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.*
(Most of this work was carried out while this author was affiliated to LIPN [4])
diegodd@gmail.com

[2] *Istituto di Analisi dei Sistemi ed Informatica "A. 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: January 23, 2020

Abstract

A proper coloring of a given graph is an assignment of colors (integer numbers) to its vertices such that two adjacent vertices receives 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. This paper presents 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. Extensive computational experiments, on synthetic and benchmark graphs from the literature, show that the new algorithm outperforms a natural compact IP formulations in terms of: (*i*) number of solved instances, (*ii*) running times and (*iii*) dual gaps obtained when optimality is not achieved. The new exact method is able to solve to optimality 8 out of the 17 (yet unclosed) hard DIMACS instances tested in this work.

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 $\{V_1, \ldots, V_k\}$ of V into k *stable sets* [1]. All vertices belonging to V_i are colored with color i ($i \in \{1, \ldots, k\}$), i.e., color i is

[1] A stable set is a subsets of pairwise non-adjacent vertices.

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 it is called the *chromatic sum* of G (see [16]). 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 [27], scheduling problems [9, 15] and resource allocation [1]. For instance, the MSCP models scheduling problems with jobs incompatibilities (see [9]). 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 [16]) and it is 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 represent 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 $C = \{V_1, \ldots, V_k\}$ is an optimal sum coloring of a graph G, then $|V_i| \geq |V_j|$ for $i, j \in \{1, \ldots, k\}$ 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 $C = \{V_1, \ldots, V_k\}$ is an optimal sum coloring of a graph G, then for $j \in \{1, \ldots, k\}$, V_j is a maximal stable set in the subgraph $G[\cup_{i=j}^{k} V_i]$.*

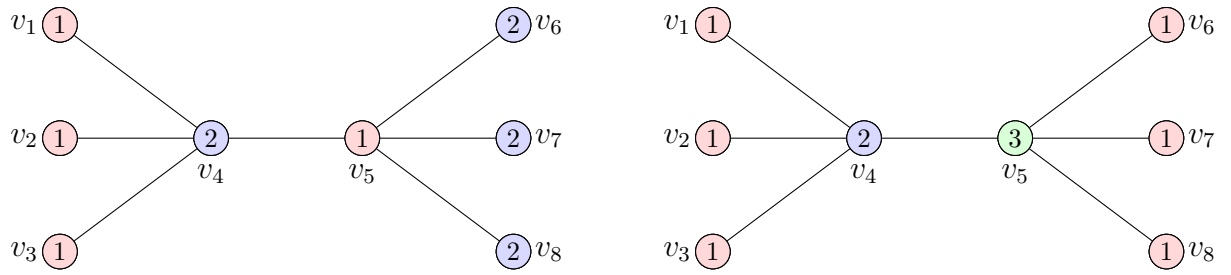

Figure 1: A coloring of a graph G using $\chi(G) = 2$ colors (left) and an optimal MSCP coloring of the same graph with $s(G) = 3$ colors and $\Sigma(G) = 11$(right).

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.

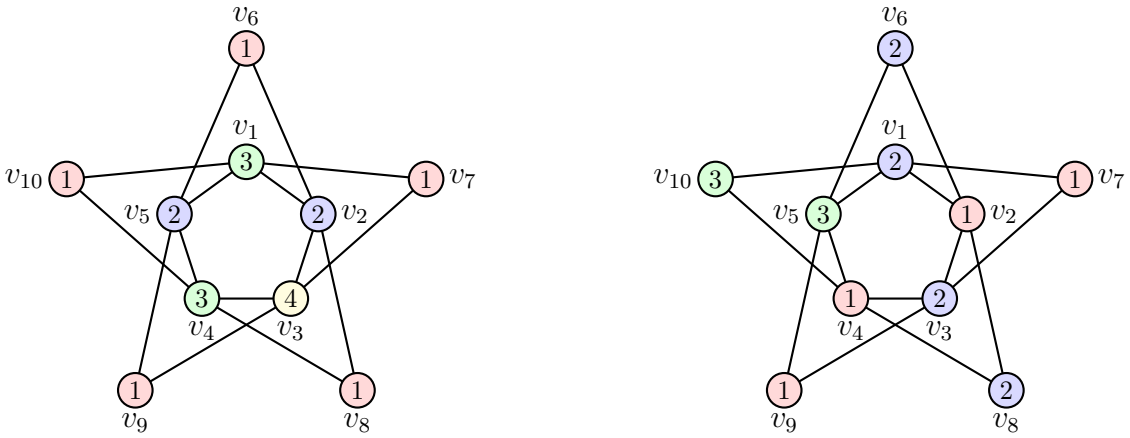

Figure 2: A coloring of a graph using maximum cardinality stable sets (left) and an optimal sum coloring of the same graph using non-maximum cardinality stable sets (right).

1.1 Literature review

We address the interested reader to [12] 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 [16]. Several articles proposed lower bounds for both $\Sigma(G)$ and $s(G)$. In [28] 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, [23] proved that $s(G) \leq \Delta(G)+1$, where $\Delta(G)$ is the maximum degree of the vertices of G, while

[8] 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 [26] the MSCP and its variants are studied, and the authors proved complexity results for specific classes of graphs, while [24] proposed a range of lower bounds for the MSCP based on clique[2] decompositions; 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 [17] greedy algorithms are developed by adapting the classical VCP greedy algorithms DSATUR and RLF. Local search algorithms have been developed as well in [11]. Later, [31, 32] 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 [13, 14].

One of the state-of-the-art algorithms for lower and upper MSCP bounds is called HESA and it has been proposed in [13]; the algorithm is based on a stochastic hybrid evolutionary search. Recently, some new lower bounds have been proposed in [18] 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 [27]. This formulation was originally proposed to tackle a generalization of MSCP in which each color has a predefined cost although no computational results were reported in [27]. Some preliminary computation results using this compact formulation can be found in [7] 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 [7] has been tackled via a dual ascent heuristic algorithm in [25].

1.2 Paper contribution

In this paper we propose the first exact algorithm for MSCP 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 obtain a branch-and-price algorithm which outperforms the direct solution of a compact IP model of the problem, and it is able to solve for the first time several hard instances from the literature.

The remainder of the paper is organized as follows. In Section 2, we review a compact integer programming formulation for MSCP. Section 3 is devoted to present the exponential-size formulation for MSCP and the corresponding branch-and-price algorithm. Finally, in Section 4 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 clique is a subsets of pairwise adjacent vertices.

2. A natural compact IP formulation

In this section, we describe a natural compact IP formulation for the MSCP originally proposed in [27], which is based on the well-known classical formulation for the VCP (see e.g., [22]). 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:

$$[IP_C] \qquad \min \sum_{i \in [k]} \sum_{v \in V} i \cdot x_v^i \tag{1}$$

$$x_u^i + x_v^i \leq 1 \qquad uv \in E, i \in [k], \tag{2}$$

$$\sum_{i \in [k]} x_v^i = 1 \qquad v \in V, \tag{3}$$

$$x_v^i \in \{0,1\} \quad v \in V, i \in [k]. \tag{4}$$

The objective function (1) minimizes the cost of the coloring. Constraints (2) 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 (3) ensure that each vertex is colored with exactly one color.

One of the main drawbacks of this formulation for the VCP is the strong symmetry presented by the associated polytopes, as each coloring can be replaced by an equivalent one by permuting the order of the colors. For the MSCP however, this symmetry does not hold since the cost of each color is different.

Replacing the integrality constraints (4) 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 $\hat{x}_v^i = 0$ $(v \in V, i \geq 3)$. $\square$

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 4, for a computational evaluation of the quality of this lower bound.

We conclude this section by mentioning that there exist other compact and pseudo-polynomial size IP formulations for the VCP. Thanks to extensive preliminary experiments, we can conclude that the adaptations to the MSCP of the *Orientation Model* (see [2]) and of the *Representatives Model* (see [3, 4]) are computationally outperformed by the natural compact formulation IP_C presented in this section.

3. 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 one for VCP introduced by [21]. This model is one of the most effective formulation for the VCP. We address the interested reader to [20, 19] for a discussion on this model for the VCP.

We denote by $\mathscr{S}_G$ the collection of all stable sets of a graph

$$\mathscr{S}_G = \{S \subseteq V : uv \notin E, \text{ for all pairs } u, v \in S\}.$$

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 $i \in [k] := \{1, \ldots, k\}$. 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, which we call IP_E:

$$[IP_E] \qquad \min \sum_{i \in [k]} \sum_{S \in \mathscr{S}_G} c_S^i \cdot \xi_S^i \tag{5}$$

$$\sum_{S \in \mathscr{S}_G} \xi_S^i \leq 1 \qquad i \in [k], \tag{6}$$

$$\sum_{i \in [k]} \sum_{S \in \mathscr{S}_G^v} \xi_S^i \geq 1 \qquad v \in V, \tag{7}$$

$$\xi_S^i \in \{0, 1\} \qquad i \in [k], S \in \mathscr{S}_G, \tag{8}$$

where $c_S^i = i \cdot |S|$ corresponds to the cost of the stable set $S \in \mathscr{S}_G$. The objective function (5) minimizes the cost of the coloring. Constraints (6) prevent from assigning more than one stable set to each color and constraints (7) ensures that each vertex is colored. Although in any optimal solution constraints (7) are satisfied by equality (see Observation 4), by defining them as inequalities we get non-negative dual variables for the pricing problem (see Section 3.1). We also observe that constraints (6) 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 3.2). Finally, we point out that IP_E can be seen as a Dantzig-Wolfe reformulation of Constraints (2) of IP_C (see e.g., [30]).

By replacing constraints (8) 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 (7) are satisfied by $\hat{\xi}$ at equality for every $v \in V$.*

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

Proposition 2 *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 (7) 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 (6). Accordingly, $\hat{x}$ satisfies constraints (2). 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 (7) are satisfied by equality by $\hat{\xi}$. Accordingly, $\hat{x}$ also satisfies constraints (3). 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 $\hat{\xi}$:

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

3.1 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 [5] 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:

$$\sum_{v \in S} \mu_v + \pi_i \leq c_S^i \qquad S \in \mathscr{S}_G, i \in [k], \tag{9}$$

where π and μ are the dual variables associated with primal constraints (6) and (7), 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 (9). A dual constraint is violated if a stable set $S \in \mathscr{S}_G$ and a color $i \in [k]$ exist such that

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

Accordingly, for a given color $i \in [k]$, it is necessary to determine if for 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 $\mu_v - i \leq 0$, 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 the number of MWSS problems which have to be solved, thus (potentially) speeding up the computational convergence.

Proposition 3 *Let $\hat{\xi}$ be an optimal solution of the RMP. Let $\bar{i}$ be the largest color for which a variable in $\hat{\xi}$ has strictly positive value*

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

If no negative reduced cost variable exists for a color $j > \bar{i}$, 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 $j \in \{\bar{i}+1, \ldots, k\}$. If no negative reduced cost variable exists for a color $j > \bar{i}$, then $\sum_{v \in S}(\mu_v - j) \leq -\pi_j = 0$ 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. □

Thanks to Proposition 3, 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 3 is met.

3.2 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 extend the branching rule introduced by Mehrotra and Trick [21] for the VCP. 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 there exists an integer solution $\bar{\xi}$ of LP_E with the same cost.*

Proof. 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 Proposition 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

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

where the inequality is given by the sum of constraints (6) 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. □

As a further remark, we note that the proof of Lemma 1 suggests 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.

The branching rule we propose is based on the following Proposition.

Proposition 4 *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 (7) 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 4. 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 $\hat{\xi}^i_S$ 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.

Proposition 4 ensures that a couple of vertices for branching exists if the solution of the RMP is fractional, thus it is a complete branching scheme. In line with the procedure proposed in [21], we find a specific couple 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 $0 < \phi(S_1) = \sum_{i\in[k]} \hat{\xi}^i_{S_1} < 1$ 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 3. 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.

In our implementation of the branch-and-price algorithm, we manage the branching decisions by maintaining a *merged graph* data structure on each node of the branch and bound tree, which allows us to efficiently perform the branching and to efficiently translate to the original graph the stable sets containing *merged vertices*.

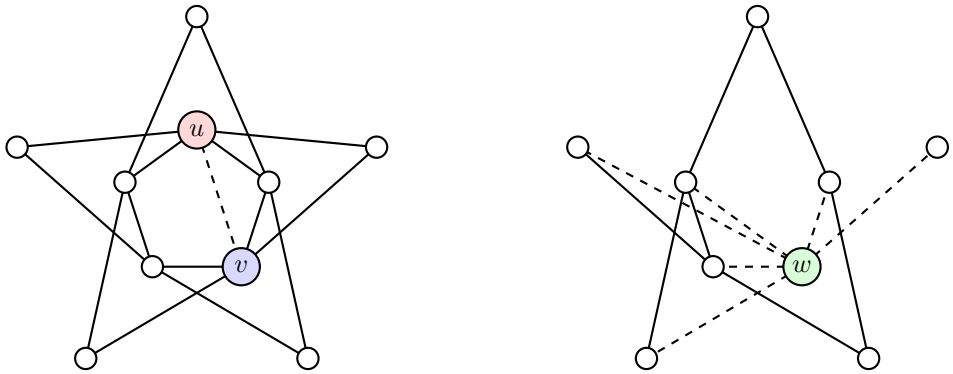

Figure 3: The two subproblem graphs obtained after branching.

4. Computational results

In this section, we present the results of the computational experiments comparing the performance of the branch-and-price algorithm based on formulation IP_E, denoted as BP in the remainder, with the direct use of two MIP solvers on IP_C. Namely, we use IBM CPLEX 12.8 and SCIP 6.0.1, with CPLEX as linear programming solver, and we denote these two approaches by IP_C^{cpx} and IP_C^{scp}, respectively, in the reminder. The goal of these experiments is to assess 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 formulations IP_C and IP_E, beyond the theoretical dominance discussed in the previous sections.

We consider two sets of instances, i.e., Erdös-Rényi graphs defined according to a specified edge probability (see Section 4.1), and the classical DIMACS instances commonly used to test the performance of coloring algorithms (see Section 4.2).

The BP algorithm was implemented in C++ by using the API provided by SCIP and using CPLEX to solve the linear relaxations. The pricing problems are solved by the combinatorial branch-and-bound algorithm for MWSS proposed by Held, Cook and Sewell [10]. 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 initial set of variables of the RMP must contain a feasible solution in order to start the CG procedure, see Section 3.1. 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 (6). 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.

4.1 Random Erdös-Rényi graphs

The first test set is composed by random 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. A time limit of 1800 seconds is set for each run.

We start this section by tuning the configuration of our BP algorithm, determining 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 extreme configurations, 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 (RMC). 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 [6]. 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 entire test-bed of 225 instances, using one column per pricing step does not yield good results and 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.

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

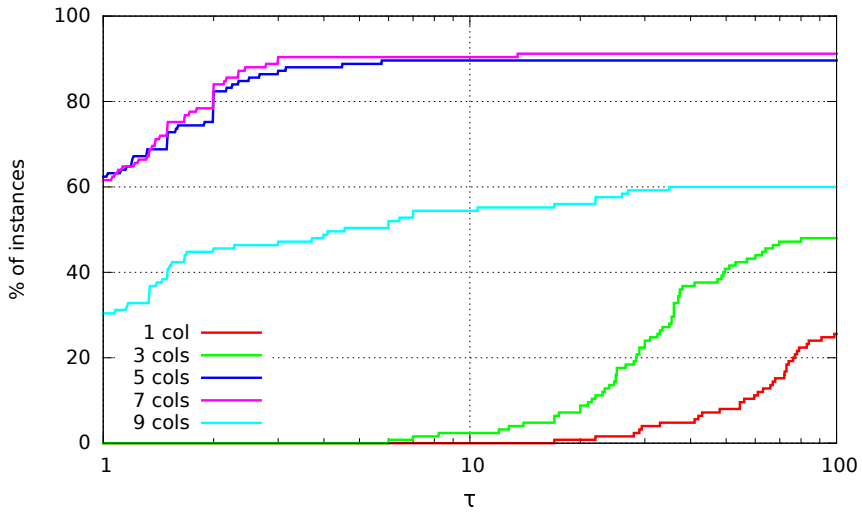

Figure 4: Performance profile of the different configurations of the BP algorithm.

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 1 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 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 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 1 clearly shows that IP_C^{cpx} outperforms IP_C^{scp} for this tests bed of

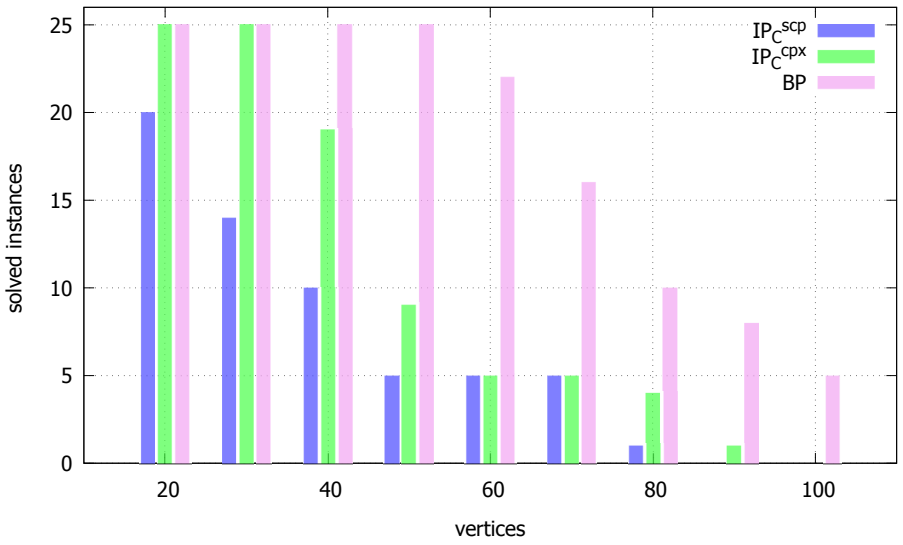

Figure 5: Number of instances solved to optimality by IP_C^{cpx}, IP_C^{scp} and BP for random Erdös-Rényi graphs with $|V| \in \{20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

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

Table 1 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 2 (considering the same set of instances as in Table 1). 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 2 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.

4.2 DIMACS graphs

The second set is composed by the DIMACS instances which have been proposed in the literature (see [29]) to test the performance of algorithms for the VCP. We performed a preliminary test to asses if these instances are hard for the MSCP as well and we run IP_C^{cpx} on all DIMACS graphs with up to 200 vertices (53 graphs in total). More in details, we tested IP_C^{cpx} with a time limit of 10 seconds and we identified 33 instances which can be classified as "easy" for the MSCP. The results of these tests are reported in Table 3. The table reports the instance name, then it reports the instance features in terms of number of vertices (column $|V|$), number of edges (column $|E|$) and percentage edge density (column δ). The table finally reports the best known upper and lower bound reported in the literature in [18] (columns UB_b and LB_b). The last columns of the table report the Chromatic Sum of the graph (column Sol), the computing time taken by IP_C^{cpx} to solve the instance (column $Time$), the number of branching nodes (column $Nodes$) and the dual bound computed at the root node (as in Table 5). Table 3 shows that these 33 instances can be easily solved by IP_C^{cpx} with very limited computational effort; to the best of our knowledge. It is wort mentioning that BP is able to solve all but 2 instances of this set with an average computing time of of 1123 seconds. The fact that IP_C^{cpx} outperforms BP on this set of instance is not surprising since specialized branch-and-price algorithms like BP are mostly competitive when the direct solution of compact models is difficult, while the computational cost of performing

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

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

						IP_C^{cpx}			
Instance	$\|V\|$	$\|E\|$	δ	UB_b	LB_b	Sol	Time	Nodes	LB_r
1-FullIns_3	30	100	23 %	54	53	54	0	1	54.00
1-FullIns_4	93	593	14 %	166	161	166	3	51	158.19
1-Insertions_4	67	232	10 %	119	117	119	3	107	109.03
2-FullIns_3	52	201	15 %	93	91	93	0	1	92.00
2-Insertions_3	37	72	11 %	62	62	62	0	9	58.04
2-Insertions_4	149	541	5 %	249	243	249	10	77	233.77
3-FullIns_3	80	346	11 %	145	141	145	0	1	143.61
3-Insertions_3	56	110	7 %	92	92	92	0	16	86.10
4-FullIns_3	114	541	8 %	205	198	205	0	1	204.00
4-Insertions_3	79	156	5 %	127	126	127	0	21	121.14
5-FullIns_3	154	792	7 %	280	269	280	2	11	277.25
anna	138	493	5 %	276	273	276	0	1	276.00
david	87	406	11 %	237	234	237	0	1	237.00
games120	120	638	9 %	443	442	443	1	5	442.00
huck	74	301	11 %	243	243	243	0	1	243.00
jean	80	254	8 %	217	216	217	0	1	217.00
miles250	128	387	5 %	325	318	325	0	1	324.00
mug100_1	100	166	3 %	202	202	202	1	2243	195.50
mug100_25	100	166	3 %	202	202	202	2	2748	195.67
mug88_1	88	146	4 %	178	178	178	1	3586	173.11
mug88_25	88	146	4 %	178	178	178	1	2243	172.50
mulsol.i.1	197	3925	20 %	1957	1957	1957	2	1	1957.00
mulsol.i.2	188	3885	22 %	1191	1191	1191	2	1	1191.00
mulsol.i.3	184	3916	23 %	1187	1187	1187	2	1	1187.00
mulsol.i.4	185	3946	23 %	1189	1189	1189	2	1	1189.00
mulsol.i.5	186	3973	23 %	1160	1160	1160	2	1	1160.00
myciel3	11	20	36 %	21	21	21	0	3	19.08
myciel4	23	71	28 %	45	44	45	0	28	40.39
myciel5	47	236	22 %	93	89	93	1	278	80.63
queen5_5	25	160	53 %	75	75	75	0	1	75.00
queen7_7	49	476	40 %	196	196	196	1	1	196.00
queen8_12	96	1368	30 %	624	624	624	4	115	624.00
r125.1	125	209	3 %			257	0	1	257.00

Table 3: DIMACS instances solved by IP_C^{cpx} in at most 10 seconds.

a decomposition does not pay in general when compact models can be tackled with limited effort. It is worth mentioning that 14 of these instances were open, in some cases with large gaps between the lower and upper bounds reported in the literature (see cite [18]).

Tables 4 and 5 show the results on the remaining 19 "hard" DIMACS instances with an extended *time-limit* of 7200 seconds. The structure is the same as for the Tables 1 and 2, with the additional information on the best lower and upper bounds reported in the literature [18]. A hyphen on the gap or bounds columns means either that the formulation could 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 dual gap of 4.98%. On the other hand, the average dual gap achieved by IP_C^{cpx} on the 12 unsolved instances is 16.59% (some of the instances showing gaps of above 50%). These results suggest that the linear relaxation of IP_E provides much tighter bounds than those computed by IP_C^{cpx}, fact that is also evidenced by the lower bounds at the root nodes displayed in Table 5. We should remark that the 7 instances solved to optimality by the

				IP_C^{scp}			IP_C^{cpx}			BP			
Instance	$\|V\|$	$\|E\|$	δ	Time	Gap	nodes	Time	Gap	nodes	Time	Gap	nodes	cols
DSJC125.1	125	736	9 %	7200	24.54 %	95592	7200	21.71 %	190914	7200	11.46 %	25	9136
DSJC125.5	125	3891	50 %	7200	388.92 %	446	7200	97.29 %	8376	7200	7.24 %	530	11000
DSJC125.9	125	6961	90 %	7200	1121.56 %	95	7200	53.46 %	2181	25	0.00 %	29	6281
miles1000	128	3216	40 %	7200	606.67 %	403	125	0.00 %	504	56	0.00 %	8	7481
miles1500	128	5198	64 %	7200	1269.82 %	255	73	0.00 %	530	8	0.00 %	1	3508
miles500	128	1170	14 %	7200	95.78 %	10563	25	0.00 %	1089	667	0.00 %	22	27798
miles750	128	2113	26 %	7200	307.22 %	1330	66	0.00 %	984	268	0.00 %	47	24130
myciel6	95	755	17 %	443	0.00 %	4946	15	0.00 %	1125	7200	4.85 %	589	87803
myciel7	191	2360	13 %	7200	13.56 %	7278	7200	5.52 %	28920	7200	8.33 %	148	98797
queen10_10	100	1470	30 %	7200	149.28 %	2545	7200	1.64 %	275223	7200	2.55 %	711	15000
queen11_11	121	1980	27 %	7200	199.98 %	1279	7200	1.93 %	137796	7200	4.27 %	128	37989
queen12_12	144	2596	25 %	7200	249.57 %	783	7200	3.85 %	128289	7200	- %	1	13210
queen13_13	169	3328	23 %	7200	292.05 %	438	7200	2.11 %	103190	7200	- %	1	12165
queen14_14	196	4186	22 %	7200	332.41 %	231	7200	5.31 %	59147	7200	- %	1	20025
queen6_6	36	290	46 %	7200	15.51 %	962266	294	0.00 %	101801	0	0.00 %	4	736
queen8_8	64	728	36 %	7200	44.88 %	49964	7200	0.69 %	482695	3	0.00 %	4	2206
queen9_9	81	1056	33 %	7200	89.28 %	7283	7200	0.99 %	369164	7200	0.87 %	1478	87223
r125.1c	125	7501	97 %	7200	2438.98 %	17	645	0.00 %	1	6	0.00 %	1	4118
R125.5	125	3838	50 %	7200	709.45 %	314	7200	4.56 %	9710	7200	0.25 %	1391	184299

Table 4: Execution times, achieved dual gap and nodes in the branch & bound tree for IP_C^{scp}, IP_C^{cpx} and BP on hard DIMACS instances.

compact formulation IP_C^{cpx} were reported yet as unsolved in the literature. When comparing the bounds from BP against the 17 instances of this group with best known bounds on the literature (yet unsolved) we can see that BP solves to optimality 8 of these 17 instances. Additionally, BP improved the best known lower bound on 4 of the remaining 9 instances.

5. 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. In addition, experiments on the DIMACS set allowed the new algorithm to solve to optimality 8 out of 17 hard and yet unsolved instances.

					IP_C^{scp}			IP_C^{cpx}			BP		
Instance	$\|V\|$	$\|E\|$	UB_b	LB_b	UB	LB	LB_r	UB	LB	LB_r	UB	LB	LB_r
DSJC125.1	125	736	326	303	339	273	235.67	346	285	258.75	350	315	313.05
DSJC125.5	125	3891	1012	924	1340	275	256.18	1179	598	571.05	1059	988	977.76
DSJC125.9	125	6961	2503	2124	3164	260	258.19	2667	1738	1727.16	2503	2503	2499.46
miles1000	128	3216	1666	1623	1965	279	271.16	1666	1666	1656.14	1666	1666	1666.00
miles1500	128	5198	3354	3239	3657	267	266.48	3354	3354	3250.00	3354	3354	3354.00
miles500	128	1170	705	686	760	389	289.12	705	705	700.00	705	705	705.00
miles750	128	2113	1173	1145	1321	325	282.37	1173	1173	1153.64	1173	1173	1172.17
myciel6	95	755	189	177	189	189	155.39	189	189	159.64	189	181	175.38
myciel7	191	2360	381	350	381	336	307.88	382	362	317.22	381	352	348.06
queen10_10	100	1470	553	551	630	253	208.61	559	550	550.00	564	550	550.00
queen11_11	121	1980	733	726	861	288	250.66	740	726	726.00	757	726	726.00
queen12_12	144	2596	943	936	1122	321	298.26	972	936	936.00	960	-	-
queen13_13	169	3328	1191	1183	1428	365	345.17	1208	1183	1183.00	1206	-	-
queen14_14	196	4186	1482	1470	1763	408	399.03	1548	1470	1470.00	1499	-	-
queen6_6	36	290	138	127	138	120	74.02	138	138	126.00	138	138	138.00
queen8_8	64	728	291	289	297	205	133.91	291	289	288.00	291	291	291.00
queen9_9	81	1056	409	406	460	244	169.99	409	405	405.00	409	406	405.00
r125.1c	125	7501			6597	260	258.86	2184	2184	2179.78	2184	2184	2184.00
R125.5	125	3838			2175	269	265.30	1844	1764	1758.16	1824	1820	1819.34

Table 5: Upper and lower bounds and lower bound at root node achieved by IP_C^{scp}, IP_C^{cpx} and BP on hard DIMACS instances.

Acknowledgments

Diego Delle Donne was partially supported by the MathSTIC Research Federation. Enrico Malaguti was partially supported by the European Union's EU Framework Programme for Research and Innovation Horizon 2020 under the Marie Skłodowska-Curie Actions Grant Agreement No 764759.

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] Denis Cornaz, Fabio Furini, and Enrico Malaguti. Solving vertex coloring problems as maximum weight stable set problems. *Discrete Applied Mathematics*, 217:151–162, 2017.

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

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

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

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

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

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

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

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

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

[30] 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.

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

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