Exact methods for the minimum sum coloring problem

Clément Lecat, Chu Min Li, Corinne Lucet, Yu Li

Exact methods for the minimum sum coloring problem

Clément Lecat (student), Chu-Min Li (advisor), Corinne Lucet (advisor), Yu Li

University of Picardie Jules Verne, MIS Laboratory
33 rue St Leu, 80000 Amiens, France

{clement.lecat, chu-min.li, corinne.lucet, yu.li}@u-picardie.fr

Abstract. The Minimum Sum Coloring Problem (*MSCP*) is an extension of the well known Graph Coloring Problem (*GCP*). Both NP-hard problems consist in assigning a color to each vertex of a graph while respecting the neighborhood constraints: two adjacent vertices cannot share a same color. For *MSCP*, a weight is associated with each color. While the objective of *GCP* is *to minimize the number of colors*, the objective of *MSCP* is *to minimize the sum of the color cost*. In this paper, we propose and compare four exact methods to solve *MSCP*. The first one is a Branch-and-Bound method which uses upper and lower bounds to reduce search space. The second, the third and the fourth methods encode a *MSCP* instance into a MaxSAT, MinSAT and COP instance respectively, and solve the instance with dedicated solvers.

1 Introduction

The Graph Coloring Problem (*GCP*) is an important NP-hard combinatorial problem. A lot of effort has been devoted to study it. Two main types of algorithms (also called solvers) are developed for solving *GCP*: exact methods and approximate methods. The exact methods aim at finding an optimal solution of the problem, including the approaches based on branch-and-bound schema (*e.g.* [20]), on graph decomposition (*e.g.* [18]), and on SAT solving by encoding the problem into an equivalent propositional formula. The approximate methods aim at finding an upper bound or a lower bound of the optimal solution of the problem, including: greedy algorithms such as the famous DSATUR [5], and various heuristic or meta-heuristic algorithms [22, 7, 17]. In the literature, these methods are usually evaluated on standard benchmarks such as DIMACS and COLOR [6, 9].

The Minimum Sum Coloring Problem *MSCP* is derived from *GCP* and is introduced in 1989 by Kubicka et Schwenk [12]. The main results on *MSCP* include the theoretical bounds [10, 1, 19, 3] and the structural properties relative to the graph families for which efficient algorithms exist. Recently, heuristics and meta-heuristics are proposed in [16] and [23, 11, 4, 21] respectively, giving bounds for DIMACS and COLOR graphs.

In this paper, we focus essentially on exact methods for *MSCP*. Concretely, we propose and analyze 4 exact methods for *MSCP*: a branch-and-bound algorithm for *MSCP* using relevant upper and lower bounds, and an encoding of *MSCP* into MaxSAT, MinSAT and COP (Constraint Optimization Problem), respectively. The 4 methods are evaluated and analyzed using a set of DIMACS and COLOR graphs and random graphs.

This paper is organized as follows. In Section 2, we formally define the *GCP* and *MSCP* problems. In Section 3, we propose a branch-and-bound algorithm for *MSCP* called BBMSCP. In Section 4, we propose an encoding of *MSCP* to MaxSAT, MinSAT and COP respectively. In Section 5, we compare these 4 exact methods on a set of DIMACS and COLOR graphs and random graphs.

2 The Minimum Sum Coloring problem

2.1 Preliminary definitions

We consider an undirected graph *G=(V,E)*, where *V* is the set of vertices ($|V| = n$) and $E \subseteq V \times V$ the set of edges ($|E| = m$). The neighbourhood $\mathcal{N}(v)$ of a vertex $v \in V$ is defined as : $\mathcal{N}(v) = \{u \in V \mid (v,u) \in E\}$. The degree of a vertex *v* is the number of neighbors of *v*, denoted as $\delta(v)$. The degree of a graph *G* is $\Delta(G) = max\{\delta(v); \forall v \in V\}$. A subgraph $G' = (V', E')$ induced from *G=(V,E)* is a graph so that $V' \subseteq V$ and $\forall (u,v) \in E$ if $u \in V'$ and $v \in V'$ then $(u,v) \in E'$. A clique $C = (V', E')$ of a graph *G* is a complet subgraph of G, for which if $u \in V'$ and if $v \in V'$ then $(u,v) \in E'$. $G \backslash v$ represents the subgraph of G obtained by removing the vertex v from G.

2.2 Coloring and sum coloring

A coloring of a graph *G* is a function $c : V \mapsto \{1, 2, \ldots, k\}$ that assigns to each vertex $v \in V$ a color $c(v)$ represented by an integer. A coloring is said to be valid, if $\forall (u,v) \in E,\ c(u) \neq c(v)$. We denote a coloring of *G* as $X = X_1, X_2, \ldots, X_k$, where $X_i = \{v \in V \mid c(v) = i\}$ is called color class i ($1 \leq i \leq k$). *GCP* consists in finding a valid coloring with the minimum number of colors. This number is called *chromatic number* of *G*, denoted by $\chi(G)$. Note that a clique of n' vertices requires n' colors. All colors in a valid coloring of G are symmetric, because exchanging any two colors gives a valid coloring.

For *MSCP*, each color i is associated with a cost p_i. For a given coloring X of *G*, the corresponding sum coloring $P(X)$ equals to the sum of costs of colors used by X.

$$P(X) = p_1 \times |X_1| + p_2 \times |X_2| + \ldots + p_k \times |X_k|$$

MSCP consists in finding a valid coloring of G such that $P(X)$ is minimum. This minimum sum is called the *chromatic sum* of *G* and is denoted by $\Sigma(G)$.

$$\Sigma(G) = min\{P(X) \mid X \ is \ a \ valid \ coloring \ of \ G\}$$

Kubicka et Schwenk [12] proved that *MSCP* is *NP-hard*. The minimum number of colors used to color *G* in an optimal solution for *MSCP* is called *strength* of *G* and is denoted by $s(G)$. In our study, as in most work in the literature, we consider that : $\forall i,\ p_i = i$. We obviously have:

Propertie 1 *Let $X^\theta = X_1, X_2, \ldots, X_k$ be a valid coloring, where $|X_1| \geq |X_2| \geq \ldots \geq |X_k|$ is verified, and X be any valid coloring that can be obtained by exchanging colors of X^θ, then $P(X^\theta) \leq P(X)$. X^θ is called* the dominant coloring *of its symmetric class.*

It is easy to find the dominant coloring from any coloring of G by just ordering the color classes in the descending order of their cardinality. So, we will always report the dominant coloring X^θ when evaluating a solution.

3 Branch-and-Bound for MSCP

In this section, we propose a Branch-and-Bound algorithm that finds an optimal solution of a MSCP instance G by developing a search tree, as illustrated in Figure 2 for the coloring of the graph in Figure 1. Each node of the tree corresponds to a vertex of the graph (a, b or c), and each branch corresponds to an assignment of a color (1, 2 or 3) to a vertex. A path of n nodes from the root to a leaf corresponds a valid coloring X whose associated weight is the weight of the dominant coloring X^θ.

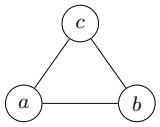

Fig. 1: A simple graph

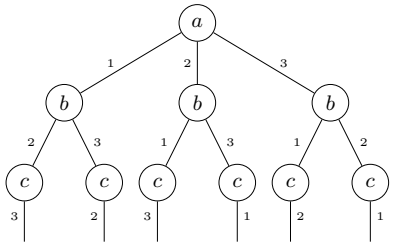

Fig. 2: A search tree for the graph of Figure 1 with 3 colors

The algorithm maintains a global variable UB, which is initialized with $\text{UB}_{MDSATUR}$, computed with the dedicated greedy algorithm MDSATUR [16]. UB represents the best dominant sum coloring found so far in G. As the strength $s(G)$ is always smaller than or equal to $\Delta(G)$+1, the number of colors is fixed to $\Delta(G)$+1. At each node of the search tree, the algorithm computes a lower bound LB which is equal to the sum of the colors assigned to the vertices already colored and an underestimation of the sum of colors that will be assigned to the remaining vertices. If LB$\geq$UB, a better solution obviously cannot be obtained from the node, search is then pruned. Otherwise, let v_i be the vertex corresponding to the search tree node. A branch is developed by assigning each available color to v_i (a color is available for v_i if it is not yet assigned to any neighbor of v_i).

The performance of the algorithm crucially depends on the quality of LB. At the beginning, LB takes the value of the theoretical lower bound $\lceil\sqrt{8m}\rceil$. Then at a search tree node corresponding to vertex v_i ($1\leq i\leq n$), assume that vertices v_1, v_2, ..., v_{i-1} are already assigned a colors $c(v_t)$ ($1\leq t<i$), we propose to compute LB as follows. The uncolored subgraph G' is partitioned into cliques $C_1, C_2, \ldots, C_l$. Let $Disp(v)$ denote the set of available colors for vertex v. The set of available colors of a clique C is defined to be $Disp(C) = \bigcup_{v\in C}\{Disp(v)\}$, and the sum of the $|C|$ smallest colors of clique C is

Algorithm 1: LB(G)

```
1  begin
2      LB← 0;
3      Partition G into cliques C_1, C_2, ..., C_l;
4      foreach C_i do
5          Disp(C_i) ← ⋃_{v∈C_i} Disp(v);
6          foreach x := 1 to |C_i| do
7              j_min ← MIN{j | j ∈ Disp(C_i)}; //the minimum color of C_i
8              LB← LB+j_min;
9              Disp(C_i) ← Disp(C_i) \ {j_min};
10     return LB;
```

denoted by $P_{lb}(C)$. Then we propose LB(G')=$\sum_{t=1}^{l} P_{lb}(C_t)$. Algorithm 1 illustrates the computation of LB for an uncolored graph.

The recursive algorithm BBMSCP (see Algorithm 2) brings together and integrates all the points developed before. Parameter G is the subgraph not yet colored, σ is the sum of assigned colors, and UB the best sum coloring found so far. The first call is BBMSCP(G, 0, UB$_{MDSATUR}$) to compute the optimal sum coloring of G, each vertex of G having $\Delta(G) + 1$ available colors at the beginning.

Algorithm 2: BBMSCP(G, σ, UB)

```
   Input  : G = (V, E), the sum of colors already assigned σ, the best sum coloring UB
            found so far
   Output : Σ(G) chromatic sum
1  begin
2      if V = ∅ then
3          return σ;
4      if LB(G)+σ <UB then
5          return UB;
6      Choose a vertex v;
7      foreach i ∈ Disp(v) such that σ + i ≤ UB do
8          foreach u ∈ N(v) do
9              Disp(u) := Disp(u) \ {i};
10         UB:=BBMSCP(G\v, σ+i, UB);
11         foreach u ∈ N(v) do
12             Disp(u) := Disp(u) ∪ {i};
13     return UB;
```

4 Solving MSCP with MaxSAT, MinSAT and COP

MSCP can be encoded into MaxSAT (MinSAT, COP) and then solved using a MaxSAT (MinSAT, COP) solver. In this section, we present the MaxSAT, MinSAT and COP problems and our encodings of *MSCP* into them.

4.1 MaxSAT, MinSAT and COP

A literal is a propositional variable x or its negation $\bar{x}$. A literal x ($\bar{x}$) is satisfied if the variable x is assigned the truth value 1 (0). A clause is a disjunction of literals that can be soft or hard. A clause is satisfied if one of its literals is satisfied. Given a set of hard and soft clauses $\{h_1, ..., h_q, s_1, ..., s_r\}$ (i.e., a CNF formula), where each h_i ($1 \leq i \leq q$) is a hard clause, and each s_i ($1 \leq i \leq r$) is a soft clause associated with a weight, the weighted partial MaxSAT (MinSAT) problem is to find an assignment of truth values to propositional variables that satisfies all the hard clauses and maximizes (minimizes) the sum of weights of satisfied soft clauses. In the sequel, MaxSAT and MinSAT are always weighted partial.

A Constraint Satisfaction Problem (CSP) is defined by a triplet (*X*, *D*, *C*), where *X* is a set of variables $\{x_1, x_2, ..., x_n\}$, *D* a set of finite domains D_{x_i} ($1 \leq i \leq n$) of the variables, and *C* a set of constraints $\{c_1, c_2, ..., c_m\}$. A Constraint Optimisation Problem (COP) is an extension of CSP with an objective function $F : D_{x_1} \times D_{x_2} \times ... \times D_{x_n} \rightarrow \mathbb{N}$. A COP problem ($X$, D, C, F) is to assign a value to each variable in its domain to satisfy all the constraints and maximize (or minimize) F.

4.2 Encoding MSCP

MSCP to SAT. In order to encode a *MSCP* problem into a MaxSAT or a MinSAT problem, we define a boolean variable x_{ai} for each vertex a and each color i, which takes the value 1 if and only if vertex a is colored with the color i. The hard clauses of MaxSAT and MinSAT are the same for *MSCP*, corresponding to the constraints of the *GCP* problem with k colors:

- Each vertex a should be assigned a color : $x_{a1} \vee x_{a2} \vee ... \vee x_{ak}$
- Each vertex a is assigned at most one color : $\neg x_{ai} \vee \neg x_{aj}$ for each $i \neq j$
- Each two adjacent vertices a and b cannot be assigned the same color : $\neg x_{ai} \vee \neg x_{bi}$ for each color i

For MaxSAT, we define a unit soft clause x_{ai} for each vertex a and each color i with weight k+1-i. The satisfaction of a soft clause x_{ai} in MaxSAT means to assign color i to vertex a. For MinSAT, we define a unit soft clause $\neg x_{ai}$ for each vertex a and each color i also with weight k+1-i. The falsification of a soft clause $\neg x_{ai}$ in MinSAT means to assign color i to vertex a. Since the objective of MaxSAT (MinSAT) is to maximize (minimize) the sum of weights of satisfied soft clauses, soft clauses for smaller colors are satisfied (falsified) in priority in MaxSAT (MinSAT) whenever possible.

Given a graph G, the space complexity of the encoding of *MSCP* into MaxSAT or MinSAT is $\mathcal{O}(n \times k^2 + m \times k)$, where k is equal to $\Delta(G) + 1$. The number of soft clauses in both cases is $(n \times k)$.

Example 1 *Refer to the graph in Figure 1, k=$\Delta(G)+1$=3. The encoding of MSCP into MaxSAT and MinSAT consists of:*

Hard clauses*:* $x_{a1} \vee x_{a2} \vee x_{a3}$, $\neg x_{a1} \vee \neg x_{a2}$, $\neg x_{a1} \vee \neg x_{a3}$, $\neg x_{a2} \vee \neg x_{a3}$, ..., $\neg x_{a1} \vee \neg x_{b1}$, $\neg x_{a2} \vee \neg x_{b2}$, $\neg x_{a3} \vee \neg x_{b3}$, ..., $\neg x_{a1} \vee \neg x_{c1}$, $\neg x_{a2} \vee \neg x_{c2}$, $\neg x_{a3} \vee \neg x_{c3}$, ...

and soft clauses with their weight*:*

$MaxSAT\ encoding\ 1$	$MinSAT\ encoding\ 1$
$x_{a1}\ 3$	$\neg x_{a1}\ 3$
$x_{a2}\ 2$	$\neg x_{a2}\ 2$
$x_{a3}\ 1$	$\neg x_{a3}\ 1$
$x_{b1}\ 3$	$\neg x_{b1}\ 3$
$x_{b2}\ 2$	$\neg x_{b2}\ 2$
$x_{b3}\ 1$	$\neg x_{b3}\ 1$
$x_{c1}\ 3$	$\neg x_{c1}\ 3$
$x_{c2}\ 2$	$\neg x_{c2}\ 2$
$x_{c3}\ 1$	$\neg x_{c3}\ 1$

A dual MaxSAT (MinSAT) encoding for *MSCP* is to define a soft clause $\neg x_{ai}$ (x_{ai}) for each vertex a and each color i with weight i, as illustrated by the following example for the graph in Figure 1.

Example 2

$MaxSAT\ encoding\ 2$	$MinSAT\ encoding\ 2$
$soft\ clauses: \neg x_{a1}\ 1$	$soft\ clauses: x_{a1}\ 1$
$\neg x_{a2}\ 2$	$x_{a2}\ 2$
$\neg x_{a3}\ 3$	$x_{a3}\ 3$
$\neg x_{b1}\ 1$	$x_{b1}\ 1$
$\neg x_{b2}\ 2$	$x_{b2}\ 2$
$\neg x_{b3}\ 3$	$x_{b3}\ 3$
$\neg x_{c1}\ 1$	$x_{c1}\ 1$
$\neg x_{c2}\ 2$	$x_{c2}\ 2$
$\neg x_{c3}\ 3$	$x_{c3}\ 3$

The MaxSAT encoding 1 and the MinSAT encoding 2 appear to be more natural. However, they are surprisingly less efficient, as we will see later.

MSCP to COP. For encoding a *MSCP* instance into a COP instance, we define a variable v_i for each vertex v_i with domain $\{1,2,...,k\}$. Then we define a constraint c_{ij} : $v_i \neq v_j$ for each edge $(v_i, v_j) \in E$. The objective function to minimize is $F = \sum_{i=1}^{n} v_i$. The space complexity of this encoding is $\mathcal{O}(n+m)$.

5 Experimental analysis

We compare the 4 exact methods for *MSCP*: BBMSCP, the Branch-and-Bound algorithm proposed in Section 3; MaxSatz [14] (version 2009) and ISAC [2] (version 2013), two representative MaxSAT solvers, using the two MaxSAT encodings proposed in Section 4; MinSatz [15] (version 2013), the only MinSAT solver available to us, using the two MinSAT encodings proposed in Section 4; and Choco3 [8] (release 3.3.0), one of the best COP solver, using the COP encoding, by solving a set of random graphs and COLOR and DIMACS graphs. A random graph of density d is generated by adding an edge (u, v) with probability d for any pair of vertices u and v. The experimental results are obtained on a processor Intel Westmere Xeon E7-8837 (2.66GHz) under Linux.

Table 1 compares MaxSAT (MinSAT) encodings 1 and 2. For each graph G, we report the number of vertices (n), the number of edges (m), the chromatic sum $\Sigma(G)$, and the runtime of each solver using each of the two possible encodings to find $\Sigma(G)$. If a solver cannot find $\Sigma(G)$ in one hour, its runtime is marked by N/A. The reported results clearly show that MaxSAT encoding 2 (MinSAT encoding 1) is significantly better than MaxSAT encoding 1 (MinSAT encoding 2). Note that the soft clauses in MaxSAT encoding 2 and MinSAT encoding 1 are all negative, contrarily to the two other encodings. An explanation of their performance is the following. Since there are many negative hard binary clauses such as $\neg u \vee \neg v$, when the soft clauses are negative and unit, many pairs of them such as $\neg u$ and $\neg v$ cannot be falsified simultaneously, because the corresponding hard clause would be falsified.

				MinSatz		MaxSatz		ISAC	
				Encoding 1	Encoding 2	Encoding 1	Encoding 2	Encoding 1	Encoding 2
Graphs	n	m	$\Sigma(G)$	Time	Time	Time	Time	Time	Time
S10A23	10	23	20	0	0	0	0	0	0
S10A40	10	40	34	0	15	0	0	0	0
S10A5	10	5	14	0	0	0	0	0	0
S20A100	20	100	55	172	N/A	N/A	126	N/A	0
S20A19	20	19	27	0	2213	0	0	0	0
S20A50	20	50	39	0	N/A	8	0	0	0
S20A75	20	75	52	18	N/A	1372	144	N/A	0
S20A95	20	95	50	32	N/A	N/A	29	N/A	0
S25A100	25	100	63	150	N/A	N/A	1051	N/A	0
S25A30	25	30	40	0	N/A	0	0	0	0
S30A100	30	100	65	22	N/A	N/A	242	N/A	0
S30A44	30	44	48	0	N/A	3098	0	N/A	0
S35A60	35	60	60	0	N/A	N/A	2	N/A	0

Table 1: Runtimes in seconds of MaxSAT and MinSAT solvers using two encodings

Table 2 and Table 3 compare different solvers on a set of random and DIMACS and COLOR graphs. MaxSatz and ISAC use MaxSAT encoding 2, and MinSatz uses MinSAT encoding 1. ISAC is the best solver, followed by MinSatz, MaxSatz and BBMCP. The COP formalism does not appear so good for *MSCP*, although the COP encoding

				BBMSCP	MaxSatz	ISAC	MinSatz	Choco3
Graphs	n	m	$\Sigma(G)$	Time	Time	Time	Time	Time
S10A23	10	23	20	0	0	0	0	30
S10A40	10	40	34	0	0	0	0	48
S10A5	10	5	14	0	0	0	0	3
S20A100	20	100	55	10	126	0	172	N/A
S20A19	20	19	27	0	0	0	0	2
S20A50	20	50	39	0	0	0	0	74
S20A75	20	75	52	8	144	0	18	N/A
S20A95	20	95	50	3	29	0	32	307
S25A100	25	100	63	96	1051	0	150	N/A
S25A30	25	30	40	0	0	0	0	25
S30A100	30	100	65	105	242	0	22	N/A
S30A44	30	44	48	0	0	0	0	1551
S35A60	35	60	60	54	2	0	0	N/A

Table 2: Runtimes in seconds of different exact methods for a set of random graphs

has linear space complexity. It is worth noting that BBMCP is a preliminary implementation for *MSCP* remaining to be improved and optimized, while MinSatz and MaxSatz are mature solvers, and ISAC is a portfolio solver which selects automatically a state-of-the-art MaxSAT solver to run according to the characteristics of the instance to solve.

				BBMSCP	MaxSatz	ISAC	MinSatz	Choco3
Graphs	n	m	$\Sigma(G)$	Time	Time	Time	Time	Time
1FI3	30	100	54	2	9	0	2	707
miles250.col	128	387	334	N/A	2	0	0	N/A
miles500.col	128	1170	715	N/A	20	0	0	N/A
myciel3.col	11	20	21	0	0	0	0	3
myciel4.col	23	71	45	0	9	0	0	1944
queen5_5.col	25	160	75	3036	N/A	N/A	N/A	N/A

Table 3: Runtimes in seconds of different exact methods for a set of DIMACS and COLOR graphs

6 Conclusions and future work

In this paper, we have proposed and compared 4 exact methods for *MSCP*. The first method is a branch-and-bound algorithm called BBMSCP with promising results. To our knowledge, BBMSCP is the first exact solver dedicated for *MSCP*. We have also proposed 2 encodings of *MSCP* to MaxSAT and MinSAT respectively, and show that the encodings with negative soft clauses are substantially better than the encodings with positive soft clauses, making MaxSAT and MinSAT formalisms currently more suitable than the COP formalism for *MSCP*. The future work includes the improvement of BBMSCP by developing new bounds and symmetry breaking, the understanding of the performance of MinSAT or MaxSAT formalisms compared with the COP formalism, and adaptation of MinSAT and MaxSAT solvers for *MSCP*.

References

1. Y. Alav, P.J. Malde, and A.J. Schwenk. Tight bounds on the chromatic sum of a connected graph. *Journal of Graph Theory*, 13:353–357, 1989.
2. C. Ansótegui, Y. Malitsky, and M. Sellmann. Maxsat by improved instance-specific algorithm configuration. *Association for the Advancement of Artificial Intelligence*.
3. A. Bar-Noy and G. Kortsarz. Minimum color sum of bipartite graphs. *J. Algolrithms*, 28(2):339–365, 1998.
4. U. Benlic and J.K. Hao. A study of breakout local search for the minimum sum coloring problem. *SEAL 2012, Lecture Notes in Computer Science*, 7673:128–137, 2012.
5. D. Brélaz. New methods to color vertices of a graph. *Commun. ACM*, 22(4):251–256, 1979.
6. ftp://dimacs.rutgers.edu/pub/challenge/graph/ benchmarks/color/.
7. A. Hertz, M. Plumettaz, and N. Zufferey. Variable space search for graph coloring. *Discrete Applied Mathematics*, 156(13):2551 – 2560, 2008.
8. http://choco solver.org.
9. http://mat.gsia.cmu.edu/COLOR/instances.html.
10. K. Jansen. The optimum cost chromatic partition problem. In *CIAC*, pages 25–36, 1997.
11. Y. Jin, J.K. Hao, and J.P. Hamiez. A memetic algorithm for the minimum sum coloring problem. *Computers & OR*, 43:318–327, 2014.
12. E. Kubicka and A.J. Schwenk. An introduction to chromatic sums. In *CSC'89 : Proceedings of the 17th conference on ACM Annual Computer Science Conference*, pages 39–45, New York, NY, USA, 1989. ACM.
13. C.M. LI, F. Manya, and J. Planes. Exploiting unit propagation to compute lower bounds in branch and bound maxsat solvers. In *CP 05 : Proceedings of the 11st international conference on Principles and Practice of Constraint Programming*, pages 403–414, Sitges, SPAIN, 2005.
14. C.M. Li, F. Many, and J. Planes. New inference rules for max-sat. *Journal Of Arteficial Intetelligence Research*, 30:321–359, 2007.
15. C.M. Li, Z. Zhu, F. Many, and L. Simon. Minimum satisfiability and its applications. *IJCAI*, 2011.
16. Y. Li, C. Lucet, A. Moukrim, and K. Sghiouer. Greedy algorithms for the minimum sum coloring problem. In *International Workshop: Logistics and transport*, 2009.
17. Z. Lü and J.K. Hao. A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203(1):241 – 250, 2010.
18. C. Lucet, F. Mendes, and A. Moukrim. An exact method for graph coloring. *Computers & Operations Research*, 33(8):2189–2207, 2006.
19. M. Malafiejski. *Sum coloring of graphs*, chapter 4, pages 55–66. Graph Coloring. New-York (USA), 2004.
20. I. Méndez-Díaz and P. Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5):826–847, 2006.
21. A. Moukrim, K. Sghiouer, C. Lucet, and Y. Li. Lower bounds for the minimum sum coloring problem. *Electronic Notes in Discrete Mathematics*, 36:663–670, 2010.
22. D.C. Porumbel, J.K. Hao, and P. Kuntz. A search space "cartography" for guiding graph coloring heuristics. *Comput. Oper. Res.*, 37:769–778, April 2010.
23. K. Sghiouer, Y. Li, C. Lucet, and A. Moukrim. A memetic algorithm for the minimum sum coloring problem. In *Confrence Internationale en Recherche Oprationnelle*, 2010.