Algorithms for the minimum sum coloring problem: a review

Yan Jin∗1,2, Jean-Philippe Hamiez†2, and Jin-Kao Hao‡2,3

1School of Computer Science and Technology, Huazhong University of
Science and Technology, No. 1037, Luoyu Road, Wuhan, China
2LERIA, Université d’Angers, 2 Bd. Lavoisier, 49045 Angers, France
3Institut Universitaire de France, 1 Rue Descartes, 75231 Paris, France

Abstract

The Minimum Sum Coloring Problem (MSCP) is a variant of the well-known vertex coloring problem which has a number of AI related applications. Due to its theoretical and practical relevance, MSCP attracts increasing attention. The only existing review on the problem dates back to 2004 and mainly covers the history of MSCP and theoretical developments on specific graphs. In recent years, the field has witnessed significant progresses on approximationalgorithms and practical solution algorithms. The purpose of this review is to provide a comprehensive inspection of the most recent and representative MSCP algorithms. To be informative, we identify the general framework followed by practical solution algorithms and the key ingredients that make them successful. By classifying the main search strategies and putting forward the critical elements of the reviewed methods, we wish to encourage future development of more powerful methods and motivate new applications.

Keywords: Sum coloring, Approximation algorithms, Heuristics and meta-heuristics, Local search, Evolutionary algorithms.

1 Introduction

Given a graph G, a proper k-coloring of G is an assignment of k different colors {1, . . . , k} to the vertices of G such that two adjacent vertices receive two different colors. The classical graph vertex coloring problem (GCP) is to find a proper (or legal) k-coloring with the minimum number of colors χ(G) (i.e., the chromatic number of G) for a general graph G. The minimum sum coloring problem (MSCP) is a variant of the GCP and aims to determine a proper k-coloring while minimizing the sum of the colors assigned to the vertices. MSCP was proposed by Kubicka [29] in the field of graph theory and by Supowit [43] in the field of VLSI design. MSCP has applications in VLSI
design, scheduling and resource allocation for instance [1,6,28,36,42]. MSCP is also related to other generalizations or variants of GCP like sum multi-coloring [2], sum list coloring [5] and bandwidth coloring [25].

Like the classical vertex coloring problem, MSCP is notable for its practical applicability and theoretical intractability. Indeed, in the general case, the decision version of MSCP is NP-complete [28,29] and approximating the minimum color sum within an additive constant factor is NP-hard [32]. As a result, MSCP is a computationally challenging problem and any algorithm able to determine the optimal solution of the problem is expected to require an exponential complexity. Due to its high computational complexity, polynomial-time algorithms exist only for some special cases of the problem (see Section 3) and solving the problem in the general case remains an imposing challenge.

In the past several decades, much effort has been devoted to developing various approximation algorithms and practical solution algorithms. Approximation algorithms aim to provide solutions of provable quality while practical solution algorithms try to find sub-optimal solutions as good as possible within a bounded and acceptable computation time. The class of heuristic and metaheuristic algorithms has been mainly developed since 2009 and has enlarged our capacity of finding improved solutions on the benchmark graphs. Representative examples of the existing heuristic algorithms include greedy algorithms [35,38], tabu search [8], breakout local search [4], iterated local search [19], ant colony [12], genetic and memetic algorithms [11,23,24,26,39,45] as well as heuristics based on independent set extraction [46,48].

To the best of our knowledge, there is only one review published one decade ago in 2004 [30] that focuses on polynomial-time algorithms for specific graphs, MSCP generalizations (or variants) and applications. For the purpose of solving MSCP, the first studies essentially concerned the development of approximation algorithms and simple greedy algorithms. Research on practical solution algorithms of MSCP was relatively new and appeared around 2009. Nevertheless, important progresses have been made since that time. The purpose of this paper is thus to provide a comprehensive review of the most recent and representative MSCP algorithms. To be informative, we identify the general framework followed by the existing heuristic and metaheuristic algorithms and their key ingredients that make them successful. By classifying the main search strategies and putting forward the critical elements of the reviewed methods, we wish to encourage future development of more powerful methods and motivate new applications.

In the following sections, we first provide a general definition of MSCP, then a brief introduction of approximation algorithms in Section 3 followed by the presentation of the studied heuristics and metaheuristics in Section 4. Section 5 presents lower and upper bounds. Before concluding, Section 6 introduces MSCP benchmark instances and summarizes the computational results reported by the best performing algorithms on these instances.

2 Definitions and formulation of MSCP

Let $G = (V,E)$ be a simple undirected graph with vertex set $V = \{v_1,\ldots,v_n\}$ and edge set $E \subset V \times V$. A proper $k$-coloring $c$ of $G$ is a mapping $c : V \rightarrow \{1,\ldots,k\}$ such that $c(v_i) \neq c(v_j), \forall \{v_i,v_j\} \in E$. Equivalently, a proper $k$-coloring can be defined as a partition of $V$ into $k$ mutually disjoint independent sets (or color classes) $V_1,\ldots,V_k$ such that $\forall u,v \in V_i (i = 1,\ldots,k), \{u,v\} \notin E$. The objective of MSCP is to find a proper
$k$-coloring $c$ with a minimum sum of the colors that are assigned to the vertices of $V$. The minimum sum of colors for MSCP is called the chromatic sum of $G$, and is denoted by $\sum(G)$. The strength $s(G)$ of a graph $G$ is the smallest number of colors over all optimal sum colorings of $G$. Obviously, the chromatic number $\chi(G)$ of $G$ from the classical vertex coloring problem is a lower bound of $s(G)$, i.e., $\chi(G) \leq s(G)$.

Let $\mathcal{C}(G)$ be the set of all proper $k$-coloring of $G$ and the minimization objective $f(c)$ ($c \in \mathcal{C}(G)$) of MSCP is given by Eq. (1).

$$f(c) = \sum_{i=1}^{n} c(v_i) \text{ or } f(c) = \sum_{l=1}^{k} l|V_l|$$

where $|V_l|$ is the cardinality of $V_l$ and $|V_1| \geq \ldots \geq |V_k|$ with the chromatic sum given by:

$$\sum(G) = \min_{c \in \mathcal{C}(G)} f(c)$$

Figure 1 shows an illustrative example for MSCP. The graph has a chromatic number $\chi(G)$ of 3 (left figure), but requires 4 colors to achieve the chromatic sum (right figure). Indeed, with the given 4-coloring, we achieve the chromatic sum of 15 while the 3-coloring of left figure leads to a suboptimal sum of 18 (upper bound).

As shown in [42], MSCP can be conveniently formulated as an integer linear programming problem as follows:

$$\begin{align*}
\text{minimize} & \quad g(x) = \sum_{i=1}^{n} \sum_{l=1}^{k} l \cdot x_{il} \\
\text{subject to} & \quad \sum_{l=1}^{k} x_{il} = 1, i \in \{1, \ldots, n\} \\
& \quad x_{il} + x_{jl} \leq 1, \forall\{v_i, v_j\} \in E, l \in \{1, \ldots, k\} \\
& \quad x_{il} \in \{0, 1\}
\end{align*}$$

where $x_{il} = 1$ ($i \in \{1, \ldots, n\}, l \in \{1, \ldots, k\}$) if $v_i$ is assigned color $l$, $x_{il} = 0$ otherwise.

The first constraint of this ILP model ensures that each vertex receives a single color while the second constraint states that two adjacent vertices cannot be assigned the same color. This linear model can be solved by any ILP solver like CPLEX [45].

Finally, as shown in [45], MSCP can also be formulated as a binary quadratic programming model.
3 Polynomial-time and \( k \)-approximation algorithms for MSCP

One notes that till now no exact algorithm especially designed for MSCP was reported in the literature except the general solution approach used in [45] which applies CPLEX to the integer linear programming formulation (Eq. (3)). On the other hand, a number of polynomial-time and \( k \)-approximation algorithms have been proposed for specific classes of graphs, such as trees, interval graphs, bipartite graphs, etc [9, 16, 22, 27, 36]. These algorithms exploit particular properties of the special graphs considered. In what follows, we briefly recall the main characteristics of these specific classes of graphs:

- A **cograph**, also called \( P_4 \)-free graph, is a graph that does not contain the path \( P_4 \) for any four vertices\(^1\).
- \( P_4 \)-**reducible** graphs are a generalization of cographs where every vertex belongs to at most one \( P_4 \);
- \( P_4 \)-**sparse** graphs generalize \( P_4 \)-reducible graphs by imposing that every set of five vertices induces at most one \( P_4 \);
- **Unicyclic** graphs contain exactly one cycle;
- A **partial \( k \)-tree** \( G \) is a graph with treewidth of at most \( k \), where the treewidth is the size of the largest vertex set in a tree decomposition of \( G \);
- A graph is **outerplanar** if it is planar (it can be embedded in the plane without crossing edges) and all its vertices lie on the exterior face;
- The **line** graph \( L(G) \) of any graph \( G = (V, E) \) is such that its vertex set is \( E \) and two vertices of \( L(G) \) are adjacent if their corresponding edges in \( G \) are incident;
- In an **interval** graph, each vertex corresponds to an interval (over the set of real numbers for instance) and there is an edge between two vertices if their corresponding intervals intersect.

In the field of VLSI design, Kroon et al. [28] considered the “optimum cost chromatic partition problem” (OCCP), whose definition is similar to MSCP. For this problem, they introduced a linear-time algorithm for trees (see also [33]). Other classes of graph optimally solved in linear time include cographs [21] or unicyclic graphs [31] for instance.

In [21], Jansen found that the OCCP can be solved in polynomial time for partial \( k \)-trees. Then, Salavatipour presented a polynomial-time algorithm for \( P_4 \)-reducible graphs [41]. Furthermore, Bonomo and Valencia-Pabon studied \( P_4 \)-sparse graphs and found a large sub-family of \( P_4 \)-sparse graphs that can be solved in polynomial time [7]. A cubic algorithm has also been proposed for outerplanar graphs [31].

Bar-Noy et al. proposed a 2-approximation algorithm\(^2\) for line graphs and showed a \((\Delta + 2)/3\)-approximation algorithm for graphs with maximum degree \( \Delta \) [1]. Then, Bar-Noy and Kortsarz proposed a 10/9-approximation algorithm for bipartite graphs [3].

\(^{1}\)A path \( P_4 \) is a sequence of 4 vertices, say \((v_1, v_2, v_3, v_4)\), such that \( \{v_i, v_{i+1}\} \in E \forall i \in \{1, 2, 3\} \) and \( \{v_i, v_{i+1}\} \notin E \forall k \in \{1, 2, 3, 4\} \setminus \{i, i+1\} \).

\(^{2}\)A \( k \)-approximation algorithm ensures to return a solution whose evaluation / cost is no more than a factor \( k \) of the optimum.
This approximation ratio was next improved to 27/26 by Malafiejski et al. [37] which is the best ratio for bipartite graphs to our knowledge. For interval graphs, Nicoloso et al. presented a 2-approximation algorithm [40], the best known ratio for this class of graphs being 1.796 [17]. Let us finally mention a 2-approximation algorithm for the entire class of $P_4$-sparse graphs [6].

4 Heuristics and metaheuristics for MSCP

Since these approximability results cannot be generalized to an arbitrary graph, for practically solving MSCP in the general case, a number of heuristic and metaheuristic algorithms have been proposed recently. In this section, we review the most representative and effective MSCP heuristic and metaheuristic algorithms which belong to three large classes of methods: greedy algorithms, local search heuristics, and evolutionary algorithms. For each reviewed algorithm, we identify its key ingredients, and highlight if the search process is constrained in the feasible space or is allowed to visit infeasible regions. We also provide in Table 1 a summary of the reviewed algorithms as well as indicators about their performances.

4.1 Greedy algorithms

Greedy algorithms are among the first heuristics proposed for MSCP. These algorithms are generally fast, simple, and easy to implement. Nevertheless, they usually achieve results of poor quality. On the other hand, given their particular features (speed and simplicity), they can advantageously be integrated into other more elaborated approaches where the greedy heuristic is used to generate an initial solution and seeds the search process. For instance, they can be used to provide initial upper bounds for an exact algorithm or to build the initial solution(s) for local search heuristics and evolutionary algorithms.

Two families of greedy algorithms for MSCP are proposed in [35]: MDSAT($n$) and MRLF($n$). They are based on the two well-known greedy coloring heuristics DSATUR [10] and RLF [34].

The original DSATUR heuristic employs the saturation degree $dsat$ of a vertex $v$ as the selection criterion to dynamically determine the next vertex to color. MDSAT($n$) improves DSATUR by considering the impact of coloring a vertex where the impact is measured based on the number of vertices whose $dsat$ would (not) be changed. The original RLF heuristic follows the partition perspective of a vertex coloring. It colors as many non-adjacent vertices as possible with one color before going to another color. MRLF($n$) which extends RLF is based on the idea of selecting the next candidate vertex $v$ for coloring such that it reduces the chance of using a new color next and keeps the current color class as large as possible. To achieve this goal, MRLF($n$) implements sophisticated greedy rules which rely on the cardinality of a subset of uncolored vertices that could be colored with and without using a new color.

A more complicated greedy heuristic (EXSCOL) is proposed in [46, 48]. It is based on independent set extraction and is highly effective for hard and large graphs. At each iteration, EXSCOL first identifies an independent set $S$ as large as possible by using a tabu search procedure. Secondly, it searches as many independent sets as possible of the same size $|S|$ to build a pool of candidate independent sets. Then, EXSCOL

$^3dsat(v_i)$ is the number of colors used to color the vertices adjacent to $v_i$. 
determines a maximum number of disjoint independent sets by solving a maximum set packing problem. Finally, the vertices of each extracted independent set receive the same smallest available color to form a color class. The above process is repeated until the graph becomes empty. Notice that there is no procedure to reconsider the extracted independent sets such that it is impossible for EXSCOL to attain an optimal solution once a “bad” independent set has been extracted.

4.2 Local search heuristics

Local search (or neighborhood search) heuristics progressively modify a candidate solution $c$ by local transformations until a stop condition is reached [15]. The two key components of a local search procedure are the evaluation function and the move (or transformation) operator which are defined on a given search space.

The evaluation function is used to assess the quality of a given coloring. The existing MSCP algorithms employ one of two types of evaluation function according to whether feasible or infeasible colorings are visited. For algorithms that explore only feasible solutions (i.e., proper colorings), the minimization function $f$ (i.e., the sum of colors, Eq. (1)) of the MSCP problem is directly used. On the other hand, algorithms that visit both feasible and infeasible solutions usually call for an augmented evaluation function $f_p$ which combines the objective function $f$ and a penalty function $p$.

In local search algorithms, one iteratively uses one or more move operators to transform the incumbent solutions $c$ to generate new neighboring solutions $c'$ for all solutions. The set of neighboring solutions that can be reached by applying a move operator ($mv$) to the current solution forms the neighborhood (denoted by $N_mv$). We describe the commonly used operators as follows.

- **One-move** changes the color of a vertex in the current solution by moving a vertex $v$ from its current color class $V_i$ to another color class $V_j$ ($i \neq j$). This operator can generate both proper or improper colorings and thus can be used to explore feasible and infeasible regions of the coloring search space;

- **Swap** displaces a vertex $v$ from its current color class $V_i$ to another color class $V_j$ (as One-move) and then moves all adjacent vertices $u$ of $v$ to $V_i$. This operator can generate both proper or improper colorings;

- **Exchange** swaps a subset of vertices $A \subset V_i (|A| > 1)$ and another subset of vertices $B \subset V_j (|B| > 1) (i \neq j)$ such that the subgraph induced by $A \cup B$ is a connected component [24]. The new solution $c'$ is feasible (respectively infeasible) if the starting solution $c$ is feasible (infeasible).

In what follows, we classify the representative local search algorithms into two categories according to the adopted neighborhood(s): single neighborhood search and multi-neighborhood search. Since local search can get stuck in local optima, most local search algorithms for MSCP use some diversification techniques to help the search to escape local optima encountered during the search. This is typically achieved by applying one or more perturbation operators to change a local optimum in a random or dedicated way.

4.2.1 Single neighborhood search

The tabu search (TS) algorithm proposed in [8] adapts the tabu algorithm designed for the classic vertex coloring problem [13, 20]. It starts with a random coloring and
visits both proper and improper colorings with the neighborhood $N_{One-move}$ induced by the $One-move$ operator. If there exist conflicting vertices, TS chooses a best move (according to its evaluation function $f_p$) to change the color of a conflicting vertex. Otherwise, TS picks a (non-conflicting) vertex and change its color at random. The above steps are repeated until a stopping criterion is satisfied. This algorithm relies simply on the tabu list for its diversification and does not call for other perturbation mechanism. This algorithm only showed limited computational results.

The breakout local search (BLS) algorithm described in [4] jointly uses two descent methods and an adaptive multi-perturbation strategy to escape local optima. The basic idea of BLS is to use descent-based local search to discover local optima and employ adaptive perturbations to continually visit different search regions in the search space. BLS explores both feasible and infeasible solutions with the help of the $One-move$ operator. At each iteration, if the current solution $c$ is a feasible coloring, BLS applies a first descent search procedure to attain a local optimum in terms of the objective function $f$. If $c$ is an infeasible coloring (i.e., with conflicting vertices), BLS applies another descent search procedure guided by an augmented evaluation function which takes into account both the objective function $f$ and the conflicting vertices. BLS is characterized by its adaptive perturbation strategy which, upon the discover of a local optimum, triggers dedicated perturbation operations to escape the local optimum trap. Based on the information on the search state, the perturbation strategy of BLS introduces a varying degree of diversification by dynamically determining the number of perturbation moves to be applied and by adaptively selecting the suitable moves (random or directed perturbations).

4.2.2 Multi-neighborhood search

The MDS(5)+LS algorithm [19] applies an iterated multi-neighborhood search and also explores feasible and infeasible regions of the search space. It first employs the $Swap$ operator until no further improvement exists in terms of its augmented evaluation function. Note that the obtained solution is not necessarily a proper coloring. If this is the case, MDS(5)+LS switches then to the $One-move$ operator to repair the solution. Additional colors can be used to guarantee that the final coloring is proper at the end of this search phase. Finally, it assigns all the vertices with their smallest legal color and changes the color labels according to the sorted cardinality of the color classes $V_i (|V_1| \geq ... \geq |V_k|)$. Afterward, a random perturbation operator is applied which consists in moving some vertices from their current color class to another color class at random. This perturbed solution is then used as the starting point of the next round of the search procedure.

4.3 Evolutionary algorithms

Different from local search algorithms which are based on a single solution, evolutionary algorithms use a pool of solutions and try to find gradually better solutions by applying genetic operators (e.g., crossover, mutation, ...) to solutions of the population [15].

The most popular evolutionary algorithms for MSCP follow the hybrid evolution framework called the memetic algorithm which jointly uses a recombination operator and a local search improvement to explore the search space [15]. They include, for instance, the MASC algorithm [24], MA-MSCP algorithm [39] and the HESA hybrid search algorithm [23]. Besides, an early parallel genetic algorithm PGA [26] employs
| Algorithm name | Reference | Type of approach | Neighborhoods | Perturbation | Comments on performance |
|----------------|-----------|------------------|--------------|-------------|-------------------------|
| MDSAT($n$)     | [35](2009) | Greedy search    | -            | -           | A family of improved greedy algorithms based on the well-known greedy coloring strategies DSATUR and RLF. |
| MRLF($n$)      |           |                  |              |             |                         |
| TS             | [8](2010)  | Local search     | $N_{One-move}$ | No          | A very simple tabu search but the results are better than those of the greedy algorithms MDSAT($n$) and MRLF($n$). |
| MDS(5)+LS      | [19](2011) | Local search     | $N_{One-move}$ & $N_{Swap}$ | Yes         | An iterated multi-neighborhood search combined with a random perturbation procedure achieving better results than MDSAT($n$), MRLF($n$) and TS. |
| BLS            | [4](2012)   | Local search     | $N_{One-move}$ | Yes         | A breakout local search combining a greedy descent strategy with an adaptive perturbation step. It performs well on the small DIMACS graphs. |
| EXSCOL         | [46, 48](2012) | Greedy + tabu search | No | No          | A complicated greedy algorithm, based on independent sets extraction with tabu search, which is quite effective for large graphs. |
| MASC           | [24](2014)  | Evolutionary search | $N_{One-move}$ & $N_{Exchange}$ | Yes         | A memetic algorithm based on a double-neighborhood tabu search and a multi-parent crossover operator. Most results are better than those of the neighborhood search heuristics. |
| MA-MSCP        | [39](2013)  | Evolutionary search | $N_{One-move}$ | Yes         | A genetic algorithm with a two-parents crossover operator combined with a local search based on a hill climbing and a “destroy & repair” procedures. Results are comparable to those of MASC. |
| HESA           | [23](2016)  | Evolutionary search | $N_{One-move}$ | Yes         | A hybrid search algorithm based on a jointly use of two crossover operators and an iterated double-phase tabu search procedure. The lower and upper bounds obtained by the HESA are highly competitive with the best known results in the literature. |
assignment and partition crossovers, first-fit mutation, and proportional selection without any local search improvement.

The MASC memetic algorithm [24] follows the design guidelines of memetic algorithms for discrete optimization [18] and combines a multi-parent crossover operator (called MG PX) and a double-neighborhood tabu search procedure. MG PX is a variant of the well-known GPX crossover originally proposed for the classical vertex coloring problem [13]. It builds the color classes of the offspring (which is always a proper coloring) one by one and transmits entire color classes as large as possible until all vertices of the offspring are colored. Besides, the tabu search procedure applies the two different and complementary neighborhoods induced by Exchange and One-move in a token-ring way to find good local optima (according to the objective function f) until the search is stagnating. MASC employs a dedicated perturbation operator to diversify the search. MASC only explores the feasible search space of MSCP.

MA-MSCP is another hybrid genetic algorithm [39] that also focuses on the feasible search space. It includes a two-parent crossover operator (yet another adaptive variant of GPX [13]), a hill-climbing local search algorithm and a “destroy & repair” procedures. During the local search phase, the hill-climbing procedure is first applied to improve the current solution by using the One-move operator. To escape local optima, MA-MSCP then applies the “destroy & repair” strategy, which randomly removes some vertices and re-inserts each of them into its largest available color class while keeping the solution feasible. If there is no such a color class, the vertex is moved to a new color class. MA-MSCP employs the above two procedures alternately until no further improvement can be obtained.

HESA is also a hybrid search algorithm [23] that alternates between feasible and infeasible regions of the search space. HESA relies on a double-crossover recombination method and an iterated double-phase tabu search procedure. The recombination method jointly uses a diversification-guided crossover and a grouping-guided crossover to generate promising offspring solutions. During the double-phase tabu search procedure, it first checks if the given solution c is a proper coloring. If c is proper, the first tabu search is called to improve its sum of colors. Otherwise, another tabu search is used to attain a proper coloring which is further improved by the first tabu search to obtain a better sum of colors. The double-phase tabu search only explores the NOne-move neighborhood. For the purpose of search diversification, HESA applies a conditional mixed perturbation strategy: 1) apply the Swap operator to a randomly chosen vertex to transform the incumbent solution, or 2) replace the current solution by the last local optimum.

Table 1 summarizes the reviewed existing heuristic algorithms with their main characteristics including the type of search paradigm, the neighborhood(s) and the presence or absence of a perturbation strategy together with a comment on their relative performance.

Finally, we mention the BQP-PR evolutionary algorithm [45] which relies on a binary quadratic programming formulation of the problem (see Section 2) and combines a path relinking approach with a tabu search procedure.

5 Bounds for MSCP

We will refer here to “theoretical” (lower and upper) bounds if they are formally proved, see Section 5.1. By opposition, “computational” bounds introduced in Section 5.2 designate those obtained running approximate algorithms.
5.1 Theoretical bounds

Recall that for any undirected simple graph \( G = (V, E) \) with \( n = |V| \) vertices and \( m = |E| \) edges, the chromatic number \( \chi(G) \) is the smallest number of colors needed to color the vertices of \( G \) such that a proper \( k \)-coloring exists and the chromatic sum \( \sum(G) \) is the minimum sum of the colors assigned to all vertices among all proper \( k \)-colorings of \( G \). In this section, we list the current known theoretical lower and upper bounds of MSCP according to [26, 39, 44].

\[
\sum(G) \leq n + m \leq \left\lceil \sqrt{8m} \right\rceil \leq \sum(G) \leq \frac{n\chi(G) - 1}{2} \leq \sum(G) \leq \left\lfloor n\chi(G) + \frac{1}{2} \right\rfloor
\]

From Eq. (4), one easily observes that the best theoretical lower and upper bounds available for MSCP are respectively

\[
\text{LB}_t = \max \left\{ \left\lceil \sqrt{8m} \right\rceil, n + \chi(G)(\chi(G) - 1) \right\}
\]

and

\[
\text{UB}_t = \min \left\{ n + m, \left\lfloor \frac{n\chi(G) + 1}{2} \right\rfloor \right\}
\]

5.2 Computational bounds

Given that MSCP is to find a proper \( k \)-coloring while minimizing the sum of the colors assigned to the vertices, Eq. (1) gives a computational upper bound for MSCP.

Let \( G' = (V, E') \subseteq (V, E) \) be any partial graph of \( G \). Any proper coloring of \( G \) must be a proper coloring of \( G' \):

\[
\sum(G') \leq \sum(G)
\]

Partial graphs considered in the literature to estimate the computational lower bound \( f_{LB} \) include bipartite graphs (trees and paths) \([14, 28]\) and cliques \([38, 48]\), while graph decomposition into cliques \([19, 23, 39, 48]\) provide better bounds according to \([38]\). Let \( c = \{S_1, S_2, \ldots, S_k\} \) be a clique decomposition of \( G \), then Eq. (5) gives a computational lower bound for MSCP since there is a single way of coloring any clique \( S_l \) (with \( |S_l| \) colors) and the sum of colors of \( S_l \) is \( |S_l|(|S_l| + 1)/2 \).

\[
f_{LB}(c) = \sum_{l=1}^{k} \frac{|S_l|(|S_l| + 1)}{2}
\]

Figure 2 shows an illustrative lower bound via clique decomposition. We decompose \( G \) into six cliques by ignoring some edges of the original graph \( G \) and obtain the chromatic sum \( \sum(G') = 13 \) (right figure). Clearly, this is a lower bound for MSCP while the chromatic sum \( \sum(G) = 15 \) (left figure).

To obtain a clique decomposition, one popular approach is to find a proper coloring of the complementary graph \( \bar{G} \) of \( G \) \([19, 23, 39, 48]\), since each color class of \( \bar{G} \) is a clique of \( G \).

6 Benchmark and performance evaluation

In this section, we first introduce a set of MSCP instances (benchmarks) that are commonly used to assess the performance of MSCP algorithms and then provide indications about the performances of the reviewed MSCP algorithms. Due to many different

---

4A clique is a complete graph where all the vertices are pairwise adjacent. A clique decomposition of a graph is a partition of the vertex set \( V \) into a collection of cliques.
factors (programming languages, running platforms, experimental protocols...), it is quite difficult to draw definitive conclusions. Nevertheless, we try to provide some useful indications with respect to their performance in terms of best and average results.

6.1 Benchmark

There exists a set of 94 frequently used benchmark instances often used for performance evaluation of MSCP algorithms. 58 instances are part of the COLOR 2002–2004 competition\(^5\) while the remaining 36 instances come from the second DIMACS challenge\(^6\). Compared to the well-known DIMACS instances, the COLOR 2002-2004 instances are relatively easy except the four large “wap” graphs. These instances refer to various topologies and densities, which can be classified into the 13 following types:

- Twelve classical random graphs (DSJC\(n,d, n \in \{125, 250, 500, 1000\}, d \in \{1, 5, 9\})
- Three geometric graphs (DSJR\(500.d, d \in \{1c, 1, 5\})
- Six flat graphs (flat\(300.\chi.0, \chi \in \{20, 26, 28\}\) and flat\(1000.\chi.0, \chi \in \{50, 60, 76\}\)
- Twelve Leighton graphs (le\(450.\chi.a, \chi \in \{5, 15, 25\}\)
- Four latin square graph (latin\(sqr.10, qg.order.\chi, \chi \in \{30, 40, 50\}\)
- Two very large random graphs (C\(2000.5\) and C\(4000.5\)
- Fourteen graphs based on register allocation (fpsol2.i.a, initx.i.a, zeroin.i.a, mulsol.i.b, a \in \{1, 2, 3\} and b \in \{1, 2, 3, 4, 5\})
- Two graphs from the scheduling area (school1 and school1\_nsh)
- Twenty four graphs from the Donald Knuth’s Stanford GraphBase (miles\(n\) with \(n \in \{250, 500, 750, 1000, 1500\}\), anna, david, huck, jean, homer, games\(120\), queen\(8.12\) and queen\(na.a, a \in \{5, …, 16\}\)

\(^5\)http://mat.gsia.cmu.edu/COLOR02
\(^6\)http://dimacs.rutgers.edu/Challenges/
• Five graphs based on the Mycielski transformation (myciela, \( a \in \{3, 4, 5, 6, 7\} \));

• Four graphs that have a hard-to-find four clique embedded (mug\( a, n \in \{88, 100\}, a \in \{1, 25\} \));

• Two “insertion” graphs (2-\text{Insert}_{3} and 3-\text{Insert}_{3});

• Four graphs from real-life optical network design problems (wap05, wap06, wap07, and wap08).

| Graph | \( n \) | \( d \) | \( \chi(G) \) | \( LB \) | \( UB \) |
|-------|------|------|----------|------|------|
| myciel3 | 11 | 20 | 0.36 | 4 | 17 | 27 |
| myciel4 | 23 | 71 | 0.28 | 5 | 33 | 69 |
| myciel5 | 47 | 236 | 0.22 | 6 | 62 | 164 |
| myciel6 | 95 | 755 | 0.17 | 7 | 116 | 380 |
| myciel7 | 191 | 2360 | 0.13 | 8 | 219 | 859 |
| anna | 138 | 493 | 0.05 | 11 | 193 | 631 |
| david | 87 | 406 | 0.11 | 11 | 142 | 493 |
| luck | 74 | 301 | 0.11 | 11 | 129 | 375 |
| homer | 561 | 1628 | 0.01 | 11 | 639 | 2189 |
| queen5.5 | 25 | 160 | 0.53 | 5 | 36 | 75 |
| queen6 | 56 | 290 | 0.46 | 7 | 57 | 144 |
| queen7 | 49 | 476 | 0.40 | 7 | 70 | 196 |
| queen8.8 | 64 | 728 | 0.36 | 9 | 100 | 320 |
| queen8.12 | 96 | 1368 | 0.30 | 12 | 62 | 144 |
| queen9.9 | 81 | 1056 | 0.33 | 10 | 126 | 445 |
| queen10.10 | 100 | 1470 | 0.30 | 11 | 155 | 600 |
| queen11.11 | 121 | 1980 | 0.27 | 11 | 178 | 726 |
| queen12.12 | 144 | 2596 | 0.25 | 12 | 210 | 936 |
| queen13.13 | 169 | 3328 | 0.23 | 13 | 247 | 1183 |
| queen14.14 | 196 | 4186 | 0.22 | 14 | 287 | 1470 |
| queen15.15 | 255 | 5180 | 0.21 | 15 | 330 | 1800 |
| queen16.16 | 256 | 6320 | 0.19 | 16 | 376 | 2176 |
| school1 | 385 | 19005 | 0.26 | 14 | 476 | 2887 |
| school1-mu | 352 | 14612 | 0.24 | 14 | 443 | 2640 |
| miles250 | 128 | 387 | 0.05 | 8 | 156 | 515 |
| miles500 | 128 | 1170 | 0.14 | 20 | 318 | 1298 |
| miles750 | 128 | 2113 | 0.26 | 31 | 593 | 2048 |
| miles1000 | 128 | 3216 | 0.40 | 42 | 989 | 2752 |
| miles1500 | 128 | 5198 | 0.64 | 73 | 2756 | 4736 |
| fpsl2.1 | 496 | 11654 | 0.09 | 65 | 2576 | 12150 |
| fpsl2.2 | 540 | 1691 | 0.08 | 30 | 886 | 6990 |
| fpsl2.3 | 425 | 8688 | 0.10 | 30 | 860 | 6587 |
| mug88a | 88 | 146 | 0.04 | 4 | 94 | 220 |
| mug88b | 88 | 146 | 0.04 | 4 | 94 | 220 |
| mug100 | 100 | 163 | 0.03 | 4 | 106 | 250 |
| mug100Q | 100 | 163 | 0.03 | 4 | 106 | 250 |
| mug100Q2 | 100 | 163 | 0.03 | 4 | 106 | 250 |
| 2 Inserts | 72 | 72 | 0.11 | 4 | 43 | 92 |
| 3 Inserts | 56 | 110 | 0.07 | 4 | 62 | 140 |
| initix.1 | 864 | 18707 | 0.05 | 54 | 2295 | 19571 |
| initix.2 | 645 | 13979 | 0.07 | 31 | 1110 | 10320 |
| mulsol.1 | 197 | 3925 | 0.20 | 49 | 1373 | 4122 |
| mulsol.2 | 188 | 3885 | 0.22 | 31 | 653 | 3008 |
| mulsol.3 | 184 | 3916 | 0.23 | 31 | 649 | 2944 |
| mulsol.4 | 185 | 3946 | 0.23 | 31 | 650 | 2960 |

Table 2 gives the detailed characteristics of the benchmark graphs. Columns 2–5 and 9–12 indicate the number \( n \) of vertices, the number \( m \) of edges, the density \( d = 2m/n(n−1) \) and the chromatic number \( \chi(G) \) of each graph. Columns 6–7 and 13–14 show the best theoretical lower and upper bounds of the chromatic sum (\( LB \) and \( UB \), respectively). Underlined entries (in all tables) indicate that theoretical upper
bounds equal the computational upper bounds while no theoretical lower bound equals the computational lower bound. Note that, since the chromatic number $\chi(G)$ of some difficult graphs are still unknown, we use the minimum $k$ for which a $k$-coloring has been reported for $G$ in the literature instead of $\chi(G)$ to compute $LB_t$ and $UB_t$ using the min/max equations introduced in Section 5.1.

6.2 Performance of MSCP algorithms

Based on the benchmark introduced in the previous section, Table 3 (see the Appendix) summarizes the computational results of six representative and effective MSCP algorithms presented in Section 4: BLS [4], MASC [24], MDS(5)+LS [19], EXSCOL [46, 48], MA-MSCP [39] and HESA [23]. Columns 1–3 present the tested graph and its best known lower and upper bounds ($f^p_{LB}$ and $f^p_{UB}$ respectively, in bold face when optimality is proved), the following 18 columns give the detailed computational results of the six algorithms. “–” marks for the reference algorithms mean non-available results. The results in terms of solution quality (best / average lower and upper bounds, $f^*_{LB}$ / $f_{LB}$ and $f^*_{UB}$ / $f_{UB}$ respectively) are directly extracted from the original papers. Computing times are not listed in the table due to the difference of experimental conditions (platforms, programming languages, stop conditions...). Nevertheless, the second and third lines of the heading respectively indicate the main computer characteristic (processor frequency) and the stop condition to have an idea of the maximum amount of search used by each approach. Note that there is no specific stop condition for EXSCOL since its extraction process ends when the current graph becomes empty. Furthermore, some heuristics can halt before reaching the stop criterion, when a known (lower) bound is reached for instance.

From Table 3 one observes that only HESA reports results for all the 94 graphs of the benchmark. Besides, MDS(5)+LS, EXSCOL, MA-MSCP, and HESA provide lower and upper bounds while BLS and MASC only give an upper bound. Additionally, Figure 3 provides performance information of each of the six algorithms compared to the best known upper and lower bounds. One observes that no algorithm can reach all the best known results. BLS and MASC attain the best upper bounds for 17 graphs out of the 27 tested graphs and for 56 graphs out of the 77 tested graphs respectively. MDS(5)+LS reaches the best lower (upper) bound for 24 (26) instances out of 38. EXSCOL reaches the best lower and upper bounds for 38 (out of 62 graphs) and 24 (out of 52 graphs) respectively. MA-MSCP reaches the best lower / upper bound for 51 / 53 graphs out of 81. HESA equals the best lower (upper) bound for 86 (85) instances out of 94.

Since the number of tested graphs differs from one algorithm to another, the performance of these algorithms cannot be compared from a statistical viewpoint. However, from Table 3 and Figure 3 we can roughly conclude that BLS, MASC, MDS(5)+LS, EXSCOL, MA-MSCP and HESA are currently the most effective algorithms for solving the MSCP problem.

From the theoretical and computational bounds reviewed above, we can make the following observations:

- Optimality is proved for 21 instances out of the 94 tested graphs since the best upper bounds are equal to the best lower bounds (see entries in bold in Table 3).
- 12 theoretical upper bounds equal the computational upper bounds while no theoretical lower bound equals the computational lower bound (underlined in Tables 2, 3).
• The theoretical upper bounds of queen $a$ ($a \in \{11, 12, 13, 14, 15, 16\}$) are equal to the best computational lower bounds meaning optimal results;

• Table 3 shows that the best computational lower bounds of some easy graphs (myciela, $a \in \{3, 4, 5, 6\}$, for instance) are not equal to the optimal upper bounds (optimality proved with CPLEX [45]). Hence, the method of decomposing the graph introduced in Section 5.2 is not good enough in some cases and should be improved.

7 Perspectives and conclusion

This review is dedicated to recent approximation algorithms and practical solution algorithms designed for the minimum sum coloring problem which attracted increasing attention in recent years. MSCP is a strongly constrained combinatorial optimization problem which is theoretically important and computationally difficult. In addition to its relevance as a typical model to formulate a number of practical problems, MSCP can be used as a benchmark problem to test constraint satisfaction algorithms and solvers.

Based on this review, we discuss some perspective research directions.

• Evaluation function and search space: as introduced in Section 2, the aim of MSCP is twofold: (1) find a proper $k$-coloring $c$ of a graph and (2) ensure that the sum of the colors assigned to the vertices is minimized. An evaluation function combining these two objectives has been proposed in [19]:

$$f'(c) = \sum_{l=1}^{k} l|V_l| + M|E(V_l)|$$

where $E(V_l)$ is the set of conflicting edges in $V_l$ and $M > 0$ is a sufficiently large natural number. Since the evaluation function is used to guide the heuristic search process, it would be interesting to design other effective evaluation function based on a better recombination of the two parts of $f'$.

Another possibility could be to explore only the feasible graph coloring search space, like in the competitive MASC and MA-MSCP approaches [24, 39], using more effective (multi-)neighborhood structures.

Besides, the combination of the above two ingredients in a proper way may lead to improved MSCP algorithms.

• Maximum independent sets extraction: As shown in Section 4.1, EXSCOL is a greedy heuristic based on the independent sets extraction that is quite effective for large graphs. Its major deficiency is that it does not include a procedure to reconsider “bad” independent sets that has been extracted. Hence, one possibility is to devise a backtracking procedure when a “bad” independent set has been identified as proposed for the graph coloring problem [47].

• Exact algorithms: There is no exact algorithm especially designed for MSCP except the general approach which applies CPLEX to solve the integer linear programming formulation of MSCP [45]. However, as shown in [45], this approach is only applicable to easy DIMACS instances. On the other hand, some exact algorithms for the classical vertex coloring problem successfully solved a subset of the hard DIMACS graphs. Hence, it would be important to fill the gap by designing exact algorithms for MSCP.
Figure 3: The performance of six representative MSCP algorithms. The y-axis shows the number of graphs for which an algorithm attains a result equal to or worse than the best known reported bound.
To conclude, the minimum sum coloring problem, like the classical coloring problem, is a generic and useful model. Advances in solution methods (both exact and heuristic methods) for these coloring problems will help find satisfying solutions to many practical problems. Given the increasing interest in the sum coloring problem and their related coloring problems, it is reasonable to believe that research in these domains will become even more intense and fruitful in the forthcoming years.

Acknowledgment

We are grateful to the anonymous referees for valuable suggestions and comments which have helped us to improve the paper. This work was partially supported by the LigeRO (2009-2014, Pays de la Loire Region), PGMO (2014-2016, Jacques Hadamard Mathematical Foundation) projects and the National Natural Science Foundation Program of China (Grant No. 61472147).

References

[1] Bar-Noy, A., Bellare, M., Halldórsson, M.M., Shachnai, H., Tamir, T. (1998). On chromatic sums and distributed resource allocation. Information and Computation 140, 183–202.

[2] Bar-Noy, A., Halldórsson, M.M., Kortsarz, G., Salman, R., Shachnai, H. (1999). Sum multi-coloring of graphs. In: J. Nesetril (ed.), 7th Annual European Symposium on Algorithms, vol. 1643 of Lecture Notes in Computer Science, Springer, Berlin / Heidelberg, Germany, pp. 390–401.

[3] Bar-Noy, A., Kortsarz, G. (1998). Minimum color sum of bipartite graphs. Journal of Algorithms 28(2): 339–365.

[4] Benlic, U., Hao, J.-K. (2012). A study of breakout local search for the minimum sum coloring problem. In: L. Bui, Y. Ong, N. Hoai, H. Ishibuchi, P. Suganthan (eds.), Simulated Evolution and Learning, vol. 7673 of Lecture Notes in Computer Science, Springer, Berlin / Heidelberg, Germany, pp. 128–137.

[5] Berliner, A., Bostelmann, U., Brualdi, R.A., Deaett, L. (2006). Sum list coloring graphs. Graphs and Combinatorics 22(2): 173–183.

[6] Bonomo, F., Durán, G., Napoli, A., Valencia-Pabon, M. (2015). A one-to-one correspondence between potential solutions of the cluster deletion problem and the minimum sum coloring problem, and its application to $P_4$-sparse graphs. Information Processing Letters 115(6–8): 600–603.

[7] Bonomo, F., Valencia-Pabon, M. (2014). On the Minimum Sum Coloring of $P_4$-sparse graphs. Graphs and Combinatorics 30(2): 303–314.

[8] Bouziri, H., Jouini, M. (2010). A tabu search approach for the sum coloring problem. Electronic Notes in Discrete Mathematics 36: 915–922.

[9] Borodin, A., Ivan, I., Ye, Y., Zimny, B. 2012. On sum coloring and sum multi-coloring for restricted families of graphs. Theoretical Computer Science 418: 1–13.
[10] Brélaz, D. (1979). New methods to color the vertices of a graph. Communications of the ACM 22(4): 251–256.

[11] Douiri, S., Elbernoussi, S. (2011). New algorithm for the sum coloring problem. International Journal of Contemporary Mathematical Sciences 6: 453–463.

[12] Douiri, S., Elbernoussi, S. (2012). A new ant colony optimization algorithm for the lower bound of sum coloring problem. Journal of Mathematical Modelling and Algorithms 11(2): 181–192.

[13] Galinier, P., Hao, J.-K. (1999). Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization 3(4): 379–397.

[14] Garey, M.R., Johnson, D.S. (1979). Computers and intractability. A Guide to the Theory of NP-Completeness, Ed. Freeman and Company, New York.

[15] Gendreau, M., Potvin J.Y. (2010). Handbook of metaheuristics. International Series in Operations Research & Management Science, Springer, New York.

[16] Hajialiobhassan, H., Mehrabadi, M.L., Tusserkani, R. (2000). Minimal coloring and strength of graphs. Discrete Mathematics 215(13): 265–270.

[17] Halldórsson, M.M., Kortsarz, G., Shachnai, H. (2003). Sum coloring interval graphs and k-claw free graphs with applications for scheduling dependent jobs. Algorithmica 37, 187–209.

[18] Hao, J.-K. (2012). Memetic algorithms in discrete optimization. In F. Neri, C. Cotta, P. Moscato (eds.) Handbook of Memetic Algorithms. Studies in Computational Intelligence 379, Chapter 6, pages 73–94, Springer.

[19] Helmar, A., Chiarandini, M. (2011). A local search heuristic for chromatic sum. In L. Di Gaspero, A. Schaerf, T. Stützle (eds.). Proceedings of the 9th Metaheuristics International Conference, pp. 161–170.

[20] Hertz, A., de Werra, D. (1987). Using tabu search techniques for graph coloring. Computing 39, 345–351.

[21] Jansen, K. (2000). Approximation results for the optimum cost chromatic partition problem. Journal of Algorithms 34(1): 54–89.

[22] Jiang, T., West, D. (1999). Coloring of trees with minimum sum of colors. Journal of Graph Theory 32(4): 354–358.

[23] Jin, Y., Hao, J.-K. (2016). Hybrid evolutionary search for the minimum sum coloring problem of graphs. Information Sciences 352-353, 15-34.

[24] Jin, Y., Hao, J.-K., Hamiez, J.P. (2014). A memetic algorithm for the Minimum Sum Coloring Problem. Computers & Operations Research 43: 318–327.

[25] Johnson, D.S., Mehrotra, A., Trick M.A. (Eds.) (2008). Special issue on computational methods for graph coloring and its generalizations. Discrete Applied Mathematics, 156(2).
[26] Kokosiński, Z., Kwarciany, K. (2007). On sum coloring of graphs with parallel genetic algorithms. In B. Beliczynski, A. Dzielinski, M. Iwanowski, B. Ribeiro (eds.), Adaptive and Natural Computing Algorithms, vol. 4431 of Lecture Notes in Computer Science, Springer, Berlin / Heidelberg, Germany, pp. 211–219.

[27] Kosowski, A. (2009). A note on the strength and minimum color sum of bipartite graphs. Discrete Applied Mathematics 157(11): 2552-2554.

[28] Kroon, L.G., Sen, A., Deng, H., Roy, A. (1996). The optimum cost chromatic partition problem for trees and interval graphs. International Workshop on Graph Theoretical Concepts in Computer Science, Lecture Notes in Computer Science, 1197: 279–292.

[29] Kubicka, E. (1989). The chromatic sum of a graph. Ph.D. Thesis, Western Michigan University.

[30] Kubicka, E. (2004). The chromatic sum of a graph: history and recent developments. International Journal of Mathematics and Mathematical Sciences 30: 1563–1573.

[31] Kubicka, E. (2005). Polynomial algorithm for finding chromatic sum for unicyclic and outerplanar graphs. Ars Combinatoria 76: 193-201.

[32] Kubicka, E., Kubicki, G., Kountanis, D. (1991). Approximation algorithms for the chromatic sum. First Great Lakes Computer Science Conference on Computing in the 90’s, Lecture Notes in Computer Science, 507: 15–21.

[33] Kubicka, E., Schwenk, A.J. (1989). An introduction to chromatic sums. Proceedings of the 17th conference on ACM Annual Computer Science Conference, CSC ’89, pages 39–45, New York, NY, USA. ACM.

[34] Leighton, F.T. (1979). A graph coloring algorithm for large scheduling problems. Journal of Research of the National Bureau of Standards 84(6): 489–506.

[35] Li, Y., Lucet, C., Moukrim, A., Sghiouer, K. (2009). Greedy algorithms for the minimum sum coloring problem. Logistique et Transports Conference, https://hal.archives-ouvertes.fr/hal-00451266/document

[36] Malafiejski, M. (2004). Sum coloring of graphs, in: M. Kubale (ed.), Graph Colorings, vol. 352 of Contemporary mathematics, American Mathematical Society, New Providence (Rhode Island) USA, pp. 55–65.

[37] Malafiejski, M., Giaro, K., Janczewski, R., Kubale, M. (2004). Sum coloring of bipartite graphs with bounded degree. Algorithmica 40(4):235–244.

[38] Moukrim, A., Sghiouer, K., Lucet, C., Li, Y. (2010). Lower bounds for the minimal sum coloring problem. Electronic Notes in Discrete Mathematics 36: 663–670.

[39] Moukrim, A., Sghiouer, K., Lucet, C., Li, Y. (2013). Upper and lower bounds for the minimum sum coloring problem, submitted for publication. https://www.hds.utc.fr/~moukrim/dokuwiki/doku.php?id=en:mscp

[40] Nicoloso, S., Sarrafzadeh, M., Song, X. (1999). On the sum coloring problem on interval graphs. Algorithmica 23, 109-126.
[41] Salavatipour, M.R. (2003). On sum coloring of graphs. Discrete Applied Mathematics 127(3): 477–488.

[42] Sen, A., Deng, H., Guha, S. (1992). On a graph partition problem with application to VLSI layout. Information Processing Letters 43(2): 87–94.

[43] Supowit, K.J. (1987). Finding a maximum planar subset of a set of nets in a channel. IEEE Trans. Comput. Aided Design CAD 6(1): 93–94.

[44] Thomassen, C., Erdős, P., Alavi, Y., Malde, P., Schwenk, A. (1989). Tight bounds on the chromatic sum of a connected graph. Journal of Graph Theory, 13: 353–357.

[45] Wang, Y., Hao, J.-K., Glover, F., Lü, Z. (2013). Solving the minimum sum coloring problem via binary quadratic programming. CoRR abs/1304.5876.

[46] Wu, Q., Hao, J.-K. (2012). An effective heuristic algorithm for sum coloring of graphs. Computers & Operations Research 39(7): 1593–1600.

[47] Wu, Q., Hao, J.-K. (2012). Improving the extraction and expansion method for large graph coloring. Discrete Applied Mathematics 160(16-17): 2397–2407.

[48] Wu, Q., Hao, J.-K. (2013). Improved lower bounds for sum coloring via clique decomposition. CoRR abs/1303.6761.
Appendix

For the purpose of completeness, this Appendix, which reproduces and extends the results given in [23], shows a performance summary of the six main heuristic algorithms for the set of 94 DIMACS benchmark graphs in terms of the lower and upper bounds of the MSCP problem.
Table 3: The performance of six heuristics and metaheuristics for the lower and upper bounds of MSCP

| Graph    | BLS [4] | MASC [24] | MDS(5)+LS [19] | EXSCOL [46, 48] | MA-MSCP [19] | HESA [23] |
|----------|---------|-----------|----------------|-----------------|---------------|-----------|
|          | 2.83 GHz| 2.7 GHz   | 2.93 GHz       | 2.8 GHz         | 2.83 GHz      | 2.83 GHz  |
|          | 2 hours | 50 generations | 1 hour | No stop condition | 2 hours | 2 hours |
| f         | b       | LB        | f               | b               | LB            | f         |
| myciel3   | 16      | 21        | 21.0             | 16              | 21.0          | 16        |
| myciel4   | 34      | 45        | 45.0             | 34              | 45.0          | 34        |
| myciel5   | 70      | 93        | 93.0             | 70              | 93.0          | 70        |
| myciel6   | 142     | 189       | 196.6            | 189             | 196.0         | 189.0     |
| myciel7   | 286     | 381       | 393.8            | 381             | 393.0         | 393.0     |
| anna      | 273     | 276       | 276.0            | 273             | 276.0         | 276.0     |
| david     | 243     | 243       | 243.0            | 243             | 243.0         | 243.0     |
| hawk      | 216     | 217       | 217.0            | 217             | 217.0         | 217.0     |
| homer     | 1129    | 1150      | -                | 1155            | 1158.5        | -         |
| queen5.5  | 75      | 75        | 75.0             | 75              | 75.0          | 75.0      |
| queen6    | 126     | 138       | 138.0            | 126             | 138.0         | 138.0     |
| queen7.7  | 196     | 196       | 196.0            | 196             | 196.0         | 196.0     |
| queen8.8  | 288     | 291       | 291.0            | 291             | 291.0         | 291.0     |
| queen12   | 624     | 624       | -                | 624             | 624.0         | 624.0     |
| queen9.9  | 405     | 409       | -                | 409             | 410.5         | -         |
| queen10.10| 550     | 553       | -                | 550             | 553.0         | -         |
| queen11.11| 726     | 733       | -                | 726             | 733.0         | -         |
| queen12.12| 936     | 943       | -                | 936             | 943.0         | -         |
| queen13.13| 1183    | 1191      | -                | 1183            | 1192.0        | -         |
| queen14.14| 1470    | 1482      | -                | 1470            | 1482.0        | -         |
| queen15.15| 1800    | 1814      | -                | 1800            | 1814.0        | -         |
| queen16.16| 2176    | 2176      | -                | 2176            | 2176.0        | -         |
| school1   | 2439    | 2674      | -                | 2439            | 2674.0        | -         |
| school1-nsh| 2176   | 2392      | -                | 2176            | 2392.0        | -         |
| miles250  | 318     | 325       | 327              | 328.8           | 325           | 325.0     |
| miles500  | 686     | 705       | 710              | 713.3           | 705.0         | 686.0     |
| miles750  | 1145    | 1173      | -                | 1173            | 1183.9        | -         |
| miles1000 | 1623    | 1666      | -                | 1666            | 1670.5        | -         |
| miles1500 | 3239    | 3354      | -                | 3354            | 3357.2        | -         |
| fpos2.1   | 3403    | 3403      | -                | 3403            | 3403.0        | -         |
| fpos2.1.2 | 1668    | 1668      | -                | 1668            | 1668.0        | -         |
| fpos2.1.3 | 1636    | 1636      | -                | 1636            | 1636.0        | -         |
| mug88     | 164     | 178       | 178.0            | 178             | 178.0         | -         |
| mug88.2   | 162     | 178       | 178.0            | 178             | 178.0         | -         |
| mug100.1  | 188     | 202       | 202.0            | 202             | 202.0         | -         |

Continued on next page
Table 3 Continued from previous page

| Graph | BLS | MBSC | MDS(5)+LS | EXSCOL | MA-MSCP | HESA |
|-------|-----|------|-----------|--------|---------|------|
|        |     |      |           |        |         |      |
| Name   |      |      |           |        |         |      |
| Name   |      |      |           |        |         |      |
|        |      |      |           |        |         |      |
| Name   |      |      |           |        |         |      |
| Name   |      |      |           |        |         |      |
| Name   |      |      |           |        |         |      |
| Name   |      |      |           |        |         |      |
| Name   |      |      |           |        |         |      |
| Name   |      |      |           |        |         |      |
| Name   |      |      |           |        |         |      |

Continued on next page
| Name     | f | b | LB | f | UB | f | UB | f | UB | f | UB | f | UB |
|----------|---|---|----|---|----|---|----|---|----|---|----|---|----|
| flat300  | 1548 | 3966 | -   | 3966 | 3966.0 | -   | 1525 | 1511.4 | 3966 | 3966.0 | 1548 | 1530.3 | 3966 | 3966.0 |
| flat300  | 1547 | 4238 | -   | 4238 | 4313.4 | -   | 1532 | 1515.3 | 4282 | 4286.1 | 1541 | 1503.9 | 4261 | 4389.4 |
| flat300  | 6601 | 25500 | -   | 25500 | 25500.0 | -   | 6600 | 6571.8 | 25500 | 25500.0 | 6433 | 6121.5 | 25500 | 25500.0 |
| flat300  | 6640 | 30100 | -   | 30100 | 30100.0 | -   | 6640 | 6608.5 | 30100 | 30100.0 | 6402 | 6347.7 | 30100 | 30100.0 |
| flat300  | 6632 | 37164 | -   | 37167 | 37167.0 | -   | 6632 | 6832.3 | 37167 | 37213.2 | 6330 | 6074.6 | 38213 | 39722.7 |
| flat300  | 1193 | 1350 | -   | 1350 | 1350.0 | -   | -   | -   | -   | -   | -   | -   | -   |
| flat300  | 1189 | 1350 | -   | 1350 | 1350.0 | -   | -   | -   | -   | -   | -   | -   | -   |
| flat300  | 1278 | 1350 | -   | 1350 | 1350.0 | -   | -   | -   | -   | -   | -   | -   | -   |
| flat300  | 1282 | 1350 | -   | 1350 | 1350.0 | -   | -   | -   | -   | -   | -   | -   | -   |
| flat300  | 2331 | 2832 | -   | 2706 | 2742.6 | -   | 2329 | 2313.7 | 2632 | 2641.9 | 2329 | 2324.3 | 2661 | 2733.1 |
| flat300  | 2348 | 2632 | -   | 2724 | 2756.2 | -   | 2343 | 2315.7 | 2642 | 2643.4 | 2348 | 2335.0 | 2690 | 2736.0 |
| flat300  | 2610 | 3487 | -   | 3491 | 3491.0 | -   | 2591 | 2545.3 | 3866 | 3868.9 | 2593 | 2569.1 | 3943 | 4048.4 |
| flat300  | 2628 | 3550 | -   | 3506 | 3511.8 | -   | 2610 | 2572.4 | 3921 | 3928.5 | 2622 | 2587.2 | 3926 | 4032.4 |
| flat300  | 3003 | 3153 | -   | 3166 | 3176.8 | -   | 2997 | 2964.4 | 3153 | 3159.4 | 3003 | 3000.4 | 3178 | 3204.3 |
| flat300  | 3305 | 3365 | -   | 3366 | 3375.1 | -   | 3305 | 3304.1 | 3366 | 3371.9 | 3305 | 3304.1 | 3379 | 3416.2 |
| flat300  | 3657 | 4515 | -   | 4700 | 4773.3 | -   | 3619 | 3597.1 | 4515 | 4525.4 | 3638 | 3617.0 | 4648 | 4700.7 |
| flat300  | 3698 | 4544 | -   | 4722 | 4805.7 | -   | 3684 | 3627.4 | 4544 | 4550.0 | 3697 | 3683.2 | 4696 | 4740.3 |
| latin    | 40950 | 41444 | -   | 41444 | 41481.5 | -   | 40950 | 40950.0 | 42223 | 42327.2 | -   | -   | -   | -   |
| latin    | 15091 | 132483 | -   | -   | -   | -   | 15091 | 15077.6 | 132515 | 132682.0 | -   | -   | -   | -   |
| latin    | 33033 | 473234 | -   | -   | -   | -   | 33033 | 33018.8 | 473234 | 473213.0 | -   | -   | -   | -   |
| latin    | 442 | 443 | 443.0 | 443 | 443.0 | 442 | 444 | 441.4 | 443 | 447.9 | 442 | 442.0 | 443 | 443.0 | 442 | 442.0 | 443 | 443.0 | 442 | 442.0 | 443 | 443.0 | 442 | 442.0 | 443 | 443.0