Solving Vertex Coloring Problems as Maximum Weight Stable Set Problems

Denis Cornaz

LAMSADE, Université Paris-Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris, France
denis.cornaz@dauphine.fr

Fabio Furini

LAMSADE, Université Paris-Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris, France
fabio.furini@dauphine.fr

Enrico Malaguti

DEI, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
enrico.malaguti@unibo.it

Abstract

In Vertex Coloring Problems, one is required to assign a color to each vertex of an undirected graph in such a way that adjacent vertices receive different colors, and the objective is to minimize the cost of the used colors. In this work we solve four different coloring problems formulated as Maximum Weight Stable Set Problems on an associated graph. We exploit the transformation proposed by Cornaz and Jost [5], where given a graph G, an auxiliary graph $\hat{G}$ is constructed, such that the family of all stable sets of $\hat{G}$ is in one-to-one correspondence with the family of all feasible colorings of G. The transformation in [5] was originally proposed for the classical Vertex Coloring and the Max-Coloring problems; we extend it to the Equitable Coloring Problem and the Bin Packing Problem with Conflicts. We report extensive computational experiments on benchmark instances of the four problems, and compare the solution method with the state-of-the-art algorithms. By exploiting the proposed method, we largely outperform the state-of-the-art algorithm for the Max-coloring Problem, and we are able to solve, for the first time to proven optimality, 14 Max-coloring and 2 Equitable Coloring instances.

Keywords: Vertex Coloring, Stable Set, Mixed Integer Linear Programming, Computational Experiments

1. Introduction

In Vertex Coloring Problems, one is required to assign a color to each vertex of an undirected graph in such a way that adjacent vertices receive different colors, and the objective is to minimize the cost of the used colors. In the classical Vertex Coloring Problem (VCP), all colors have the same cost, hence, the

objective is to minimize the number of used colors. The Equitable Coloring Problem, is a VCP with the additional restriction that subsets of vertices receiving the same color, denoted as *color classes*, differ in cardinality of at most one unit. The Max-coloring Problem is defined as a VCP where each vertex has a positive weight, and the cost of a color is given by the maximum weight of the vertices in the corresponding color class. Finally, by considering a VCP with a positive weight associated with each vertex, and imposing that the total weight of the vertices in a color class does not to exceed a given capacity, we obtain a VCP with capacity constraints. Because the problem generalizes the Bin Packing Problem as well, it is known in the literature as Bin Packing Problem with Conflicts. All mentioned problems are NP-hard.

Coloring problems are very challenging from the computational viewpoint and have several relevant applications, including, just to mention a few, scheduling [14], timetabling [6], frequency assignment [11], register allocation [4] and communication networks [23] (see Malaguti and Toth, [17]).

Problems definition. Let G be a graph with vertex set $V(G)$ and edge set $E(G)$. A stable set $S \subseteq V(G)$ is a subset of vertices containing no edge, a clique $K \subseteq V(G)$ is a subset of vertices inducing a complete subgraph. The stable sets of G are the cliques of the complementary graph $\overline{G}$ of G.

Given a weight $c \in \mathbb{Z}^{V(G)}$, the *Max Weight Stable Set Problem (MWSSP)* is to determine a stable set S of G maximizing $\sum_{v \in S} c(v)$. We denote by $\alpha(G, c)$ the optimum of MWSSP. The MWSSP can be naturally formulated as a (Mixed-)Integer Linear Program (MIP) since $\alpha(G, c) = \max c^\top x$ over x in $STAB(G)$, that is, the set of vectors of $\mathbb{R}^{V(G)}$ satisfying

$$\begin{cases} x_u + x_v \leq 1 & uv \in E(G) \\ x_v \in \{0, 1\} & v \in V(G) \end{cases}$$

A *coloring of* G is a partition $\mathcal{S} = S_1, \ldots, S_p$ of $V(G)$ into stable sets S_i. The *Vertex Coloring Problem (VCP)* is to find a coloring of G with a minimum number p of stable sets. We denote by $\chi(G)$ the optimum of VCP. The VCP is a very challenging problem from the computational viewpoint, for which state-of-the-art exact methods can fail in optimally solving instances with more than 200 vertices. The best performing exact methods for the VCP are based on the Set Covering formulation of the problem, where binary variables are associated with stable sets of G. Since stable sets of an arbitrary graph G are in exponential number with respect to the graph size, Set Covering formulations require column generation techniques and Branch-and-Price algorithms for managing the exponentially many variables. The recent algorithms by Malaguti, Monaci and Toth [16], Gualandi and Maluccelli [12], and Held, Cook and Sewell [13], are all very sophisticated implementations of a Branch-and-Price algorithm, embedding specialized (meta)heuric procedures [16], constraint programming techniques [12] and improved algorithms for the

column generation subproblem [13]. We also mention the Branch-and-Cut approach by Méndez-Díaz and Zabala [20], which is effective for some special classes of graphs.

Given a graph G, a coloring $\mathcal{S} = S_1, \ldots, S_p$ is *equitable* if $|S_i| - |S_j| \leq 1$ for each $i, j = 1, \ldots, p$. The *Equitable Coloring Problem* (ECP) is to find an equitable coloring with minimum p. We denote by $\chi_{\text{eq}}(G)$ the optimum of ECP. The most recent mathematical programming contributions to the exact solution of the ECP include the Branch-and-Cut algorithm by Bahiense et al [1], which exploits an asymmetric formulation (proposed by Campêlo, Corrêa and Campos [3] for the VCP); and the MIP formulation by Méndez-Díaz, Nasini and Savarín [19], which is strengthened by valid inequalities derived from a polyhedral study, and can be solved directly by a MIP solver. In addition, a Branch-and-Bound algorithm was recently proposed by Méndez-Díaz, Nasini and Savarín [18], extending to the ECP the famous DSATUR algorithm for the VCP by Brélaz [2]. To the best of our knowledge, no extended formulation with variables associated with stable sets of G was proposed. A possible reason is the non straightforward definition of equitable cardinality constraints for the color classes in this setting.

Given a graph G and vertex weights $c \in \mathbb{Z}^{V(G)}$, the *Max-coloring Problem* (also known as *Weighted VCP*, see [15, 10]) is the problem of determining a coloring $\mathcal{S} = S_1, \ldots, S_p$ of G which minimizes $\psi(\mathcal{S}) := \sum_{i=1}^{p} c_i$ where $c_i = \max_{v \in S_i} c(v)$. We denote by $\chi_{\max}(G, c)$ the optimum of Max-col. When c is a unit vector, $\psi(\mathcal{S}) = p$ and Max-col reduces to the VCP. The best performing exact method for the Max-col is the Branch-and-Price algorithm by Furini and Malaguti [10], which can solve instances with up-to 100 vertices.

Given a graph G, vertex weights $c \in \mathbb{Z}^{V(G)}$, and an nonnegative integer κ, *the Bin Packing Problem with Conflict (BPPC)* is to determine a coloring $\mathcal{S} = S_1, \ldots, S_p$ of G which minimizes p so that the weight of the stable sets $c(S_i) := \sum_{v \in S_i} c(v)$ does not exceed κ. The optimum is denoted by $\chi_{\text{bp}}(G, c, \kappa)$. State-of-the-art algorithms for the BPPC, recently proposed by Fernandes-Muritiba et al. [9], Elhedhli et al. [7], Sadykov and Vanderbeck [22], exploit a Set Covering formulation and implement a Branch-and-Price framework. Apparently, the presence of capacity constraints on the cardinality of the color classes reduces the practical difficulty of solving the BPPC. The mentioned algorithms can solve to optimality instances with up to 500 vertices.

Despite the effectiveness of the approaches solving exponential-size formulations, in general it is a tough task to design, implement and tune an algorithm based on the Branch-and-Price or Branch-and-Cut framework. Unfortunately, no direct formulation for coloring problems having a polynomial number of variables is known to be effective, unless it is strengthened by valid inequalities, aimed at improving the quality of the associated linear relaxation and at removing part of the symmetry affecting these formula-

tions (see, Mendez-Diaz and Zabala [20] for the VCP, and Bahiense et al. [1] and Mendez-Diaz et al. [19] for the ECP).

In this work we solve the VCP, ECP, Max-col and BPPC by reformulating them as MWSSPs on an associated graph. We exploit the transformation proposed by Cornaz and Jost [5], where given a graph G, an auxiliary graph $\hat{G}$ is constructed, such that the family of all stable sets of $\hat{G}$ is in one-to-one correspondence with the family of all feasible colorings of G. The transformation was originally proposed for the VCP and for the Max-col. We extend the transformation to the ECP and the BPPC; in these cases additional constraints have to be defined for the MWSSP. The advantage of this approach relies on simplicity: it allows us to solve coloring problems by solving MWSSPs, which can be tackled by problem specific algorithms, or can be formulated as MIPs of polynomial size, and solved by a general purpose MIP solver.

Given the transformation we exploit, where $\hat{G}$ has one vertex for each edge in the complement graph of G, the method is notably promising for instances where G is dense, thus producing auxiliary graphs $\hat{G}$ of manageable size.

2. The transformation

In [5], Cornaz and Jost explain how to transform a weighted graph (G, c) to an auxiliary weighted graph $(\hat{G}, \hat{c})$ so that:

Theorem 1 ([5]). $\alpha(\hat{G}, \hat{c}) + \chi_{\max}(G, c) = \mathbf{1}^\top c$.

The transformation starts by considering the line-graph $L(\overline{G})$ of the complementary graph $\overline{G}$ of G, that is, $L(\overline{G})$ has vertex-set the edge-set of $\overline{G}$ and two vertices are linked in $L(\overline{G})$ if the two corresponding edges are adjacent in $\overline{G}$. We let $\overrightarrow{G}$ be an acyclic orientation of $\overline{G}$ so that each edge uv of $\overline{G}$ is orientated from u to v whenever $c_u > c_v$. (Hence, $\overrightarrow{G}$ is not necessarily uniquely defined if some vertices have the same weight). Each edge of $L(\overline{G})$ corresponds to a pair of incident arcs of $\overrightarrow{G}$. The auxiliary graph $\hat{G}$ is obtained from $L(\overline{G})$ by removing each edge of $L(\overline{G})$ corresponding to a simplicial pair of arcs of $\overrightarrow{G}$, that is, two arcs with the same tail and with their heads linked by another arc. The weight $\hat{c}_e$ of a vertex e of $\hat{G}$ (arc of $\overrightarrow{G}$) is c_v where v is the head of $e = uv$ (i.e., the vertex of minimum weight).

A *simplicial star of* $\overrightarrow{G}$ is a subset of arcs all having the same tail and whose heads are pairwise linked. For any coloring $S_1, \ldots, S_p$ of G, each S_i is a clique of $\overline{G}$, and, since the orientation of $\overline{G}$ is acyclic, in each of these cliques S_i there is a unique vertex v_i which is the center of a simplicial star $\overrightarrow{S_i}$ of $\overrightarrow{G}$ which

spans S_i. Remark that

$$|S_i| - |\overrightarrow{S_i}| = 1 \tag{1}$$

Moreover,

$$\sum_{v \in S_i} c_v - \sum_{e \in \overrightarrow{S_i}} \hat{c}_e = c_{v_i} \quad \text{and} \quad c_{v_i} = \max_{v \in S_i} c_v \tag{2}$$

A *simplicial stellar forest* of $\overrightarrow{G}$ is a subset of arcs of $\overrightarrow{G}$ which can be partitioned into vertex-disjoint simplicial stars. Observe that given a simplicial stellar forest $\overrightarrow{F} = \overrightarrow{S_1} \cup \ldots \cup \overrightarrow{S_p}$ of $\overrightarrow{G}$, this corresponds to a stable set in $\hat{G}$. Since the converse holds as well, it follows that there is a 1-to-1 correspondence between the stable sets of $\hat{G}$ and the clique partition of $\overline{G}$, and hence with the colorings of G. In the following, we let $v_1, \ldots, v_p$ be the centers of the simplicial stars of $\overrightarrow{G}$ associated with the coloring $S_1, \ldots, S_p$ of G, and we will use the following

Observation 1.

$$\text{If } v \in \{v_1, \ldots, v_p\}, \text{ then } \delta^-(v) \cap \overrightarrow{F} = \emptyset, \text{ otherwise } |\delta^-(v) \cap \overrightarrow{F}| = 1 \tag{3}$$

where $\delta^-(v)$ is the set of all arcs of $\overrightarrow{G}$ the head of which is v, and an isolated vertex is the center of an empty star. Similarly, $\delta^+(v)$ is the set of all arcs of $\overrightarrow{G}$ the tail of which is v.

Theorem 1 follows from (2) by summing over all centers v_i. It allows to formulate any instance of Max-col as an instance of MWSSP. In the two following propositions, we give MIP formulations for ECP and BPPC. Both are obtained from the basic MWSSP formulation in $\hat{G}$ by adding constraints.

Proposition 1. $\chi_{\text{eq}}(G) = \min \mathbf{1}^\top x$ over all $x \in STAB(\hat{G})$ satisfying

$$x(\delta^+(u)) - x(\delta^+(v)) \leq 1 + x(\delta^-(v))\Big(|\delta^+(u)| - 1\Big) \quad \text{for all } u, v \in V(G) \tag{4}$$

where $x(A)$ stands for $\sum_{a \in A} x_a$.

Proof of Proposition 1. If u is not a center, then $x(\delta^+(v)) = 0$ and the inequality is trivially satisfied. If v is not a center, then by (3), we have $x(\delta^-(v)) = 1$ and again the inequality is trivially satisfied. When both u and v are centers, the constraints ensures that $|\overrightarrow{S_u}| - |\overrightarrow{S_v}| \leq 1$ where $\overrightarrow{S_u}, \overrightarrow{S_v}$ are the simplicial stars of u, v. By (1), the coloring is 1-balanced. *(end of proof)*

Proposition 2. $\chi_{\text{bp}}(G, c, \kappa) = \min \mathbf{1}^\top x$ over all $x \in STAB(\hat{G})$ satisfying

$$c_v + \sum_{w:vw \in \delta^+(v)} c_w x_{vw} \leq \kappa \quad \text{for all vertex } v \in V(G) \tag{5}$$

Proof of Proposition 2. If v is not a center, then $c_v \leq \kappa$ otherwise the problem is unfeasible. If v is a center of a simplicial star $\overrightarrow{S_v}$, then the left-hand-side of the constraint is equal to the weight $c(S_v)$ of the corresponding stable set. *(end of proof)*

In Figure 1, we depict an example of the transformation. On the left of the figure we report a graph $\overline{G}$ with 9 vertices and 12 edges. For each vertex v we report the weight c_v, and the orientation $\overrightarrow{G}$ is represented as well. The simplicial stellar forest $\overrightarrow{F}$ is depicted by bold arrows. On the right of the figure, we depict the transformed graph $\hat{G}$, where for each vertex u, v we report the corresponding weight $c_{u,v} = c_v$. Vertices in the stable set corresponding to $\overrightarrow{F}$ are represented by larger circles.

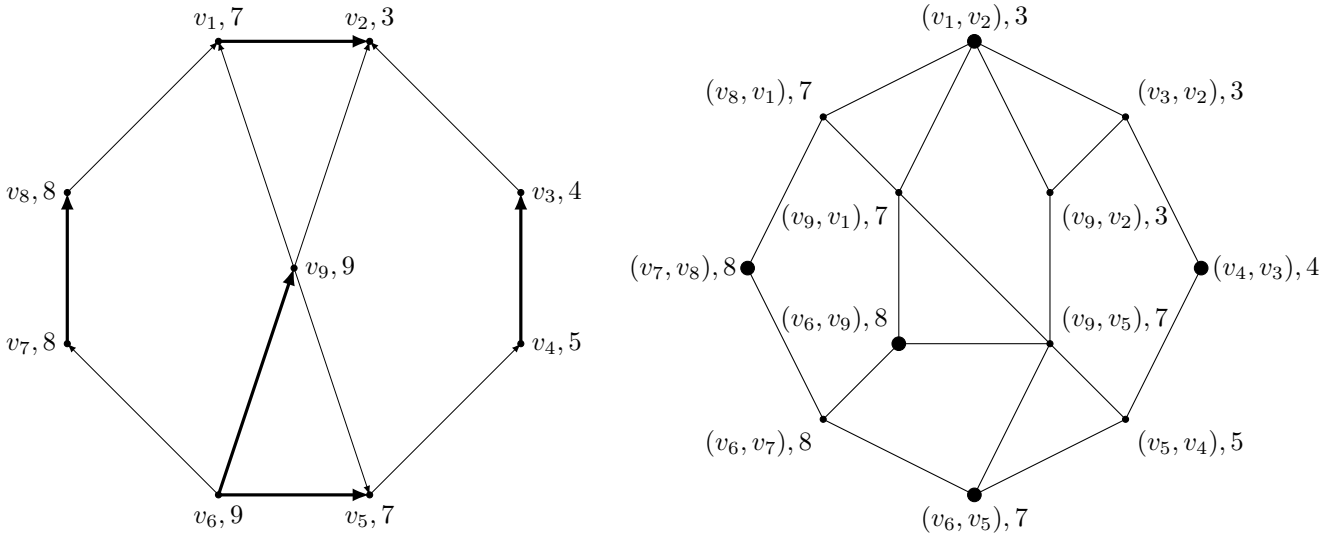

Figure 1: Complementary graph $\overline{G}$ (left) and auxiliary graph $\hat{G}$ (right).

3. Computational Experiments

We considered VCP, Max-col, ECP and BPPC instances, we derived the auxiliary graph $\hat{G}$ as described in the previous section, and solved the resulting MWSSP (with additional constraints (4) and (5) in the case of the ECP and BPPC, respectively). In the following, we denote this solution procedure by $MWSS(\hat{G})$. Experiments were performed on a single core of a i5 linux PC with 8 GB RAM. Results are compared with state-of-the-art algorithms for the corresponding problems, all based on Branch-and-Price for the VCP, Max-col and BPPC, and Branch-and-Cut algorithms and a Branch-and-Bound algorithm for the

ECP. An important parameter in these experiments is the *density* of the graph, denoted by δ, which is the number of edges divided by the maximum possible number of edges (so $\delta(G) + \delta(\overline{G}) = 1$).

3.1. Algorithms for the Maximum Weight Stable Set Problem.

MWSSPs can be solved by means of specialized algorithms, or formulated as a MIP (see Section 1) and directly tackled by a MIP solver.

In our preliminary experiments we considered the following algorithms for solving MWSSPs: i) the Cliquer algorithm by Ostergard, [21]; ii) the Combinatorial Branch and Bound algorithm by Held et al. [13]; iii) the MIP formulation of the MWSSP solved by CPLEX 12.6.

According to our experiments, the best choice is to tackle directly the MIP formulation of the MWSSP by CPLEX. This result is partially in contrast to what could be expected, since specialized algorithms are known to be faster when a MWSSP has to be solved as a column generation subproblem in a Branch-and-Price algorithm (see, e.g., [16, 13, 10]). Note that in the column generation case, the MWSSP to be solved are defined on smaller-size graphs. From our computational experience, for very large graphs, like the ones obtained from the transformation, specialized algorithms tend to be slower. For example, for the 53 VCP instances considered in the next section, within a time limit of 1h of computing time, CPLEX was able to solve to optimality the MWSSP on graph $\hat{G}$ for 38 instances, versus the 3 and 6 instances solved to optimality by Cliquer [21] and the Branch-and-Bound algorithm in [13], respectively. In the following we report results obtained by solving the MIP formulation of the MWSSP (with additional constraints (4) and (4) in the case of the ECP and BPPC, respectively) by CPLEX 12.6.

3.2. Computational results for the VCP.

None of the Branch-and-Price algorithms [16], [12] and [13] clearly dominates the others, hence, we compare our results with [16], which reports an extensive set of computational results for benchmark instances. We tested the 115 DIMACS benchmark instances (`ftp://dimacs.rutgers.edu/pub/challenge/graph/`) considered in [16], and restrict the set to 58 instances for which the auxiliary graph $\hat{G}$ has at most 30,000 vertices. In Table 1 we report the cardinality of the vertex sets $|V(G)|$ and $|V(\hat{G})|$, and of the edge sets $|E(G)|$ and $|E(\hat{G})|$, respectively, for the original and transformed graphs. Clearly, the size of $V(\hat{G})$ remains limited for dense graphs G, which are the instances where the method we study is more promising. In the table we then report the results obtained by solving the MWSSP on the transformed graph $\hat{G}$ by CPLEX 12.6, with a limit of 10 hours of computing time (columns $MWSS(\hat{G})$). Namely we report, for each considered instance, a lower

bound LB and an upper bound UB on the value $\chi(G)$, and the corresponding computing time (tl when time limit is reached), which includes the (usually very small) time for applying the transformation of G to $\hat{G}$. In the next columns, we report the results of the Branch-and-Price algorithm in [16] (columns BP [16]), obtained within the same 10 hours of computing time on a slightly slower Pentium 4 PC with 2 GB RAM. On the considered 58 instances, the $MWSS(\hat{G})$ method performs very well, solving to optimality 38 instances, compared with the 39 instances optimally solved by [16]. In particular the described method can solve instance DSJC250.9, which was only recently solved to optimality by [13] with almost 3 hours of computing time.

3.3. Computational results for the Max-col.

The best performing algorithm for Max-col, i.e., the Branch-and-Price algorithm [10], can solve instances with up to 100 vertices. This makes the $MWSS(\hat{G})$ method an excellent candidate alternative for solving the Max-col, since we expect the weighted auxiliary graph $\hat{G}$ being not too large, and we do not have to spoil the MWSSP formulation with additional constraints. We considered all the instances in [10]. This includes 46 instances from the COLOR02/03/04 benchmark (`http://mat.gsia.cmu.edu/COLOR02`) and two sets of instances from matrix-decomposition problems. In Tables 2, 4 and 5 we report the size of the original and transformed graphs, and the results obtained by the presented method (columns $MWSS(\hat{G})$), followed by the results from [10], obtained on the same computer. Time limits are set to 1 hour of computing time for the first to two sets (DIMACS and p) and to 4 hours for the last set (R), as in [10].

Clearly the results obtained by the $MWSS(\hat{G})$ method largely outperform those in [10] (obtained on the same computer): concerning the 46 COLOR02/03/04 benchmark, 38 instances are solved to optimality, i.e., 2 more than [10]. By allowing up to 10 hours of computing time (see Table 3), the $MWSS(\hat{G})$ method can additionally solve to optimality instances R100_1g and R100_1gb, and for unsolved instances, the optimality gaps are further reduced and always smaller than those obtained by [10] (while the Branch-and-Price algorithm in [10] does not benefit from additional computing time). Concerning the p instances, which are all solved to optimality by both methods, the average computing time is reduced of one order of magnitude, from 6.83 to 0.35 seconds. Finally, for the R instances, [10] reports only 19 optimal solutions, while all the 30 instances are optimally solved by the $MWSS(\hat{G})$ method.

3.4. Computational results for the ECP

For solving the ECP, we consider the MWSSP on graph $\hat{G}$ and the additional balance constraints (4). Constraints (4) are $O(n^2)$ in total, since one constraint is defined for each pair of vertices $u, v \in V$, but

	G		$\hat{G}$		$MWSS(\hat{G})$			BP [16]		
instance	$\|V(G)\|$	$\|E(G)\|$	$\|V(\hat{G})\|$	$\|E(\hat{G})\|$	LB	UB	$time$	LB	UB	$time$
DSJC125.1	125	736	7,014	545,250	5	6	tl	5	5	142.0
DSJC125.5	125	3,891	3,859	197,166	14	19	tl	17	17	18,290.0
DSJC125.9	125	6,961	789	9,503	44	44	0.9	44	44	3,916.0
DSJC250.1	250	3,218	27,907	4,350,878		250	tl	6	8	tl
DSJC250.5	250	15,668	15,457	1,588,249	16	43	tl	20	28	tl
DSJC250.9	250	27,897	3,228	79,902	72	72	3,393.6	71	72	tl
DSJC500.9	500	112,437	12,313	584,336	122	134	tl	123	127	tl
DSJR500.1c	500	121,275	3,475	39,563	85	85	2.8	85	85	288.5
queen5_5	25	160	140	1,240	5	5	0.0	5	5	0.2
queen6_6	36	290	340	5,092	7	7	0.6	7	7	6.0
queen7_7	49	476	700	15,736	7	7	0.4	7	7	0.2
queen8_8	64	728	1,288	40,344	9	9	496.6	9	9	3.6
queen9_9	81	1,056	2,184	90,672	10	10	21,348.7	10	10	36.6
queen10_10	100	1,470	3,480	184,600	10	12	tl	11	11	687.0
queen11_11	121	1,980	5,280	348,040	12	13	tl	12	12	1,875.7
queen12_12	144	2,596	7,700	616,940	12	15	tl	12	13	tl
queen13_13	169	3,328	10,868	1,039,688	13	16	tl	13	14	tl
queen14_14	196	4,186	14,924	1,679,580	14	17	tl	14	15	tl
queen15_15	225	5,180	20,020	2,617,720	15	29	tl	15	16	tl
queen16_16	256	6,320	26,320	3,955,952	16	27	tl	16	17	tl
myciel3	11	20	35	150	4	4	0.0	4	4	4.7
myciel4	23	71	182	2,051	5	5	0.3	5	5	123.0
myciel5	47	236	845	21,810	6	6	330.9	4	6	tl
myciel6	95	755	3,710	206,615	6	7	tl	4	7	tl
myciel7	191	2,360	15,785	1,840,230	4	8	tl	5	8	tl
mulsol.i.1	197	3,925	15,381	1,805,912	49	49	10.4	49	49	0.2
mulsol.i.2	188	3,885	13,693	1,498,197	31	31	8.4	31	31	4.7
mulsol.i.3	184	3,916	12,920	1,371,624	31	31	6.8	31	31	0.2
mulsol.i.4	185	3,946	13,074	1,396,293	31	31	6.8	31	31	0.2
mulsol.i.5	186	3,973	13,232	1,421,732	31	31	8.5	31	31	6.0
zeroin.i.1	211	4,100	18,055	2,303,047	49	49	15.3	49	49	3.8
zeroin.i.2	211	3,541	18,614	2,383,090	30	30	17.1	30	30	4.4
zeroin.i.3	206	3,540	17,575	2,185,681	30	30	16.2	30	30	4.5
anna	138	493	8,960	790,966	11	11	7.0	11	11	3.6
david	87	406	3,335	178,437	11	11	1.0	11	11	0.2
huck	74	301	2,400	108,649	11	11	0.4	11	11	0.2
jean	80	254	2,906	144,975	10	10	0.8	10	10	0.2
games120	120	638	6,502	487,295	9	9	8.0	9	9	0.2
miles250	128	387	7,741	634,554	8	8	5.7	8	8	5.0
miles500	128	1,170	6,958	541,403	20	20	127.8	19	20	3.7
miles750	128	2,113	6,015	436,137	31	31	3.3	31	31	0.2
miles1000	128	3,216	4,912	323,889	42	42	1.5	42	42	0.2
miles1500	128	5,198	2,930	158,110	73	73	1.3	73	73	0.1
1-Insertions_4	67	232	1,979	80,730	4	5	tl	3	5	tl
1-Insertions_5	202	1,227	19,074	2,461,400	4	6	tl	3	6	tl
2-Insertions_4	149	541	10,485	1,001,021	4	5	tl	3	5	tl
4-Insertions_3	79	156	2,925	146,146	4	4	29,459.3	3	4	tl
1-FullIns_4	93	593	3,685	205,688	5	5	1,907.4	4	5	tl
1-FullIns_5	282	3,247	36,374	6,487,330	4	6	tl	4	6	tl
2-FullIns_3	52	201	1,125	34,190	5	5	4.0	5	5	2.9
2-FullIns_4	212	1,621	20,745	2,791,046	4	6	tl	5	6	tl
3-FullIns_3	80	346	2,814	137,405	6	6	726.9	6	6	2.9
4-FullIns_3	114	541	5,900	420,462	7	7	27,559.8	7	7	3.4
5-FullIns_3	154	792	10,989	1,073,628	7	8	tl	8	8	4.6
mug88_1	88	146	3,682	206,951	4	4	9.3	4	4	9.6
mug88_25	88	146	3,682	206,953	4	4	14.9	4	4	10.6
mug100_1	100	166	4,784	307,175	4	4	41.5	4	4	14.4
mug100_25	100	166	4,784	307,175	4	4	26.2	4	4	12.0

Table 1: Results for the VCP, DIMACS benchmark instances.

	G		$\hat{G}$		$MWSS(\hat{G})$			BP [16]		
instance	$\|V(G)\|$	$\|E(G)\|$	$\|V(\hat{G})\|$	$\|E(\hat{G})\|$	LB	UB	$time$	LB	UB	$time$
DSJC125_1g	125	736	7,014	545,250	20	30	tl	21	27	tl
DSJC125_1gb	125	736	7,014	545,250	76	104	tl	82	96	tl
DSJC125_5g	125	3,891	3,859	197,166	62	82	tl	68	82	tl
DSJC125_5gb	125	3,891	3,859	197,166	209	284	tl	230	273	tl
DSJC125_9g	125	6,961	789	9,503	169	169	0.3	169	169	2
DSJC125_9gb	125	6,961	789	9,503	604	604	0.4	604	604	5
GEOM100	100	547	4,403	271,017	65	65	3.8	65	65	12
GEOM100a	100	992	3,958	230,557	89	89	2.4	89	89	2
GEOM100b	100	1,050	3,900	225,378	32	32	3.8	32	32	1
GEOM110	110	638	5,357	364,224	68	68	59.5	68	69	tl
GEOM110a	110	1,207	4,788	307,120	97	97	12.9	97	97	7
GEOM110b	110	1,256	4,739	302,283	37	37	5.0	37	37	2
GEOM120	120	773	6,367	472,507	72	72	12.4	72	72	12
GEOM120a	120	1,434	5,706	400,114	105	105	7.0	105	105	11
GEOM120b	120	1,491	5,649	393,915	35	35	16.5	35	35	1
GEOM30b	30	81	354	5,923	12	12	0.0	12	12	0
GEOM40b	40	157	623	14,023	16	16	0.1	16	16	0
GEOM50b	50	249	976	27,722	18	18	0.1	18	18	0
GEOM60b	60	366	1,404	48,110	23	23	0.5	23	23	0
GEOM70	70	267	2,148	91,726	47	47	0.3	47	47	1
GEOM70a	70	459	1,956	79,625	73	73	0.4	73	73	1
GEOM70b	70	488	1,927	77,712	24	24	0.9	24	24	0
GEOM80	80	349	2,811	137,702	66	66	1.1	66	66	2
GEOM80a	80	612	2,548	118,686	76	76	1.1	76	76	1
GEOM80b	80	663	2,497	114,959	27	27	2.5	27	27	1
GEOM90	90	441	3,564	197,010	61	61	2.0	61	61	4
GEOM90a	90	789	3,216	168,570	73	73	4.8	73	73	1
GEOM90b	90	860	3,145	162,945	30	30	1.7	30	30	1
R100_1g	100	509	4,441	273,695	19	23	tl	20	25	tl
R100_1gb	100	509	4,441	273,695	76	83	tl	76	90	tl
R100_5g	100	2,456	2,494	102,327	53	62	tl	56	65	tl
R100_5gb	100	2,456	2,494	102,327	203	225	tl	212	233	tl
R100_9g	100	4,438	512	5,033	141	141	0.1	141	141	0
R100_9gb	100	4,438	512	5,033	518	518	0.3	517	518	5
R50_1g	50	108	1,117	34,029	14	14	0.8	14	14	1
R50_1gb	50	108	1,117	34,029	53	53	1.6	53	53	0
R50_5g	50	612	613	12,251	37	37	1.4	36	37	2
R50_5gb	50	612	613	12,251	135	135	1.5	131	135	22
R50_9g	50	1,092	133	654	74	74	0.0	74	74	0
R50_9gb	50	1,092	133	654	262	262	0.0	262	262	0
R75_1g	70	251	2,164	92,467	18	18	132.8	18	18	12
R75_1gb	70	251	2,164	92,467	70	70	192.2	67	70	447
R75_5g	75	1,407	1,368	41,115	51	51	1,056.0	49	51	2,136
R75_5gb	75	1,407	1,368	41,115	186	186	989.3	179	187	tl
R75_9g	75	2,513	262	1,724	110	110	0.0	109	110	0
R75_9gb	75	2,513	262	1,724	396	396	0.0	396	396	1

Table 2: Results for the Max-col, COLOR02/03/04 instances, time limit of 1 hour.

	G		$\hat{G}$		$MWSS(\hat{G})$		
instance	$\|V(G)\|$	$\|E(G)\|$	$\|V(\hat{G})\|$	$\|E(\hat{G})\|$	LB	UB	$time$
DSJC125_1g	125	736	7,014	545,250	20	25	tl
DSJC125_1gb	125	736	7,014	545,250	78	95	tl
DSJC125_5g	125	3,891	3,859	197,166	63	78	tl
DSJC125_5gb	125	3,891	3,859	197,166	212	263	tl
R100_1g	100	509	4,441	273,695	21	21	28,788.5
R100_1gb	100	509	4,441	273,695	81	81	9,362.2
R100_5g	100	2,456	2,494	102,327	55	59	tl
R100_5gb	100	2,456	2,494	102,327	210	225	tl

Table 3: Results for the Max-col, COLOR02/03/04 instances, time limit of 10 hours.

the constraint is active only if u, v are the centers of simplicial stars associated with two distinct colors (see proposition 1). Hence, there is a large number of constraints which are rarely activated. In this case, better results are obtained by defining (4) as *lazy constraints* in the CPLEX MIP solver, i.e., constraints that are not added to the formulation, but checked only when an incumbent (integer) solution is found (and eventually added to the formulation when violated).

The three recent papers [19], [18] and [1] report computational experiments on randomly generated graphs and the DIMACS benchmark instances. We consider the same random graphs used for testing the Branch-and-Bound algorithm in [18], which also summarizes results obtained by solving the compact formulation from [19], having from 60 to 120 vertices and densities $\delta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, as reported in Table 6. Each row of the table reports average results over 30 instances for the size of the original and transformed graphs. The table also reports the number of instances solved to optimality (opt) within 2 hours of computing time by the approach $MWSS(\hat{G})$, and the average computing time (for solved instances only). In the table we also report the number of optimally solved instances and the average computing time for [19] and [18] (obtained with a PC similar to the one we use).

The approach we propose can solve all instances with 60 vertices with the exception of 3 instances with $\delta = 0.3$, while only 10 instances with 70 vertices and $\delta = 0.3$ are solved, and no instance with 70 vertices and $\delta = 0.5$ is solved. When 80 or more vertices are considered, instances with $\delta = 0.1, 0.7, 0.9$ remain more tractable, while instances with $\delta = 0.3, 0.5$ confirm to be difficult for the approach. Considering the same instances, complete results for the formulation in [19] are available up to 70 vertices (300 instances in total). The formulation in [19] can be solved to optimality for 217 instances, while the approach we propose can solve 257 of the same instances. The Branch-and-Bound algorithm [18] performs very well on these random problems, indeed, it can solve to optimality all the 300 instances with up to 70 vertices, and

	G		$\hat{G}$		$MWSS(\hat{G})$			BP [16]		
instance	$\|V(G)\|$	$\|E(G)\|$	$\|V(\hat{G})\|$	$\|E(\hat{G})\|$	LB	UB	$time$	LB	UB	$time$
P06	16	38	82	608	565	565	0.0	565	565	0
P07	24	92	184	2,112	3,771	3,771	0.0	3,771	3,771	0
p08	24	92	184	2,112	4,049	4,049	0.0	4,049	4,049	0
p09	25	100	200	2,400	3,388	3,388	0.0	3,388	3,388	0
p10	16	32	88	688	3,983	3,983	0.0	3,983	3,983	0
p11	18	48	105	904	3,380	3,380	0.0	3,380	3,380	0
p12	26	90	235	3,118	657	657	0.0	657	657	0
p13	34	160	401	7,048	3,220	3,220	0.0	3,220	3,220	1
p14	31	110	355	5,899	3,157	3,157	0.0	3,157	3,157	0
p15	34	136	425	7,760	341	341	0.0	341	341	0
p16	34	134	427	7,816	2,343	2,343	0.0	2,343	2,343	0
p17	37	161	505	10,090	3,281	3,281	0.0	3,281	3,281	0
p18	35	143	452	8,523	3,228	3,228	0.0	3,228	3,228	0
p19	36	156	474	9,196	3,710	3,710	0.0	3,710	3,710	0
p20	37	142	524	10,715	1,830	1,830	0.0	1,830	1,830	1
p21	38	155	548	11,471	3,660	3,660	0.0	3,660	3,660	0
p22	38	154	549	11,498	1,912	1,912	0.0	1,912	1,912	0
p23	44	204	742	18,179	3,770	3,770	0.1	3,770	3,770	0
p24	34	104	457	8,723	661	661	0.0	661	661	0
p25	36	120	510	10,308	504	504	0.0	504	504	0
p26	37	131	535	11,086	520	520	0.0	520	520	0
p27	44	174	772	19,356	216	216	0.1	216	216	2
p28	44	164	782	19,766	1,729	1,729	0.1	1,729	1,729	0
p29	53	254	1,124	34,271	3,470	3,470	0.1	3,470	3,470	0
p30	60	317	1,453	50,540	4,891	4,891	0.2	4,891	4,891	1
p31	47	179	902	24,580	620	620	0.1	620	620	0
p32	51	221	1,054	31,117	2,480	2,480	0.1	2,480	2,480	0
p33	56	258	1,282	41,840	3,018	3,018	0.3	3,018	3,018	1
p34	74	421	2,280	100,077	1,980	1,980	0.6	1,980	1,980	2
p35	86	566	3,089	158,255	2,140	2,140	0.6	2,140	2,140	2
p36	101	798	4,252	256,244	7,210	7,210	1.4	7,210	7,210	4
p38	87	537	3,204	167,359	2,130	2,130	1.2	2,130	2,130	10
p40	86	497	3,158	163,820	4,984	4,984	1.0	4,984	4,984	13
p41	116	900	5,770	406,504	2,688	2,688	3.2	2,688	2,688	31
p42	138	1,186	8,267	698,948	2,466	2,466	3.2	2,466	2,466	171

Table 4: Results for the Max-col, p instances.

	G		$\hat{G}$		$MWSS(\hat{G})$			BP [16]		
instance	$\|V(G)\|$	$\|E(G)\|$	$\|V(\hat{G})\|$	$\|E(\hat{G})\|$	LB	UB	$time$	LB	UB	$time$
r01	144	1,280	9,016	796,427	6,724	6,724	17.6	6,724	6,724	171
r02	142	1,246	8,765	763,268	6,771	6,771	11.3	6,771	6,771	200
r03	139	1,188	8,403	716,330	6,473	6,473	10.4	6,473	6,473	250
r04	151	1,406	9,919	919,467	6,342	6,342	11.4	6,342	6,342	565
r05	142	1,266	8,745	760,628	6,408	6,408	12.3	6,408	6,408	458
r06	148	1,381	9,497	861,204	7,550	7,550	10.3	7,550	7,550	136
r07	141	1,253	8,617	743,949	6,889	6,889	13.7	6,889	6,889	210
r08	138	1,191	8,262	698,277	6,057	6,057	4.1	6,057	6,057	289
r09	129	1,027	7,229	571,102	6,358	6,358	9.5	6,358	6,358	241
r10	150	1,409	9,766	898,183	6,508	6,508	13.1	6,508	6,508	227
r11	208	2,247	19,281	2,501,166	7,654	7,654	57.2	7,654	7,654	2,973
r12	199	2,055	17,646	2,189,139	7,690	7,690	53.7	7,690	7,690	3,161
r13	217	2,449	20,987	2,841,217	7,500	7,500	105.2	7,500	-	tl
r14	214	2,387	20,404	2,723,361	8,254	8,254	65.3	8,254	8,254	1,420
r15	198	2,055	17,448	2,152,308	8,021	8,021	25.1	8,021	8,021	1,157
r16	188	1,861	15,717	1,839,320	7,755	7,755	26.7	7,755	7,755	712
r17	213	2,392	20,186	2,679,713	7,979	7,979	79.3	7,979	7,979	2,257
r18	200	2,079	17,821	2,221,870	7,232	7,232	58.2	7,232	7,232	2,502
r19	185	1,803	15,217	1,751,992	6,826	6,826	32.7	6,826	6,826	1,349
r20	217	2,447	20,989	2,841,629	8,023	8,023	104.1	8,023	8,023	993
r21	281	3,554	35,786	6,339,798	9,284	9,284	390.0	-	-	tl
r22	285	3,684	36,786	6,607,860	8,887	8,887	302.6	-	-	tl
r23	288	3,732	37,596	6,827,707	9,136	9,136	375.3	-	-	tl
r24	269	3,284	32,762	5,552,048	8,464	8,464	201.7	8,464	-	tl
r25	266	3,177	32,068	5,376,273	8,426	8,426	225.6	-	-	tl
r26	284	3,629	36,557	6,546,162	8,819	8,819	439.6	-	-	tl
r27	259	3,019	30,392	4,959,581	7,975	7,975	248.1	7,975	-	tl
r28	288	3,765	37,563	6,818,692	9,407	9,407	222.7	9,407	-	tl
r29	281	3,553	35,787	6,340,075	8,693	8,693	388.0	8,693	-	tl
r30	301	4,122	41,028	7,785,349	9,816	9,816	346.9	9,816	-	tl

Table 5: Results for the Max-col, R instances.

	G		$\hat{G}$		$MWSS(\hat{G})$		MIP [19]		BB [18]	
δ	$\|V(G)\|$	$\|E(G)\|$	$\|V(\hat{G})\|$	$\|E(\hat{G})\|$	opt	$time$	opt	$time$	opt	$time$
10	60	179.1	1,590.9	58,084.5	30	4.6	30	0.0	30	0.0
30	60	534.8	1,235.2	38,362.7	27	592.5	30	506.0	30	0.0
50	60	883.2	886.8	21,465.4	30	679.0	19	1,825.0	30	0.6
70	60	1,234.6	535.4	8,442.3	30	30.3	27	942.0	30	1.6
90	60	1,589.3	180.7	1,044.4	30	0.0	30	1.0	30	0.0
10	70	242.6	2,172.4	93,040.1	30	51.7	30	0.0	30	0.0
30	70	719.1	1,695.9	62,018.6	0		15	4,005.0	30	0.1
50	70	1,205.8	1,209.2	34,317.8	10	4,237.6	0		30	6.7
70	70	1,688.7	726.3	13,353.5	30	68.0	6	236.0	30	21.0
90	70	2,177.5	237.5	1,544.3	30	0.0	30	258.0	30	0.1
10	80	312.7	2,847.3	140,007.9	30	544.0	30	1.3	30	0.0
30	80	950.5	2,209.5	92,399.3	0				30	22.7
50	80	1,573.0	1,587.0	51,780.9	0				28	324.0
70	80	2,217.2	942.8	19,745.3	29	1,153.3			21	1,817.0
90	80	2,843.8	316.2	2,385.2	30	0.1	27	659.0	30	6.6
10	90	402.1	3,602.9	199,697.2	21	3,202.4			30	0.0
30	90	1,200.1	2,804.9	132,520.9	0				30	52.0
50	90	2,005.4	1,999.6	73,237.9	0					
70	90	2,799.7	1,205.3	28,736.1	23	2,261.1				
90	90	3,598.9	406.1	3,502.0	30	0.1			30	161.0
10	100	504.7	4,445.3	274,114.9	18	3,441.0	30	15.5	30	0.0
10	120	715.8	6,424.2	477,499.4	0		15	1,673.0	30	2.6

Table 6: Results for the ECP, random instances from [18].

141 instances with 80 vertices, while our approach can solve 89 instances with 80 vertices. Our approach seems to have a better performance on dense graphs: it is from one to four orders of magnitude faster for instances with $\delta = 0.9$ and 70 or more vertices (both approaches solve all instances in these classes), and can solve 29 instances with 80 vertices and $\delta = 0.7$, while the algorithm in [18] can solve only 21 instances in this class.

Concerning the Branch-and-Cut algorithm in [1], it considers 5 instances per class, with up to 70 vertices. It can solve 2 instances with 60 vertices and $\delta = 0.3$, and 3 instances with 60 vertices and $\delta = 0.5$, while no instance with 70 vertices and $\delta = 0.3$ or $\delta = 0.5$ is solved within a time limit of 2 hours on a 1.8GHz PC with 1 GB RAM. This allows us to conclude that the method we propose has a better performance than [1] on randomly generated instances.

We also experimented our method on 54 DIMACS instances and 4 $kneser$ graphs considered in [19], and restrict the set to 42 instances for which the auxiliary graph $\hat{G}$ has at most 30,000 vertices. A subset of these instances was considered in [1] as well. Table 7 reports the size of the original and transformed

graphs, and the results obtained by the presented method (columns $MWSS(\hat{G})$), followed by the computing times from [19], [1] and [18], from the respective papers. The approach we propose performs satisfactorily on these instances, and solves to optimality 23 out of 54 instances within 2 hours of computing time. It worth mention that we are able to solve to optimality, for the first time, instance DSJC125.9, and, by allowing 22312 seconds of computing time on a multi-core cpu, instance DSJC250.9, whose optimal solution value is 72 (reported time is the sum for all the cpu cores). The Branch-and-Bound algorithm [19] is the best performing approach for this set of instances, and can solve 31 instances to optimality within the same time limit. The Branch-and-Bound algorithm [18] can solve 26 instances (the miles750 and miles1000 instances, where time limit is marked tl^*, can be solved by choosing a different strategy in the algorithm). Finally, the algorithm in [1] solves all the 19 considered instances.

3.5. Computational results for the BPPC.

State-of-the-art algorithms for the BPPC, proposed by Fernandes-Muritiba et al. [9], Elhedhli et al. [7] and Sadykov and Vanderbeck [22] are all based on the Branch-and-Price framework. All these algorithms are tested on BPPC instances with up to 500 items (vertices), obtained by adding random incompatibility graphs with densities in the range $0.1 - 0.9$ to 80 BPP instances from Falkenauer [8], for a total of 720 instances. Mentioned Branch-and-Price algorithms can solve approximately 700 out of 720 instances. The most difficult ones are those with densities in the range $0.3 - 0.4$. The approach we propose, that is, formulating the MWSSP on graph $\hat{G}$ with the additional capacity constraints (5), can solve almost all instances with density 0.9, and a few instances with density 0.8, within 1 hour of computing time. However, these instances are easily solved by Branch-and-Price algorithms. Thus, we conclude that, although the method we propose can solve BPPC instances, it is not competitive with state-of-the-art Branch-and-Price algorithms.

4. Conclusions

We exploited the idea of solving Coloring problems on a graph by solving Maximum Weight Stable Set problems on an auxiliary graph, the solutions of which are in one-to-one correspondence with the colorings of the original graph. The contribution of the paper is twofold: first, the extensive computational experiments we performed showed that we could solve some very hard instances of the Vertex Coloring Problem on dense graphs, while for the Max-coloring Problem, the current state-of-the-art algorithm is largely outperformed; second, we extended the idea to the Equitable Coloring Problem and to the Bin Packing Problem with Conflicts, and performed extensive computational experiments for these

	G		$\hat{G}$		$MWSS(\hat{G})$			MIP [19]	BC [1]	BB [18]
instance	$\|V(G)\|$	$\|E(G)\|$	$\|V(\hat{G})\|$	$\|E(\hat{G})\|$	LB	UB	$time$	$time$	$time$	$time$
miles750	128	2,113	6,015	436,137	31	31	4.6	0	171	tl*
miles1000	128	3,216	4,912	323,889	42	42	17.7	0	267	tl*
miles1500	128	5,198	2,930	158,110	73	73	4.2	0	13	0
zeroin.i.1	211	4,100	18,055	2,303,047	49	50	tl	0	50	0
zeroin.i.2	211	3,541	18,614	2,383,090	30	37	tl	2	510	tl
zeroin.i.3	206	3,540	17,575	2,185,681	30	37	tl	5	491	tl
queen6_6	36	290	340	5,092	7	7	1.0	1	1	0
queen7_7	49	476	700	15,736	7	7	0.4	0	0	0
queen8_8	64	728	1,288	40,344	9	10	tl	654	441	6
queen8_12	96	1,368	3,192	162,036	12	12	30.0	5		3,079
queen9_9	81	1,056	2,184	90,672	9	11	tl	tl		475
queen10_10	100	1,470	3,480	184,600	10	13	tl	tl		tl
david	87	406	3,335	178,437	11	30	tl	0	13	0
1-FullIns_3	30	100	335	5,342	4	4	0.3	0	2	0
2-FullIns_3	52	201	1,125	34,190	5	5	10.0	0	25	1
3-FullIns_3	80	346	2,814	137,405	6	6	429.9	0	85	tl
4-FullIns_3	114	541	5,900	420,462	6	7	tl	0	72	tl
5-FullIns_3	154	792	10,989	1,073,628	7	8	tl	0	268	tl
1-FullIns_4	93	593	3,685	205,688	5	5	1,269.2	28		1,404
mug88_1	88	146	3,682	206,951	4	4	27.6	1		109
mug88_25	88	146	3,682	206,953	4	4	28.4	0		56
mug100_1	100	166	4,784	307,175	4	4	78.7	1		4,425
mug100_25	100	166	4,784	307,175	4	4	49.3	1		4,978
mulsol.i.1	197	3,925	15,381	1,805,912	49	49	545.3	1		0
mulsol.i.2	188	3,885	13,693	1,498,197	31	37	tl	tl		tl
1-Insertions_4	67	232	1,979	80,730	4	5	tl	tl		1,055
2-Insertions_3	37	72	594	13,020	4	4	5.1	0		0
3-Insertions_3	56	110	1,430	49,500	4	4	503.5	8		1
4-Insertions_3	79	156	2,925	146,146	3	4	tl	836		1,615
DSJC125.1	125	736	7,014	545,250	4	6	tl	214		0
DSJC125.5	125	3,891	3,859	197,166	14	22	tl	tl		tl
DSJC125.9	125	6,961	789	9,503	44	44	0.7	tl		tl
DSJC250.1	250	3,218	27,907	4,350,878		250	tl	tl		tl
DSJC250.5	250	15,668	15,457	1,588,249		250	tl	tl		tl
DSJC250.9	250	27,897	3,228	79,902	71	73	tl	tl		tl
myciel4	23	71	182	2,051	5	5	0.4	0	5	0
myciel5	47	236	845	21,810	6	6	2,279.5	149		0
myciel6	95	755	3,710	206,615	5	7	tl	tl		tl
flat300_20_0	300	21,375	23,475	3,021,152		300	tl	tl		tl
kneser7_2	21	105	105	770	6	6	0.3	0	6	0
kneser7_3	35	70	525	10,780	3	3	0.3	0	2	0
kneser9_4	126	315	7,560	611,940	3	3	261.0	0	809	0

Table 7: Results for the ECP, DIMACS instances.

problems as well. The experiments showed that the proposed method performs well for the Equitable Coloring Problem, while it is not competitive with state-of-the-art algorithms for the Bin Packing Problem with Conflicts. Most of the best performing algorithms for Graph Coloring problems are based on Branch-and-Price or Branch-and-Cut frameworks, which are generally tough tasks to implement and tune. Our approach is instead based on the idea of generating an effective polynomial size Mixed-Integer Programming formulation and then solving it by using a general purpose solver as a black box, exploiting in this manner the effectiveness of modern Mixed-Integer Programming solvers. The continuous development and improvement of these solvers would automatically reflects on the performances of the presented method.

[1] L. Bahiense, Y. Frota, T. F. Noronha, and C. C. Ribeiro. A branch-and-cut algorithm for the equitable coloring problem using a formulation by representatives. *Discrete Appl Math*, 164:34–46, 2014.

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

[3] M. Campêlo, V.A. Campos, and R.C. Corrêa. On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Appl Math*, 156(7):1097–1111, 2008.

[4] F.C. Chow and J.L. Hennessy. The priority-based coloring approach to register allocation. *ACM T Prog Lang Sys*, 12(4):501–536, 1990.

[5] D. Cornaz and V. Jost. A one-to-one correspondence between colorings and stable sets. *Oper Res Lett*, 36(6):673 – 676, 2008.

[6] D. de Werra. An introduction to timetabling. *Eur J Oper Res*, 19:151–162, 1985.

[7] S. Elhedhli, L. Li, M. Gzara, and J. Naoum-Sawaya. A branch-and-price algorithm for the bin packing problem with conflicts. *INFORMS J Comput*, 23(3):404–415, 2011.

[8] E. Falkenauer. A hybrid grouping genetic algorithm. *J Heuristics*, 2:5–30, 1996.

[9] A. E. Fernandes Muritiba, M. Iori, E. Malaguti, and P. Toth. Algorithms for the bin packing problem with conflicts. *INFORMS J Comput*, 22(3):401–415, 2010.

[10] F. Furini and E. Malaguti. Exact weighted vertex coloring via branch-and-price. *Discrete Optim*, 9(2):130–136, 2012.

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

[12] S. Gualandi and F. Malucelli. Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS J Comput*, 24(1):81–100, 2012.

[13] S. Held, W. Cook, and E.C. Sewell. Maximum-weight stable sets and safe lower bounds for graph coloring. *Math Program Comput*, 4:363–381, 2012.

[14] F.T. Leighton. A graph coloring algorithm for large sceduling problems. *J Res Nat Bur Stand*, 84(6):489–503, 1979.

[15] E. Malaguti, M. Monaci, and P. Toth. Models and heuristic algorithms for a weighted vertex coloring problem. *J Heuristics*, 15:503–526, 2009.

[16] E. Malaguti, M. Monaci, and P. Toth. An exact approach for the vertex coloring problem. *Discrete Optim*, 8:174–190, 2011.

[17] E. Malaguti and P. Toth. A survey on vertex coloring problems. *Int T Oper Res*, 17:1–34, 2010.

[18] I. Méndez-Díaz, G. Nasini, and D. Severín. An exact dsatur-based algorithm for the equitable coloring problem. *arXiv preprint arXiv:1405.7011*, 2014.

[19] I. Méndez-Díaz, G. Nasini, and D. Severín. A polyhedral approach for the equitable coloring problem. *Discrete Appl Math*, 164:413–426, 2014.

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

[21] P. Ostergård. A fast algorithm for the maximum clique problem. *Discrete Appl Math*, 120:197 – 207, 2002.

[22] R. Sadykov and F. Vanderbeck. Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS J Comput*, 25(2):244–255, 2013.

[23] T.K. Woo, S.Y.W. Su, and R. Newman Wolfe. Resource allocation in a dynamically partitionable bus network using a graph coloring algorithm. *IEEE T Commun*, 39(12):1794–1801, 2002.