparison between Lower Bounds on $\chi(G)$ for DIMACS instances

form a clique), and then $\hat{C}_i \leftarrow \tilde{C}_j \cup \{\hat{C}'_i \setminus \{\tilde{v}_j\}\}$. By construction, $\hat{C}_i$ is a stable set in G. $\square$

Trivially, we have: $\chi_{\tilde{C}}(G^{\tilde{C}}) = \chi(G^{\tilde{C}})$. Thus, from lemma 1, a lower bound denoted by $LB^{\tilde{C}}$ for $\chi(G^{\tilde{C}})$ is a lower bound for $\chi_{\tilde{C}}(G)$.

Improved DSATUR-based Branch and Bound. We present now the improved DSATUR-based Branch and Bound. The new DSATUR algorithm, denoted DSATUR-χ_{G_A}, is obtained by replacing in Algorithm 1 the call to Algorithm 2 by a call to the new Algorithm 3. The key element of the new algorithm is the update of the lower bound $\chi_{G_A}(G^{\tilde{C}})$, computed thanks to the reduced graph $G^{\tilde{C}}$ at the nodes of the branching tree.

Since updating the lower bound can be computationally expensive, we derive strategies to compute it only in "promising" nodes of the branching tree. The term "promising" is linked to two different aspects. Clearly pruning during the first levels of the branching tree is more likely to produce a larger reduction of the branching nodes. Secondly, the lower bound is likely to prune when the difference between the node lower and upper bounds is small. Accordingly, we introduce the function $\phi(\tilde{n}, UB - \tilde{k})$ which decides if we update or not the lower bound in function of the input parameters. The first parameter $\tilde{n}$ corresponds to the number of colored vertices in $\tilde{C}$, i.e., the depth of the node in the branching tree. The second parameter $UB - \tilde{k}$ corresponds to the gap between the incumbent value and the lower bound $\tilde{k}$. The effectiveness of this algorithm will be discussed in Section 5.

Algorithm 3: DSATUR($\tilde{C}$)-χ_{G_A}

if *all the vertices are colored* **then**
    if $\tilde{k} < UB$ **then**
        $C^* \leftarrow \tilde{C}, UB \leftarrow \tilde{k}$;
    end
else
    if $\max\{\tilde{k}, LB\} < UB$ **then**
        if $\phi(\tilde{n}, UB - \tilde{k}) = \texttt{true}$ **then**
            if $\chi_{G_A}(G^{\tilde{C}}) < UB$ **then**
                select an uncolored vertex v;
                for *every feasible color* $i \in \tilde{C}$ *plus a new one* **do**
                    $\widehat{C} \leftarrow \tilde{C}$, add v in $\widehat{V}_i$;
                    DSATUR($\widehat{C}$);
                end
            end
        end
    end
end

5. Computational results

Algorithms 1, 2 and 3 are coded in C/C++, and run on a PC with an Intel(R) Core(TM) i7-4770 CPU at 3.40GHz and 16 GB RAM memory, under Linux Ubuntu 14.04 64-bit. Since χ_{G_A} is efficient for high density graphs, we extend the set of instances adding larger high density graphs, i.e., 5 instances per

$n = 85, 90, 95, 100, 110, 120, 130$ and $d = 0.7, 0.8, 0.9$ plus 5 instances for $n = 140$ and $d = 0.9$. The final testbed is then composed of 160 instances. The entire benchmark of instances can be downloaded at the following address `http://www.lamsade.dauphine.fr/coloring`.

To reduce the impact of the quality of the initial UB on the execution of the algorithms, in all the computational tests presented in this section we initialize UB with the best heuristic solution computed by DSATUR in 3600 seconds. In case DSATUR is able to prove optimality within that time limit, UB corresponds to the chromatic number $\chi(G)$ of the instance. Finally for all the tests we set a time limit of 3600 seconds and in case of time limit we report "tl".

The goal of this computational section is twofold. First, we test the full impact of the proposed bounding procedure updating the lower bound at each node of the branching scheme (Subsection 5.1). Then we discuss possible enhancements based on the function ϕ in order to select a promising subset of nodes in which we update the lower bound (Subsection 5.2).

5.1 Updating the lower bound at each node of DSATUR

In this section, we discuss the results obtained updating the lower bound $\chi_{G_A}(G)$ at each node of the branching tree, thus in Algorithm 3, the function ϕ always returns true. Tables 6 and 7 are divided in three parts: in the first we present the instances' features, in the second we present the results obtained by DSATUR and in the third we present the results obtained by DSATUR-χ_{G_A}. Each line of Table 6 reports the average values of 5 random instances of a given size n and a given density d, while each line of Table 7 reports instead the results of the subset of DIMACS instances also discussed in Table 5. The average values are computed considering only the subset of instances solved to proven optimality, i.e., excluding the "tl" cases. In the following we explain the meaning of the tables' columns:

- *OPT** : the chromatic number or the best UB in case of time limit.
- *nodes* : the total number of processed nodes.
- *time* : the total computing time (*tl* means a time limit of 3600 seconds).
- *time*$_G$: the computing time to generate the reduced graph $G^{\tilde{C}}$ for DSATUR-$\chi_{G_A}(G)$ (only reported for Table 7 since negligible for the random instances).
- *time*$_B$: the computing time of $\chi_{G_A}(G)$ for DSATUR-$\chi_{G_A}(G)$.
- *max/min* : the maximim/minimum total computing time (only reported for Table 6).
- *#bounds* : the number of times the lower bound $\chi_{G_A}(G)$ is computed (potentially lower than the total number of processed nodes in case some of the nodes are pruned by the standard bound $\max\{\tilde{k}, LB\}$).
- *#cuts* : the number of times that $\chi_{G_A}(G)$ is able to prune.
- *fail* : the number of instances that can not be solved in less than 3600 seconds (only reported for Table 6).

The values in bold highlight the best computing time or number of nodes between DSATUR and DSATUR-χ_{G_A}. First of all, we can conclude that the bound $\chi_{G_A}(G)$ is effective since the number of nodes in the branching tree of DSATUR-χ_{G_A} is significantly smaller. The computing time of these bounds is significant since it represents more than 80% of the total time. The computing time necessary to build $G^{\tilde{C}}$ and G_A is less than 1% of the total computing time.

Instance			DSATUR					DSATUR-χ_{G_A}							
n	d	OPT*	nodes	time	max	min	fail	nodes	time$_B$	time	max	min	#bounds	#cuts	fail
70	0.1	4.0	30	**0.00**	**0.00**	**0.00**	0	**1**	4.11	4.18	5.70	3.32	1	1	0
70	0.2	6.0	2648	**0.00**	**0.00**	**0.00**	0	**288**	975.10	983.43	1929.17	335.39	180	62	0
70	0.3	8.0	**372720**	**0.91**	**1.17**	**0.74**	**0**	-	-	tl	tl	tl	-	-	5
70	0.4	10.0	**3510691**	**12.23**	**13.75**	**9.78**	**0**	-	-	tl	tl	tl	-	-	5
70	0.5	12.0	7989873	**36.66**	**65.10**	**12.89**	**0**	**2048**	2441.24	2469.34	tl	2469.34	1584	973	4
70	0.6	14.0	9428697	**55.82**	**126.65**	**23.51**	0	**1078**	762.80	773.27	1613.80	116.67	863	511	0
70	0.7	17.6	18992632	**142.48**	**324.19**	46.69	0	**812**	167.29	171.56	567.48	**0.57**	664	396	0
70	0.8	21.8	4901009	44.01	83.15	19.19	0	**1**	0.14	**0.14**	**0.17**	**0.09**	1	1	0
70	0.9	28.6	149337	1.65	3.05	0.72	0	**1**	0.01	**0.01**	**0.04**	**0.00**	1	1	0
75	0.1	4.0	7	**0.00**	**0.00**	**0.00**	0	**1**	7.84	7.94	13.36	3.16	1	1	0
75	0.2	6.0	3395	**0.00**	**0.01**	**0.00**	**0**	**174**	1248.61	1255.72	tl	215.12	112	43	1
75	0.3	8.0	**293236**	**0.77**	**1.49**	**0.28**	**0**	-	-	tl	tl	tl	-	-	5
75	0.4	10.0	**12605807**	**45.62**	**87.54**	**17.73**	**0**	-	-	tl	tl	tl	-	-	5
75	0.5	12.4	**28613624**	**149.70**	**227.03**	**44.04**	**0**	-	-	tl	tl	tl	-	-	5
75	0.6	15.0	95301936	**634.90**	**1192.44**	**322.49**	**0**	**1928**	2184.92	2210.59	tl	1551.23	1600	1022	2
75	0.7	18.0	83548744	698.86	1037.27	341.07	0	**728**	256.26	**261.99**	**776.67**	**27.18**	614	379	0
75	0.8	22.4	28783000	291.25	619.22	125.05	0	**11**	2.23	**2.30**	**10.99**	**0.04**	10	5	0
75	0.9	31.0	6581082	79.17	199.18	20.03	0	**1**	0.01	**0.01**	**0.05**	**0.00**	1	1	0
80	0.1	4.8	5709	**0.00**	**0.01**	**0.00**	**0**	**668**	1752.58	1781.86	tl	13.73	392	109	1
80	0.2	7.0	**301583**	**0.60**	**1.10**	**0.24**	**0**	-	-	tl	tl	tl	-	-	5
80	0.3	9.0	**16277818**	**49.05**	**72.96**	**22.98**	**0**	-	-	tl	tl	tl	-	-	4
80	0.4	11.0	**186461982**	**788.48**	**2572.13**	**201.71**	**0**	-	-	tl	tl	tl	-	-	5
80	0.5	13.0	**106724150**	**614.49**	**1757.23**	**184.47**	**0**	-	-	tl	tl	tl	-	-	5
80	0.6	16.0	181208034	**1361.59**	tl	**367.73**	**2**	**1540**	1891.82	1915.89	tl	1915.89	1251	755	4
80	0.7	19.2	23675526	228.07	tl	17.60	1	**18**	10.26	**10.46**	tl	**0.43**	16	6	1
80	0.8	24.4	33677431	391.34	720.18	158.11	0	**1**	0.13	**0.13**	**0.16**	**0.10**	1	1	0
80	0.9	34.0	36839	0.53	1.26	0.05	0	**1**	0.01	**0.01**	**0.01**	**0.00**	1	1	0
85	0.7	20.2	-	tl	tl	tl	5	-	-	tl	tl	tl	-	-	5
85	0.8	24.2	75694582	1011.09	tl	361.94	2	**1133**	238.41	**148.14**	**593.03**	**0.17**	914	548	**0**
85	0.9	33.0	9000991	146.92	412.32	3.13	0	**1**	0.04	**0.04**	**0.10**	**0.02**	1	1	0
90	0.7	21.4	-	tl	tl	tl	5	-	-	tl	tl	tl	-	-	5
90	0.8	25.2	-	tl	tl	tl	5	**5067**	746.60	**776.27**	**1763.40**	**80.23**	4262	2723	**0**
90	0.9	33.8	20907764	363.95	1719.21	0.20	0	**1**	0.03	**0.03**	**0.06**	**0.01**	1	1	0
95	0.7	22.2	-	tl	tl	tl	5	-	-	tl	tl	tl	-	-	5
95	0.8	27.2	-	tl	tl	tl	5	**10730**	2054.92	**2132.24**	tl	**1599.88**	8795	5344	**3**
95	0.9	35.2	25246065	501.99	tl	501.99	4	**57**	10.35	**2.23**	**5.45**	**0.02**	56	11	**0**
100	0.7	23.8	-	tl	tl	tl	5	-	-	tl	tl	tl	-	-	5
100	0.8	27.4	-	tl	tl	tl	5	**2696**	624.52	**645.33**	tl	**193.76**	2308	1424	**3**
100	0.9	37.0	-	tl	tl	tl	5	**613**	29.02	**30.73**	**105.85**	**2.36**	524	246	**0**
105	0.7	24.6	-	tl	tl	tl	5	-	-	tl	tl	tl	-	-	5
105	0.8	29.2	-	tl	tl	tl	5	**10084**	2506.64	**2605.62**	tl	**2605.62**	8612	5398	**4**
105	0.9	38.2	-	tl	tl	tl	5	**203**	4.02	**4.51**	**7.76**	**0.06**	169	41	**0**
110	0.7	25.2	-	tl	tl	tl	5	-	-	tl	tl	tl	-	-	5
110	0.8	30.2	-	tl	tl	tl	5	-	-	tl	tl	tl	-	-	5
110	0.9	39.2	-	tl	tl	tl	5	**970**	57.10	**61.00**	**216.56**	**7.62**	823	410	**0**
120	0.7		-	tl	tl	tl	5	-	-	tl	tl	tl	-	-	5
120	0.8	27.2	-	tl	tl	tl	5	-	-	tl	tl	tl	-	-	5
120	0.9	33.2	-	tl	tl	tl	5	**3922**	271.35	**290.53**	**1010.43**	**21.32**	3314	1853	**0**
41.6 140	0.9	4.0	-	tl	tl	tl	5	**16648**	1666.87	**1810.92**	tl	**1478.52**	14047	8348	**3**

Table 6: DSATUR-χ_{G_A} for random VCP instances

Instance				DSATUR		DSATUR-χ_{G_A}					
name	n	d	OPT*	nodes	time	nodes	time$_G$	time$_B$	time	#bounds	#cuts
queen5_5	25	0.53	5	7	**0.00**	**1**	0.00	0.00	**0.00**	1	1
queen6_6	36	0.46	7	432	**0.00**	**1**	0.00	0.28	0.28	1	1
queen7_7	49	0.40	7	9	**0.00**	**1**	0.00	0.22	0.22	1	1
queen8_12	96	0.36	12	14	**0.00**	**1**	0.00	3.43	3.62	1	1
queen8_8	64	0.30	9	1992673	**10.17**	**319**	2.11	563.94	568.62	235	144
queen9_9	81	0.33	10	-	tl	tl	-	-	tl	816	454
queen10_10	100	0.30	11	-	tl	tl	-	-	tl	220	114
queen11_11	121	0.27	13	-	tl	tl	-	-	tl	287	110
queen12_12	144	0.25	14	-	tl	tl	-	-	tl	19	0
myciel3	11	0.36	4	50	0.00	**1**	0.00	0.00	0.00	1	1
myciel4	23	0.28	5	1579	**0.00**	**338**	0.00	2.80	2.86	220	82
myciel5	47	0.22	6	**1287849**	**1.48**	tl	-	-	tl	95327	32519
myciel6	95	0.17	7	-	tl	tl	-	-	tl	16723	5710
miles250	128	0.05	8	10	**0.00**	**1**	0.00	3.22	4.29	1	1
miles500	128	0.14	20	**27**	**0.00**	**27**	14.65	106.57	126.83	23	0
miles750	128	0.26	31	33	**0.01**	**1**	0.00	1.93	2.58	1	1
miles1000	128	0.40	42	191	**0.05**	**1**	0.00	0.82	1.26	1	1
miles1500	128	0.64	73	**75**	**0.24**	**75**	7.36	22.06	33.79	73	0
anna	138	0.05	11	13	**0.00**	**1**	0.00	4.05	6.05	1	1
huck	74	0.11	11	-	tl	**1**	0.00	0.20	**0.29**	1	1
jean	80	0.08	10	39335667	69.42	**1**	0.00	0.46	**0.61**	1	1
david	87	0.11	11	13	**0.00**	**1**	0.00	0.64	0.85	1	1
games120	120	0.09	9	-	tl	**1**	0.00	4.43	**5.17**	1	1
mug88_1	88	0.04	4	72832198	**50.35**	**46746**	304.27	2541.35	3119.92	27745	8740
mug88_25	88	0.04	4	**532869960**	**310.51**	tl	-	-	tl	161950	38681
mug100_1	100	0.03	4	**1083098474**	**697.15**	tl	-	-	tl	72776	25526
mug100_25	100	0.03	4	-	tl	tl	-	-	tl	65061	18697
mulsol.i.1	197	0.20	49	-	tl	**1**	0.00	5.59	**9.56**	1	1
mulsol.i.2	188	0.22	31	53	**0.01**	**1**	0.00	4.39	7.90	1	1
mulsol.i.3	184	0.23	31	53	**0.00**	**1**	0.00	3.56	6.43	1	1
mulsol.i.4	185	0.23	31	53	**0.01**	**1**	0.00	3.53	6.48	1	1
mulsol.i.5	186	0.23	31	53	**0.01**	**1**	0.00	4.33	7.35	1	1
1-FullIns_3	30	0.23	4	128	**0.00**	**9**	0.00	0.50	0.51	6	1
1-FullIns_4	93	0.14	5	-	tl	tl	-	-	tl	13855	4600
2-FullIns_3	52	0.15	5	376103299	**347.15**	**16680**	11.80	933.97	971.95	10329	3975
3-FullIns_3	80	0.11	6	-	tl	tl	-	-	tl	34726	21256
4-FullIns_3	114	0.08	7	-	tl	tl	-	-	tl	17687	11345
5-FullIns_3	154	0.07	8	-	tl	tl	-	-	tl	1437	930
1-Insertions_4	67	0.10	5	-	tl	tl	-	-	tl	90981	28273
2-Insertions_3	37	0.11	4	37564	**0.01**	**7350**	0.72	72.22	76.52	4216	1068
2-Insertions_4	149	0.05	5	-	tl	tl	-	-	tl	12	0
3-Insertions_3	56	0.07	4	8705001	**6.71**	tl	-	-	tl	173236	42915
4-Insertions_3	79	0.05	4	-	tl	tl	-	-	tl	194277	49245
DSJC125.9	125	0.90	44	-	tl	**9747**	35.75	903.06	**964.47**	8426	5119
DSJC250.9	250	0.90	84	-	tl	tl	-	-	tl	19046	11599
DSJR500.1c	500	0.97	85	-	tl	**1**	0.00	1.67	**1.93**	1	1
r125.1c	125	0.97	46	48	0.15	**1**	0.00	0.01	**0.01**	1	1
r125.1	125	0.03	5	7	**0.00**	**1**	0.00	2.11	3.14	1	1
r125.5	125	0.50	36	-	tl	**135**	10.37	109.51	**125.23**	133	9
r250.1c	250	0.97	64	7820	9.75	**1**	0.00	0.02	**0.04**	1	1
r250.1	250	0.03	8	-	tl	**1**	0.00	112.50	**134.64**	1	1
r250.5	250	0.48	66	-	tl	tl	-	-	tl	47	0
r1000.1c	1000	0.97	101	-	tl	tl	-	-	tl	12	0
zeroin.i.1	211	0.19	49	51	**0.05**	**1**	0.00	8.23	13.95	1	1
zeroin.i.2	211	0.16	30	32	**0.02**	**1**	0.00	8.92	14.87	1	1
zeroin.i.3	206	0.17	30	32	**0.02**	**1**	0.00	8.44	13.72	1	1

Table 7: DSATUR-χ_{G_A} for DIMACS instances

For high density instances ($d = 0.7, 0.8, 0.9$), the computing time is significantly reduced. For $n = 85, d = 0.7$, $n = 90, d = 0.7, 0.8$, $n = 95, d = 0.7, 0.8$ and for instances $n \geq 95$, DSATUR-χ_{G_A} can solve to proven optimality some of the instances while DSATUR always fails. For very high density instances ($d = 0.9$), the gap between the computing times of DSATUR and DSATUR-χ_{G_A} is significant: for $n = 120$, DSATUR-χ_{G_A} computes the optimal solution for the 5 instances in less than 300 seconds in average while DSATUR is not able to solve any of them. However for low density instances ($d = 0.2, 0.3, 0.4, 0.5$), the computing time of the lower bounds at each node of the branching tree is very high and DSATUR-χ_{G_A} is not able to solve these instances within the time limit.

The results obtained for DIMACS instances are similar: DSATUR-χ_{G_A} is very efficient in solving high density instances, solving 2 more instances (38 instead of 36) compared to DSATUR. We recall that the missing DIMACS instances are the ones in which $\chi_{G_a}(G)$ cannot be computed within 3600 seconds. Accordingly for those instances DSATUR-χ_{G_A} cannot be executed. It is worth mentioning that when the column node is 1 it means that the the lower bound $\chi_{G_A}(G)$ is able to prove the optimality of the initial UB at the root node of the branching tree. We also investigate DSATUR-$\omega(G)$, i.e., replacing the lower bound $\chi_{G_A}(G)$ with $\omega(G)$, the results are discussed in the Appendix of the manuscript. Let us remark that only the number of nodes can be significantly reduced while the computing time of DSATUR-$\omega(G)$ is always greater than the computing time of DSATUR.

5.2 Updating the lower bound at promising nodes of DSATUR

In this subsection, we define $\phi(\tilde{n}, UB - \tilde{k})$ in order to reduce the computing time of DSATUR-$\chi_{G_A}(G)$. As shown in Table 6, *#cuts* is significantly lower than *#bounds*. It means that a lot of the calculated lower bounds are not able to prune the potential subtree at the current node.

In Figure 4a, the horizontal axis is the node depth $\tilde{n}$ in the branching tree while in Figure 4b it is the gap between UB and $\tilde{k}$. In both figures, the vertical axis is the number of times a bound has been computed (green curve) or the number of times a bound has cut a node (red curve) on 5 instances for $n = 70$ and d varying from 0.1 to 0.9. We observe in Figure 4a that $\chi_{G_A}(G)$ is very efficient when it is computed at the depth node between $\tilde{n} = 18\% * n$ and $\tilde{n} = 35\% * n$ (with an optimal success rate at $\tilde{n} = 28\% * n$). In Figure 4b, we observe that $\chi_{G_A}(G)$ is able to cut an important number of nodes when the gap between UB and $\tilde{k}$ is equal to 1. Moreover, when the gap is greater to 10, the computation of $\chi_{G_A}(G)$ is useless. Following these observations, we define $\phi^*(\tilde{n}, UB - \tilde{k})$ as follows:

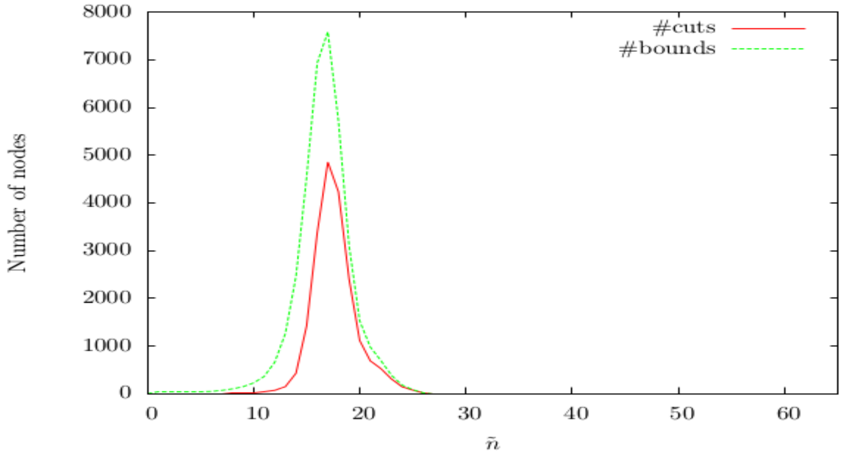
Algorithm 4: $\phi^*(\tilde{n}, UB - \tilde{k})$

```
if 0.18 n ≤ ñ ≤ 0.35 n and UB − k̃ ≤ 10 then
    return true
else
    return false
end
```

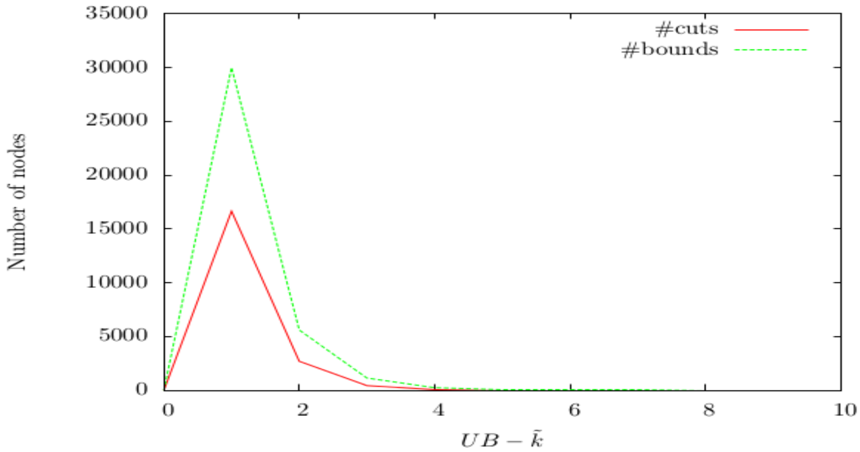
With this function ϕ^*, the computing time for solving random VCP instances with $n = 70, 75, 80$ and $d = 0.7$ is reduced as shown in Table 8.

6. Conclusion

In this paper we have considered a series of different lower bounding techniques for DSATUR. The principal idea is to exploit lower bounds for the Vertex Coloring Problem in order to prune the implicit enu-



(a)



(b)

Figure 4: Statistics on χ_{G_A} during an execution of DSATUR-χ_{G_A}

Instance			DSATUR-χ_{G_A} with ϕ								DSATUR-χ_{G_A} with ϕ^*							
n	d	OPT*	nodes	time$_B$	time	max	min	#bounds	#cuts	fail	nodes	time$_B$	time	max	min	#bounds	#cuts	fail
70	0.7	17.6	**812**	167.29	171.56	567.48	0.57	664	396	0	3209893	74.74	**102.04**	**325.79**	**0.56**	262	109	0
75	0.7	18.0	**728**	256.26	261.99	776.67	27.18	614	379	0	3574556	148.80	**182.79**	**521.65**	**18.15**	313	154	0
80	0.7	19.2	**3845**	706.56	**728.52**	tl	**0.43**	3004	1796	1	32724255	409.34	**725.43**	tl	0.44	1212	470	1

Table 8: DSATUR-χ_{G_A} on selected nodes

meration scheme. In the literature, all the efforts have been made in the direction of a better selection of the branching node. In this paper we have shown instead the potential of exploiting fast but strong lower bounds within the branching scheme. Thanks to the new lower bound based on the 1-to-1 mapping between VCPs and Stable Set Problems, we have successfully reduced both the computing time and the number of nodes for high density random instances and for a subset of high density DIMACS instances.

Acknowledgment

The authors want to thank Denis Cornaz and Enrico Malaguti for stimulating discussions on the topic and for their contributions.

References

[1] A. Hoffman. On eigenvalues and colorings of graphs. *Graph Theory and Its Applications, Proc. Adv. Sem., Math. Research Center, Univ. of Wisconsin, Madison, WI, 1969, Academic Press, New York* : 79-91, 1970.

[2] A. Amin, S. Hakimi. Upper Bounds on the Order of a Clique of a Graph. *Society for Industrial and Applied Mathematics* 22(4): 569-573, 1972.

[3] D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM (CACM)* 22(4): 251-256, 1979.

[4] F. Leighton. A Graph Coloring Algorithm for Large Scheduling Problems. *Journal of Research of the National Bureau of Standards* 84(6): 489-506, 1979.

[5] M. Garey, D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*.

[6] M. Grötschel,L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169-197, 1981.

[7] F. Chow, J. Henessy. The priority-based coloring approach to register allocation. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 12(4): 501-536, 1990.

[8] A. Mehrotra, M. Trick. A column generation approach for graph coloring *INFORMS Journal on Computing* , 8(4): 344-54, 1996.

[9] E. Sewell. An improved algorithm for exact graph coloring. *Cliques, coloring, and satisfiability. Proceedings of the second DIMACS implementation challenge*, 26: 359-373, 1996.

[10] M. Caramia, and P. Dell'Olmo. Solving the minimum-weighted coloring problem *Networks*, 38(2):88-101, 2001.

[11] PRJ Ostergärd. A fast algorithm for the maximum clique problem *Discrete Applied Mathematics*, 120(1):197-207, 2002.

[12] N. Barnier, P. Brisset. Graph coloring for air traffic flow management *Annals of operations research*, 130(1-4): 163-178, 2004.

[13] M. Lübbecke and J. Desrosiers. Selected Topics in Column Generation *Operations Research*, 53(6): 1007-1023, 2004.

[14] I. Méndez-Díaz and P. Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5):826-847, 2006.

[15] J. Konc, D. Janezic. A Branch and Bound Algorithm for Matching Protein Structures. *ICANNGA*,(2): 399-406, 2007.

[16] M. Gamachea, A. Hertzb, J. Olivier Ouelletb,. A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. *Computers & Operations Research*, 34(8): 2384-2395, 2007.

[17] R. Opsut and F. Roberts. I-Colorings, I-Phasings, and I-Intersection assignments for graphs, and their applications Networks, 13(3): 327-345, 2007.

[18] I. Méndez-Díaz and P. Zabala. A cutting plane algorithm for graph coloring. *Discrete Applied Mathematics*, 156(2): 159-179, 2008.

[19] D. Cornaz, V. Jost. A one-to-one correspondence between colorings and stable sets. *Operations Research Letters* 36(6): 673-676, 2008.

[20] M.Campelo, V. Campos, R. Correa On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics*, 156(7): 1097-1111, 2008.

[21] G. Palubeckis. On the recursive largest first algorithm for graph colouring. *International Journal of Computer Mathematics*, 85(2): 191-200, 2008.

[22] N. Zufferey , P. Amstutz, P. Giaccari. Graph colouring approaches for a satellite range scheduling problem. *Journal of Scheduling*, 11(4): 263-277, 2008.

[23] C. Bentz, M.C. Costa, D. de Werra, C. Picouleau and B. Ries. On a graph coloring problem arising from discrete tomography. *Networks*, 51(4): 256-267, 2008.

[24] P. Hansen, M. Labbé, D. Schindl. Set covering and packing formulations of graph coloring: Algorithms and first polyhedral results. *Discrete Optimization*, 6(2): 135-147, 2009.

[25] E. Malaguti and P. Toth. A survey on vertex coloring problems. *International Transactions in Operational Research*, 17(1): 1-34, 2010.

[26] C. Li and Z. Quan. An Efficient Branch-and-Bound Algorithm Based on MaxSAT for the Maximum Clique Problem *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 128-133, 2010.

[27] P. San Segundo. A new DSATUR-based algorithm for exact vertex coloring. *Computers & Operations Research* 39(7): 1724-1733, 2012.

[28] E. Malaguti, M. Monaci, and P. Toth. An exact approach for the vertex coloring problem. *Discrete Optimization*, 8(2): 174-190, 2011.

[29] S. Gualandi and F. Malucelli. Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS Journal on Computing*, 24(1): 81-100, 2012.

[30] S. Held, W. Cook, and E. Sewell. Maximum-weight stable sets and safe lower bounds for graph coloring. *Mathematical Programming Computation*, 4(4): 363-381, 2012.

[31] S. Das, G. Ghidini, A. Navarra and C. Pinotti. Localization and scheduling protocols for actor-centric sensor networks. *Networks*, 59(3): 299-319, 2012.

[32] P. San Segundo, F. Mata, D. Rodrguez-Losada, M. Hernando. An improved bit parallel exact maximum clique algorithm. *Optimization Letters*, 7(3): 467-479, 2013.

[33] D. Morrison, J. Sauppe, E. Sewell, and S. Jacobson. A wide branching strategy for the graph coloring problem. *INFORMS Journal on Computing*, 26(4): 704–717, 2014.

[34] A. Schrijver. Combinatorial optimization. *Springer-Verlag, Berlin Heidelberg* : 1095-1096, 2002.

[35] K. Smith-Miles and D. Baatar. Exploring the role of graph spectra in graph coloring algorithm performance. *Discrete Applied Mathematics*, 176(0): 107-121, 2014.

[36] J. Hao and Q. Wu. A review on algorithms for maximum clique problems. *European Journal of Operational Research*, 242(3): 693-709, 2015.

[37] E. Malaguti, I. Méndez-Diaz, J. Miranda-Bront and P. Zabala. A branch-and-price algorithm for the (k,c)-coloring problem. *Networks*, 65(4): 353-366, 2015.

[38] I. Méndez-Diaz, G. Nasini, D. Severin. A DSATUR-based algorithm for the Equitable Coloring Problem. *Computers & OR*, 57(4): 41-50, 2015.

[39] F. Furini, V. Gabrel, I. C. Ternier. Lower Bounding Techniques for DSATUR-based Branch and Bound. *Electronic Notes in Discrete Mathematics (INOC 2015)*, (to appear), 2015.

A. DSATUR-$\omega(G)$

Instance			DSATUR					DSATUR-$\omega(G)$							
n	d	OPT*	nodes	time	max	min	fail	nodes	time$_B$	time	max	min	#bounds	#cuts	fail
70	0.1	4.0	28	0.00	0.00	0.00	0	**7**	0.00	0.00	0.00	0.00	4	1	0
70	0.2	6.0	2648	**0.00**	**0.00**	**0.00**	0	**1727**	0.04	0.10	0.13	0.05	934	128	0
70	0.3	8.0	372720	**0.91**	**1.17**	**0.74**	0	**222859**	7.01	16.64	21.31	12.83	121306	19458	0
70	0.4	10.0	3510691	**12.23**	**13.75**	**9.78**	0	**1710667**	63.21	164.26	186.19	130.69	953755	191248	0
70	0.5	12.0	7989873	**36.66**	**65.10**	**12.89**	0	**3041566**	128.38	366.38	657.17	124.39	1741156	420366	0
70	0.6	14.0	9428697	**55.82**	**126.65**	**23.51**	0	**2430588**	124.00	377.05	761.72	164.71	1449691	427421	0
70	0.7	17.6	18992632	**142.48**	**324.19**	**46.69**	0	**4364458**	224.69	729.50	1797.41	195.99	2613759	762523	0
70	0.8	21.8	4901009	**44.01**	**83.15**	**19.19**	0	**787189**	50.90	162.16	320.43	76.60	483569	150377	0
70	0.9	28.6	149337	**1.65**	**3.05**	**0.72**	0	**17202**	1.65	3.82	4.88	2.25	10563	2885	0
75	0.1	4.0	7	0.00	0.00	0.00	0	**1**	0.00	0.00	0.00	0.00	1	1	0
75	0.2	6.0	3395	**0.00**	**0.01**	**0.00**	0	**1921**	0.06	0.14	0.22	0.03	1048	164	0
75	0.3	8.0	293236	**0.77**	**1.49**	**0.28**	0	**162682**	5.96	14.84	27.97	6.07	89330	15693	0
75	0.4	10.0	12605807	**45.62**	**87.54**	**17.73**	0	**6620961**	269.36	720.57	1407.13	264.19	3658431	681264	0
75	0.5	12.4	28613624	**149.70**	**227.03**	**44.04**	0	**10543331**	547.32	1618.04	2540.29	453.87	6053194	1489730	0
75	0.6	15.0	95301936	**634.90**	**1192.44**	**322.49**	**0**	**19157017**	1085.91	3402.98	tl	2614.89	11273677	3141421	4
75	0.7	18.0	83548744	**698.86**	**1037.27**	**341.07**	**0**	**14083047**	969.26	3038.20	tl	1996.62	8506062	2593911	3
75	0.8	22.4	28783000	**291.25**	**619.22**	**125.05**	0	**3419914**	300.48	889.08	1864.20	359.28	2122965	702682	0
75	0.9	31.0	6581082	**79.17**	**199.18**	**20.03**	0	**1073304**	91.35	271.69	649.51	79.55	645095	164999	0
80	0.1	4.8	5709	**0.00**	**0.01**	0.00	0	**4398**	0.12	0.23	0.38	0.00	2308	212	0
80	0.2	7.0	301583	**0.60**	**1.10**	**0.24**	0	**202053**	6.41	14.88	26.43	6.51	107713	13281	0
80	0.3	9.0	16277818	**49.05**	**72.96**	**22.98**	0	**9357071**	377.96	976.23	1437.22	456.29	5107741	849373	0
80	0.4	11.0	186461982	**788.48**	**2572.13**	**201.71**	**0**	**26703290**	1244.36	3500.43	tl	3102.13	14852444	2943274	4
80	0.5	13.0	106724150	**614.49**	**1757.23**	**184.47**	**0**	**17468890**	1041.02	3144.70	tl	2369.82	9993432	2383151	3
80	0.6	16.0	303043923	**2256.96**	tl	**367.73**	**2**	**16872198**	1081.40	3412.40	tl	3028.86	9869519	2655098	3
80	0.7	19.2	95698561	**902.46**	tl	**17.60**	1	**6961383**	539.52	1671.40	tl	108.03	4182121	1232137	1
80	0.8	24.4	33677431	**391.34**	**720.18**	**158.11**	0	**3030072**	417.60	1049.45	1497.34	590.31	1899559	629662	0
80	0.9	34.0	36839	**0.53**	**1.26**	**0.05**	0	**9661**	1.91	3.46	9.79	0.30	5665	998	0

Table 1: DSATUR-$\omega(G)$ for random VCP instances

In Table 1, we compare DSATUR and DSATUR-$\omega(G)$ for random VCP instances on the same benchmark as bound $\chi_{G_A}(G)$, using the same notations.

DSATUR-$\omega(G)$ systematically produces less nodes than DSATUR for each group of random VCP instances, cutting on average about 50% of the nodes. The time spent computing the bounds covers only half of the total time which is far less than bound $\chi_{G_A}(G)$ but since it does not cut as often, it does not improve the algorithm.

In Table 2 and 3, we compare DSATUR and DSATUR-$\omega(G)$ for DIMACS instances. Since a great number of bounds compared to $\chi_{G_A}(G)$, time$_G$ is now relevant, taking up 50% of the total time. The bound computing time is just under 50% of the total time. This distribution happens because the bound is rarely effectively cutting, producing many useless bounds.

Instance				DSATUR		DSATUR-$\omega(G)$					
name	n	d	OPT*	nodes	time	nodes	time$_G$	time$_B$	time	#bounds	#cuts
queen5_5	25	0.50	5	7	**0.00**	**1**	0.00	0.00	**0.00**	1	1
queen6_6	36	0.50	7	432	**0.00**	**146**	0.00	0.00	**0.00**	88	19
queen7_7	49	0.40	7	9	**0.00**	**1**	0.00	0.00	**0.00**	1	1
queen8_12	96	0.30	12	14	**0.00**	**1**	0.00	0.00	**0.00**	1	1
queen8_8	64	0.40	9	1992673	**10.17**	**633997**	20.44	19.12	48.43	358992	83980
queen9_9	81	0.30	10	-	tl	tl	-	-	tl	18375311	3731935
queen10_10	100	0.30	11	-	tl	tl	-	-	tl	12093227	2540494
queen11_11	121	0.30	13	-	tl	tl	-	-	tl	11607207	1946952
queen12_12	144	0.30	14	-	tl	tl	-	-	tl	7667695	1372549
queen13_13	169	0.20	15	-	tl	tl	-	-	tl	6447933	1006276
queen14_14	196	0.20	17	-	tl	tl	-	-	tl	6820787	925208
queen15_15	225	0.20	18	-	tl	tl	-	-	tl	4954101	839427
queen16_16	256	0.20	19	-	tl	tl	-	-	tl	4045660	634146
myciel3	11	0.40	4	50	**0.00**	**38**	0.00	0.00	**0.00**	23	4
myciel4	23	0.30	5	1579	**0.00**	**1252**	0.00	0.00	**0.00**	694	109
myciel5	47	0.20	6	1287849	**1.48**	**1049548**	10.82	9.64	23.88	565256	79441
miles250	128	0.00	8	10	**0.00**	**1**	0.00	0.00	**0.00**	1	1
miles500	128	0.10	20	27	**0.00**	**1**	0.00	0.00	**0.00**	1	1
miles750	128	0.30	31	33	0.01	**1**	0.00	0.00	**0.00**	1	1
miles1000	128	0.40	42	191	0.05	**1**	0.00	0.00	**0.00**	1	1
miles1500	128	0.60	73	75	0.24	**1**	0.00	0.00	**0.00**	1	1
anna	138	0.10	11	13	**0.00**	**1**	0.00	0.00	**0.00**	1	1
david	87	0.10	11	13	**0.00**	**1**	0.00	0.00	**0.00**	1	1
homer	561	0.00	13	-	tl	**1**	0.00	0.00	**0.00**	1	1
huck	74	0.10	11	-	tl	**1**	0.00	0.00	**0.00**	1	1
jean	80	0.10	10	39335667	69.42	**1**	0.00	0.00	**0.00**	1	1
fpsol2.i.1	496	0.10	65	-	tl	**1**	0.00	0.00	**0.00**	1	1
fpsol2.i.2	451	0.10	30	15342	0.28	**1**	0.00	0.00	**0.00**	1	1
fpsol2.i.3	425	0.10	30	15342	0.28	**1**	0.00	0.01	**0.01**	1	1
inithx.i.1	864	0.10	54	67301	5.88	**1**	0.00	0.01	**0.01**	1	1
inithx.i.2	645	0.10	31	-	tl	**1**	0.00	0.00	**0.00**	1	1
inithx.i.3	621	0.10	31	-	tl	**1**	0.00	0.01	**0.01**	1	1
mug88_1	88	0.00	4	72832198	**50.35**	**27768646**	312.78	221.22	590.78	16384661	5000672
mulsol.i.1	197	0.20	49	-	tl	**1**	0.00	0.00	**0.00**	1	1
mulsol.i.2	188	0.20	31	53	0.01	**1**	0.00	0.00	**0.00**	1	1
mulsol.i.3	184	0.20	31	53	**0.00**	**1**	0.00	0.00	**0.00**	1	1
mulsol.i.4	185	0.20	31	53	0.01	**1**	0.00	0.00	**0.00**	1	1
mulsol.i.5	186	0.20	31	53	0.01	**1**	0.00	0.00	**0.00**	1	1
school1	385	0.30	14	16	**0.05**	**1**	0.00	17.66	17.66	1	1
school1_nsh	352	0.20	14	16	**0.03**	**1**	0.00	7.32	7.32	1	1
le450_15a	450	0.10	16	-	tl	tl	-	-	tl	1049670	203435
le450_15b	450	0.10	16	-	tl	tl	-	-	tl	1471390	258508
le450_15c	450	0.20	22	-	tl	tl	-	-	tl	807532	79501
le450_15d	450	0.20	23	-	tl	tl	-	-	tl	951701	107603
le450_25a	450	0.10	25	27	**0.00**	**1**	0.00	0.00	**0.00**	1	1
le450_25b	450	0.10	25	27	**0.00**	**1**	0.00	0.00	**0.00**	1	1
le450_25c	450	0.20	27	-	tl	tl	-	-	tl	795654	128379
le450_25d	450	0.20	27	-	tl	tl	-	-	tl	731981	133878
le450_5a	450	0.10	5	7	**0.00**	**1**	0.00	0.00	**0.00**	1	1
le450_5b	450	0.10	8	-	tl	tl	-	-	tl	981325	67736
le450_5c	450	0.10	5	7	**0.00**	**1**	0.00	0.00	**0.00**	1	1
le450_5d	450	0.10	5	7	**0.00**	**1**	0.00	0.00	**0.00**	1	1
abb313GPIA	1557	0.00	10	-	tl	tl	-	-	tl	134953	50021
ash331GPIA	662	0.00	4	34	**0.00**	**26**	0.04	0.03	0.08	15	2
ash608GPIA	1216	0.00	5	tl	tl	tl	-	-	tl	1009	4
ash958GPIA	1916	0.00	4	60	**0.02**	**48**	0.98	0.55	1.57	26	2

Table 2: DSATUR-$\omega(G)$ for DIMACS instances

Instance				DSATUR		DSATUR-$\omega(G)$					
name	n	d	OPT*	nodes	time	nodes	time$_G$	time$_B$	time	#bounds	#cuts
1-FullIns_3	30	0.20	4	128	0.00	**104**	0.00	0.00	0.00	57	8
1-FullIns_5	282	0.10	6	-	tl	tl	-	-	tl	9928304	937481
2-FullIns_3	52	0.20	5	376103299	347.15	**375944**	3.60	2.61	**7.38**	226555	77163
2-FullIns_4	212	0.10	6	-	tl	tl	-	-	tl	37813835	12879162
2-FullIns_5	852	0.00	7	-	tl	tl	-	-	tl	1910578	650516
3-FullIns_4	405	0.00	7	-	tl	tl	-	-	tl	10193901	5603340
3-FullIns_5	2030	0.00	8	-	tl	tl	-	-	tl	1001	0
4-FullIns_4	690	0.00	8	-	tl	tl	-	-	tl	2923326	1770784
4-FullIns_5	4146	0.00	9	-	tl	tl	-	-	tl	1001	0
5-FullIns_4	1085	0.00	9	-	tl	tl	-	-	tl	1666	276
1-Insertions_6	607	0.00	7	-	tl	tl	-	-	tl	1451788	254893
2-Insertions_3	37	0.10	4	37564	**0.01**	**32032**	0.11	0.14	0.27	16945	1844
2-Insertions_5	597	0.00	6	-	tl	tl	-	-	tl	2284104	316270
3-Insertions_3	56	0.10	4	8705001	**6.71**	**7231689**	40.70	30.27	82.66	3861409	491104
3-Insertions_5	1406	0.00	6	-	tl	tl	-	-	tl	208231	35187
4-Insertions_4	475	0.00	5	-	tl	tl	-	-	tl	4157812	468152
DSJC125.1	125	0.10	5	19	0.00	**14**	0.00	0.00	0.00	9	1
DSJC125.5	125	0.50	19	-	tl	tl	-	-	tl	5352841	997918
DSJC125.9	125	0.90	46	-	tl	tl	-	-	tl	2016532	639100
DSJC250.1	250	0.10	9	-	tl	tl	-	-	tl	3718382	248301
DSJC250.5	250	0.50	33	-	tl	tl	-	-	tl	1656757	232664
DSJC500.1	500	0.10	15	-	tl	tl	-	-	tl	1326586	73455
DSJC500.5	500	0.50	63	-	tl	tl	-	-	tl	466539	58342
DSJR500.1	500	0.00	12	14	0.00	**1**	0.00	0.00	0.00	1	1
DSJC1000.1	1000	0.10	25	-	tl	tl	-	-	tl	318576	13105
DSJC1000.5	1000	0.50	112	-	tl	tl	-	-	tl	15	0
flat300_20_0	300	0.50	39	-	tl	tl	-	-	tl	1586081	270917
flat300_26_0	300	0.50	40	-	tl	tl	-	-	tl	1877375	278455
flat300_28_0	300	0.50	40	-	tl	tl	-	-	tl	1785660	240087
flat1000_50_0	1000	0.50	112	-	tl	tl	-	-	tl	23	0
flat1000_60_0	1000	0.50	112	-	tl	tl	-	-	tl	19	0
games120	120	0.10	9	-	tl	**1**	0.00	0.00	**0.00**	1	1
r125.1c	125	1.00	46	48	0.15	**1**	0.00	0.00	**0.00**	1	1
r125.1	125	0.00	5	7	0.00	**1**	0.00	0.00	0.00	1	1
r125.5	125	0.50	36	-	tl	**135**	0.03	0.02	**0.09**	133	9
r250.1c	250	1.00	64	7820	**9.75**	**1**	0.00	656.96	656.96	1	1
r250.1	250	0.00	8	-	tl	**1**	0.00	0.00	**0.00**	1	1
r250.5	250	0.50	66	-	tl	tl	-	-	tl	565796	148170
r1000.1	1000	0.00	20	59	0.02	**1**	0.00	0.01	**0.01**	1	1
wap01a	2368	0.00	47	-	tl	tl	-	-	tl	4655	991
wap02a	2464	0.00	46	-	tl	tl	-	-	tl	1001	0
wap04a	5231	0.00	48	-	tl	tl	-	-	tl	1001	0
wap05a	905	0.10	50	-	tl	**908**	4.93	2.62	**8.80**	906	2
wap06a	947	0.10	47	-	tl	tl	-	-	tl	214024	167258
wap07a	1809	0.10	45	-	tl	tl	-	-	tl	4225	719
wap08a	1870	0.10	45	tl	tl	-	-	-	tl	1397	211
zeroin.i.1	211	0.20	49	51	0.05	**1**	0.00	0.00	**0.00**	1	1
zeroin.i.2	211	0.20	30	32	0.02	**1**	0.00	0.00	**0.00**	1	1
zeroin.i.3	206	0.20	30	32	0.02	**1**	0.00	0.00	**0.00**	1	1

Table 3: DSATUR-$\omega(G)$ for DIMACS instances