Received December 19, 2019, accepted January 9, 2020, date of publication February 10, 2020, date of current version April 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2973122

A Column Generation-Based Lower Bound for the Minimum Sum Coloring Problem

MEHDI MRAD [1], OLFA HARRABI [2], JOUHAINA CHAOUACHI SIALA [3], AND ANIS GHARBI [1]
[1]Department of Industrial Engineering, College of Engineering, King Saud University, Riyadh 11421, Saudi Arabia
[2]Higher Institute of Management of Tunis, ISG Tunis, Tunis University, Bardo 2000, Tunisia
[3]Institute of Advanced Business Studies of Carthage, Carthage University, IHEC Carthage Presidency, Carthage 2017, Tunisia

Corresponding author: Mehdi Mrad (mmrad@ksu.edu.sa)

This research was funded by the Deanship of Scientific Research at King Saudi University through the Research Group Project no RGP-296.

ABSTRACT The objective of this paper is to derive a tight and efficient lower bound for the minimum sum coloring problem. This NP-hard problem is a variant of the classical graph coloring problem where the objective is to minimize the sum of the colors. A column generation approach is proposed to solve the linear relaxation of a set partition-based formulation. Various enhancements are proposed in order to efficiently obtain attractive columns while avoiding as much as possible the exact solution of the huge number of the NP-hard pricing problems. Experimental results conducted on 42 hard benchmark instances show an average reduction of 89.73% of the gap between the best known lower and upper bounds, including 14 new optimality results.

INDEX TERMS Chromatic sum, graph coloring, column generation, lower bound.

I. INTRODUCTION

In this paper, we consider the Minimum Sum Coloring Problem (*MSCP*) which can be formally defined as follows: Let $G = (V, E)$ be an undirected graph, where $V = \{v_1, \ldots, v_n\}$ is a set of n vertices and E is a set of m edges. Given a set of K colors $\{1, \ldots, K\}$, the *MSCP* consists in assigning a color to each vertex such that no adjacent vertices have the same color, while minimizing the sum of assigned colors. In other words, we need to find a mapping $c : V \to \{1, \ldots, K\}$ such that:

- $c(v_i) \neq c(v_j)$ for all $(v_i, v_j) \in E$.
- the *chromatic sum* $\sum_{i=1}^{n} c(v_i)$ is minimized.

Interestingly, the *MSCP* has a variety of practical applications including:

- *VLSI design:* It consists in minimizing the total wiring length that is used in connecting nets to terminals and tracks so that no overlapping nets can be routed within the same track ([1], [2] and [3]).
- *Resource allocation:* The objective is to minimize the average response time of a system of processors that compete over resources where no two jobs with conflicting requirements are executed simultaneously [4].
- *Job scheduling*: Dependent jobs are to be scheduled on multiple dedicated machines, where each job requires the exclusive use of at most k machines. The objective is to minimize the total completion time [5].
- *Clustering*: The cluster deletion problem consists in finding the minimum number of edges that can be removed from an input graph to obtain a cluster graph, i.e. where every connected component is a clique [6].
- *Fixed interval scheduling with machine-dependent processing costs:* A feasible non-preemptive schedule with minimum total processing costs is to be found in a parallel machine environment where jobs are processed during fixed time intervals and incur some machine-dependent processing costs [7].

The *MSCP* models many graph coloring generalizations such as the sum multi-coloring [8], the sum list coloring [9], and the bandwidth coloring [10]. Actually, the *MSCP* is a variant of the classical graph coloring problem. The latter problem consists in finding the so-called *chromatic number* of a graph G, which is the minimum number of colors $\chi(G)$ for which no two adjacent nodes of G have the same color. It is worth noting that the number of colors needed to get the smallest chromatic sum is called the *strength* $s(G)$ of the graph G which is actually an upper bound on $\chi(G)$. For illustrative purpose, the graph depicted in Figure 2 has a chromatic number $\chi(G) = 2$ (Figure 1a.), but requires 3 colors to achieve the chromatic sum of 11 i.e. $s(G) = 3$ (Figure 1b). Note that using only 2 colors leads to a suboptimal sum of 12.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wei.

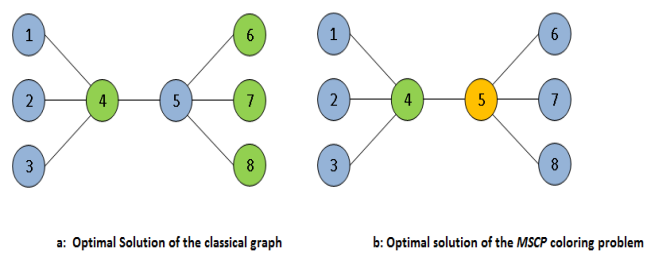

1
2
3
4
5
6
7
8
a: Optimal Solution of the classical graph
b: Optimal solution of the *MSCP* coloring problem

FIGURE 1. Difference between the classical graph coloring problem and the MSCP.

Kubicka and Schwenk [11] have shown that the *MSCP* is NP-hard in general, but polynomial time solvable for trees. The dynamic programming algorithm for trees can be extended to partial k-trees, block graphs and cographs [12]. Moreover, the *MSCP* can be solved in polynomial time on a wide subclass of graphs ([13] and [14]).

Polynomial-time algorithms and k-approximations algorithm have been proposed for some special cases of graphs, such as interval graphs and bipartite graphs ([7] and [15]). Numerous heuristic approaches have been proposed for the *MSCP*. Li et al [16] presented two families of greedy algorithms for the *MSCP* and improved the two previous state-of-the-art greedy algorithms of Brelaz [17] and Leighton [18]. Later on, an effective heuristic which is based on independent set extraction has been proposed by Wu and Hao [19]. This method has shown a potential to perform well on large graphs with more than 500 vertices improving numerous best known upper bounds on benchmark instances. During the last decade, various local search heuristics and metaheuristics have been proposed for the *MSCP*. These include the Tabu Search algorithm of Bouziri and Jouini [20], the Multi-Neighbourhood search of Helmar and Chiarandini [21], the Breakout Local Search algorithm of Benlic and Hao [22], the Memetic Algorithms of Moukrim *et al.* [23] and Jin *et al.* [24], the Hybrid Evolutionary Search of Jin and Hao [25], the D-Wave Quantum Computer-based approach of Mahasinghe *et al.* [26], the Parallel Metaheuristics of Kokosiński and Bała [27], and the bi-objective genetic algorithm of Harrabi and Siala [28].

As far as we know, the only exact approaches that have been proposed for the *MSCP* are the quadratic programming model of Wang *et al.* [29] and the branch-and-bound algorithm of Lecat *et al.* [30]. For more references on *MSCP*, the reader is referred to the recent review paper of Jin *et al.* [31].

The objective of this paper is to derive a tight and efficient lower bound for the *MSCP*. For this purpose, a set partition-based formulation is proposed and is shown that it yields a linear relaxation that outperforms those of the edge-based and the vertex-based formulations. Note that the set partitioning formulation has been successfully applied for the classical graph coloring problem (see for instance Mehrotra and Trick [32], Hansen *et al.* [33], Malaguti *et al.* [34], Gualandi and Malucelli [35], and Held *et al.* [36]). Although its extension to the *MSCP* looks quite straightforward, it seems that the graph coloring research community apprehended its use due to its high computational burden. Indeed, the set partition-based formulation for the *MSCP* involves much more columns than that for the classical graph coloring problem. More importantly, it requires a significantly larger number of pricing problems that need to be solved. In this paper, an efficient column generation approach is proposed to solve the linear relaxation of the set partition-based formulation. Various enhancements are devised in order to significantly reduce the number of pricing problems to be solved. These include the following procedures:

- A graph reduction procedure that substantially reduces the size of the graph by removing a set of unnecessary vertices along with their incident edges.
- An unattractiveness detection procedure that enables to preliminary check the nonexistence of an attractive column for a given color without solving the corresponding pricing problem.
- An attractiveness checking procedure that takes benefit from already generated columns for a given color to derive attractive ones for larger colors without solving their corresponding pricing problems.
- A heuristic solution that potentially generate attractive column without resorting to solve the pricing problem exactly.

Our experimental results provide strong evidence of the proposed enhancement since only 16.99% of the pricing problems had to be solved exactly, on average. Moreover, our proposed lower bound provides results that are larger than or equal to the best known values of the state-of-the-art lower bounds for all of the considered 42 benchmark instances. Indeed, consistent reductions of the gap between the best known lower and upper bounds have been obtained (89.73% on average). The optimal solution has been reached for 33 out of 42 instances, including 14 new optimality results. The best known lower bounds being substantially improved for 7 out of the 9 remaining instances.

The remaining of this paper is organized as follows. Section 2 describes three mathematical formulations of the *MSCP*, along with some dominance relationships between their respective linear relaxations. Section 3 details the proposed column generation approach for the *MSCP* with several enhancements for an efficient convergence. Section 4 presents the computational results obtained on various benchmark instances. Section 5 is devoted to the conclusion and some directions for future research.

II. STATE-OF-THE-ART LOWER BOUNDS

In this section, we describe the existing lower bounds of the literature that have been proposed for the *MSCP*.

A. ALGEBRAIC LOWER BOUNDS

Based on minimum bipartite graph computation, Thomassen *et al.* [37] have shown that $\left\lceil \sqrt{8m} \right\rceil$ is a valid lower bound for the *MSCP* (recall that m denotes the number of edges in the considered graph). To the best of our knowledge, this is the first lower bound that has been proposed so far for the *MSCP*. Later on, Kokosinski and Kwarciany [38] have remarked that since the minimum number of colors to get the chromatic sum is $s(G)$, then a lower bound on the minimum sum coloring with exactly $s(G)$ can be obtained by assigning color i to each node $i = 1, \ldots, s(G)$ and color 1 to each of the remaining $n - s(G)$ nodes. Therefore, $\sum_{i=1}^{s(G)} i + (n - s(G)) = n + s(G)(s(G)-1)/2$ is a valid lower bound for the chromatic sum. Since $s(G)$ is at least equal to the chromatic number $\chi(G)$, then valid lower bound is $n + \chi(G)(\chi(G) - 1)/2$.

In the *MSCP* community, the bound $LB_t = \max\{\lceil\sqrt{8m}\rceil, n+\chi(G)(\chi(G)-1)/2\}$ is referred to as the *theoretical* lower bound.

Recently, Lecat *et al.* [39] used the so-called *motif* concept to derive a clever algebraic lower bound for the MSCP. This concept was originally proposed to compute upper bounds for the chromatic strength [40]. The obtained lower bound, denoted hereafter by $LB_{M\sum}$, exhibited a very good performance on various sets of benchmark instances.

B. CLIQUE PARTITION-BASED LOWER BOUNDS

Moukrim *et al.* [41] observed that the chromatic sum of any partial graph G' of G is a lower bound for the chromatic sum of G. Therefore, they focused on extracting some special partial graphs for which the chromatic sum can be efficiently computed, such as path unions, trees and cliques. They showed that path unions and trees do not yield promising lower bounds. Indeed, their experimental results showed that even an upper bound on the chromatic sum of such partial graphs gives very weak values. However, they found interesting to partition the graph G into disjoint cliques $G_1, G_2, \ldots, G_l$ so as to maximize $\Sigma_{i=1}^{l} \frac{n_i(n_i+1)}{2}$, where n_i denotes the number of nodes in clique G_i $(i = 1, \ldots, l)$ (note that the optimal chromatic sum of a clique G_i is equal to $\frac{n_i(n_i+1)}{2}$ which is obtained by assigning color h to each node $h = 1, 2, \ldots, n_i$). By observing that any coloring of the complementary graph $G' = (V, V^2 \backslash E)$ provides a partion of G into cliques, it turns out that a lower bound for the *MSCP* can be obtained by solving a coloring problem on G'. Moukrim *et al.* [41] solved the latter problem by using a Repeated Modified Degree Saturation algorithm (RMDS). It consists in repeating n times on the graph G', each time starting from a different vertex, a greedy algorithm which is based on the so-called ''modified degree saturation'' rule developed by Li *et al.* [16].

The idea of clique decomposition has been later revisited by several authors. Helmar and Chiarandini [21] solved the coloring problem that is defined on G' by using the fifth variant of the Modified Degree Saturation greedy algorithm proposed by Li *et al.* [16], denoted by MDS(5), together with a local search algorithm that is based on variable neighborhood search and iterated local search. Douiri and Elbernoussi [42] used an ant colony optimization algorithm (*ANT*) to solve the same problem. Wu and Hao [43] proposed a clique extraction approach (*EXCLIQUE*) that iteratively identifies and removes the maximum number of independent cliques having the largest size. Moukrim *et al.* [23] devised a Memetic Algorithm (*MA* − *MSCP*) hybridizing a simple genetic algorithm with local search. The best results obtained so far have been obtained by the Hybrid Evolutionary Search Algorithm (*HESA*) of Jin and Hao [25] which integrates several special features to ensure a high search efficiency, including an original recombination mechanism to generate offspring solutions and an iterated double-phase tabu search procedure to ensure local optimization. For more information about tabu search algorithm, the reader is referred to [44]. Recently, Wu et al [45] proposes an extraction and backward expansion approach (*EBES*) to compute upper and lower bounds of the sum coloring problem. Based on the principle of "reduce-and-solve", the approach first investigates an extraction phase that mainly reduces the size of the given graph by extracting as many collections of color classes as possible. Then, during the backward extension phase, the heuristic approach recovers the intermediate graphs by adding the extracted independent sets. To get tight bounds for the sum coloring problem, EBES optimizes the coloring of each intermediate graph.

III. MATHEMATICAL PROGRAMMING FORMULATIONS

A. A VERTEX-BASED FORMULATION

Define the neighbourhood of a vertex v_i, denoted by δ_i, as the set of adjacent vertices to vertex v_i (i.e. $\delta_i = \{v_j \in V : (v_i, v_j) \in E\}$). Let $d_i = |\delta_i|$ denote the degree of vertex v_i. Let $x_{ik} = 1$ if vertex v_i is colored with color k, and $x_{ik} = 0$ otherwise. The sum coloring problem can be modeled using the following so called vertex-based formulation (Recall that n is the number of vertices and K is the number of colors):

$$(P_V) : Minimize \sum_{i=1}^{n}\sum_{k=1}^{K} k x_{ik} \tag{1}$$

$$\text{subject to: } \sum_{k=1}^{K} x_{ik} = 1 \quad \forall i = 1, \ldots, n \tag{2}$$

$$\sum_{v_j \in \delta_i} x_{jk} \leq d_i(1 - x_{ik}) \quad \forall i = 1, \ldots, n; \quad \forall k = 1, \ldots, K \tag{3}$$

$$x_{ik} \in \{0, 1\} \quad \forall i = 1, \ldots, n; \quad \forall k = 1, \ldots, K \tag{4}$$

The objective function (1) is to minimize the total sum of coloring. Constraints (2) ensure that each vertex is colored only once. Constraints (3) state that for each vertex $v_i \in V$, no adjacent vertex can have the same color as v_i. Finally, Constraints (4) indicate that the decision variables x_{ik} are binary-valued.

B. AN EDGE-BASED FORMULATION

Following the same notations of the vertex-based model, one could restate Constraints (3) so as the two extremities of each edge cannot receive the same color. This yields the following edge-based mathematical program:

$$(P_E): \textit{Minimize} \sum_{i=1}^{n}\sum_{k=1}^{K} kx_{ik} \quad (5)$$

$$\text{subject to: } \sum_{k=1}^{K} x_{ik} = 1 \quad \forall i = 1,\ldots,n \quad (6)$$

$$x_{ik} + x_{jk} \leq 1 \quad \forall k = 1,\ldots,K; \; \forall (v_i, v_j) \in E \quad (7)$$

$$x_{ik} \in \{0,1\} \quad \forall i = 1,\ldots,n; \; \forall k = 1,\ldots,K \quad (8)$$

It is worth noting that the edge-based formulation has been first proposed by Sen *et al.* [2] for the sum coloring problem. It is worth noting that relaxing Constraint (4) by setting $x_{ik} \in [0,1]$ ($\forall i = 1,\ldots,n; \forall k = 1,\ldots,K$) yields a larger feasible solutions space and its optimal solution will therefore constitute a valid lower bound for the *MSCP*. Such a transformation is referred to as the *linear relaxation* of the original integer program. Interestingly, the following result shows that the linear relaxation of the edge-based model dominates the one of the vertex-based program.

Proposition 1: Let Z_E^* and Z_V^* denote the optimal objective value of the linear relaxation of the edge-based and the vertex-based formulations, respectively. We have:

$$Z_E^* \geq Z_V^*$$

Proof: It suffices to show that (P_E) is the result of applying the Reformulation and Linearization Technique (RLT) on (P_V).

- *The Reformulation Phase:* It is easy to show that Constraints (3) can be re-written as the following nonlinear constraints:

$$x_{ik} \sum_{v_j \in \delta_i} x_{jk} = 0 \quad \forall i = 1,\ldots,n; \; \forall k = 1,\ldots,K \quad (9)$$

Constraints (4) imply that:

$$x_{ik} \leq 1; \quad \forall i = 1,\ldots,n; \; \forall k = 1,\ldots,K \quad (10)$$

Consequently, if we multiply Constraints (10) from both sides by $(1 - x_{jk})$; $\forall v_j \in \delta_i$, then we obtain:

$$(1 - x_{jk}) * x_{ik} \leq 1 - x_{jk} \quad \forall i = 1,\ldots,n; \forall k = 1,\ldots,K; \forall v_j \in \delta_i \quad (11)$$

$$\Leftrightarrow x_{ik} - x_{jk} * x_{ik} \leq 1 - x_{jk} \quad \forall i = 1,\ldots,n; \forall k = 1,\ldots,K; \forall v_j \in \delta_i \quad (12)$$

- *The Linearization Phase:* Let $t_{ij}^k = x_{jk} * x_{ik}$. Therefore, Constraints (12) are equivalent to:

$$x_{ik} - t_{ij}^k + x_{jk} \leq 1 \quad \forall (v_i, v_j) \in E; \; \forall k = 1,\ldots,K \quad (13)$$

On the other hand, Constraints (9) are equivalent to:

$$\sum_{v_j \in \delta_i} t_{ij}^k = 0 \quad \forall i = 1,\ldots,n; \; \forall k = 1,\ldots,K \quad (14)$$

From (14), we get:

$$t_{ij}^k = 0 \quad \forall (v_i, v_j) \in E; \; \forall k = 1,\ldots,K \quad (15)$$

Using (15) in (13), we can conclude that:

$$x_{ik} + x_{jk} \leq 1 \quad \forall (v_i, v_j) \in E; \; \forall k = 1,\ldots,K \quad (16)$$

Therefore, applying RLT on (P_V) yields (P_E). According to Sherali and Adams [46] the convex hull of the obtained RLT formulation (P_E) is always included in the convex hull of the original formulation (P_V). That is, the linear relaxation of (P_E) dominates that of (P_V). Consequently, we have $Z_E^* \geq Z_V^*$.

C. A SET PARTITIONING FORMULATION

In this section, we propose a set partitioning formulation which is based on the concept of maximum independent set. In graph theory, an independent set is a set of vertices that does not include any adjacent vertices. The main idea of this formulation is to assign each vertex in the graph to an independent set. The objective is therefore to minimize the sum of costs related to all independent sets.

Let R denote the total number of independent sets in V. Define $S_r \subseteq V$ as the r^{th} independent set and $C_{kr} = k * |S_r|$ as the cost of coloring set S_r with color k. For any given $v_i \in V$ and $r = 1,\ldots,R$, let $a_{ir} = 1$ if $v_i \in S_r$ and 0 otherwise. Now, we define the decision variable X_{kr} by $X_{kr} = 1$ if color k is assigned to the vertices of S_r and 0 otherwise. The minimum sum coloring problem can be then formulated by the following integer linear programming model:

$$(P_S): \textit{Minimize} \sum_{k=1}^{K}\sum_{r=1}^{R} C_{kr}X_{kr} \quad (17)$$

$$\text{Subject to: } \sum_{k=1}^{K}\sum_{r=1}^{R} a_{ir}X_{kr} = 1 \quad \forall i = 1,\ldots,n \quad (18)$$

$$\sum_{r=1}^{R} X_{kr} \leq 1 \quad \forall k = 1,\ldots,K \quad (19)$$

$$X_{kr} \in \{0,1\} \quad \forall k = 1,\ldots,K, \; \forall r \in R \quad (20)$$

The objective (17) is to minimize the total cost of independent sets. Constraints (18) stipulate that among all sets including a given vertex v_i, only one set S_r will be assigned a color. Note that assigning a color to a unique independent set does not require this set to be maximal. Indeed, as the reader may appreciate from Figure 1b, the independent set $S_1 = \{5\}$ is not maximal since it is included in the independent set $S_2 = \{1, 2, 3, 5\}$. However, the optimal sum coloring is obtained by assigning a color to S_1 which is different from that of $\{1, 2, 3\}$. Constraints (19) ensure that each color k should be assigned to at most one set S_r. Finally, Constraints (20) indicate that the decision variables X_{kr} are binary-valued.

The following result shows that the linear relaxation of the set partitioning formulation dominates the one of the edge-based model.

Proposition 2: Let Z_S^* denote the optimal objective value of the linear relaxation of the set partitioning formulation. We have:

$$Z_S^* \geq Z_E^*$$

Proof: It suffices to show that the linear relaxation of (P_S) corresponds to the Dantzig-Wolfe decomposition of the linear relaxation of (P_E). Denote by X_k the vector $(x_{1k}, x_{2k}, \ldots, x_{nk})$ for $k = 1, \ldots, K$, by C_k the vector of n elements all equal to k, $\mathbb{1}_n$ a vector of n elements all equal to one, $\mathbb{1}_m$ a vector of m elements all equal to one and by A the edge-vertex incidence matrix of the graph. The linear relaxation of (P_E) can be written as follows:

$$(LP_E) : Minimize \sum_{k=1}^{K} C_k^T X_k \tag{21}$$

$$\text{subject to: } \sum_{k=1}^{K} X_k = \mathbb{1}_n \tag{22}$$

$$AX_k \leq \mathbb{1}_m \quad \forall k = 1, \ldots, K \tag{23}$$

$$X_k \in [0, 1]^n \quad \forall k = 1, \ldots, K \tag{24}$$

Now, consider the feasible region $\Pi_k = \{X_k : AX_k \leq \mathbb{1}_m, X_k \in [0, 1]^n\}$ and denote by $\{X_k^0, X_k^1, \ldots, X_k^{R_k}\}$ the set of its extreme points. Note that any point $X_k \in \Pi_k$ can be expressed as a convex combination of $X_k^0, X_k^1, \ldots, X_k^{R_k}$. That is:

$$X_k = \sum_{j=0}^{R_k} \lambda_j^k X_k^j \tag{25}$$

$$\sum_{j=0}^{R_k} \lambda_j^k = 1 \tag{26}$$

$$\lambda_j^k \in [0, 1] \quad \forall j \in \{0, \ldots, R_k\} \tag{27}$$

It is clear that ${}_n$ is an extreme point of Π_k (say without loss of generality $X_k^0 = {}_n$). Therefore, Equations (25)-(27) may be written as follows:

$$X_k = \sum_{j=1}^{R_k} \lambda_j^k X_k^j \tag{28}$$

$$\sum_{j=1}^{R_k} \lambda_j^k \leq 1 \tag{29}$$

$$\lambda_j^k \in [0, 1] \quad \forall j \in \{1, \ldots, R_k\} \tag{30}$$

By using Equations (28)-(30) in (21)-(24), we obtain the following mathematical program which is nothing but the linear relaxation of (P_S):

$$Minimize \sum_{k=1}^{K} C_k^T \sum_{j=1}^{R_k} \lambda_j^k X_k^j \tag{31}$$

$$\text{subject to: } \sum_{k=1}^{K} \sum_{j=1}^{R_k} \lambda_j^k X_k^j = \mathbb{1}_n \tag{32}$$

$$\sum_{j=1}^{R_k} \lambda_j^k = 1 \tag{33}$$

$$\lambda_j^k \in [0, 1] \quad \forall k = 1, \ldots, K, \; j \in \{1, \ldots, R_k\} \tag{34}$$

At this point, it should be noticed that a set partitioning formulation has been proposed for the classical graph coloring problem and was efficiently solved using column generation and Branch-and-Price techniques (Malaguti et al. [34], Gualandi and Malucelli [35], Held *et al.* [36]). However, to the best of our knowledge, these techniques have never been used for solving the set partitioning model for the minimum sum coloring problem. This is most probably due to the huge number of pricing problems that need to be solved compared to those generated for the classical graph coloring problem. Indeed, while solving only one pricing problem at each iteration is enough for the classical coloring problem, the *MSCP* requires solving a separate pricing problem at each iteration for each color $k = 1, \ldots, K$ in order to guarantee the convergence of the column generation algorithm. As it will be shown in the next section, an enhanced column generation approach is proposed to efficiently deal with the huge number of pricing problems.

IV. A COLUMN GENERATION APPROACH

The objective of this paper is to derive strong lower bounds for the minimum sum coloring problem. For that purpose, we propose a column generation model to solve the linear relaxation of the set partitioning formulation. Column generation technique is a powerful approach that has been successfully used to solve large-sized linear programs. It consists in starting with solving the so called Master Restricted Problem (*MRP*), which is a reduced version of the linear relaxation of the original problem including only L independent sets (L << R). Actually, the *MRP* includes much less decision variables (or columns) than the original one. Then, an optimality test is performed in order to check the existence/nonexistence of some attractive columns i.e. some variables that would improve the solution if they are added to the *MRP*. For that purpose, an optimization problem, referred to as the *pricing problem*, is solved with the objective of finding a column with minimum reduced cost. If such a column has a negative reduced cost, then it it will be included in the *MRP* which will be solved again. Otherwise, we can conclude that there is no column with negative reduced cost, which means that the currently found solution is optimal.

Let $\mathcal{C}$ denote the set of columns with negative reduced costs that are obtained after solving the pricing problems. The proposed column generation procedure is detailed in Algorithm 1:

The initial set of columns of the *MRP* is constructed as follows. Let MIS_G denote the Maximum Independent Set

Algorithm 1 General Framework of the Column Generation Procedure

```
**Input:** An undirected graph $G = (V, E)$.
**Output:** A lower bound $Z^*_S$.
Step 0: Generate a feasible solution on the graph G. Let $S = \{S_1, S_2, \ldots, S_L\}$ denote the obtained set of independent subsets.
Step 1: Solve the MRP on the set S using a commercial solver and Set $Z^*_{MRP}$ to its optimal objective value. Set $\mathcal{C} = \varnothing$.
Step 2: Solve the pricing problem corresponding to each color $k = 1, \ldots, K$. Add all obtained attractive columns (if any) to $\mathcal{C}$.
Step 3: If $\mathcal{C} \neq \varnothing$, then set $S = S \cup \mathcal{C}$ and go to Step 1.
Else, Set $Z^*_S = Z^*_{MRP}$ and Stop.
```

problem defined on a graph $G = (V, E)$. It consists in finding a subset of vertices $V' \subseteq V$ with maximum cardinality such that no two vertices of V' are adjacent. Clearly, all vertices of V' can be colored with the same color. The Maximum Independent Set problem can be mathematically formulated as follows:

$$(MIS_G) : Maximize \sum_{i=1}^{n} y_i \tag{35}$$

Such that:

$$y_i + y_j \leq 1 \quad \forall (v_i, v_j) \in E \tag{36}$$
$$y_i \in \{0, 1\} \quad \forall v_i \in V \tag{37}$$

The initial set of columns of the *MRP* is constructed using the following simple constructive heuristic which requires iteratively solving a maximum independent set problem (Algorithm 2):

Algorithm 2 Initialization of the Column Generation Procedure

```
**Input:** An undirected graph $G = (V, E)$ and its corresponding MRP.
**Output:** A feasible coloring $c(v_i)$ for all $i \in V$.
Step 0: Set k=1.
Step 1: Solve $MIS_G$ and Set $V' = \{v_i \in V : y_i = 1\}$. Let $c(v_i) = k$ for all $v_i \in V'$ and include the column corresponding to k in the MRP.
Step 2: Update the graph G by setting $V = V \backslash V'$ and $E = E \backslash \{(v_i, v_j) \in E : v_i \in V'$ or $v_j \in V'\}$.
Step 3: If $V \neq \varnothing$, then set $k = k + 1$ and go to Step 1.
Else, Stop.
```

After solving the *MRP*, one needs to check the optimality of the obtained solution. For this aim, we solve the pricing problem in order to detect attractive columns. Since we are dealing with a minimization problem, only columns with negative reduced costs will be added to the master restricted problem. We denote by α_i the dual variables of the constraint (32) and β_k the dual variables of the constraint (33). A column A_k associated to color k is attractive if it is feasible and its reduced cost is negative. Feasibility means that there is no two adjacent vertices in the subset of vertices selected in this column *i.e.* $a_{ik} + a_{jk} \leq 1, \quad \forall (v_i, v_j) \in E$. The reduced cost of a column is computed as follows:

$$\sum_{i=1}^{n} (k - \alpha_i) * a_{ik} - \beta_k \tag{38}$$

Hence, our pricing problem can be expressed as follows:

$$(P_k) : Minimize\ Z_k = \sum_{i=1}^{n} (k - \alpha_i).a_{ik} \tag{39}$$

Such that:

$$a_{ik} + a_{jk} \leq 1 : \forall (v_i, v_j) \in E \tag{40}$$
$$a_{ik} \in \{0, 1\} : \forall v_i \in V, \forall k = 1, \ldots, K \tag{41}$$

Let Z^*_k denote the optimal objective value of (P_k). Clearly, if $Z^*_k < \beta_k$, then the optimal values of (a_{ik}) constitute an attractive column A_k. Note that the pricing problem is equivalent to a maximum weighted independent set problem which turns out to be NP-hard [47]. Although it is quite efficiently solved by commercial solvers, it makes the column generation procedure time consuming due to the large number of generated pricing problems. In the following, various enhancement procedures are proposed to accelerate the convergence of the proposed approach by avoiding the exact solution of the generated pricing problems as much as possible.

A. GRAPH REDUCTION

Interestingly, the size of the graph can be substantially reduced using the following observation:

Observation 1: At a given iteration, all vertices v_i such that $\alpha_i \leq k$ can be removed from the graph, together with their incident edges.

Proof: If $\alpha_i < k$, then any feasible solution of (P_k) such that $a_{ik} = 1$ is clearly suboptimal. Indeed, modifying such a solution by setting $a_{ik} = 0$ yields a feasible solution with better objective value. Therefore, vertex v_i will never appear in the optimal solution of (P_k). If $\alpha_i = k$, then any optimal solution of (P_k) such that $a_{ik} = 1$ will remain optimal if a_{ik} is set equal to zero. Therefore, removing vertex v_i from the graph will not change the optimal objective value of (P_k). Moreover, since vertex v_i satisfies $\alpha_i \leq k'$ for any $k' > k$, then removing vertex v_i from the graph will not change the optimal solution of the pricing problems $(P_{k'})$ for all $k' = k + 1, \ldots, K$.

In the sequel, it is assumed without loss of generality that all vertices of the graph satisfy $\alpha_i > k$.

B. UNATTRACTIVENESS DETECTION

The following result provides a simple test that enables to preliminary check the nonexistence of an attractive column for a given color k without solving the pricing problem (P_k).

Observation 2: If $\sum_{v_i \in V}(k - \alpha_i) \geq \beta_k$, then there is no attractive column for color k.

Proof: Since all vertices v_i of the reduced graph satisfy $\alpha_i > k$, then $\sum_{v_i \in V}(k - \alpha_i) \leq \sum_{v_i \in V}(k - \alpha_i).a_{ik}$ for any feasible solution (a_{ik}) of (P_k). Therefore, if $\sum_{v_i \in V}(k - \alpha_i) \geq \beta_k$, then the reduced cost of column A_k will be positive for any solution of (P_k). Consequently, color k will have no attractive column.

Interestingly, the following observation shows that the optimal solution of a given pricing problem (P_k) can be useful to detect the nonexistence of an attractive column for larger values of k without solving the corresponding pricing problems.

Observation 3: For a given $k' > k$, if $Z_k^* \geq \beta_{k'}$, then there is no attractive column for color k'.

Proof: Clearly, we have $Z_k^* \leq Z_{k'}^*$ for all $k' > k$. Therefore, if $Z_k^* \geq \beta_{k'}$, then all feasible solutions of $(P_{k'})$ will satisfy $\sum_{v_i \in V}(k' - \alpha_i).a_{ik} \geq \beta_{k'}$. Consequently, color k' will have no attractive column.

C. ATTRACTIVENESS CHECKING

Let A_k denote an attractive column for color k, i.e. a feasible solution (a_{ik}) of (P_k) such that $\sum_{v_i \in V}(k - \alpha_i) < \beta_k$. Algorithm 3 describes a procedure that takes benefit from A_k in order to derive attractive columns for colors $k' > k$ without solving the pricing problems $(P_{k'})$.

Algorithm 3 Attractiveness Checking Procedure

```
Input: An undirected graph $G = (V, E)$ and an attractive column $A_k$ for color $k$.
Output: A set of attractive columns $A_{k'}$ for all $k' \geq k$.
Set $V' = V$.
For $k' = k + 1, \ldots, K$
Set $V' = V' \backslash \{v_i \in V' : \alpha_i \leq k'\}$.
If $\sum_{v_i \in V'}(k' - \alpha_i) < \beta_{k'}$ then $A_{k'} = A_k \cap V'$ is an attractive column for color $k'$.
End (for)
```

D. HEURISTIC SOLUTION

If the pricing problem of a given color $k > 1$ is not identified to be useless to solve (using the unattractiveness detection procedure), or no attractive column could be derived from previous obtained columns (using the attractiveness checking procedure), then one has to solve it. In this section, we propose an ultimate attempt to potentially generate an attractive column without resorting to solve the pricing problem exactly. It consists in using the following greedy algorithm to find a (maximum) weighted independent set σ (Algorithm 4). Let $\Lambda(v_i)$ denote the set of adjacent vertices of node v_i.

Now if $\sum_{v_i \in \sigma}(k - \alpha_i) < \beta_k$, then an attractive column A_k is obtained by setting $a_{ik} = 1$ for $v_i \in \sigma$ and $a_{ik} = 0$ otherwise. Actually, Step 2 of Algorithm 1 is replaced by Algorithm 5 that generates attractive columns after embedding all the proposed enhancements, where $\mathcal{P}$ denotes the set of pricing problems that need to be solved, and E_i denotes the set of edges that are incident to vertex $v_i \in V$.

Algorithm 4 Greedy Algorithm for the Pricing Problem

```
Input: An undirected graph $G = (V, E)$ and $\Lambda(v_i) = \{v_j \in V : (v_i, v_j) \in E\}$ for all $v_i \in V$.
Output: A maximum weighted independent set $\sigma$.
Step 1. Let $v_0$ be the node of $V$ with largest $\alpha_i$.
Step 2. Set $\sigma = \sigma \cup \{v_0\}$ and $V = V \backslash \Lambda(v_0)$.
Step 3. If $V = \emptyset$, then STOP. Else, go to Step 1.
```

Algorithm 5 Enhanced Generation of Attractive Columns

```
Input: An undirected graph $G = (V, E)$ and the optimal solution of its MRP.
Output: An attractive column $A_k$ for each color $k = 1, \ldots, K$.
Step 0. Set $k = 1$, $\mathcal{C} = \emptyset$ and $\mathcal{P} = \bigcup_{k=1}^{K} P_k$.
Step 1. While ($P_k \notin \mathcal{P}$ and $k \leq K$)
            $k = k + 1$
    End (while)
    If $k = K + 1$ then Stop.
Step 2. For $v_i \in V$
            If $\alpha_i \leq k$, then set $V = V \backslash \{v_i\}$ and $E = E \backslash E_i$
    End (for)
Step 3. If $\sum_{v_i \in V}(k - \alpha_i) \geq \beta_k$, then set $\mathcal{P} = \mathcal{P} \backslash P_k$, $k = k + 1$ and go to Step 1.
Step 4. Apply the Heuristic Solution procedure. If an attractive column $A_k$ is obtained, then set $\mathcal{C} = \mathcal{C} \cup \{A_k\}$, $\mathcal{P} = \mathcal{P} \backslash P_k$ and go to Step 7.
Step 5. Solve $(P_k)$ using a commercial solver and set $\mathcal{P} = \mathcal{P} \backslash P_k$. If $Z_k^* < \beta_k$ then an attractive column $A_k$ is obtained and set $\mathcal{C} = \mathcal{C} \cup \{A_k\}$. Else go to Step 8.
Step 6. For $k' = k + 1, \ldots, K$
            If $Z_k^* \geq \beta_{k'}$ then set $\mathcal{P} = \mathcal{P} \backslash P_{k'}$
    End (for)
Step 7. Set $V' = V$.
    For $k' = k + 1, \ldots, K$
        Set $V' = V' \backslash \{v_i \in V' : \alpha_i \leq k'\}$.
        If $\sum_{v_i \in V'}(k' - \alpha_i) < \beta_{k'}$ then set $\mathcal{C} = \mathcal{C} \cup \{A_k \cap V'\}$, $\mathcal{P} = \mathcal{P} \backslash P_{k'}$
    End (for)
Step 8. Set $k = k + 1$ and go to Step 1.
```

The following example illustrates one iteration of the column generation procedure after embedding the enhancement procedures.

Example 1: Consider the 6 vertex-9 edge graph that is displayed in Figure 2a. Figure 2b depicts the initial solution with cost equal to 12 (generated by Algorithm 2). The obtained sets are $S_1 = \{1, 2, 4\}$, $S_2 = \{3\}$, $S_3 = \{5\}$ and $S_4 = \{6\}$. After solving the first *MRP*, we get $\beta_k = 0$ for all $k = 1, .., K$ and

$$\alpha_1 = 3, \alpha_2 = 0, \alpha_3 = 2, \alpha_4 = 0, \alpha_5 = 3, \alpha_6 = 4$$

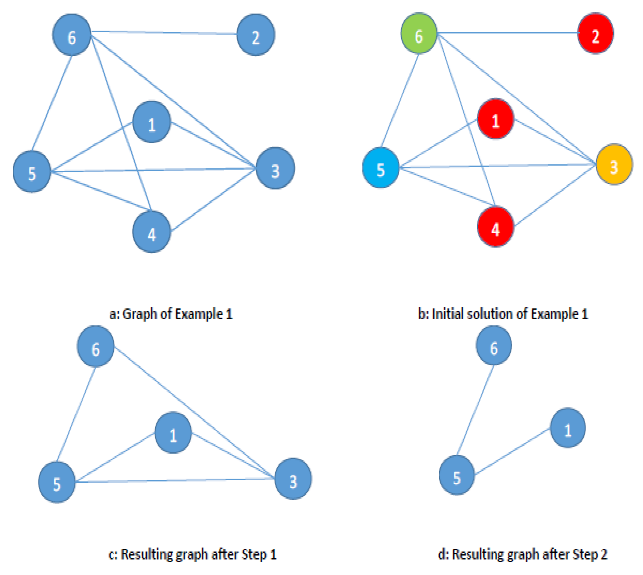

6
2
1
5
3
4
a: Graph of Example 1
b: Initial solution of Example 1
c: Resulting graph after Step 1
d: Resulting graph after Step 2

FIGURE 2. **An illustrative example.**

k = 1 : According to Observation 1, vertices 2 and 4 will be removed from the graph. The resulting graph is displayed in Figure 2c. Note that no unattractiveness could be detected according to Observation 2 since $\sum_{v_i \in V}(k - \alpha_i) = -8 < \beta_1$. Also, the attractiveness checking cannot be applied since there is no previously found attractive column in this step. After applying the greedy heuristic to the pricing problem, we obtain

$$a_{11} = 1, a_{21} = 0, a_{31} = 0, a_{41} = 0, a_{51} = 0, a_{61} = 1$$

The obtained column is considered attractive since its reduced cost is equal to −5.

k = 2 : Since $\alpha_3 = 2 \leq k$, then vertex 3 will be removed from the graph according to Observation 1. The resulting graph is displayed in Figure 2d. Now, after applying attractiveness checking, we observe that the column generated for the previous color ($k = 1$) defined by $a_{11} = 1, a_{21} = 0, a_{31} = 0, a_{41} = 0, a_{51} = 0, a_{61} = 1$ remains attractive since $\sum_{v_i \in V}(k - \alpha_i) = -3 < \beta_2$.

k = 3 : Now the graph will only include vertex 6 since vertices 1 and 5 will be removed thanks to Observation 1. Since $3 - \alpha_6 = -1 < \beta_3$, then the column defined by $a_{11} = 0, a_{21} = 0, a_{31} = 0, a_{41} = 0, a_{51} = 0, a_{61} = 1$ is attractive according to the attractiveness checking procedure.

k = 4 : Since $\alpha_6 = 4$, then vertex 6 is removed from the graph which becomes empty. The procedure moves to the next iteration of the column generation procedure after appending all the generated attractive columns to the *MRP*. Note that the exact solution of the pricing problem has not been called during the whole iteration.

V. COMPUTATIONAL EXPERIMENTS

All of the proposed procedures have been coded using C++ and compiled with Visual studio 2012. The linear programs have been solved by Cplex 12.6. Computational experiments were conducted on an i7 processor with 2.6 Ghz and 16 Gb of available memory.

Our experiments have been conducted on a set of 42 benchmark instances. Some of these instances come from the second DIMACS challenge [1] while the others are part of COLOR 2002-2004 competitions.[2] We limited the size of the considered graphs to 225 vertices or 7000 edges. Indeed, the runtime of the proposed approach on larger graphs significantly exceeded our time limit of 1800 seconds. The number of colors K has been set to the state-of-the-art upper bound on the chromatic strength, namely the bound UB_S proposed by Lecat *et al.* [48]. The considered instances refer to various topologies and densities, which can be classified into the following types:

- Five graphs based on the Mycielski transformation: myciel$a$ with $a \in \{3; 4; 5; 6; 7\}$.
- Twenty one graphs from the Donald Knuth's Stanford GraphBase: miles$n$ with $n \in \{250; 500; 750; 1000; 1500\}$, anna, david, huck, jean, games120, queen$a.a$ with $a \in \{5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15\}$, and queen8.12.
- Six graphs based on register allocation: mulsol.i.a with $a \in \{1; 2; 3; 4; 5\}$ and zeroin.i.3.
- Four graphs that have a hard-to-find four clique embedded: mugn_a with $n \in \{88; 100\}$ and $a \in \{1; 25\}$.
- Two "insertion" graphs: 2-insert_3 and 3-insert_3.
- Three classical random graphs: DSJC125.a with $a \in \{1; 5; 9\}$.

A. RESULTS OF THE STATE-OF-THE-ART LOWER BOUNDS

Table 1 displays the results of each lower bound of the literature on the considered benchmark instances. In this table, we provide for each instance:

- $|V|$: the number of vertices.
- $|E|$: the number of edges.
- UB^*: the best known upper bound of the literature [31].
- LB_t : the value of the theoretical lower bound.
- LB_{RMDS}: the value of the lower bound of Moukrim *et al.* [41].
- $LB_{MDS(5)+LS}$: the value of the lower bound of Helmar and Chiarandini [21].
- LB_{ANT}: the value of the lower bound of Douiri and Elbernoussi [42].
- $LB_{EXCLIQUE}$: the value of the lower bound of Wu and Hao [43].
- $LB_{MA-MSCP}$: the value of the lower bound of Moukrim *et al.* [23].
- LB_{HESA}: the value of the lower bound of Jin and Hao [25].
- $LB_{M\sum}$: the value of the lower bound of Lecat et al [39].
- LB^*: the maximum value over all of the lower bounds of the literature.

[1] http://dimacs.rutgers.edu/Challenges/
[2] http://mat.gsia.cmu.edu/COLOR02

TABLE 1. Results of the State-of-the-art lower bounds.

Instances	|V|	|E|	UB^*	LB_t	LB_{RMDS}	$LB_{MDS(5)+LS}$	LB_{ANT}	$LB_{EXCLIQUE}$	$LB_{MA-MSCP}$	LB_{HESA}	$LB_{M\sum}$	LB^*
myciel3	11	20	21	17	16	16	16	16	16	16	**20**	**20**
myciel4	23	71	45	33	34	34	34	34	34	34	**41**	**41**
myciel5	47	236	93	62	70	70	70	70	70	70	**81**	**81**
myciel6	95	755	189	116	142	142	142	142	142	142	**158**	**158**
myciel7	191	2360	381	219	286	286	286	286	286	286	**308**	**308**
miles250	128	387	325	156	316	**318**	316	**318**	**318**	**318**	267	**318**
miles500	128	1170	705	318	677	**686**	677	677	**686**	**686**	609	**686**
miles750	128	2113	1173	593	-	-	-	-	**1145**	**1145**	973	**1145**
miles1000	128	3216	1666	989	-	-	-	-	**1623**	**1623**	1475	**1623**
miles1500	128	5198	3354	2756	-	-	-	-	**3239**	**3239**	3107	**3239**
queen5.5	25	160	75	36	**75**	**75**	**75**	**75**	**75**	**75**	**75**	**75**
queen6.6	36	290	138	57	126	126	126	126	126	126	**127**	**127**
queen7.7	49	476	196	70	**196**	**196**	**196**	**196**	**196**	**196**	**196**	**196**
queen8.8	64	728	291	100	288	288	288	288	288	288	**289**	**289**
queen9.9	81	2112	409	126	-	-	-	-	405	405	**406**	**406**
queen10.10	100	1470	553	155	-	-	-	-	550	550	**551**	**551**
queen11.11	121	1980	726	178	-	-	-	-	**726**	**726**	**726**	**726**
queen12.12	144	2596	936	210	-	-	-	-	**936**	**936**	**936**	**936**
queen13.13	169	3328	1183	247	-	-	-	-	**1183**	**1183**	**1183**	**1183**
queen14.14	196	4186	1470	287	-	-	-	-	**1470**	**1470**	**1470**	**1470**
queen15.15	225	5180	1800	330	-	-	-	-	**1800**	**1800**	**1800**	**1800**
queen8.12	96	1368	624	162	-	-	-	-	**624**	**624**	**624**	**624**
anna	138	493	276	193	272	**273**	272	**273**	**273**	**273**	241	**273**
david	87	406	237	142	**234**	**234**	**234**	229	**234**	**234**	189	**234**
huck	74	301	243	129	**243**	**243**	**243**	**243**	**243**	**243**	180	**243**
jean	80	254	217	125	**216**	**216**	**216**	**216**	**216**	**216**	158	**216**
games120	120	638	443	156	**442**	**442**	**442**	**442**	**442**	**442**	396	**442**
mulsol.i.1	197	3925	1957	1373	-	-	-	-	**1957**	**1957**	1422	**1957**
mulsol.i.2	188	3885	1191	653	-	-	-	-	**1191**	**1191**	722	**1191**
mulsol.i.3	184	3916	1187	649	-	-	-	-	**1187**	**1187**	718	**1187**
mulsol.i.4	185	3946	1189	650	-	-	-	-	**1189**	**1189**	720	**1189**
mulsol.i.5	186	3973	1160	651	-	-	-	-	**1160**	**1160**	720	**1160**
zeroin.3	206	3540	998	641	**998**	**998**	997	**998**	**998**	**998**	705	**998**
mug88-1	88	146	178	94	163	164	163	164	-	164	**178**	**178**
mug88-25	88	146	178	94	161	162	162	162	-	162	**178**	**178**
mug100-1	100	166	202	106	186	188	187	188	-	188	**202**	**202**
mug100-25	100	166	202	106	183	186	185	186	-	186	**202**	**202**
2-insert-3	37	72	62	43	55	55	55	55	-	55	**59**	**59**
3-insert-3	56	110	92	62	84	84	84	84	-	84	**88**	**88**
DSJC125.1	125	736	326	135	238	238	-	246	247	247	**297**	**297**
DSJC125.5	125	3891	1012	261	504	493	-	536	549	549	**851**	**851**
DSJC125.9	125	6961	2503	1071	1600	1621	-	1664	1689	1691	**2108**	**2108**

It should be noted that the lower bound of Wu et al [45] has not been computed on any of the considered benchmark instances. It is clearly observed from Table 1 that, for all instances, the best values are obtained by LB_{HESA} and $LB_{M\sum}$. Moreover, despite the various methods that have been proposed in the literature, there is still a need to close the gap for the considered medium-sized benchmark instances. The scope of this paper is to contribute in closing this gap.

B. EFFECTIVENESS OF THE PROPOSED COLUMN GENERATION PROCEDURE

Table 2 shows the results of the lower bounds that are provided by each of the presented mathematical formulations. For each formulation, we provide:

- – LB_h: the obtained lower bound, with $h \in \{E, V, S\}$.
  – Gap_h: the obtained relative gap defined by $Gap_h = 100\frac{UB^*-LB_h}{LB_h}$, with $h \in \{E, V, S\}$.
  – Red_h: the obtained gap reduction (if any) with respect to the best known bounds of the literature, defined by $Red_h = 100\frac{LB_h-LB^*}{UB^*-LB^*}$, with $h \in \{E, V, S\}$.

In Table 2, the italic entries denote the improved best known results and the bold entries denote the optimal values. Entries that are marked with an asterisk denote new optimality results.

Table 2 provides strong evidence of the good quality of the proposed column generation procedure. From this table, we observe that the set partioning lower bound consistently outperforms the edge-based and the vertex-based bounds. Indeed, the average relative gap of LB_S is only 0.80% whereas those of LB_E and LB_V are equal to 268.99% and 275.40%, respectively. These empirical results strongly support the theoretical dominance between the mathematical formulations.

As a matter of fact, LB_S provides lower bounds that are larger than or equal to the best known values of the

TABLE 2. Effectiveness of the proposed column generation procedure.

Instances	|V|	|E|	LB^*	UB^*	Edge-based formulation			Node-based formulation			Set partitioning formulation		
					LB_E	Gap_E	Red_E	LB_V	Gap_V	Red_V	LB_S	Gap_S	Red_S
myciel3	11	20	20	21	17	23.53%	-	16	31.25%	-	**21***	0.00%	100.00%
myciel4	23	71	41	45	35	28.57%	-	33	36.36%	-	*44*	2.27%	75.00%
myciel5	47	236	81	93	71	30.99%	-	66	40.91%	-	*88*	5.68%	58.33%
myciel6	95	755	158	189	143	32.17%	-	131	44.27%	-	*176*	7.39%	58.06%
myciel7	191	2360	308	381	287	32.75%	-	261	45.98%	-	*349*	9.17%	56.16%
miles250	128	387	318	325	190	71.05%	-	186	74.73%	-	**325***	0.00%	100.00%
miles500	128	1170	686	705	192	267.19%	-	192	267.19%	-	**705***	0.00%	100.00%
miles750	128	2113	1145	1173	192	510.94%	-	192	510.94%	-	**1173***	0.00%	100.00%
miles1000	128	3216	1623	1666	192	767.71%	-	192	767.71%	-	**1666***	0.00%	100.00%
miles1500	128	5198	3239	3354	192	1646.88%	-	192	1646.88%	-	**3354***	0.00%	100.00%
queen5.5	25	160	75	75	38	97.37%	-	38	97.37%	-	**75**	0.00%	-
queen6.6	36	290	127	138	54	155.56%	-	54	155.56%	-	**138***	0.00%	100.00%
queen7.7	49	476	196	196	74	164.86%	-	74	164.86%	-	**196**	0.00%	-
queen8.8	64	728	289	291	96	203.13%	-	96	203.13%	-	**291***	0.00%	100.00%
queen9.9	81	2112	406	409	122	235.25%	-	122	235.25%	-	405	0.99%	-
queen10.10	100	1470	551	553	150	268.67%	-	150	268.67%	-	550	0.55%	-
queen11.11	121	1980	726	726	182	298.90%	-	182	298.90%	-	**726**	0.00%	-
queen12.12	144	2596	936	936	216	333.33%	-	216	333.33%	-	**936**	0.00%	-
queen13.13	169	3328	1183	1183	254	365.75%	-	254	365.75%	-	**1183**	0.00%	-
queen14.14	196	4186	1470	1470	294	400.00%	-	294	400.00%	-	**1470**	0.00%	-
queen15.15	225	5180	1800	1800	338	432.54%	-	338	432.54%	-	**1800**	0.00%	-
queen8.12	96	1368	624	624	144	333.33%	-	144	333.33%	-	**624**	0.00%	-
anna	138	493	273	276	199	38.69%	-	183	50.82%	-	**276***	0.00%	100.00%
david	87	406	234	237	127	86.61%	-	122	94.26%	-	**237***	0.00%	100.00%
huck	74	301	243	243	111	118.92%	-	105	131.43%	-	**243**	0.00%	-
jean	80	254	216	217	115	88.70%	-	108	100.93%	-	**217***	0.00%	100.00%
games120	120	638	442	443	180	146.11%	-	180	146.11%	-	**443***	0.00%	100.00%
mulsol.i.1	197	3925	1957	1957	266	635.71%	-	260	652.69%	-	**1957**	0.00%	-
mulsol.i.2	188	3885	1191	1191	275	333.09%	-	258	361.63%	-	**1191**	0.00%	-
mulsol.i.3	184	3916	1187	1187	271	338.01%	-	254	367.32%	-	**1187**	0.00%	-
mulsol.i.4	185	3946	1189	1189	273	335.53%	-	256	364.45%	-	**1189**	0.00%	-
mulsol.i.5	186	3973	1160	1160	274	323.36%	-	256	353.13%	-	**1160**	0.00%	-
zeroin.3	206	3540	998	998	285	250.18%	-	264	278.03%	-	**998**	0.00%	-
mug88-1	88	146	178	178	132	34.85%	-	132	34.85%	-	**178**	0.00%	-
mug88-25	88	146	178	178	132	34.85%	-	132	34.85%	-	**178**	0.00%	-
mug100-1	100	166	202	202	150	34.67%	-	150	34.67%	-	**202**	0.00%	-
mug100-25	100	166	202	202	150	34.67%	-	150	34.67%	-	**202**	0.00%	-
2-insert-3	37	72	59	62	56	10.71%	-	54	14.81%	-	**62***	0.00%	100.00%
3-insert-3	56	110	88	92	84	9.52%	-	82	12.20%	-	**92***	0.00%	100.00%
DSJC125.1	125	736	297	326	188	73.40%	-	186	75.27%	-	*314*	3.82%	58.62%
DSJC125.5	125	3891	851	1012	188	438.30%	-	188	438.30%	-	*978*	3.48%	78.88%
DSJC125.9	125	6961	2108	2503	188	1231.38%	-	188	1231.38%	-	*2500*	0.12%	99.24%
Average	-	-	-	-	-	**268.99%**	-	-	**275.40%**	-	-	**0.80%**	**89.73%**

state-of-the-art lower bounds for all of the 42 instances. Furthermore, thanks to the proposed column generation procedure, the gap between the best known lower and upper bounds has been reduced by 89.73%, on average. More interestingly, the set partitioning lower bound is able to reach the optimal solution for 33 instances out of 42, among which 14 results are new optimality ones (i.e. $LB^* < LB_S = UB^*$). The best known lower bounds being substantially improved for 7 out of the 9 remaining instances. For illustrative purpose, the gap between the best known upper and lower bounds has been reduced by 99.24% for one of the largest instances, namely DSJC125.9.

C. EFFICIENCY OF THE PROPOSED COLUMN GENERATION PROCEDURE

The aim of this section is two fold. First, we want to measure how practical is the computation of the set partitioning lower bound. Second, we want to assess the impact of the implemented enhancements on the column generation procedure. Table 3 depicts, for each instance, the number of solved pricing problems, the required CPU time to compute LB_S (in seconds) and the ratio of the CPU time of the *rough* version of the column generation procedure (i.e. where all the pricing problems are solved exactly) over the CPU time of the enhanced version.

From Table 3, we can see that the proposed column generation procedure requires an average computation time of 247.71 seconds, which is quite reasonable for obtaining such close gaps for hard benchmark instances. Actually, the required CPU time was less than one minute for 64% of the instances (27 out of 42). Moreover, we remark that the CPU tends to be higher when the number of vertices is larger. Indeed, Table 3 clearly shows that, for the same family, the number of solved pricing problems increases with

TABLE 3. Efficiency of the proposed column generation procedure.

Instances	|V|	|E|	Pricing	Time	Ratio
myciel3	11	20	21	0.10	1.84
myciel4	23	71	57	1.02	1.33
myciel5	47	236	250	14.91	1.11
myciel6	95	755	858	80.28	1.58
myciel7	191	2360	4107	991.14	1.17
miles250	128	387	2664	97.98	0.60
miles500	128	1170	1669	23.37	1.49
miles750	128	2113	1410	10.64	3.86
miles1000	128	3216	1791	12.96	3.93
miles1500	128	5198	775	5.73	7.94
queen5.5	25	160	45	0.16	2.44
queen6.6	36	290	55	0.90	4.05
queen7.7	49	476	88	1.22	3.40
queen8.8	64	728	177	4.42	3.47
queen9.9	81	2112	212	5.44	6.58
queen10.10	100	1470	229	11.79	5.02
queen11.11	121	1980	291	15.82	5.65
queen12.12	144	2596	641	31.38	4.93
queen13.13	169	3328	869	58.42	3.36
queen14.14	196	4186	1201	106.33	3.24
queen15.15	225	5180	26956	140.03	3.55
queen8.12	96	1368	144	2.02	8.34
anna	138	493	31369	389.99	0.36
david	87	406	2354	11.15	0.99
huck	74	301	267	1.19	2.83
jean	80	254	2895	10.29	0.89
games120	120	638	466	31.85	1.01
mulsol.i.1	197	3925	187934	1520.86	0.81
mulsol.i.2	188	3885	72637	594.57	1.19
mulsol.i.3	184	3916	65017	487.09	1.18
mulsol.i.4	185	3946	63950	511.38	1.17
mulsol.i.5	186	3973	72156	608.48	1.07
zeroin.3	206	3540	138660	1763.73	1.22
mug88-1	88	146	1309	5.48	6.87
mug88-25	88	146	1323	4.92	6.47
mug100-1	100	166	1553	7.62	6.33
mug100-25	100	166	1605	7.54	6.71
2-insert-3	37	72	493	2.99	2.69
3-insert-3	56	110	2357	21.32	2.34
DSJC125.1	125	736	1211	1723.28	1.16
DSJC125.5	125	3891	642	932.78	2.43
DSJC125.9	125	6961	390	151.40	12.29
Average	-	-	16502.33	247.71	3.31

the number of vertices. However, for a constant number of vertices, the number of solved pricing problems decreases when the number of edges increases. As it will be detailed in Table 4, this could be indeed explained by the fact the ‘‘Unattractiveness detection’’ procedure tends to be much more effective for a fixed number of vertices and an increasing number of edges. This can be clearly seen for instances *miles* and *DSJC*. It is worth noting that solving the edge-based and vertex-based formulations required an average CPU time of 0.12 and 0.03 seconds, respectively.

It is interesting to mention that the proposed enhancements have clearly contributed in reducing the overall computational effort. Indeed, embedding these enhancements enabled the column generation procedure to be, on average, 3.31 times faster than its rough version. It should be mentioned that the latter version was rarely faster than the enhanced one (5 cases out of 42). This is mainly due to the fact that the total computation time for solving the master restricted problems was longer for the enhanced version. Nevertheless, the total dedicated time for solving the pricing problems was much less for all the instances when the enhancements were implemented.

Pushing our analysis a step further, Table 4 depicts the impact of the different proposed enhancement procedures in terms of avoidance of the exact solution of the pricing problems. In this table, we provide for each instance:

- Unattractiveness detection: the percentage of pricing problems that are identified not to provide any attractive column without recourse to their exact resolution (i.e. thanks to Observations 2 and 3).
- Attractiveness checking: the percentage of times an attractive column has been generated without solving its corresponding pricing problem.
- Heuristic solution: the percentage of attractive columns that have been generated using the proposed heuristic procedure for the pricing problem (i.e. without resorting to the exact algorithm for the maximum independent set).
- Exact solution: the percentage of pricing problems that needed to be solved exactly by the optimization solver.

Table 4 provides strong evidence of the worth of implementing the proposed enhancements. Indeed, the need of an exact procedure for solving the pricing problem occured in only 16.99% of the cases, on average. This percentage could be as small as 2% in some instances such as *queen15.15* where 96.72% of the pricing problems have been avoided thanks to the ‘‘Unattractiveness detection’’ and ‘‘Attractiveness checking’’ procedures. The most difficult set of instances seems to be *mulsol* where the percentage of exactly solving the pricing problems was the highest one (61.41% on average). Moreover, it appears that the ‘‘Unattractiveness detection’’ procedure has the greatest impact since it served to identify unworthy pricing problems in 54.26% of the cases, on average. Interestingly, this percentage could reach high values such as 87.48% for one of the largest instances, namely *miles1500*. Recall that the lower bound of this 128 vertex-5198 edge instance has been obtained in only 5.73 seconds, and is proved to be optimal for the first time. The ‘‘attractiveness checking’’ procedure has the second impacting rank since it made feasible to detect attractive columns in an average of 22.30% of the cases without solving the pricing problem. It is worth noting that, in some cases, the ‘‘attractive checking’’ procedure was even more impacting than the ‘‘Unattractiveness detection’’ one. These include all instances of *mulsol* set and particularly *3-insert_3* instance where 54.56% of the generated attractive columns have been found using the ‘‘attractiveness checking’’ procedure. It is worth noting that even our weakest enhancement procedure, namely the ‘‘heuristic solution’’, was able to avoid the exact solution of the pricing problem in 41.5% of the cases where it has been called.

TABLE 4. Impact of the enhancement procedures.

Instances	|V|	|E|	Unattractiveness detection	Attractiveness checking	Heuristic solution	Exact solution
myciel3	11	20	47.62%	19.05%	13.33%	20.00%
myciel4	23	71	53.44%	17.20%	14.29%	15.08%
myciel5	47	236	48.70%	25.76%	11.41%	14.12%
myciel6	95	755	46.96%	30.94%	11.14%	10.96%
myciel7	191	2360	40.43%	41.60%	8.92%	9.05%
miles250	128	387	47.97%	31.15%	9.83%	11.05%
miles500	128	1170	66.97%	17.06%	6.13%	9.84%
miles750	128	2113	71.16%	14.45%	4.04%	10.36%
miles1000	128	3216	73.33%	11.71%	2.52%	12.44%
miles1500	128	5198	87.48%	2.50%	1.88%	8.14%
queen5.5	25	160	70.22%	10.89%	8.89%	10.00%
queen6.6	36	290	74.46%	11.86%	7.02%	6.66%
queen7.7	49	476	72.41%	16.50%	6.51%	4.58%
queen8.8	64	728	67.27%	19.95%	7.94%	4.84%
queen9.9	81	2112	67.16%	22.62%	6.76%	3.46%
queen10.10	100	1470	67.14%	25.37%	4.72%	2.78%
queen11.11	121	1980	64.51%	28.87%	4.52%	2.10%
queen12.12	144	2596	62.73%	30.03%	3.87%	3.37%
queen13.13	169	3328	59.68%	33.49%	3.68%	3.15%
queen14.14	196	4186	56.50	36.92%	3.62%	2.96%
queen15.15	225	5180	59.57%	37.15%	1.28%	2.00%
queen8.12	96	1368	73.72%	17.98%	6.11%	2.19%
anna	138	493	15.63%	31.31%	3.02%	50.03%
david	87	406	46.06%	20.21%	8.37%	25.36%
huck	74	301	68.70%	9.23%	15.30%	6.77%
jean	80	254	36.84%	26.61%	8.45%	28.10%
games120	120	638	63.16%	25.19%	8.02%	3.63%
mulsol.i.1	197	3925	10.08%	14.60%	0.82%	74.49%
mulsol.i.2	188	3885	15.86%	25.41%	0.62%	58.10%
mulsol.i.3	184	3916	15.90%	25.23%	0.59%	58.28%
mulsol.i.4	185	3946	17.10%	24.75%	0.68%	57.47%
mulsol.i.5	186	3973	15.94%	24.85%	0.52%	58.69%
zeroin.3	206	3540	12.18%	21.68%	0.68%	65.46%
mug88-1	88	146	76.55%	2.87%	13.39%	7.19%
mug88-25	88	146	76.70%	2.90%	13.16%	7.25%
mug100-1	100	166	76.31%	3.24%	12.83%	7.62%
mug100-25	100	166	77.16%	2.60%	13.70%	6.54%
2-insert-3	37	72	44.96%	42.14%	4.62%	8.29%
3-insert-3	56	110	34.51%	54.56%	2.66%	8.26%
DSJC125.1	125	736	47.95%	36.87%	8.82%	6.35%
DSJC125.5	125	3891	66.12%	26.18%	3.70%	4.00%
DSJC125.9	125	6961	81.99%	13.20%	2.17%	2.64%
Average	-	-	**54.26%**	**22.30%**	**6.44%**	**16.99%**

D. THEORETICAL VERSUS PRACTICAL PERFORMANCE OF THE PROPOSED PROCEDURE

It is worth noting that, due to the NP-hardness of the problem, the computational complexity of the proposed approach is not surprisingly exponential. Indeed, except Algorithm 3 and Algorithm 4, which run in $O(Kn)$ and $O(n)$ time, respectively, all the proposed algorithms run in exponential time. More precisely, the complexity of Algorithm 1 is $O(K2^n)$ since it may require generating all possible attractive columns. Algorithm 2 runs in $O(2^n)$ time since it requires solving the maximum independent set problem using a 0-1 mathematical formulation. Similarly, the complexity of Algorithm 5 is $O(2^n)$ since it is bounded by the exact solution of the maximum weighted independent set problem. Consequently, the whole approach requires a *theoretical* number of $O(K2^n)$ iterations. However, the proposed column generation procedure includes enough *smart* components that make it very efficient in practice. In our experimental study, we found that the proposed column generation procedure empirically requires a number of pricing problems which constitutes, on average, around 0.01% of the expected theoretical number. For the sake of illustration, consider the instance on which our column generation procedure required the largest CPU time, namely instance zeroin.3. The generated number of pricing problems is 138,660 which represents a fraction of about 10^{-58} of the corresponding theoretical number. Actually, this kind of exponential algorithms that have empirical performances much better than their theoretical complexities is very common in the combinatorial optimization community. One of the most famous examples of such algorithms may be the simplex method which, despite its exponential theoretical complexity and the existence of polynomial-time methods

TABLE 5. List of used symbols.

Symbols	Meanings
n	number of vertices
K	number of colors
$\chi(G)$	chromatic number
$s(G)$	strength of the graph G
δ_i	the neighbourhood of a vertex v_i
d_i	the degree of vertex v_i
P_V	the vertex-based formulation
P_E	the edge-based formulation
LP_E	the linear relaxation of (P_E)
P_S	the set partitioning formulation
Z_E^*	the optimal objective value of the linear relaxation of the edge-based formulation
Z_V^*	the optimal objective value of the linear relaxation of the vertex-based formulations
Z_S^*	the optimal objective value of the linear relaxation of the set partitioning formulation
R	the total number of independent sets in V
S_r	the r^{th} independent set
C_{kr}	the cost of coloring set S_r with color k
$\mathcal{C}$	the set of columns with negative reduced costs obtained after solving the pricing problems
(P$_k$)	the pricing problem
Z_k^*	the optimal objective value of (P_k)
A_k	a column associated to color k
σ	(maximum) weighted independent set

(such as the interior point method), is still implemented as the default LP solver by CPLEX - one of the most powerful optimization software [49].

VI. CONCLUSION

In the present paper, the minimum sum coloring problem is addressed. This problem commonly arises in many domains such as scheduling and resource allocation. A new tight lower bound based on a set partition formulation is shown to dominate the linear relaxation of both edge-based and vertex-based formulations. This lower bound is efficiently computed by using a column generation approach that is empowered by different procedures that substantially help avoiding the exact solution of the pricing problems. Experimental results conducted on hard benchmark instances show significant reductions of the gap between the best known lower and upper bounds, including 14 new optimality results out of 42 instances.

The obtained results strongly motivate future research to focus on designing a Branch-and-Price algorithm for the exact solution of the minimum sum coloring problem. Also, designing improved lower bounds for large real-life minimum sum coloring instances could be a challenging future topic. Another interesting practical application of the minimum sum coloring problem that is worth of future investigation could be the optimization of classroom scheduling where the weight of the color represents the faculty preferences.

APPENDIX
LIST OF USED SYMBOLS

See Table 5.

REFERENCES

[1] K. J. Supowit, "Finding a maximum planar subset of a set of nets in a channel," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. CAD-6, no. 1, pp. 93–94, Jan. 1987.

[2] A. Sen, H. Deng, and S. Guha, "On a graph partition problem with application to VLSI layout," *Inf. Process. Lett.*, vol. 43, no. 2, pp. 87–94, 1992.

[3] T. Szkaliczki, "Routing with minimum wire length in the dogleg-free manhattan model is $\mathcal{NP}$-complete," *SIAM J. Comput.*, vol. 29, no. 1, pp. 274–287, Jan. 1999.

[4] A. Bar-Noy, M. Bellare, M. M. Halldórsson, H. Shachnai, and T. Tamir, "On chromatic sums and distributed resource allocation," *Inf. Comput.*, vol. 140, no. 2, pp. 183–202, Feb. 1998.

[5] M. M. Halldórsson, G. Kortsarz, and H. Shachnai, "Sum coloring interval and k-claw free graphs with application to scheduling dependent jobs," *Algorithmica*, vol. 37, no. 3, pp. 187–209, Nov. 2003.

[6] F. Bonomo, G. Duran, A. Napoli, and M. Valencia-Pabon, "A one-to-one correspondence between potential solutions of the cluster deletion problem and the minimum sum coloring problem, and its application to P4-sparse graphs," *Inf. Process. Lett.*, vol. 115, nos. 6–8, pp. 600–603, Jun. 2015.

[7] L. G. Kroon, A. Sen, H. Deng, and A. Roy, "The optimal cost chromatic partition problem for trees and interval graphs," in *Proc. Int. Workshop Graph-Theoretic Concepts Comput. Sci.* Berlin, Germany: Springer, 1996, pp. 279—292.

[8] A. Bar-Noy, M. M. Halldórsson, G. Kortsarz, R. Salman, and H. Shanhnai, "Sum multi-coloring of graphs," in *Proc. Eur. Symp. Algorithms*. Berlin, Germany: Springer, 1999, pp. 390–401.

[9] A. Berliner, U. Bostelmann, R. A. Brualdi, and L. Deaett, "Sum list coloring graphs," *Graphs Combinatorics*, vol. 22, no. 2, pp. 173–183, Jun. 2006.

[10] D. S. Johnson, A. Mehrotra, and M. A. Trick, "Special issue on computational methods for graph coloring and its generalizations," *Discrete Appl. Math.*, vol. 156, no. 2, pp. 145–146, Jan. 2008.

[11] E. Kubicka and A. J. Schwenk, "An introduction to chromatic sums," in *Proc. 17th Conf. ACM Annu. Comput. Sci. Conf.*, 1989, pp. 39–45.

[12] K. Jansen, "Complexity results for the optimum cost chromatic partition problem," Tech. Rep., 1997.

[13] M. Malafiejski, K. Giaro, R. Janczewski, and M. Kubale, "Sum coloring of bipartite graphs with bounded degree," *Algorithmica*, vol. 40, no. 4, pp. 235–244, 2004.

[14] F. Bonomo and M. Valencia-Pabon, "Minimum sum coloring of P4-sparse graphs," *Electron. Notes Discrete Math.*, vol. 35, pp. 293–298, Dec. 2009.

[15] A. Bar-Noy and G. Kortsarz, "Minimum color sum of bipartite graphs," *J. Algorithms*, vol. 28, no. 2, pp. 339–365, Aug. 1998.

[16] Y. Li, C. Lucet, A. Moukrim, and K. Sghiouer, "Greedy algorithms for the minimum sum coloring problem," in *Proc. Logistique et Transp.*, 2009, p. LT-027.

[17] D. Brélaz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[18] F. T. Leighton, "A graph coloring algorithm for large scheduling problems," *J. Res. Nat. Bureau Standards*, vol. 84, no. 6, pp. 489–506, 1979.

[19] Q. Wu and J.-K. Hao, "An effective heuristic algorithm for sum coloring of graphs," *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1593–1600, Jul. 2012.

[20] H. Bouziri and M. Jouini, "A tabu search approach for the sum coloring problem," *Electron. Notes Discrete Math.*, vol. 36, pp. 915–922, Aug. 2010.

[21] A. Helmar and M. Chiarandini, "A local search heuristic for chromatic sum," in *Proc. 9th Metaheuristics Int. Conf.*, vol. 1101, 2011, pp. 161–170.

[22] U. Benlic and J.-K. Hao, "A study of breakout local search for the minimum sum coloring problem," in *Proc. Asia–Pacific Conf. Simulated Evol. Learn.* Berlin, Germany: Springer, 2012, pp. 128–137.

[23] A. Moukrim, K. Sghiouer, C. Lucet, and Y. Li, "Upper and lower bounds for the minimum sum coloring problem," 2014.

[24] Y. Jin, J.-K. Hao, and J.-P. Hamiez, "A memetic algorithm for the minimum sum coloring problem," *Comput. Oper. Res.*, vol. 43, pp. 318–327, Mar. 2014.

[25] Y. Jin and J.-K. Hao, "Hybrid evolutionary search for the minimum sum coloring problem of graphs," *Inf. Sci.*, vols. 352–353, pp. 15–34, Jul. 2016.

[26] M. J. Dinneen, A. Mahasinghe, and K. Liu, "Finding the chromatic sums of graphs using a D-Wave quantum computer," *J. Supercomput.*, vol. 75, no. 8, pp. 4811–4828, Aug. 2019.

[27] Z. Kokosiński and M. Bała, "Solving graph partitioning problems with parallel metaheuristics," in *Recent Advances in Computational Optimization*. Springer, 2018, pp. 89–105.

[28] O. Harrabi and J. C. Siala, "An effective parameter tuning for a bi-objective genetic algorithm to solve the sum coloring problem," in *Soft Computing for Problem Solving*. Singapore: Springer, 2019, pp. 107–119.

[29] Y. Wang, J.-K. Hao, F. Glover, and Z. Lü, "Solving the minimum sum coloring problem via binary quadratic programming," 2013, *arXiv:1304.5876*. [Online]. Available: http://arxiv.org/abs/1304.5876

[30] C. Lecat, C.-M. Li, C. Lucet, and Y. Li, "Exact methods for the minimum sum coloring problem," Tech. Rep., 2015.

[31] Y. Jin, J.-P. Hamiez, and J.-K. Hao, "Algorithms for the minimum sum coloring problem: A review," *Artif. Intell. Rev.*, vol. 47, no. 3, pp. 367–394, Mar. 2017.

[32] A. Mehrotra and M. A. Trick, "A column generation approach for graph coloring," *Informs J. Comput.*, vol. 8, no. 4, pp. 344–354, Nov. 1996.

[33] P. Hansen, M. Labbé, and D. Schindl, "Set covering and packing formulations of graph coloring: Algorithms and first polyhedral results," *Discrete Optim.*, vol. 6, no. 2, pp. 135–147, May 2009.

[34] E. Malaguti, M. Monaci, and P. Toth, "An exact approach for the Vertex Coloring Problem," *Discrete Optim.*, vol. 8, no. 2, pp. 174–190, May 2011.

[35] S. Gualandi and F. Malucelli, "Exact solution of graph coloring problems via constraint programming and column generation," *Informs J. Comput.*, vol. 24, no. 1, pp. 81–100, Feb. 2012.

[36] S. Held, W. Cook, and E. C. Sewell, "Maximum-weight stable sets and safe lower bounds for graph coloring," *Math. Prog. Comput.*, vol. 4, no. 4, pp. 363–381, Dec. 2012.

[37] C. Thomassen, P. Erdös, Y. Alavi, P. J. Malde, and A. J. Schwenk, "Tight bounds on the chromatic sum of a connected graph," *J. Graph Theory*, vol. 13, no. 3, pp. 353–357, Jul. 1989.

[38] Z. Kokosiński and K. Kwarciany, "On sum coloring of graphs with parallel genetic algorithms," in *Proc. Int. Conf. Adapt. Natural Comput. Algorithms*. Berlin, Germany: Springer, 2007, pp. 211–219.

[39] C. Lecat, C. Lucet, and C.-M. Li, "New lower bound for the minimum sum coloring problem," in *Proc. 31st AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, 2017, pp. 853–859.

[40] C. Lecat, C. Lucet, and C.-M. Li, "Sum coloring: New upper bounds for the chromatic strength," 2016, *arXiv:1609.02726*. [Online]. Available: http://arxiv.org/abs/1609.02726

[41] A. Moukrim, K. Sghiouer, C. Lucet, and Y. Li, "Lower bounds for the minimal sum coloring problem," *Electron. Notes Discrete Math.*, vol. 36, pp. 663–670, Aug. 2010.

[42] S. M. Douiri and S. Elbernoussi, "A new ant colony optimization algorithm for the lower bound of sum coloring problem," *J. Math. Modelling Algorithms*, vol. 11, no. 2, pp. 181–192, Jun. 2012.

[43] Q. Wu and J.-K. Hao, "Improved lower bounds for sum coloring via clique decomposition," 2013, *arXiv:1303.6761*. [Online]. Available: http://arxiv.org/abs/1303.6761

[44] F. Glover, "Tabu search—Part I," *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, 1989.

[45] Q. Wu, Q. Zhou, Y. Jin, and J.-K. Hao, "Minimum sum coloring for large graphs with extraction and backward expansion search," *Appl. Soft Comput.*, vol. 62, pp. 1056–1065, Jan. 2018.

[46] H. D. Sherali and W. P. Adams, *A Reformulation-Linearization Technique for Solving Discrete Continuous Nonconvex Problems*, vol. 31. Springer, 2013.

[47] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. Boston, MA, USA: Springer, 1972, pp. 85–103.

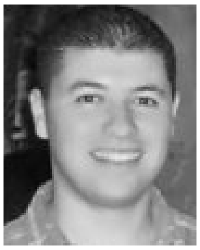
[48] C. Lecat, C. Lucet, and C.-M. Li, "Minimum sum coloring problem: Upper bounds for the chromatic strength," *Discrete Appl. Math.*, vol. 233, pp. 71–82, Dec. 2017.

[49] *Deciding Which CPLEX's Numerous Linear Programming Algorithms for Fastest Performance*. Accessed: Apr. 14, 2019. [Online]. Available: https://www-01.ibm.com/support/docview.wss?uid=swg21399934

MEHDI MRAD received the Ph.D. degree in operations research from the University of Tunis. He is currently an Associate Professor with the Department of Industrial Engineering, King Saud University. His interests include network design, vehicle routing, project management, and scheduling and cutting problems. He is attracted by the application of different optimization techniques to model and solve real-life complex problems from different engineering fields.

OLFA HARRABI received the Ph.D. degree in computer science from the University of Tunis, Tunisia, in 2019. She is currently an Assistant of computer Science with the Faculty of Sciences of Tunis (FST) Tunis, Manar University. Her main research interests include artificial intelligence, operations research, optimization, metaheuristics, and decision support systems.

JOUHAINA CHAOUACHI SIALA received the Ph.D. degree in computer science from the High Institute of Management (ISG) Tunis, University of Tunis. Since 1998, she has been continuing in the academic streamline. She is currently a Full Professor with the Quantitative Methods Department, Business School of Carthage (IHEC), University of Carthage. She has published a number of articles in international journals and conferences. Her areas of interests include operations research, metaheuristics, transportation problems, vehicle routing problems, and decision support systems.



ANIS GHARBI received the B.Sc. degree in mathematics from the Faculty of Science of Tunis, University of Tunis, the master's degree in operations research from the High Institute of Management of Tunis, University of Tunis, the Ph.D. degree in management science from the High Institute of Management of Tunis, University of Tunis, and the Habilitation degree in management science from the High Institute of Management, University of Tunis. He is currently a Professor of operations research with the Industrial Engineering Department, College of Engineering, King Saud University, Saudi Arabia. His research interest focuses on the design, implementation, and analysis of optimization algorithms for hard combinatorial problems, with applications on machine scheduling and bin packing. He has conducted his research within the Combinatorial Optimization Research Group-ROI, Polytechnic School of Tunisia.