A branch-and-price algorithm for the Minimum Sum Coloring Problem

A Branch-and-Price Algorithm for the Minimum Sum Coloring Problem

Diego Delle Donne[1], Fabio Furini[2], Enrico Malaguti[3] and Roberto Wolfler Calvo[4]

[1] *LIX CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau, France.*
delledonne@lix.polytechnique.fr

[2] *Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" – Consiglio Nazionale delle Ricerche (IASI-CNR), Roma, Italy.*
fabio.furini@iasi.cnr.it

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

[4] *LIPN, Université Paris 13, CNRS UMR 7030, F-93430, Villetaneuse, France.*
wolfler@lipn.univ-paris13.fr

Last update: September 7, 2020

Abstract

A proper coloring of a given graph is an assignment of a positive integer number (color) to each vertex such that two adjacent vertices receive different colors. This paper studies the Minimum Sum Coloring Problem (MSCP), which asks for finding a proper coloring while minimizing the sum of the colors assigned to the vertices. We propose the first branch-and-price algorithm to solve the MSCP to proven optimality. The newly developed exact approach is based on an Integer Programming (IP) formulation with an exponential number of variables which is tackled by column generation. We present extensive computational experiments, on synthetic and benchmark DIMACS graphs from the literature, to compare the performance of our newly developed branch-and-price algorithm against three compact IP formulations. On synthetic graphs, our algorithm outperforms the compact formulations in terms of: (*i*) number of solved instances, (*ii*) running times and (*iii*) exit gaps obtained when optimality is not achieved. For the DIMACS instances, our algorithm is competitive with the best compact formulation and provides very strong dual bounds.

keywords: Minimum Sum Coloring, Vertex Coloring, Integer Linear Programming, Column Generation, Branch-and-Price Algorithm.

1. Introduction

Let $G = (V, E)$ be a simple undirected graph with $n = |V|$ vertices and $m = |E|$ edges, a proper *coloring* C (or simply a coloring) of G is a partition $\{V_1, \dots, V_k\}$ of V into k *stable sets*, i.e., subsets of pairwise non-adjacent vertices. All vertices belonging to V_i are colored with color i ($i \in \{1, \dots, k\}$), i.e., color i is denoted by the integer number i. The *sum of the colors* of a coloring C is given by the function

$$f(C) = \sum_{i=1}^{k} i \cdot |V_i|.$$

The *Minimum Sum Coloring Problem* (MSCP) consists in finding a coloring C of G with the minimum value of the function $f(C)$. This minimum value is denoted by $\Sigma(G)$ and is called the *chromatic sum* of G (see [18]). The smallest number of colors (or equivalently stable sets) associated with $\Sigma(G)$ is called the *strength* of the graph and it is denoted by $s(G)$.

The MSCP models relevant applications in several areas including VLSI design [29], scheduling problems [10, 17] and resource allocation [1]. For instance, the MSCP models scheduling of jobs incompatibilities (see [10]). The incompatibilities can be represented by a graph where the vertices are the jobs and the edges represent the conflicts which forbid scheduling jobs at the same time (e.g., if the jobs requires the same non-sharable resource). Assuming that jobs have unitary execution time, a schedule of the jobs corresponds to a coloring of the graph where the integer numbers representing the colors are the completion times of the jobs. In this context, the MSCP corresponds to the minimization of the average completion time of the jobs.

The MSCP is $\mathcal{NP}$-Hard (see [18]) and related to the classical *Vertex Coloring Problem* (VCP). The VCP asks for a coloring of a graph G with the minimum number $\chi(G)$ of colors, the *chromatic number* of G. We recall in what follows the main features of MSCP optimal solutions which help in understanding the peculiarities of the problem with respect to other coloring problems.

Observation 1 *Given a graph G, the strength $s(G)$ can be greater than the chromatic number $\chi(G)$.*

An example graph where Observation 1 applies is depicted in Figure 1, where two colorings are illustrated. The integer numbers on the vertices encode the assigned colors. The coloring C_1 (on the left part of the figure) uses $\chi(G) = 2$ colors with $f(C_1) = 12$, while the coloring C_2 (on the right part of the figure) is characterized by $f(C_2) = 11$ and it uses an extra color. For this example graph, the strength $s(G)$ is equal to 3. This example can be extended by using t pendant vertices on each side of the central edge (instead of 3) thus obtaining $f(C_1) = 3t + 3$ and $f(C_2) = 2t + 5$. With this configuration, the *loss* incurred by using only $\chi(G)$ colors grows linearly with the size of the graph.

The following observation states that the stable sets are ordered by non-increasing cardinality in any optimal MSCP solution.

Observation 2 *If $C = \{V_1, \dots, V_k\}$ is an optimal sum coloring of a graph G, then $|V_i| \geq |V_j|$ for $i, j \in \{1, \dots, k\}$ and $j > i$.*

We denote by $G[W]$ the subgraph of G induced by a vertex subset $W \subseteq V$, i.e., $G[W] = (W, E_W)$, where $E_W = \{uv \in E : u, v \in W\}$. The following observation holds:

Observation 3 *If $C = \{V_1, \dots, V_k\}$ is an optimal sum coloring of a graph G, then for $j \in \{1, \dots, k\}$, V_j is a maximal stable set in the subgraph $G[\cup_{i=j}^{k} V_i]$.*

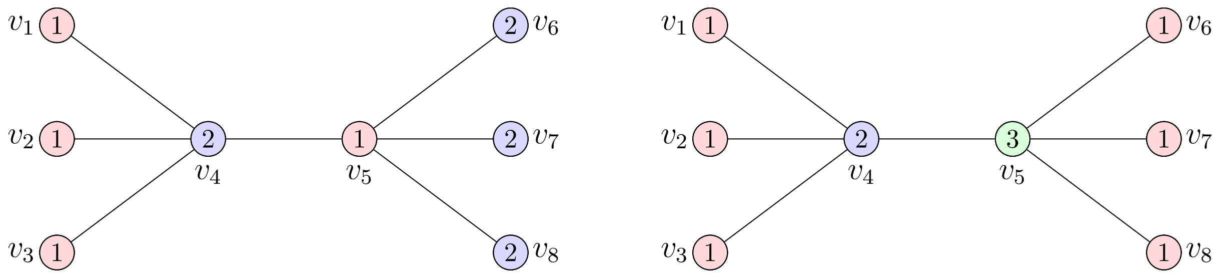

Figure 1: A coloring of a graph G using $\chi(G) = 2$ colors (left) and an optimal MSCP coloring of the same graph with $s(G) = 3$ colors and $\Sigma(G) = 11$(right).

Given a optimal sum coloring C, Observation 3 states that each stable set V_j ($j = \{1, \ldots, k\}$) is a maximal stable set in the subgraph $G[\cup_{i=j}^{k} V_i]$, but it is not necessarily a maximum cardinality stable set of this subgraph. An example is given in Figure 2 where two colorings of a graph are illustrated; the coloring C_1 (on the left part of the figure) uses maximum cardinality stable sets (i.e., each V_j is of maximum cardinality in $G[\cup_{i=j}^{k} V_i]$) and has $f(C_1) = 19$, while the coloring C_2 (on the right part of the figure) is characterized by $f(C_2) = 18$ and it uses maximal, but non-maximum, cardinality stable sets. As in Figure 1, the integer numbers on the vertices represent the assigned colors.

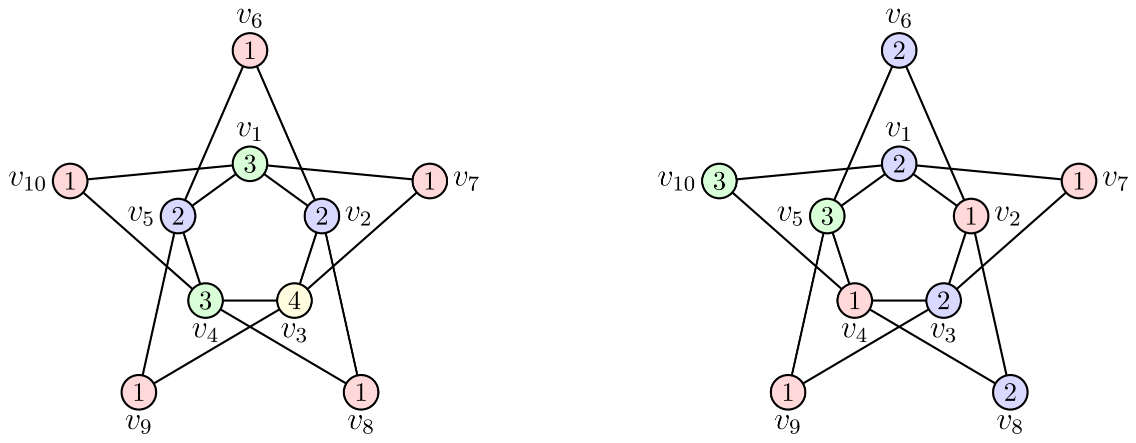

Figure 2: A coloring of a graph using maximum cardinality stable sets (left) and an optimal MSCP solution of the same graph using non-maximum cardinality stable sets (right).

1.1 Literature review

We address the interested reader to [14] for a complete survey on MSCP topics and we briefly mention in this section some of the main results and algorithms. The MSCP has been originally introduced in [18]. Several articles proposed lower bounds for both $\Sigma(G)$ and $s(G)$. In [30] it is shown that $\lceil \sqrt{8m} \rceil \leq \Sigma(G) \leq \lfloor \frac{3}{2}(m+1) \rfloor$ holds for any graph G. Later, [25] proved that $s(G) \leq \Delta(G)+1$, where $\Delta(G)$ is the maximum degree of the vertices of G, while

[9] proved that $s(G) \leq \lceil \frac{\Delta(G)+col(G)}{2} \rceil$ where $col(G)$ is an invariant based on linear orderings of the vertices. Based on this bound, the authors also conjecture that $s(G) \leq \lceil \frac{\Delta(G)+\chi(G)}{2} \rceil$ holds. In [28] the MSCP and its variants are studied, and the authors proved complexity results for specific classes of graphs, while [26] proposed a range of lower bounds for the MSCP based on clique decomposition; for every clique partition $\{K_1, \ldots, K_t\}$ of a graph G, a lower bound on $\Sigma(G)$ is given by $\sum_{j=1}^{t} \frac{|K_j|(|K_j|-1)}{2}$. This bound is valid since each clique K_j must receive at least $|K_j|$ different colors.

Many heuristic algorithms have also been developed over the years to find good upper bounds for the MSCP. In [19] greedy algorithms are developed by adapting the classical VCP greedy algorithms DSATUR and RLF. Local search algorithms have been developed as well in [13]. Later, [33, 34] devised a sophisticated greedy algorithm combined with tabu search that showed particularly effective for large graphs. Different genetic algorithms and tabu search algorithms have been developed in [15, 16].

One of the state-of-the-art algorithms for lower and upper MSCP bounds is called HESA and it has been proposed in [15]; the algorithm is based on a stochastic hybrid evolutionary search. Recently, some new lower bounds have been proposed in [20] which improved some of the results presented in previous works.

To the best of our knowledge, little attention has been given to exact methods for the MSCP. A compact *Integer Programming* (IP) formulation for the MSCP has been firstly proposed in [29]. This formulation was originally proposed to tackle a generalization of MSCP in which each color has a predefined cost. No computational results were reported in [29]. Some preliminary computational results using this compact formulation can be found in [6] where also an exponential-size formulation has been proposed, but no exact solution algorithm for the latter is given. We finally mention that the linear relaxation of the exponential-size formulation proposed in [6] has been tackled via a dual ascent heuristic algorithm in [27].

1.2 Paper contribution

In this paper we propose the first exact algorithm for MSCP. It is based on an exponential-size IP formulation. The linear relaxation of the formulation provides very strong lower bounds, improving on several best known bounds from the literature. By combining column generation with a suited branching strategy, we design the first full branch-and-price algorithm for MSCP.

The remainder of the paper is organized as follows. In Section 2, we review a compact integer programming formulation for MSCP and, in Section 3, we propose two additional compact formulations. Section 4 is devoted to present the exponential-size formulation for MSCP and the corresponding branch-and-price algorithm. Finally, in Section 5 we report on computational experiments performed on a set of VCP benchmark instances and a set of random instances comparing the models and the algorithms addressed in this work.

2. A natural compact IP formulation

In this section, we describe a natural compact IP formulation for the MSCP originally proposed in [29], which is based on the well-known classical formulation for the VCP (see e.g., [24]). This model uses a binary variable x_v^i for each vertex $v \in V$ and each (potential) color $i \in [k] := \{1, \ldots, k\}$, specifying whether the vertex v is assigned to color i or not. The value k represents an upper bound on the strength $s(G)$ of the graph. With this encoding, the MSCP can be formulated by the following compact IP model, which we call IP_C:

$$
\begin{aligned}
[IP_C] \qquad & \min \sum_{i \in [k]} \sum_{v \in V} i \cdot x_v^i && && \text{(1a)}\\
& x_u^i + x_v^i \leq 1 && uv \in E, i \in [k], && \text{(1b)}\\
& \sum_{i \in [k]} x_v^i = 1 && v \in V, && \text{(1c)}\\
& x_v^i \in \{0,1\} && v \in V, i \in [k]. && \text{(1d)}
\end{aligned}
$$

The objective function (1a) minimizes the cost of the coloring. Constraints (1b) ensure that for each edge $uv \in E$ and each color $i \in [k]$, at most one endpoint of the edge receives color i. Constraints (1c) ensure that each vertex is colored with exactly one color.

In the VCP, one of the main drawbacks of the use of variables x_v^i, assigning color i to vertex v, is the strong symmetry presented by the associated polytope, as each coloring can be replaced by an equivalent one by picking a different set of colors to be used and by permuting the order of the colors. For the MSCP however, this symmetry does not hold because the cost of each color is different, and, according to Observation 2, in any optimal solution sets of vertices receiving the same color are ordered by non-increasing cardinality. Equivalent solutions can only be obtained by swapping colors that are used for exactly the same number of vertices.

By replacing the integrality constraints (1d) with $x_v^i \geq 0$, for $v \in V$ and $i \in [k]$, we obtain the linear relaxation of IP_C, denoted as LP_C, whose optimal solution value $Z(LP_C)$ provides a lower bound on the optimal solution value $Z(IP_C)$ of IP_C. The following proposition provides an upper bound for $Z(LP_C)$:

Proposition 1 $Z(LP_C) \leq n + \frac{n}{2}$.

Proof. A feasible LP_C solution $\hat{x}$ of value $n + \frac{n}{2}$ can be obtained as follows: $\hat{x}_v^1 = \hat{x}_v^2 = \frac{1}{2}$ $(v \in V)$ and $\hat{x}_v^i = 0$ $(v \in V, i \geq 3)$. □

Proposition 1 shows that $Z(LP_C)$ cannot be greater than $n + \frac{n}{2}$. However, the optimal solution value $Z(IP_C)$ may raise up to the quadratic term $\frac{n^2-n}{2}$ (e.g., for complete graphs), fact that shows that $Z(LP_C)$ can provide a very weak lower bound. See also Section 5, for a computational evaluation of the quality of this lower bound.

3. Additional compact formulations

In this section, we introduce other two additional compact IP formulations for MSCP, both based on well-known VCP formulations from the literature. In particular, in Section 3.1 we introduce the *orientation* formulation and in Section 3.2, the *representatives* formulation.

3.1 Orientation formulation

The orientation formulation for VCP [2] uses an integer variable x_v for each vertex $v \in V$ to determine the color assigned to v. In addition, a binary variable y_{vw} is used for each edge $vw \in E$ with $v < w$ (assuming a total ordering for V, e.g., $V = \{1, 2, \ldots, n\}$) which takes value 1 if and only if $x_v < x_w$ (i.e., the color assign to v is smaller than the color assigned to w). With this encoding, MSCP can be formulated by the following IP model, which we call IP_O:

$$
\begin{aligned}
[IP_O] \qquad & \min \sum_{v \in V} x_v && && \text{(2a)}\\
& x_u - x_v \geq 1 - k\, y_{uv} && uv \in E, u < v, && \text{(2b)}\\
& x_v - x_u \geq 1 - k\, (1 - y_{uv}) && uv \in E, u < v, && \text{(2c)}\\
& x_v \in [k] && v \in V, && \text{(2d)}\\
& y_{uv} \in \{0, 1\} && uv \in E, u < v. && \text{(2e)}
\end{aligned}
$$

The objective function (2a) minimizes the cost of the colors assigned to each vertex. Constraints (2b) and (2c) impose $|x_u - x_v| \geq 1$ for each edge $uv \in E$. If $y_{uv} = 0$ then constraint (2b) implies $x_u > x_v$ (and constraint (2c) does not impose any condition), whereas if $y_{uv} = 1$ then constraint (2c) implies $x_v > x_u$ (and constraint (2b) does not impose any condition). As in the compact formulation IP_C, this model also presents a strong symmetry in the context of VCP, but as we stated before, this symmetry is not a drawback for MSCP since each color has a different cost.

By replacing constraints (2d) and (2e), with $1 \leq x_v \leq k$, for $v \in V$ and $0 \leq y_{uv} \leq 1$, for each $uv \in E, u < v$, respectively, we obtain the linear relaxation of IP_O, denoted by LP_O. We denote by $Z(LP_O)$ the optimal solution value of IP_O. The following proposition characterizes the value of $Z(LP_O)$, which provides in general a very weak lower bound.

Proposition 2 *For any $k \geq 2$, $Z(LP_O) = n$, where $n = |V|$.*

Proof. Let $(\hat{x}, \hat{y})$ be such that:

$$
\begin{aligned}
\hat{x}_v &= 1, && \forall v \in V,\\
\hat{y}_{uv} &= \frac{1}{2}, && \forall uv \in E, u < v,
\end{aligned}
$$

and all other variables set to 0. For this vector, both right hand sides of constraints (2b) and (2c) become $1 - \frac{k}{2}$ which is never greater than 0, when $k \geq 2$. Then, assigning color 1 for each vertex trivially satisfies constraints (2b) and (2c) for any graph. Since each variable x_v is at its lower bound, we conclude that $\hat{x}$ is an optimal solution, and the objective function on $\hat{x}$ equals n, thus proving the result. □

3.2 The asymmetric representatives formulation

For each vertex $u \in V$, let $\overline{N}(u) := \{v \in V \ : \ uv \notin E\}$ be the *anti-neighborhood* of the vertex. Given an (arbitrary) *total order relation* $\prec$ of the vertices of the graph G, the anti-neighborhood of a vertex $u \in V$ can be partitioned into its *lower* anti-neighborhood $\overline{N}_L(u) := \{v \in \overline{N}(u) : v \prec u\} \cup \{u\}$ and its *upper* anti-neighborhood $\overline{N}_U(u) := \{v \in \overline{N}(u) : u \prec v\} \cup \{u\}$ (with $\overline{N}^*_U(u) := \overline{N}_U(u) \setminus \{u\}$).

In the *asymmetric representatives* formulation, which has been originally introduced for the VCP in [3], a coloring is determined by selecting a representative vertex for each color. This representative vertex is the vertex with the smallest index for the color. Let x^i_{uv} be a binary variable for each color $i \in [k]$ and each pair of vertices $u \in V$ and $v \in \overline{N}_U(u)$, which takes value 1 if and only if vertex u is the *representative* of vertex v for color i. With this encoding, the MSCP can be formulated by the following IP model, which we call IP_R:

$$[IP_R] \qquad \min \sum_{i\in[k]} \sum_{u\in V} \sum_{v\in\overline{N}_U(u)} i \cdot x^i_{uv} \tag{3a}$$

$$\sum_{i\in[k]} \sum_{v\in\overline{N}_L(u)} x^i_{vu} = 1 \qquad u \in V, \tag{3b}$$

$$x^i_{uv} + x^i_{uw} \leq x^i_{uu} \qquad u \in V,\ v, w \in \overline{N}^*_U(u), vw \in E, i \in [k], \tag{3c}$$

$$x^i_{uv} \leq x^i_{uu} \qquad u \in V, v \in \overline{N}^*_U(u), i \in [k], \tag{3d}$$

$$\sum_{u\in V} x^i_{uu} \leq 1 \qquad i \in [k], \tag{3e}$$

$$x^i_{uv} \in \{0, 1\} \qquad i \in [k], u \in V, v \in \overline{N}_U(u). \tag{3f}$$

The objective function (3a) minimizes the cost of the colors. Constraints (3b) ensure that each vertex has exactly one representative vertex (potentially itself). Constraints (3c) avoid that two adjacent vertices have the same representative vertex (i.e., the same color), and prevent that any of these two vertices is represented by a non representative vertex. Constraints (3d) avoid that a vertex in the anti-neighborhood $\overline{N}^*_U(u)$ can be represented by a non representative vertex. Finally, constraints (3e) impose that each color is represented by at most one vertex.

By replacing constraints (3f) with $x^i_{uv} \geq 0$ for $i \in [k], u \in V, v \in \overline{N}(u)$, we obtain the linear relaxation of IP_R, which we denote by LP_R. In line with previous notation, we denote by $Z(LP_R)$ the optimal solution value of LP_R.

4. An exponential-size IP formulation

An exponential-size formulation is a model where variables are in correspondence with an exponential set of elements. In this section, we describe an exponential-size IP formulation for the MSCP, inspired by the Set Covering formulation for VCP introduced by [23]. This model is one of the most effective formulation for the VCP, and was successfully exploited to design the best performing exact algorithms for the problem by [21, 7, 12]. In [11], the authors propose an alternative Set Packing formulation for the VCP, although computational experiments did not show a clear superiority of one model with respect to the other. We address the interested reader to [22] for a discussion on most important models and algorithmic approaches for the VCP and some generalizations.

We denote by $\mathscr{S}_G$ the collection of all stable sets of a graph $\mathscr{S}_G = \{S \subseteq V : uv \notin E$, for all pairs $u, v \in S\}$. Given a vertex $v \in V$, we define $\mathscr{S}_G^v := \{S \in \mathscr{S}_G : v \in S\}$ as the collection of all stable sets containing vertex v. The exponential-size formulation for the MSCP uses a binary variable ξ_S^i for each stable set $S \in \mathscr{S}_G$ and each color $i \in [k] := \{1, \dots, k\}$. The value k has the same meaning as for IP_C and it represents an upper bound on the strength $s(G)$ of the graph. These variables state whether color i is assigned to the stable set S, or not. With this encoding, the MSCP can be formulated by the following exponential-size IP model, which we call IP_E:

$$[IP_E] \qquad \min \sum_{i \in [k]} \sum_{S \in \mathscr{S}_G} c_S^i \cdot \xi_S^i \tag{4a}$$

$$\sum_{S \in \mathscr{S}_G} \xi_S^i \leq 1 \qquad i \in [k], \tag{4b}$$

$$\sum_{i \in [k]} \sum_{S \in \mathscr{S}_G^v} \xi_S^i \geq 1 \qquad v \in V, \tag{4c}$$

$$\xi_S^i \in \{0, 1\} \qquad i \in [k], S \in \mathscr{S}_G, \tag{4d}$$

where $c_S^i = i \cdot |S|$ corresponds to the cost of the stable set $S \in \mathscr{S}_G$. The objective function (4a) minimizes the cost of the coloring. Constraints (4b) prevent from assigning more than one stable set to each color and constraints (4c) ensures that each vertex is colored. Although in any optimal solution constraints (4c) are satisfied by equality (see Observation 4), by defining them as inequalities we get non-negative dual variables for the pricing problem (see Section 4.1). We observe that constraints (4b) can be stated as equalities by allowing the use of the empty set as a valid stable set. This fact is used to develop a complete branching scheme (see Section 4.2). We also point out that IP_E can be seen as a Dantzig-Wolfe reformulation of Constraints (1b) of IP_C (see e.g., [32]). Concerning the relationship of IP_E with the Set Covering formulation for the classical VCP, in the latter problem all the colors are identical and have unitary cost regardless the number of vertices receiving the color. This has two important modeling consequences: first, in the VCP the color index is not relevant, and hence index i and constraint (4b) are dropped from the model. Second, in the VCP it is possible to consider maximal stable sets only, thus obtaining a Set Covering formulation with a much smaller, though exponential, number of variables (from each solution where a vertex is colored more than once, a proper coloring of the same cost can be trivially obtained). In the MSCP instead the number of vertices receiving a color impacts the color cost, and hence, for each color $i \in [k]$, we have to consider all the stables sets of G.

By replacing constraints (4d) with $\xi_S^i \geq 0$, for $i \in [k]$ and $S \in \mathscr{S}_G$, we obtain the linear relaxation of IP_E, denoted as LP_E, whose optimal solution value $Z(LP_E)$ gives a lower bound on the optimal solution value $Z(IP_E)$ of IP_E. In the remaining of this section, we make use of the following observation which characterizes the optimal solutions of LP_E:

Observation 4 *If $\hat{\xi}$ is an optimal solution of LP_E, then constraints* (4c) *are satisfied by $\hat{\xi}$ at equality for every $v \in V$.*

The relationship between the lower bounds provided by the linear relaxation of IP_C and IP_E is established by the following proposition.

Proposition 3 *The lower bound provided by LP_E is at least as strong as the lower bound provided by LP_C, i.e., $Z(LP_E) \geq Z(LP_C)$ and there are instances in which the inequality is tight, i.e., $Z(LP_E) > Z(LP_C)$.*

Proof. Given an optimal solution $\hat{\xi}$ of LP_E, we construct the following vector $\hat{x}$:

$$\hat{x}_v^i = \sum_{S \in \mathscr{S}_G^v} \hat{\xi}_S^i, \qquad v \in V, i \in [k].$$

We first prove that $\hat{x}$ is a feasible solution for LP_C. By Proposition 4, constraints (4c) are satisfied at equality by $\hat{\xi}$, thus implying that each $\hat{x}_v^i \in [0, 1]$. For each edge $uv \in E$ and each color $i \in [k]$, we have

$$\hat{x}_u^i + \hat{x}_v^i = \sum_{S \in \mathscr{S}_G^u} \hat{\xi}_S^i + \sum_{S \in \mathscr{S}_G^v} \hat{\xi}_S^i \leq \sum_{S \in \mathscr{S}_G} \hat{\xi}_S^i \leq 1,$$

and the last inequality is given by constraints (4b). Accordingly, $\hat{x}$ satisfies constraints (1b). For each vertex $v \in V$, we have

$$\sum_{i \in [k]} \hat{x}_v^i = \sum_{i \in [k]} \sum_{S \in \mathscr{S}_G^v} \hat{\xi}_S^i = 1,$$

since constraints (4c) are satisfied by equality by $\hat{\xi}$. Accordingly, $\hat{x}$ also satisfies constraints (1c). Therefore, by construction $\hat{x}$ is a feasible solution for LP_C. Finally, as far as the objective function is concerned, we have

$$Z(LP_C) \leq \sum_{i \in [k]} \sum_{v \in V} i \cdot \hat{x}_v^i = \sum_{i \in [k]} \sum_{v \in V} \left(i \cdot \sum_{S \in \mathscr{S}_G^v} \hat{\xi}_S^i \right) = \sum_{i \in [k]} \sum_{S \in \mathscr{S}_G} c_S^i \cdot \hat{\xi}_S^i = Z(LP_E).$$

To conclude the proof, we show an instance where $Z(LP_C) < Z(LP_E)$. Consider a cycle of size 5 where vertices are labeled consecutively through the cycle: v_1, v_2, v_3, v_4 and v_5. By Proposition 1, we have that $Z(LP_C) \leq 7.5$. For this cycle the optimal solution value $Z(LP_E)$ is equal to 9. An optimal LP_E solution of value 9 is $\hat{\xi}^1_{\{v_1,v_3\}} = \hat{\xi}^2_{\{v_2,v_4\}} = \hat{\xi}^3_{\{v_5\}} = 1$ and all the other variables are set to 0. □

Although (potentially) providing a tighter lower bound than the linear relaxation of LP_C, the optimal solution of LP_E can be fractional. An example is given by LP_E for the graph of Figure 2. This instance has a chromatic sum $\Sigma(G)$ equal to 18 which can be obtained with the coloring C_2 presented in the right part of the figure. An optimal integer solution

$\hat{\xi}$ of IP_E is $\hat{\xi}^1_{\{v_2,v_4,v_7,v_9\}} = \hat{\xi}^2_{\{v_1,v_3,v_6,v_8\}} = \hat{\xi}^3_{\{v_5,v_{10}\}} = 1$. However, LP_E has an optimal solution value $Z(LP_E) = \frac{52}{3} = 17.\bar{3}$, which can be obtained with the following fractional solution $\hat{\xi}$:

$$\begin{aligned}
&\text{stable sets for color 1:} \quad && \hat{\xi}^1_{\{v_2,v_4,v_7,v_9\}} = \hat{\xi}^1_{\{v_1,v_3,v_6,v_8\}} = \hat{\xi}^1_{\{v_6,v_7,v_8,v_9,v_{10}\}} = \frac{1}{3},\\
&\text{stable sets for color 2:} \quad && \hat{\xi}^2_{\{v_2,v_5,v_7,v_{10}\}} = \hat{\xi}^2_{\{v_1,v_4,v_6,v_9\}} = \hat{\xi}^2_{\{v_3,v_5,v_8,v_{10}\}} = \frac{1}{3},\\
&\text{stable sets for color 3:} \quad && \hat{\xi}^3_{\{v_2,v_5\}} = \hat{\xi}^3_{\{v_1,v_4\}} = \hat{\xi}^3_{\{v_2,v_4\}} = \hat{\xi}^3_{\{v_3,v_5\}} = \hat{\xi}^3_{\{v_1,v_3\}} = \frac{1}{6}.
\end{aligned}$$

4.1 Solving the linear relaxation of IP_E

Since IP_E has exponentially many variables, *column generation* (CG) techniques are needed to solve its linear relaxation LP_E. The CG procedure starts with a *restricted master problem* (RMP), i.e., LP_E initialized with a subset of variables containing a feasible solution. Then, iteratively, new additional variables are generated until optimality can be proved.

At each CG iteration we are given an optimal primal-dual solution of the RMP, and the *pricing problem* (PP) is solved to check if new variables with negative reduced costs have to be added to the RMP, which is then re-optimized. The procedure is iterated until no reduced cost variable can be added, implying that the current primal solution is optimal for LP_E. We refer the interested reader to [4] for further details on CG.

We describe in what follows how the PP associated to LP_E is derived by reasoning on the separation of the dual constraints. The dual constraints of LP_E are:

$$\sum_{v \in S} \mu_v + \pi_i \le c^i_S \qquad S \in \mathscr{S}_G, i \in [k], \tag{5}$$

where π and μ are the dual variables associated with primal constraints (4b) and (4c), respectively. Since the RMP contains a subset of the variables, its dual contains a subset of the constraints. A primal variable with a negative reduced cost is associated to a violated dual constraint (5). A dual constraint is violated if a stable set $S \in \mathscr{S}_G$ and a color $i \in [k]$ exist such that

$$0 > c^i_S - \left(\sum_{v \in S} \mu_v + \pi_i \right) = i \cdot |S| - \sum_{v \in S} \mu_v - \pi_i.$$

Accordingly, for a given color $i \in [k]$, it is necessary to determine if a stable set $S \in \mathscr{S}_G$ exists such that

$$\sum_{v \in S} \mu_v - i \cdot |S| = \sum_{v \in S} (\mu_v - i) > -\pi_i.$$

Summarizing, the PP corresponds to the series of *Maximum Weight Stable Set* (MWSS) problems one for each color $k \in [k]$. The weight of a vertex $v \in V$ is defined as $\mu_v - i$. A negative reduced cost variable is found whenever the total weight $\sum_{v \in S}(\mu_v - i)$ of a stable set S is greater than $-\pi_i$. Note that when $\mu_v - i \le 0$, the corresponding vertex v can be discarded.

Since the weight of the vertices depends on the color $i \in [k]$, up to k pricing problems may have to be solved at each CG iteration. The following result allows us to reduce their number, thus (potentially) speeding up the computational convergence.

Proposition 4 *Let $\hat{\xi}$ be an optimal solution of the RMP. Let $\bar{\imath}$ be the largest color for which a variable in $\hat{\xi}$ has strictly positive value*

$$\bar{\imath} = \max_{i\in[k]}\{i \mid \exists S \in \mathscr{S}_G,\ \hat{\xi}^i_S > 0\}.$$

If no negative reduced cost variable exists for a color $j > \bar{\imath}$, no negative reduced cost variable exists for all colors $r > j$.

Proof. By complementary slackness, the dual solution associated with $\hat{\xi}$ has $\pi_j = 0$, for every $j \in \{\bar{\imath}+1, \dots, k\}$. If no negative reduced cost variable exists for a color $j > \bar{\imath}$, then $\sum_{v\in S}(\mu_v - j) \le -\pi_j = 0$ for every $S \in \mathscr{S}_G$. Therefore, $\sum_{v\in S}(\mu_v - r) < 0$ for every $r > j$, since the weights of the vertices decrease increasing the value of the color. $\square$

Thanks to Proposition 4, the MWSS problems can be solved in order of colors from 1 to k, stopping each iteration of the CG procedure as soon as the condition of Proposition 4 is met. In our implementation, we resort to the combinatorial branch-and-bound algorithm for MWSS problem proposed by Held, Cook and Sewell [12]. This algorithm is one the best performing algorithm for the MWSS problem which has been designed to solve the pricing problems for a branch-and-price algorithm to solve the classical vertex coloring problem. We refer to [12] for further details on this algorithm.

4.2 A robust branching rule for IP_E

We embed the CG procedure within a branching scheme, thus obtaining a *branch-and-price algorithm*. The branching step should be handled with care as the addition of arbitrary branching constraints can destroy the structure of the PP. A *robust branching rule* is a rule that preserves the PP, which is the MWSS problem for IP_E. In order to obtain a robust branching, we branch by means of the rule originally proposed by [35], which was used by [23] for the VCP, and can be extended to other variants of the VCP as well, as discussed in [8]. We first introduce a Lemma which will be helpful in the remaining of this section.

Lemma 1 *Let $\hat{\xi}$ be an optimal fractional solution of LP_E. If $\sum_{i\in[k]}\hat{\xi}^i_S \in \{0,1\}$ for every $S \in \mathscr{S}_G \setminus \{\emptyset\}$, then:*

i) there exists an integer solution $\bar{\xi}$ of LP_E with the same cost;

ii) $\hat{\xi}$ is not an extreme point of the polyhedron associated with LP_E.

Proof.

i) When a vertex v belongs to a stable set S for which $\sum_{i\in[k]}\hat{\xi}^i_S = 1$, then v cannot belong to any other stable set (by Observation 4). Then, the collection $T := \{S \in \mathscr{S}_G \mid \sum_{i\in[k]}\hat{\xi}^i_S = 1\}$ defines a partition of V into stable sets. Note that

$$|T| = \sum_{S\in\mathscr{S}_G}\sum_{i\in[k]}\hat{\xi}^i_S \le k,$$

where the inequality is given by the sum of constraints (4b) over every color $i \in [k]$. Also note that $\hat{\xi}$ uses and saturates, by optimality, colors from 1 to $|T|$. Let $S_1, \ldots, S_{|T|}$ be a total ordering of T by non-increasing cardinality, i.e., $|S_i| \geq |S_{i+1}|$ for every $i \in \{1, \ldots, |T|-1\}$. From this ordering, we define the integer solution $\bar{\xi}$ by setting $\bar{\xi}^i_{S_i} = 1$ for every $i \in \{1, \ldots, |T|\}$. We claim that $\bar{\xi}$ has the same cost as $\hat{\xi}$, hence being an equivalent optimal solution for LP_E. Indeed, $\bar{\xi}$ uses the stable sets of $\hat{\xi}$ with an optimal ordering, according to Observation 2.

ii) We have seen $\hat{\xi}$ uses and saturates, by optimality, colors from 1 to $|T|$. By hypothesis, each stable set S not saturating a single color takes at least two distinct colors. Similarly, each color i that is not assigned entirely to a single stable set S, is used for at least two distinct stable sets. Hence, we can define a sequence of pairs (S_1, i_1), (S_2, i_1), (S_2, i_2), $\ldots$, (S_l, i_l), (S_{l+1}, i_l), (S_{l+1}, i_{l+1}), $\ldots$, (S_t, i_t), (S_{t+1}, i_t), with $t \leq T$, such that: $S_{t+1} = S_1$, $0 < \hat{\xi}^{i_l}_{S_l} < 1$, $l = 1, \ldots, t$, and each stable set S_l and each color i_l appears twice in the sequence, so the sequence contains an even number of pairs. With this sequence we can define two feasible solutions $\dot{\xi}$ and $\ddot{\xi}$ as:

- $\dot{\xi} := \hat{\xi}$; $\dot{\xi}^{i_l}_{S_l} := \hat{\xi}^{i_l}_{S_l} + \epsilon,\ l = 1, \ldots, t,\ l$ odd; $\dot{\xi}^{i_l}_{S_l} := \hat{\xi}^{i_l}_{S_l} - \epsilon,\ l = 1, \ldots, t,\ l$ even;
- $\ddot{\xi} := \hat{\xi}$; $\ddot{\xi}^{i_l}_{S_l} := \hat{\xi}^{i_l}_{S_l} - \epsilon,\ l = 1, \ldots, t,\ l$ odd; $\ddot{\xi}^{i_l}_{S_l} := \hat{\xi}^{i_l}_{S_l} + \epsilon,\ l = 1, \ldots, t,\ l$ even;

and we have $\hat{\xi} = \frac{1}{2}\dot{\xi} + \frac{1}{2}\ddot{\xi}$. □

The branching rule we propose is based on the following Proposition, in which the notation $S_1 \triangle S_2$ represents the *symmetric difference* $(S_1 \cup S_2) \setminus (S_1 \cap S_2)$ between two sets S_1 and S_2.

Proposition 5 *Let $\hat{\xi}$ be an optimal fractional solution for LP_E. Either an optimal integer solution can be constructed from $\hat{\xi}$, or there exist two non-adjacent vertices u and v and two stable sets S_1 and S_2, with $u \in S_1 \cap S_2$ and $v \in S_1 \triangle S_2$, such that $0 < \sum_{i \in [k]} \hat{\xi}^i_{S_1} < 1$ and $0 < \sum_{i \in [k]} \hat{\xi}^i_{S_2} < 1$.*

Proof. Assume the conditions of Lemma 1 do not apply. Hence, there is a stable set S_1 such that $0 < \sum_{i \in [k]} \hat{\xi}^i_{S_1} < 1$. Consider a vertex $u \in S_1$. In any optimal solution, constraint (4c) is satisfied by equality for u, hence there must be a stable set S_2 with $u \in S_2$ and $0 < \sum_{i \in [k]} \hat{\xi}^i_{S_2} < 1$. Since $S_1 \neq S_2$, we have $S_1 \triangle S_2 \neq \emptyset$. □

Consider a fractional optimal solution $\hat{\xi}$ and two vertices u and v identified with the characteristics of Proposition 5. Two branching nodes can be created as follows:

- In the first child node, vertices u and v are forced to be assigned to the same color. To this end, all variables $\hat{\xi}^i_S$ such that $v \in S, u \notin S$ or $v \notin S, u \in S$, are set to 0 for all $i \in [k]$. When solving the PP, vertices u and v are *merged*, i.e., they are removed from G and a new vertex w is added to G with neighbourhood $N_G(w) = N_G(u) \cup N_G(v)$. The weight of vertex w in the MWSS problems is defined as the sum of the weights of u and v.

- In the second child node, vertices u and v are forced to be assigned to different colors. To this end, all variables $\hat{\xi}_S^i$ such that $v \in S, u \in S$ are set to 0 for all $i \in [k]$. In the PP, we then add an edge uv to graph.

Whenever the optimal solution of the RMP is fractional, Proposition 5 together with Lemma 1 ensure that either an equivalent integer solution can be recovered from it or a pair of vertices for branching exists, thus the proposed branching scheme is complete.

In line with the procedure proposed in [23], we find a specific pair of vertices for branching in the following way. Let $\hat{\xi}$ be the current fractional solution of the RMP. We select a stable set S_1 such that $0 < \phi(S_1) = \sum_{i\in[k]} \hat{\xi}_{S_1}^i < 1$ and having the $\phi(S_1)$ value closest to 0.5. For each vertex $u \in S_1$, there exists another stable set S_2 containing u and having a fractional value $\phi(S_2)$. We select as u the first vertex in S_1. We then select as S_2 the stable set containing u and having value $\phi(S_2)$ value closest to 0.5 (randomly breaking ties). Finally, we select the first vertex $v \in S_1 \triangle S_2$, thus defining the pair of vertices u and v for branching.

We show an example of the branching operation cutting a fractional optimal solution of the RMP. We consider the graph of Figure 2 and the fractional LP_E solution presented in Section 4. We select as S_1 the stable set $\{v_1, v_3, v_6, v_8\}$ (with $\phi(S_1) = \frac{1}{3}$). We then select its first vertex v_1 and we determine the second stable set $S_2 = \{v_1, v_4, v_6, v_9\}$ (containing vertex v_1 and with $\phi(S_2) = \frac{1}{3}$). Finally we select vertex v_3, thus defining $\{v_1, v_3\}$ as branching pair. In Figure 3, we depict the two subproblem graphs obtained after branching on the vertices v_1 and v_3 (represented in the figure by vertices u and v). In the left part of the figure, the new edge uv (dashed line) prevents these vertices from taking the same color. In the right part of the figure instead, the two vertices are merged into a new vertex w which is connected to all the neighbours of u and v (dashed edges). This forces the two vertices to receive the same color.

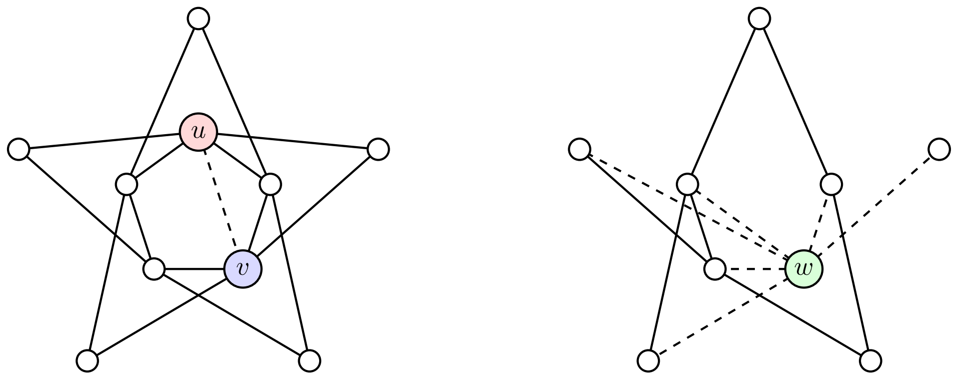

Figure 3: The two subproblem graphs obtained after branching.

As a final remark, we note that Lemma 1 shows that optimal fractional solutions satisfying the lemma hypothesis are not *basic solutions* of LP_E (i.e., they are convex combinations of other equivalent solutions). Therefore, they do not appear within the CG process if the solution of the RMP is performed by the Simplex algorithm. Indeed, given a subset of columns in the RMP, the Simplex algorithm would return an extreme point of the associated polytope. In case a newly generated column enters the basis, this can be at value 0 (degenerate solution) or strictly larger than 0. In both cases, this is the result of a pivot move in the Simplex, thus producing a new extreme point in the updated RMP. Nevertheless, we also remark that an implementation not using the Simplex algorithm is also possible; if such a solution is found within the process, Lemma 1 allows to obtain an equivalent integer solution (as in the proof of the lemma). In our implementation of the branch-and-price algorithm, we solve the RMP by the primal Simplex method, so we are not required to deal with this issue.

5. Computational results

In this section, we asses the computational performance of the three compact IP formulations presented in Sections 2 and 3, i.e., IP_C, IP_O and IP_R, in terms of computational time and number of solved instances. Moreover, we discuss the characteristics of the different compact formulations having a greater impact on the performance. Then, we compare the branch-and-price algorithm based on formulation IP_E, which is denoted as BP in the remainder of this section, against the direct use of state-of-the-art IP solvers on the best compact formulation. The goal of these experiments is also to evaluate the size of the instances that can be solved to proven optimality by the considered methods, and to measure the influence of the edge density on the computational effort. Finally, we are interested in the computational evaluation of the dual bounds provided by the four different formulations presented in this manuscript, beyond the theoretical dominance discussed in the previous sections. In our experiments, we use IBM `CPLEX` 12.8 (denoted for brevity as `CPLEX`) and `SCIP` 6.0.1 with `CPLEX` as linear programming solver (denoted as `SCIP`).

The remainder of this section is organized as follows. In Section 5.1, we describe the general settings of the experiments and the classes of tested instances, i.e., synthetic and `DIMACS` graphs. In Section 5.2, we present a computational comparison of the compact formulations on the `DIMACS` graphs followed by a discussion of the impact of the main features of `CPLEX` on the number of solved instances. In Section 5.3, we discuss the implementation details of BP, i.e., the newly developed exact algorithm. In Section 5.4, we computationally compare BP against the best compact formulation, using the synthetic class of graphs. Finally, in Section 5.5, we discuss the quality of the LP relaxation of the different IP formulations on the `DIMACS` graphs and we compare BP against the best compact formulation, always on the `DIMACS` graphs.

5.1 Experimental settings and instances

Our experimental environment is a desktop PC running Linux with an Intel Core i7-3770 processor at 3.40GHz and 8GB RAM. For all formulations, the upper bound on the strength of the graph is set to $k = \Delta(G) + 1$.

The BP algorithm was implemented in `C++` by using the API provided by `SCIP` to manage the main branch-and-price components, and using `CPLEX` to solve the linear relaxations. We resort to several dedicated data structures to efficiently handle different elements of the algorithm. In particular, we implemented a dedicated data structure, which we call *Merged Graph*, to handle the branching decisions (i.e., the merging of the vertices and addition of the edges to perform the branching operations). This structure allows us to extract the modified graph on each node of the branching tree while keeping track of the *mapping* between the vertices of the original and the merged one.

We consider two sets of instances; namely, the classical DIMACS instances commonly used to test the performance of coloring algorithms (see [31]) and the Erdös-Rényi graphs defined according to a specified edge probability. For the DIMACS instances, we restrict our computational experimentation to the instances from [31] with up to 200 vertices (52 graphs in total). The *random instances* test set is composed by Erdös-Rényi graphs of $n = |V|$ vertices, with $n \in \{20, 30, 40, 50, 60, 70, 80, 90, 100\}$, and edge densities $\delta \in \{10\%, 30\%, 50\%, 70\%, 90\%\}$. For each combination of n and δ, we generated 5 instances, for a total test-bed of 225 instances.

5.2 Performance of the three IP compact formulations on the DIMACS instances

In this section, we discuss the computational performance of the IP compact formulation IP_C presented in Section 2 and the two additional IP compact formulations IP_O and IP_R, presented in Section 3. We use for these tests the 52 DIMACS instances. The formulations are solved by using CPLEX with default parameters and we set a time limit of 1800 seconds for each of the tests. Since the three formulations are solved by CPLEX, we call them IP_C^{cpx}, IP_O^{cpx} and IP_R^{cpx}, in the reminder of this section.

The results of these tests are reported in Table 1. The first column reports the name of the instance and the second column corresponds either to the chromatic sum $\Sigma(G)$ or, if none of the three formulations is able to solve an instance, we report the best upper bound from the literature (see [20]). These best upper bounds are marked in column ub^* with an $*$ next to the values. For each formulation, we report the number of variables (columns "vars") and the time taken by CPLEX to solve the corresponding formulation (columns "time"). In case the time limit is reached, we report "t.l." in the table.

Table 1 shows then three gaps which further characterize the performance of the three formulations. The first one (column "gap") is the *exit gap* computed as $100\ (ub - lb)/lb$, where ub and lb are the upper and lower bounds computed by CPLEX for the corresponding formulation within the time limit (0 for the instances solved to proven optimality). The second one (column "gap$_\ell$") is the *LP gap* computed as $100\ (ub^* - Z(LP))/ub^*$, where ub^* is the chromatic sum $\Sigma(G)$ of the instance (or the best upper bound from the literature) and $Z(LP)$ is the value of the LP relaxation of the corresponding formulation. The third one (column "gap$_r$") is the *root node gap*, computed as $100\ (ub^* - lb_r)/ub^*$, where lb_r corresponds to the lower bound computed by CPLEX at the end of the root node of the branching tree. The lower bound lb_r corresponds to the value of the LP relaxation after the addition of several rounds of generic cuts generated by the CPLEX and the preprocessing phase of the IP solver. The formulation IP_R is characterized by a very high number of variables since these DIMACS graphs are relatively sparse. For this reason, CPLEX produces an out-of-memory error for 11 instances. In these cases, we report "*out of memory*" in Table 1. In addition, always for IP_R^{cpx}, CPLEX is not able to compute any lower bound for two instances, i.e., DSJC125.5

and queen_12_12. For this reason, we report "*" in Table 1 for the exit and the root node gaps. CPLEX is instead able to compute the value of the LP relaxations and, for this reason, the table reports the LP gap for these two instances.

Formulation IP_C^{cpx} is the best performing one for this set of instances since it is able to solve to proven optimality 40 out of the 52 DIMACS instances. The second best formulation is IP_R^{cpx} which solves 27 instances. Formulation IP_O^{cpx} is able to solve only 13 instances. It is important to notice that IP_R^{cpx} is characterized by the largest number of variables, in some instances the number of variables is 100 times larger than the number of variables of the other two formulations. On the other hand, IP_O^{cpx} has the smallest number of variables. Considering the entire test bed, IP_C^{cpx} has $\approx 8,000$ variables on average, $IP_O^{cpx} \approx 2,000$ on average, and $IP_R^{cpx} \approx 400,000$ on average. Not only IP_C^{cpx} is able to solve the largest number of instances but also it generally produces the smallest exit gaps. Most of the exits gaps of this formulation are below 20% and the largest one is 112.17%, for the instance DSJC125.5. The exit gaps of IP_O^{cpx} and IP_R^{cpx} are substantially larger. For IP_O^{cpx}, in most of the instances the exit gaps are above 20% and the largest one is 616.67%, for the instance *DSJC125.9*. The exit gaps of IP_R^{cpx} are smaller compared to the ones of IP_O but, as mentioned before, this formulation incurs in out-of-memory errors for 11 instances. It is worth noticing that in one instance, i.e, DSJC125.9, the exit gap of IP_R^{cpx} is the smallest one and it is equal to 0.14%. For this instance, IP_C^{cpx} has an exit gap equal to 67.56% and IP_O^{cpx} has an exit gap equal to 616.67%.

As far as the LP gaps are concerned, formulation IP_C^{cpx} has an average LP gap of 49.6%, IP_O^{cpx} of 66.3% and IP_R^{cpx} of 31.5% (for IP_R^{cpx} we just consider the 41 instances where the exit gap can be computed). These results show that IP_R^{cpx} has the strongest LP relaxation among the three formulations but the high number of variables (and constraints) makes this formulation not computationally competitive. Also the computational time necessary to solve the LP relaxation of IP_R^{cpx} is considerably larger than the time necessary to solve the LP relaxation of IP_O^{cpx} and IP_C^{cpx}, which is in most of the cases negligible. The average time to solve the LP relaxation for IP_C^{cpx} is 0.30 seconds and the maximum ≈ 3 seconds, for IP_O^{cpx} the average is 0.01 seconds and the maximum ≈ 0.05 seconds, for IP_R^{cpx} the average is ≈ 35 seconds and the maximum is over 300 seconds. Formulation IP_O^{cpx} has the weakest LP relaxation and its LP gaps are often larger than 50% and, for some cases, even larger than 90%. Also the LP gaps of IP_C^{cpx} are relatively large but smaller than those of IP_O^{cpx}, but in several instances the LP gaps IP_C^{cpx} are smaller than 20%. Also for IP_C^{cpx} there are instances on which the LP gap is over 90%.

As far as the root node gaps are concerned, the most important thing to notice is that the root node gaps of IP_C^{cpx} are very small. This fact shows that CPLEX is very effective in reducing the LP gaps for this formulation. The tests showed that very good quality upper and lower bound are computed by CPLEX for IP_C^{cpx}, in particular, 14 instances can be solved at the root node. CPLEX is also very effective in computing good root node gaps for IP_O^{cpx}, but this gaps are considerably larger than those of IP_C^{cpx}. The root nodes gaps of IP_R^{cpx} are also small but the computing time is large. In all the 27 instances solved to optimality by IP_R^{cpx}, CPLEX solves them at the root node. Nevertheless, IP_C^{cpx} is characterized by the best compromise between the strength of the LP relaxation and the time necessary to solve the instances to prove optimality.

CPLEX is one of the most powerful general-purpose IP solvers which implements state-

		IP_C^{cpx}					IP_O^{cpx}					IP_R^{cpx}					
Instance	ub^*	vars	time	gap	gap$_\ell$	gap$_r$	vars	time	gap	gap$_\ell$	gap$_r$	vars	time	gap	gap$_\ell$	gap$_r$	time$_\ell$
`1-FullIns_3`	54	360	0.0	0.00	16.7	0.0	130	0.2	0.00	44.4	12.0	4,380	0.2	0.00	8.3	0.0	0.0
`1-FullIns_4`	166	3,069	3.0	0.00	16.0	4.7	686	t.l.	8.63	44.0	13.9	124,674	131.4	0.00	9.6	0.0	9.5
`1-Insertions_4`	119	1,541	2.7	0.00	15.5	8.4	299	t.l.	3.45	43.7	14.7	47,058	58.0	0.00	6.7	0.0	2.3
`2-FullIns_3`	93	832	0.2	0.00	16.1	1.1	253	0.9	0.00	44.1	8.6	18,832	3.0	0.00	11.3	0.0	0.4
`2-Insertions_3`	62	370	0.2	0.00	10.5	6.4	109	0.4	0.00	40.3	9.7	6,310	1.3	0.00	3.2	0.0	0.1
`2-Insertions_4`	249	5,662	9.9	0.00	10.2	6.1	690	t.l.	9.48	40.2	10.2	404,092	t.l.	5.46	6.0	4.4	191.7
`3-FullIns_3`	145	1,600	0.5	0.00	17.2	1.0	426	31.3	0.00	44.8	7.2	57,880	37.1	0.00	14.1	0.0	2.2
`3-Insertions_3`	92	672	0.2	0.00	8.7	6.4	166	4.4	0.00	39.1	8.7	17,832	6.3	0.00	3.8	0.0	0.6
`4-FullIns_3`	205	2,736	0.4	0.00	16.6	0.5	655	182.9	0.00	44.4	5.2	144,336	130.5	0.00	14.4	0.0	12.6
`4-Insertions_3`	127	1,106	0.4	0.00	6.7	4.6	235	8.1	0.00	37.8	6.3	42,056	51.5	0.00	3.1	0.0	2.8
`5-FullIns_3`	280	4,312	1.6	0.00	17.5	1.0	946	t.l.	1.45	45.0	5.2	312,004	1054.1	0.00	15.9	0.0	36.8
`anna`	276	19,734	0.4	0.00	27.9	0.0	1,124	8.8	0.00	50.0	5.6	1,301,014	219.8	0.00	27.2	0.0	316.0
`david`	237	14,355	0.3	0.00	46.6	0.0	899	t.l.	14.38	63.3	25.1	564,630	53.8	0.00	42.3	0.0	40.7
`DSJC125.1`	326 *	3,000	t.l.	20.39	42.5	20.6	861	t.l.	55.76	61.7	30.3	171,336	t.l.	56.10	38.4	19.0	85.6
`DSJC125.5`	1012 *	9,500	t.l.	112.17	81.5	43.6	4,016	t.l.	507.86	87.6	70.9	302,784	t.l.	*	67.1	*	535.8
`DSJC125.9`	2503 *	15,125	t.l.	67.56	92.5	31.0	7,086	t.l.	616.67	95.0	75.8	110,594	t.l.	0.14	52.0	0.1	190.7
`games120`	443	3,240	0.6	0.00	59.4	0.2	1,396	t.l.	5.66	72.9	5.4	178,794	t.l.	1.22	56.3	0.5	60.1
`huck`	243	7,918	0.1	0.00	54.3	0.0	676	t.l.	11.04	69.5	19.8	264,718	19.5	0.00	50.4	0.0	13.1
`jean`	217	5,840	0.1	0.00	47.2	0.0	588	13.6	0.00	63.1	3.7	217,978	16.9	0.00	46.3	0.0	6.1
`miles1000`	1666	22,144	124.4	0.00	88.5	0.6	6,560	t.l.	21.00	92.3	9.3	871,920			out of memory		
`miles1500`	3354	27,264	73.1	0.00	94.3	3.1	10,524	t.l.	22.70	96.2	10.3	651,354			out of memory		
`miles250`	325	4,224	0.4	0.00	41.7	0.3	902	t.l.	3.77	60.6	4.6	259,677	412.3	0.00	40.6	0.0	17.0
`miles500`	705	9,856	24.9	0.00	72.8	0.7	2,468	t.l.	23.04	81.8	14.5	545,622	t.l.	11.37	71.3	1.5	181.8
`miles750`	1173	16,512	65.5	0.00	83.6	1.7	4,354	t.l.	25.46	89.1	7.6	792,447			out of memory		
`mug100_1`	202	500	1.0	0.00	25.7	3.2	266	t.l.	3.01	50.5	6.4	24,420	346.9	0.00	23.5	0.0	1.1
`mug100_25`	202	500	1.8	0.00	25.7	3.1	266	t.l.	4.66	50.5	7.4	24,420	t.l.	0.58	23.5	0.6	1.3
`mug88_1`	178	440	1.3	0.00	25.8	2.7	234	t.l.	2.89	50.6	6.2	18,850	103.3	0.00	23.3	0.0	0.7
`mug88_25`	178	440	1.0	0.00	25.8	3.1	234	t.l.	3.75	50.6	8.2	18,850	68.6	0.00	23.2	0.0	0.9
`mulsol.i.1`	1957	24,034	1.6	0.00	86.4	0.0	4,122	t.l.	23.63	89.9	11.9	1,900,516			out of memory		
`mulsol.i.2`	1191	29,516	2.3	0.00	77.0	0.0	4,073	t.l.	23.14	84.2	19.8	2,179,317			out of memory		
`mulsol.i.3`	1187	29,072	2.4	0.00	77.2	0.0	4,100	t.l.	23.07	84.5	20.3	2,070,432			out of memory		
`mulsol.i.4`	1189	29,415	2.4	0.00	77.1	0.0	4,131	t.l.	23.32	84.4	19.9	2,108,181			out of memory		
`mulsol.i.5`	1160	29,760	2.5	0.00	76.4	0.0	4,159	t.l.	24.09	84.0	20.7	2,146,880			out of memory		
`myciel3`	21	66	0.0	0.00	21.4	9.1	31	0.0	0.00	47.6	19.0	276	0.0	0.00	4.8	0.0	0.0
`myciel4`	45	276	0.2	0.00	23.3	10.2	94	1.3	0.00	48.9	21.0	2,460	0.6	0.00	5.9	0.0	0.0
`myciel5`	93	1,128	1.4	0.00	24.2	13.3	283	t.l.	5.51	49.5	23.7	21,408	9.1	0.00	8.2	0.0	0.4
`myciel6`	189	4,560	15.0	0.00	24.6	15.5	850	t.l.	21.35	49.7	24.3	182,640	t.l.	5.00	10.4	3.3	16.1
`myciel7`	381 *	18,336	t.l.	6.98	24.8	16.7	2,551	t.l.	53.78	49.9	24.7	1,533,696			out of memory		
`queen10_10`	553 *	3,600	t.l.	1.64	72.9	0.5	1,570	t.l.	33.73	81.9	14.0	128,880	t.l.	32.18	60.8	0.5	132.7
`queen11_11`	733 *	4,961	t.l.	4.41	75.2	1.0	2,101	t.l.	36.90	83.5	14.2	221,441	t.l.	27.82	64.4	1.0	413.8
`queen12_12`	943 *	6,336	t.l.	6.30	77.1	0.7	2,740	t.l.	34.54	84.7	12.9	345,136	t.l.	*	67.2	*	1395.4
`queen13_13`	1191 *	8,281	t.l.	5.92	78.7	0.7	3,497	t.l.	42.28	85.8	12.4	540,813			out of memory		
`queen14_14`	1482 *	10,192	t.l.	5.44	80.2	0.8	4,382	t.l.	52.21	86.8	11.8	786,240			out of memory		
`queen5_5`	75	825	0.0	0.00	50.0	0.0	345	221.9	0.00	66.7	20.0	5,445	0.2	0.00	19.7	0.0	0.1
`queen6_6`	138	1,404	293.5	0.00	60.9	8.7	616	t.l.	16.62	73.9	23.2	14,664	121.9	0.00	40.2	0.0	0.6
`queen7_7`	196	2,401	0.6	0.00	62.5	0.0	1,001	t.l.	30.21	75.0	17.9	36,701	21.2	0.00	43.9	0.0	3.7
`queen8_12`	624	6,240	3.7	0.00	76.9	0.0	2,832	t.l.	180.02	84.6	62.1	213,720	1111.3	0.00	63.9	0.0	129.6
`queen8_8`	291 *	1,792	t.l.	1.04	67.0	1.0	792	t.l.	32.00	78.0	15.5	37,856	t.l.	2.08	51.4	1.0	9.3
`queen9_9`	409 *	2,673	t.l.	0.99	70.3	1.0	1,137	t.l.	40.16	80.2	15.4	74,745	t.l.	3.70	56.7	1.0	36.7
`r125.1`	257	1,125	0.0	0.00	27.6	0.0	334	0.9	0.00	51.4	1.9	68,994	9.0	0.00	26.2	0.0	6.5
`r125.1c`	2184	15,625	628.6	0.00	91.4	0.2	7,626	t.l.	534.74	94.3	55.9	46,750	3.0	0.00	1.4	0.0	2.3
`R125.5`	1853 *	12,500	t.l.	5.13	89.9	5.1	3,963	t.l.	40.77	93.3	17.6	403,700	t.l.	25.94	85.4	3.8	168.5

Table 1: Computational comparison between the compact formulations IP_C^{cpx}, IP_O^{cpx} and IP_R^{cpx}.

`CPLEX` configuration	Instances solved
Default parameters	40
No cutting planes $\rightarrow$ `CPX_PARAM_CUTPASS`	37
No presolving $\rightarrow$ `CPX_PARAM_PREPASS`	41
No presolving nor other reductions $\rightarrow$ `CPX_PARAM_PREIND`	37
No probing before branching $\rightarrow$ `CPX_PARAM_PROBE`	41
No cutting planes, presolving, other reductions nor probing	16

Table 2: Effect on the performance of `CPLEX` in solving IP_C^{cpx} of removing four of its main features.

of-the-art techniques to tackle IP formulations. In this section, we analyze which are the main components of the solver which affect the most its performance in solving IP_C^{cpx}. In particular, we performed 4 additional rounds of tests, on the 52 `DIMACS` instances, disabling some of the main features of `CPLEX`: (*i*) we set the parameter `CPX_PARAM_CUTPASS = -1` to disable the generation of cutting planes; (*ii*) we set the parameter `CPX_PARAM_PREPASS = 0` to remove the *presolve* phase of `CPLEX`; (*iii*) we set the parameter `CPX_PARAM_PREIND = 0` to remove the presolve phase and the reductions of `CPLEX`; (*iv*) we set the parameter `CPX_PARAM_PROBE = -1` to disable the *probing* phase of `CPLEX`. Finally, we performed a fifth round of tests disabling altogether these features.

The results of these tests are reported in Table 2, in which we show the number of instances solved to optimality by each of the `CPLEX` configuration. The table shows that these parameters have an impact on the performance of the solver. Removing the cutting planes results in solving 3 instances less, the same as removing the presolving together with the reductions. Surprisingly, removing only the presolving or the probing, one additional instance can be solved. The most important message of this table, is that by removing altogether these techniques, `CPLEX` is only able to solve 16 instances. This means that these techniques have a complementary effect, so removing them separately is not enough to reduce drastically the performance, but once all of them are removed altogether, the performance of the solver is substantially deteriorated. Even if 1 more instance can be solved by removing only the presolving or the probing, we decide to keep the standard configuration for the tests reported in the next sections. This allows us to avoid the *overfitting* of the parameter calibration.

Summarizing, these tests show that IP_C^{cpx} is computationally superior to IP_O^{cpx} and IP_R^{cpx} for these test bed of instances. For this reason we use this formulation as a term of comparison in Section 5.4 and Section 5.5, where we discuss the performance of the newly developed branch-and-price algorithm.

5.3 Branch-and-price implementation details

The initial set of variables of the RMP must contain a feasible solution in order to start the CG procedure, see Section 4.1. The basic initialization defining a stable set for each vertex and each color might not be enough to guarantee a feasible solution since the number k of colors can be smaller than the number n of vertices. In order to overcome this issue, we include an additional *dummy* variable ξ_V^∞ which covers all the vertices of the graph at a cost $c_V^\infty = +\infty$. In addition, this variable is not present in any of the constraints (4b). Clearly, ξ_V^∞ is not active in any LP_E optimal solution, while it makes any RMP feasible. Nevertheless, it is usually preferable to start the CG process with an initial set of variables containing a proper solution. To this end, we apply a greedy heuristic to search for such a set of variables. The greedy algorithm we propose starts by searching for a maximum stable set to be colored with the first color, it removes the vertices from the graph and it repeats this procedure for the next colors. The loop stops when all vertices are colored or k stables set are generated and the corresponding variables are added to the RMP.

In the following paragraph, we discuss how to determine the the number of columns to be added at each column generation (CG) iteration. We recall that up to k different pricing subproblems can be solved, potentially generating one column for each color. However, for the convergence of the CG procedure, it is enough to add a single negative reduced cost column at each iteration. With the aim of (empirically) finding the best configuration between these two extremes, we conducted a computational experimentation where we solve the pricing subproblems for increasing colors (starting from color 1), and varying the maximum number of columns that are generated before re-optimizing the restricted master problem (RMP). Namely, we consider to add up to 1, 3, 5, 7 and 9 columns per iteration (obtaining in this way 5 different configurations of the BP algorithm). To illustrate the relative performance of these configurations, we compare them by means of a performance profile analysis [5]. For each instance and for each configuration, we compute a normalized time ratio τ as the ratio between the computing time of the considered configuration over the minimum computing time for solving the instance to optimality. The instances which cannot be solved by any configuration are considered as "time limit" and accordingly they have an infinite value of τ. In Figure 4, each configuration is represented by a curve denoted by the maximum number of added columns in the legend. The vertical axis reports, for each value of τ, the percentage of the instances for which the corresponding configuration spent at most τ times the computing time of the fastest one. The curves start from the percentage of instances in which the corresponding configuration is the fastest one and, at the right end of the chart, we can read the percentage of instances solved by a specific configuration (within the time limit of 1800 seconds). Computing times below 0.5 seconds are considered as ties. The best performance is graphically represented by the curves in the upper part the figure.

As Figure 4 depicts for the test-bed of 225 random instances, using one column per pricing step does not yield good results; it is profitable to add columns for several colors at each step. However, our tests also show that adding too many columns may also worsen the performance of the algorithm. The best performance is given by the configuration which adds up to 7 columns per iteration; accordingly, we adopt this BP configuration in all subsequent experiments.

5.4 Performance of the branch-and-price algorithm for the random instances

In this section, we compare the performance of the newly developed branch-and-price algorithm against IP_C which is the best IP formulation for the MSCP as discussed in Section 5.2.

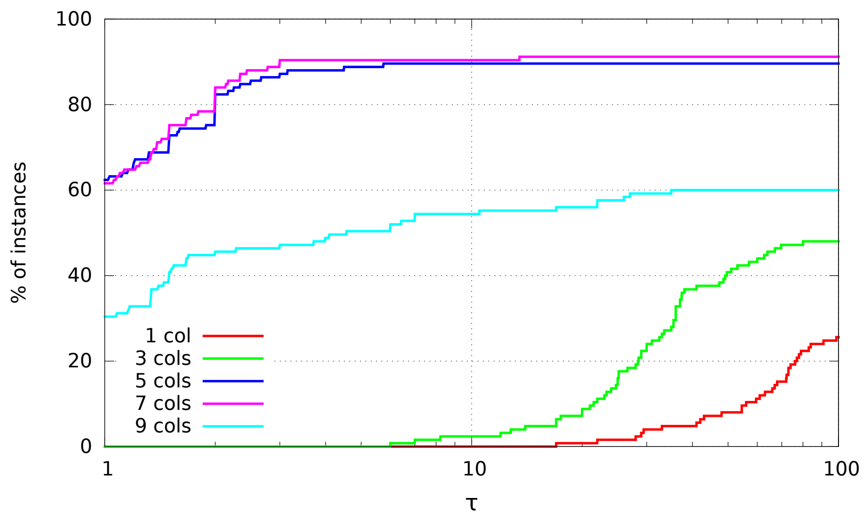

Figure 4: Performance profile of the different configurations of the BP algorithm.

We consider in the reminder of this section the 225 random instances described in Section 5.1. For these tests we set a time limit of 1800 seconds. Since the branch-and-price algorithm is implemented in SCIP we also report the results of this solver to tackle IP_C and we call this configuration IP_C^{scp}.

Figure 5 summarizes the number of instances solved to optimality by the three exact methods considered, i.e., BP, IP_C^{cpx} and IP_C^{scp}, for the complete set of random instances. Grouping the instances by graph size, the vertical bars represent the number of solved instances by the corresponding method. For each value of n, we have a total of 25 instances. Figure 5 shows that for $n = 20$, BP and IP_C^{cpx} are able to solve all the instances within the time limit of 1800 seconds, while IP_C^{scp} solves only 20. Increasing the number of vertices has a strong effect on the performance of IP_C^{scp}, which is able to solve 5 instances for each value $n \in \{50, 60, 70\}$, only 1 instance for $n = 80$ and none of the bigger ones. As far as the performance of IP_C^{cpx} is concerned, this method is able to solve all the instances with $n \in \{20, 30\}$. Its performance starts to decrease with $n = 40$, as for this group of instances IP_C^{cpx} is able to solve 19 of them. Figure 5 shows that IP_C^{cpx} computationally outperforms IP_C^{scp} on this set of instances, since IP_C^{cpx} is able to solve 93 out of the 225 instances while IP_C^{scp} only solves 60. Moreover, IP_C^{cpx} solves 4 instances with $n = 80$ and 1 instance with $n = 90$, while IP_C^{scp} solves none of them. However, IP_C^{cpx} does not solve any instance with $n = 100$. The BP method is able to solve in total 161 instances and it consistently solves all instances with $n \in \{20, 30, 40, 50\}$. For bigger instances, its performance starts decreasing but the method is able to solve 22 instances with $n = 60$, 16 with $n = 70$, 10 with $n = 80$, 8 with $n = 90$ and 5 with $n = 100$. Summarizing, Figure 5 computationally proves that BP has an overall better performance than both IP_C^{cpx} and IP_C^{scp} on the considered random Erdös-Rényi graphs.

Detailed computational results are reported in Table 3 for graphs with 20, 50, 80 and 100 vertices and different edge densities. Each line of the table reports average results for 5 instances with the similar characteristics. The first two columns of the table report the number n of vertices of the graph and the percentage edge density δ. The table is divided

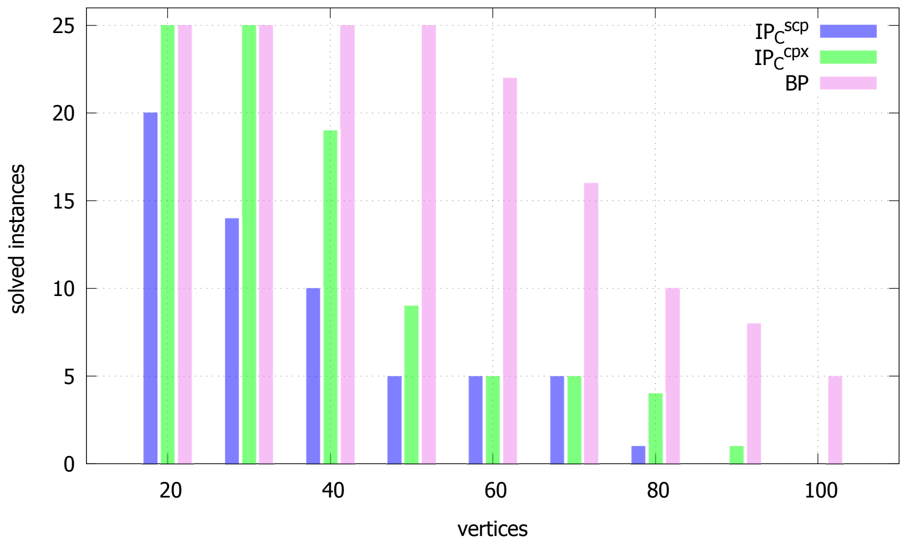

Figure 5: Number of instances solved to optimality by IP_C^{cpx}, IP_C^{scp} and BP for random Erdös-Rényi graphs with $|V| \in \{20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

in three parts which report the results of the three considered methods, i.e, BP, IP_C^{cpx} and IP_C^{scp}. For each of these methods, the table reports the number of instances solved (column *solved*), the average computing time (column *Time*), the average exit gap (column *Gap*) and the average number of nodes explored in the branching tree (column *nodes*). For the BP method, the table also report the average number of generated columns during the branch-and-price algorithm (column *cols*). The exit gap is computed as the percentage difference between the best upper and lower bounds divided by the lower bound when the time limit is reached, an average gap of 0 means that all the instances are solved by the corresponding method. In line with the results showed in Figure 5, Table 3 clearly shows that IP_C^{cpx} outperforms IP_C^{scp} for this tests bed of instances. The method IP_C^{cpx} is able to solve many more instances to proven optimality and on average explores a much smaller number of nodes. In addition, for the instances which cannot be solved within the time limit of 1800 seconds, IP_C^{cpx} guarantees much smaller exit gaps compared to the ones of IP_C^{scp}. The large difference in the computational behavior can be explained by the different and more sophisticated branch-and-cut algorithm implemented by `CPLEX` with respect to the one implemented by `SCIP`. Part of this performance difference is also explained by the better dual bounds computed at the root node by `CPLEX` with respect to the one computed by `SCIP` (see Table 4 for further details on this point). As far as the edge density impact on the performance of IP_C^{cpx} and IP_C^{scp} is concerned, the table points out that both methods tend to suffer when the density increases. For instance, IP_C^{scp} is not able to solve any instance with $n = 20$ and $\delta = 90\%$. The performance of IP_C^{cpx} is also heavily affected by high edge densities. For instance, IP_C^{cpx} is only able to solve 4 out of 5 instances with $n = 80$ and $\delta = 10\%$ while none of the instances with higher edge densities. As far as the comparison between IP_C^{cpx} and BP is concerned, the table clearly shows that BP substantially outperforms IP_C^{cpx} for this test bed of instances. The BP method consistently solves also the instances with up to 50 vertices while IP_C^{cpx} already struggles with instances with $n = 50$ and high edge densities.

		IP_C^{scp}				IP_C^{cpx}				BP				
n	δ	solved	time	gap	nodes	solved	time	gap	nodes	solved	time	gap	nodes	cols
20	10	5	0.00	0.00	1.00	5	0.01	0.00	0.00	5	0.20	0.00	1.80	191.20
	30	5	1.80	0.00	16.40	5	0.04	0.00	3.80	5	0.00	0.00	1.20	163.00
	50	5	14.00	0.00	826.20	5	0.22	0.00	13.60	5	0.20	0.00	1.60	161.60
	70	5	151.20	0.00	12373.20	5	0.15	0.00	0.00	5	0.00	0.00	1.00	162.60
	90	0	1800.00	44.94	132823.20	5	0.02	0.00	0.00	5	0.20	0.00	1.00	135.80
50	10	5	5.00	0.00	102.80	5	0.44	0.00	33.20	5	8.80	0.00	12.60	2042.80
	30	0	1800.00	18.79	60990.60	4	611.98	0.40	123770.00	5	118.80	0.00	145.80	8788.80
	50	0	1800.00	92.15	4416.00	0	1800.00	29.76	117536.00	5	47.00	0.00	96.00	6381.60
	70	0	1800.00	182.05	2906.00	0	1800.00	15.54	48078.20	5	5.60	0.00	25.80	1989.80
	90	0	1800.00	336.65	1914.80	0	1800.00	4.26	56830.40	5	1.40	0.00	3.80	899.20
80	10	1	1712.40	5.06	50477.20	4	669.42	0.40	168339.40	0	1800.00	5.39	171.40	30354.80
	30	0	1800.00	112.28	992.80	0	1800.00	37.65	31406.80	0	1800.00	5.84	237.20	41998.60
	50	0	1800.00	264.24	197.80	0	1800.00	48.70	21266.80	0	1800.00	3.46	342.00	54391.20
	70	0	1800.00	1063.65	22.20	0	1800.00	41.36	14050.80	5	310.00	0.00	226.40	18431.40
	90	0	1800.00	1526.90	1.00	0	1800.00	22.58	9133.00	5	20.00	0.00	22.60	3919.60
100	10	0	1800.00	21.63	18767.00	0	1800.00	10.00	190629.80	0	1800.00	7.67	12.60	4112.40
	30	0	1800.00	172.56	327.40	0	1800.00	55.80	14790.40	0	1800.00	8.35	63.60	16543.40
	50	0	1800.00	836.61	7.20	0	1800.00	81.28	4928.00	0	1800.00	6.99	180.00	43321.80
	70	0	1800.00	1430.98	1.00	0	1800.00	75.11	2038.00	0	1800.00	1.56	361.20	58463.60
	90	0	1800.00	2126.30	1.00	0	1800.00	41.56	1366.20	5	59.20	0.00	52.80	6211.20

Table 3: Computational results of IP_C^{scp}, IP_C^{cpx} and BP on random Erdös-Rényi graphs with $|V| \in \{20, 50, 80, 100\}$ and $\delta \in \{10\%, 30\%, 50\%, 70\%, 90\%\}$.

Also for the unsolved instances, the exit gaps guaranteed by BP are much smalled than those of IP_C^{cpx}.

Table 3 points out a completely different impact of the edge density on BP with respect to the impact on IP_C^{cpx}. The BP method is able to solve all the instances with $\delta = 90\%$ and all different values of n. On the other hand, BP tends to struggle when the edge density is smaller, fact which is also evident by looking at the number of explored nodes. By comparing the average number of nodes explored by BP and IP_C^{cpx} for the instances solved to proven optimality, one can see that BP explores on average a much smaller number of nodes. This behavior can be explained by the better quality of the dual bound provided by LP_E compared to the one provided by LP_C. As far as the number of generated columns is concerned, the table shows that BP generates on average hundreds of columns for $n = 20$ and thousands of columns with $n = 50$ and the number of columns drastically increases for larger instances. The edge density has an impact on the number of generated columns, since instances with high edge density can be solved with a much smaller number of columns on average.

Detailed computational results on the lower and upper bounds provided by BP, IP_C^{cpx} and IP_C^{scp} are shown in Table 4 (considering the same set of instances as in Table 3). This table reports three columns for each method, the average upper bound (column UB), the average lower bound (column LB) and the average lower bound computed at the root node of the branching tree (column LB_r). Clearly, for the instances solved to proven optimality the columns UB and LB coincide, otherwise the bounds are the one obtained by the corresponding method in case of time limit. The column LB_r reports for IP_C^{cpx} and IP_C^{scp} the dual bound computed at the root node, i.e., the value of the linear relaxation of IP_C strengthened by the general purpose valid inequalities generated by the solvers. On the other hand, for the BP method the column LB_r simply reports the bound provided by LP_E.

As far as the comparison between IP_C^{cpx} and IP_C^{scp} is concerned, Table 4 clearly shows that IP_C^{cpx} is able to compute better lower and upper bounds with respect to IP_C^{scp}. The column LB_r partially explains the difference in the computational behavior between IP_C^{cpx} and IP_C^{scp}. By looking at these two columns one can see that much stronger dual bounds are computed by `CPLEX` than those computed by `SCIP`. By comparing instead the root dual bounds provided by BP with those of IP_C^{cpx}, one can see that the linear relaxation bound of LP_E is much stronger than the dual bound provided by `CPLEX` at the root node even after the addition of several families of general purpose valid inequalities. The strength of the root node bound of BP plays a crucial rule on its computational performance and it is the main reason why BP outperforms both IP_C^{cpx} and IP_C^{scp} on this set of randomly generated Erdös-Rényi graphs.

		IP_C^{scp}			IP_C^{cpx}			BP		
$\|V\|$	δ	UB	LB	LB$_r$	UB	LB	LB$_r$	UB	LB	LB$_r$
20	10	29.80	29.80	29.80	29.80	29.80	29.69	29.80	29.80	29.80
	30	41.60	41.60	39.96	41.60	41.60	40.89	41.60	41.60	41.60
	50	57.00	57.00	43.24	57.00	57.00	54.52	57.00	57.00	56.75
	70	73.60	73.60	43.85	73.60	73.60	71.98	73.60	73.60	73.60
	90	117.80	81.29	45.82	117.80	117.80	117.80	117.80	117.80	117.80
50	10	93.40	93.40	88.20	93.40	93.40	90.70	93.40	93.40	92.85
	30	157.40	132.54	101.22	156.00	155.35	130.14	156.00	156.00	152.64
	50	254.60	132.72	103.71	231.80	186.71	179.30	227.00	227.00	223.86
	70	354.00	125.56	105.09	319.60	276.54	262.03	317.00	317.00	314.66
	90	566.20	130.50	105.76	524.20	503.11	484.53	524.20	524.20	523.20
80	10	180.80	172.14	147.18	181.00	180.27	157.31	186.40	176.87	175.26
	30	377.40	177.77	162.06	341.80	248.29	229.96	331.20	312.77	309.38
	50	610.20	167.57	165.42	518.00	348.21	328.24	491.80	475.34	470.59
	70	1927.00	165.90	165.55	713.20	504.61	484.52	686.80	686.80	680.64
	90	2714.00	166.85	166.85	1152.00	940.31	930.82	1125.40	1125.40	1124.70
100	10	255.40	209.97	186.33	249.00	226.37	202.82	255.60	237.41	237.07
	30	565.20	207.41	203.64	495.60	318.07	299.59	475.40	438.77	436.47
	50	1916.80	204.65	204.65	803.00	442.98	430.88	722.60	675.37	670.45
	70	3095.00	202.20	202.20	1137.60	649.73	643.59	1011.00	995.46	989.15
	90	4216.40	189.43	189.43	1815.20	1282.33	1273.19	1676.80	1676.80	1672.17

Table 4: Computational results of IP_C^{scp}, IP_C^{cpx} and BP on random Erdös-Rényi graphs with $|V| \in \{20, 50, 80, 100\}$ and $\delta \in \{10\%, 30\%, 50\%, 70\%, 90\%\}$.

5.5 Performance of the branch-and-price algorithm for the `DIMACS` instances

In this section we analyze the performance of BP on the 52 `DIMACS` instances. Firstly, we compare the bounds provided by the LP relaxation of the three IP compact formulations with the LP relaxation of the extended formulation IP_E.

In Table 5 we report the instance name, the number of vertices and edges (columns $|V|$ and $|E|$) and the best lower bound known in the literature [20] (column LB$_b$). Then the table reports the optimal value of the LP relaxation of IP_C, IP_O, IP_R and IP_E, rounded up to the closest integer value. Additionally, for IP_E we also report the computational time needed to calculate this value or "t.l." if the time limit of 7200 seconds was reached. We report in boldface text the best lower bound for each instance (ties are reported in boldface text as well). The table reports the symbol "-" in case the lower bound cannot be computed within the time limit (i.e., 7200 seconds). Specifically, the bound could not be computed for 10 instances when using formulation IP_R, and for 3 instances, i.e., `queen12_12`, `queen13_13` and `queen14_14` when using IP_E. The table shows that the strongest linear relaxation is by far the one of IP_E, which however requires a column generation approach to be solved (see Section 4.1). For all 52 instances, $Z(LP_E)$ is significantly larger than $Z(LP_C)$, $Z(LP_R)$ and $Z(LP_O)$, in some cases, by more than one order of magnitude. These very strong bounds are often stronger than the best known bounds in the literature. Specifically, $Z(LP_E)$ is strictly larger than every other bound in 25 instances, while LB$_b$ only in 8 cases. The time necessary to compute $Z(LP_E)$ can be high, indeed, in 8 cases the computational time is larger than 1800 seconds. But, on the other hand, in 23 cases this bound could be computed in less than 30 seconds.

We discuss now the results of BP used to solve the `DIMACS` instances to proven optimality. Even if `DIMACS` instances are very challenging for the classical vertex coloring problem, some of these instances may not be such for the MSCP. In Section 5.2, we showed that IP_C^{cpx} is able to solve 33 out of the 52 `DIMACS` instances in less than 10 seconds, thus implying that these instances are not very challenging for MSCP. For this reason, we drop these 33 instances from the following analysis and we focus on the remaining 19 instances which can be classified as the "hard" instances. As far as the performance of BP is concerned on the 33 "easy" instances, it is worth mentioning that IP_C^{cpx} outperforms BP, since it is on average faster and can solve all of them. Nevertheless, BP is able to solve all but 2 of these instances, namely `2-Insertions_4` and `games120`, with an average computing time of of 1123 seconds. For the two unsolved instances the exit gaps are smaller than 1%.

Tables 6 shows the results on the remaining 19 "hard" `DIMACS` instances with an extended *time-limit* of 7200 seconds. The structure of the table is the same as for Table 3. A hyphen on the gap column means either that the formulation could not find a lower bound (i.e., the linear relaxation was not solved within the time-limit) or an upper bound (i.e., no integer solution was found.). IP_C^{cpx} is able to solve 7 of these 19 "hard" instances, while BP solves 8 instances. In 3 of the 11 instances not solved by BP, this formulation could not solve the root node, but on the other 8 instances it achieves an average exit gap of 4.98%. On the other hand, the average exit gap achieved by IP_C^{cpx} on the 12 unsolved instances is 16.24% (some of the instances showing gaps of above 50%).

6. Conclusions

The Minimum Sum Coloring problem is a computationally challenging graph problem with several peculiarities which make it different from other generalizations of the Vertex Coloring problem. The problem has recently attracted much interest in the literature on heuristic and methaheuristic approaches, but, to the best of our knowledge, little attention has been given to exact solution methods. In this paper we have adapted to the Minimum Sum Coloring some compact formulations proposed in the literature for the Vertex Coloring, and we have first identified which compact formulation performs better than the others. Then, we have proposed the first branch-and-price algorithm to solve an exponential-size formulation of the problem, which is the main contribution of this paper. The performance of the developed models and algorithms have been assessed on a large set of instances, both randomly generated and `DIMACS` instances (standard benchmarks in the literature for coloring problems). Our computational results showed the superior performance of the newly developed algorithm, in particular for the solution of randomly generated graphs.

All the tests presented in this paper are performed in single-thread mode. This clean setting allows a fair comparison between the different IP formulations, removing the impact of the parallelization on the performance. As a future line of research, we mention that a parallel version of our branch-and-price algorithm could potentially results in a further

Instance	$\lvert V\rvert$	$\lvert E\rvert$	LB$_b$	Linear relaxation lower bounds $\lceil Z(LP_C)\rceil$	$\lceil Z(LP_R)\rceil$	$\lceil Z(LP_O)\rceil$	$\lceil Z(LP_E)\rceil$	time
1-FullIns_3	30	100	53	45	50	30	**54**	0.1
1-FullIns_4	93	593	161	140	150	93	**166**	21.1
1-Insertions_4	67	232	**117**	101	111	67	116	1.8
2-FullIns_3	52	201	91	78	83	52	**93**	1.3
2-Insertions_3	37	72	**62**	56	60	37	**62**	0.1
2-Insertions_4	149	541	243	224	234	149	**246**	1851.7
3-FullIns_3	80	346	141	120	125	80	**145**	12.2
3-Insertions_3	56	110	**92**	84	89	56	**92**	2.4
4-FullIns_3	114	541	198	171	176	114	**205**	67.9
4-Insertions_3	79	156	126	119	123	79	**127**	11.1
5-FullIns_3	154	792	269	231	236	154	**280**	2099.8
anna	138	493	273	199	201	138	**276**	521.3
david	87	406	234	127	137	87	**237**	12.0
DSJC125.1	125	736	303	188	201	125	**314**	1953.3
DSJC125.5	125	3891	924	188	333	125	**978**	254.7
DSJC125.9	125	6961	2124	188	1202	125	**2500**	18.0
games120	120	638	442	180	194	120	**443**	5482.4
huck	74	301	**243**	111	121	74	**243**	1.6
jean	80	254	216	115	117	80	**217**	4.7
miles1000	128	3216	1623	192	-	128	**1666**	59.2
miles1500	128	5198	3239	192	-	128	**3354**	15.1
miles250	128	387	318	190	193	128	**325**	1116.0
miles500	128	1170	686	192	202	128	**705**	379.0
miles750	128	2113	1145	192	-	128	**1173**	88.3
mug100_1	100	166	**202**	150	155	100	**202**	3310.2
mug100_25	100	166	**202**	150	155	100	**202**	1850.1
mug88_1	88	146	**178**	132	137	88	**178**	180.8
mug88_25	88	146	**178**	132	137	88	**178**	156.9
mulsol.i.1	197	3925	**1957**	266	-	197	**1957**	87.0
mulsol.i.2	188	3885	**1191**	275	-	188	**1191**	1595.0
mulsol.i.3	184	3916	**1187**	271	-	184	**1187**	1547.2
mulsol.i.4	185	3946	**1189**	273	-	185	**1189**	1312.3
mulsol.i.5	186	3973	**1160**	274	-	186	**1160**	1392.8
myciel3	11	20	**21**	17	20	11	**21**	0.1
myciel4	23	71	**44**	35	43	23	**44**	0.1
myciel5	47	236	**89**	71	86	47	88	0.3
myciel6	95	755	**177**	143	170	95	176	3.8
myciel7	191	2360	**350**	287	-	191	349	212.0
queen10_10	100	1470	**551**	150	217	100	550	188.0
queen11_11	121	1980	**726**	182	261	121	**726**	1907.3
queen12_12	144	2596	**936**	216	309	144	-	t.l.
queen13_13	169	3328	**1183**	254	-	169	-	t.l.
queen14_14	196	4186	**1470**	294	-	196	-	t.l.
queen5_5	25	160	**75**	38	61	25	**75**	0.3
queen6_6	36	290	127	54	83	36	**138**	0.1
queen7_7	49	476	**196**	74	110	49	**196**	1.0
queen8_12	96	1368	**624**	144	226	96	**624**	161.0
queen8_8	64	728	289	96	142	64	**291**	3.7
queen9_9	81	1056	**406**	122	177	81	405	24.0
r125.1	125	209	**257**	186	190	125	**257**	2078.4
r125.1c	125	7501	**6597**	188	2154	125	2184	6.8
R125.5	125	3838	**2175**	188	270	125	1820	59.0

Table 5: Comparison of the lower bounds provided by the linear relaxations of the four IP models considered, i.e., IP_C, IP_R, IP_O and IP_E against the best lower bound from the literature [20], i.e., LB_b. We report in boldface the best lower bound for each instance.

				IP_C^{cpx}			BP			
Instance	$\|V\|$	$\|E\|$	δ	time	gap	nodes	time	gap	nodes	cols
DSJC125.1	125	736	9	t.l.	17.49	190914	t.l.	11.46	25	9136
DSJC125.5	125	3891	50	t.l.	97.29	8376	t.l.	7.24	530	11000
DSJC125.9	125	6961	90	t.l.	53.46	2181	25	0.00	29	6281
miles1000	128	3216	40	125	0.00	504	56	0.00	8	7481
miles1500	128	5198	64	73	0.00	530	8	0.00	1	3508
miles500	128	1170	14	25	0.00	1089	667	0.00	22	27798
miles750	128	2113	26	66	0.00	984	268	0.00	47	24130
myciel6	95	755	17	15	0.00	1125	t.l.	4.85	589	87803
myciel7	191	2360	13	t.l.	5.52	28920	t.l.	8.33	148	98797
queen10_10	100	1470	30	t.l.	1.64	275223	t.l.	2.55	711	15000
queen11_11	121	1980	27	t.l.	1.93	137796	t.l.	4.27	128	37989
queen12_12	144	2596	25	t.l.	3.85	128289	t.l.	-	1	13210
queen13_13	169	3328	23	t.l.	2.11	103190	t.l.	-	1	12165
queen14_14	196	4186	22	t.l.	5.31	59147	t.l.	-	1	20025
queen6_6	36	290	46	294	0.00	101801	0	0.00	4	736
queen8_8	64	728	36	t.l.	0.69	482695	3	0.00	4	2206
queen9_9	81	1056	33	t.l.	0.99	369164	t.l.	0.87	1478	87223
r125.1c	125	7501	97	645	0.00	1	6	0.00	1	4118
R125.5	125	3838	50	t.l.	4.56	9710	t.l.	0.25	1391	184299

Table 6: Execution times, achieved exit gaps and nodes in the branching tree for IP_C^{cpx} and BP on hard DIMACS instances, with an extended time limit of 7200.

improvements of its performance.

Acknowledgments

The authors are grateful to Ian-Christopher Ternier and Sébastien Martin for preliminary discussions and computational experiments, and to two anonymous referees for their comments that helped to significantly improve the quality of the paper. Diego Delle Donne was partially supported by the MathSTIC Research Federation and most of this work was carried out while this author was affiliated to LIPN. Fabio Furini was partially supported by Google. Enrico Malaguti was supported by the Air Force Office of Scientific Research under award FA8655-20-1-7019.

References

[1] Amotz Bar-Noy, Mihir Bellare, Magnús M. Halldórsson, Hadas Shachnai, and Tami Tamir. On chromatic sums and distributed resource allocation. *Information and Computation*, 140(2):183 – 202, 1998.

[2] Ralf Borndörfer, Andreas Eisenblätter, Martin Grötschel, and Alexander Martin. The orientation model for frequency assignment problems. Technical Report TR-98-01, ZIB, Takustr. 7, 14195 Berlin, 1998.

[3] Manoel B. Campêlo, Victor A. Campos, and Ricardo C. Corrêa. On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics*, 156(7):1097–1111, 2008.

[4] Jacques Desrosiers and Marco E. Lübbecke. *Column Generation*, chapter A Primer in Column Generation. Springer US, Boston, MA, 2005.

[5] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[6] Fabio Furini, Enrico Malaguti, Sébastien Martin, and Ian-Christopher Ternier. ILP models and column generation for the minimum sum coloring problem. *Electronic Notes in Discrete Mathematics*, 64:215 – 224, 2018.

[7] Stefano Gualandi and Federico Malucelli. Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS Journal on Computing*, 24(1):81–100, 2012.

[8] Stefano Gualandi and Federico Malucelli. A simple branching scheme for vertex coloring problems. *Discrete Applied Mathematics*, 160(1):192 – 196, 2012.

[9] Hossein Hajiabolhassan, Medhi Mehrabadi, and Ruzbeh Tusserkani. Minimal coloring and strength of graphs. *Discrete Mathematics*, 215(1):265 – 270, 2000.

[10] Magnús M. Halldórsson, Guy Kortsarz, and Hadas Shachnai. Sum coloring interval and k-claw free graphs with application to scheduling dependent jobs. *Algorithmica*, 37(3):187–209, Nov 2003.

[11] Pierre Hansen, Martine Labbé, and David Schindl. Set covering and packing formulations of graph coloring: Algorithms and first polyhedral results. *Discrete Optimization*, 6(2):135–147, 2009.

[12] Stephan Held, William J. Cook, and Edward C. Sewell. Maximum-weight stable sets and safe lower bounds for graph coloring. *Math. Program. Comput.*, 4(4):363–381, 2012.

[13] Anders Helmar and Marco Chiarandini. A local search heuristic for chromatic sum. In *Proceedings of the 9th metaheuristics international conference*, volume 1101, pages 161–170, 2011.

[14] Yan Jin, Jean-Philippe Hamiez, and Jin-Kao Hao. Algorithms for the minimum sum coloring problem: a review. *Artificial Intelligence Review*, 47(3):367–394, 2017.

[15] Yan Jin and Jin-Kao Hao. Hybrid evolutionary search for the minimum sum coloring problem of graphs. *Information Sciences*, 352:15–34, 2016.

[16] Yan Jin, Jin-Kao Hao, and Jean-Philippe Hamiez. A memetic algorithm for the minimum sum coloring problem. *Computers & Operations Research*, 43:318–327, 2014.

[17] Leo G. Kroon, Arunabha Sen, Haiyong Deng, and Asim Roy. The optimal cost chromatic partition problem for trees and interval graphs. In Fabrizio d'Amore, Paolo Giulio Franciosa, and Alberto Marchetti-Spaccamela, editors, *Graph-Theoretic Concepts in Computer Science*, pages 279–292, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[18] Ewa Kubicka and Allen J. Schwenk. An introduction to chromatic sums. In *Computer Trends in the 1990s - Proceedings of the 1989 ACM 17th Annual Computer Science Conference, Louisville, Kentucky, USA, February 21-23, 1989*, pages 39–45, 1989.

[19] Yu Li, Corinne Lucet, Aziz Moukrim, and Kaoutar Sghiouer. Greedy Algorithms for the Minimum Sum Coloring Problem. In *Logistique et transports*, pages LT–027, Sousse, Tunisia, March 2009.

[20] Weibo Lin, Mingyu Xiao, Yi Zhou, and Zhenyu Guo. Computing lower bounds for minimum sum coloring and optimum cost chromatic partition. *Computers & Operations Research*, 109:263 – 272, 2019.

[21] Enrico Malaguti, Michele Monaci, and Paolo Toth. An exact approach for the vertex coloring problem. *Discrete Optimization*, 8(2):174–190, 2011.

[22] Enrico Malaguti and Paolo Toth. A survey on vertex coloring problems. *ITOR*, 17(1):1–34, 2010.

[23] Anuj Mehrotra and Michael A. Trick. A column generation approach for graph coloring. *INFORMS Journal on Computing*, 8(4):344–354, 1996.

[24] Isabel Méndez-Díaz and Paula Zabala. A polyhedral approach for graph coloring^1. *Electronic Notes in Discrete Mathematics*, 7:178–181, 2001.

[25] John Mitchem and Patrick Morriss. On the cost-chromatic number of graphs. *Discrete Mathematics*, 171(1-3):201–211, 1997.

[26] Aziz Moukrim, Kaoutar Sghiouer, Corinne Lucet, and Yu Li. Lower bounds for the minimal sum coloring problem. *Electronic Notes in Discrete Mathematics*, 36:663 – 670, 2010. ISCO 2010 - International Symposium on Combinatorial Optimization.

[27] Stefania Pan, Roberto Wolfler Calvo, Mahuna Akplogan, Lucas Létocart, and Nora Touati. A dual ascent heuristic for obtaining a lower bound of the generalized set partitioning problem with convexity constraints. *Discrete Optimization*, 33:146 – 168, 2019.

[28] Mohammad R. Salavatipour. On sum coloring of graphs. *Discrete Applied Mathematics*, 127(3):477–488, 2003.

[29] Arunabha Sen, Haiyong Deng, and Sumanta Guha. On a graph partition problem with application to VLSI layout. *Inf. Process. Lett.*, 43(2):87–94, 1992.

[30] Carsten Thomassen, Paul Erdös, Yousef Alavi, Paresh J. Malde, and Allen J. Schwenk. Tight bounds on the chromatic sum of a connected graph. *Journal of Graph Theory*, 13(3):353–357, 1989.

[31] Michael Trick. Operations research page - vertex coloring instances. `https://mat.tepper.cmu.edu/COLOR/instances.html`. Accessed: June 2019.

[32] François Vanderbeck. On Dantzig-Wolfe Decomposition in Integer Programming and ways to Perform Branching in a Branch-and-Price Algorithm. *Operations Research*, 48(1):111–128, 2000.

[33] Qinghua Wu and Jin-Kao Hao. An effective heuristic algorithm for sum coloring of graphs. *Computers & Operations Research*, 39(7):1593–1600, 2012.

[34] Qinghua Wu and Jin-Kao Hao. Improved lower bounds for sum coloring via clique decomposition. *arXiv preprint arXiv:1303.6761*, 2013.

[35] Alexander Aleksandrovich Zykov. On some properties of linear complexes. *Matematicheskii sbornik*, 66(2):163–188, 1949.