Hybrid evolutionary search for the minimum sum coloring problem of graphs

Yan Jin [a,b] and Jin-Kao Hao [b,c,*]

[a] *School of Computer Science and Technology, Huazhong University of Science and Technology, No. 1037,Luoyu Road, Wuhan, China*

[b] *LERIA, Université d'Angers, 2 bd Lavoisier, 49045 Angers, France*

[c] *Institut Universitaire de France, Paris, France*

Information Sciences, accpeted February 2016

Abstract

Given a graph G, a proper k-coloring of G is an assignment of k colors $\{1, \ldots, k\}$ to the vertices of G such that two adjacent vertices receive two different colors. The minimum sum coloring problem (MSCP) is to find a proper k-coloring while minimizing the sum of the colors assigned to the vertices. This paper presents a stochastic hybrid evolutionary search algorithm for computing upper and lower bounds of this NP-hard problem. The proposed algorithm relies on a joint use of two dedicated crossover operators to generate offspring solutions and an iterated double-phase tabu search procedure to improve offspring solutions. A distance-and-quality updating rule is used to maintain a healthy diversity of the population. We show extensive experimental results to demonstrate the effectiveness of the proposed algorithm and provide the first landscape analysis of MSCP to shed light on the behavior of the algorithm.

Keywords: Sum coloring; Hybrid search; Evolutionary computing; Local optimization; Tabu Search.

1 Introduction

Given a simple undirected graph $G = (V, E)$ with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set $E \subset V \times V$, let $\{1, \ldots, k\}$ be the set of k different colors. A proper k-coloring c of G is a mapping $c : V \rightarrow \{1, \ldots, k\}$ such that $c(v_i) \neq c(v_j)$, for all

* Corresponding author.
Email addresses: yanjin.china@hotmail.com (Yan Jin), jin-kao.hao@univ-angers.fr (Jin-Kao Hao).

$\{v_i, v_j\} \in E$. 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 no two vertices of a color class are linked by an edge, i.e., $\forall u, v \in V_i$ $(i = 1, \ldots, k), \{u, v\} \notin E$. The cardinality of a color class V_i is given by the number of its vertices, and is usually denoted by $|V_i|$. The conventional graph coloring problem (GCP) is to color a graph G with a minimum number $\chi(G)$ of colors, $\chi(G)$ is the so-called chromatic number of G. The minimum sum coloring problem (MSCP) studied in this paper is to find a proper k-coloring c of a graph G which minimizes the sum of the colors assigned to the vertices of G [22,36].

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

where $|V_l|$ is the cardinality of V_l, $|V_1| \geq \ldots \geq |V_k|$ and k is larger than or equal to the chromatic number $\chi(G)$ of G in the GCP. Throughout the paper, we assume that the color classes of a k-coloring are sorted in non-increasing order of their cardinality. The minimum sum of colors for MSCP is called the chromatic sum of G, and is denoted by $\sum(G)$. MSCP has practical applications in areas like VLSI design, scheduling and resource allocation [27].

Notice that although MSCP is tightly related to the conventional GCP, MSCP and GCP have different optimization objectives (minimization of the sum of colors vs. minimization of the number of colors). Figure 1 provides an example for MSCP which also illustrates the relation with GCP. 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 leads to a suboptimal sum of 18.

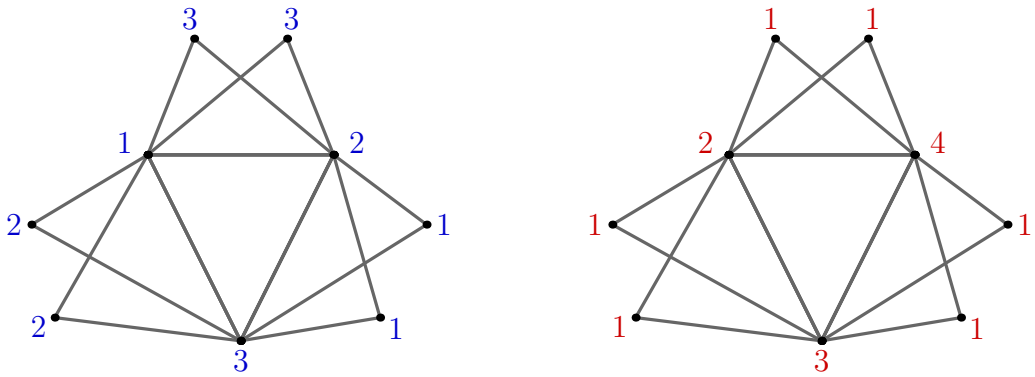

Fig. 1. An illustrative example for MSCP

From a theoretical point of view, MSCP is proved to be NP-hard [22]. There exist a number of studies on specific graphs. For instance, a polynomial time algorithm is available for finding the chromatic sum of a tree [22]. Several

approximation algorithms are also proposed for bipartite graphs, chain bipartite graphs and interval graphs, k-split graphs, and line graphs of trees [2,4,16,23,21,34]. Unfortunately, MSCP remains computationally difficult and challenging in the general case.

Given the hardness of the general MSCP, a number of heuristic algorithms have been proposed to obtain suboptimal solutions (upper bounds)or to compute lower bounds in acceptable computing time.

In 2007, Kokosiński and Kwarciany [20] presented the first parallel genetic algorithm which employs assignment and partition crossovers, first-fit mutation, and proportional selection with an island migration model. They showed the first computational results in reference to theoretical upper bounds on 16 small DIMACS graphs. In 2009, Li et al. [25] experimented two greedy algorithms (MDSAT & MRLF) which are adaptations of two well-known GCP heuristics called DSATUR [6] and RLF [24] to MSCP. Their experimental results showed that MDSAT & MRLF perform better than DSATUR & RLF. Later in 2010, Moukrim et al. [29] proposed a clique decomposition technique for computing a lower bound of MSCP. In 2010, Bouziri and Jouini [5] adapted the well-known Tabucol coloring algorithm of [15] to MSCP. The experimental results applied on seven small DIMACS instances are better than those of the greedy algorithms MDSAT & MRLF. In 2011, Helmar and Chiarandini [14] elaborated a local search heuristic (MDS(5)+LS) to find upper and lower bounds of MSCP, which combines variable neighborhood search and iterated local search to oscillate between feasible and infeasible regions. Comparative results showed that MDS(5)+LS attains new bounds for 27 out of 38 tested instances and outperforms the above three algorithms. In 2012, Benlic and Hao [3] developed a breakout local search algorithm (BLS) which jointly uses a local search and adaptive perturbation strategies. They reported improved upper bounds for 4 instances out of 27 tested graphs. In 2012, Wu and Hao [37] devised an effective heuristic using independent set extraction (EXSCOL) which performs especially well on large graphs with more than 500 vertices. Later in 2013, the same authors [38] applied this approach for lower bound computation. In 2014, Moukrim et al. [30] introduced a memetic algorithm using a two-parent crossover combined with a hill-climber and "destroy and repair" procedure (MA), and reported high quality upper and lower bounds on 81 tested instances. The same year, Jin et al. [17] presented another memetic algorithm based on tabu search and a multi-parent crossover (MASC). MASC discovered 15 new upper bounds out of 77 tested graphs.

In this work, we are interested in the computation of both upper and lower bounds of MSCP. For this, we introduce an effective stochastic hybrid evolutionary search algorithm (HESA) whose main contributions can be summarized as follows.

- From the algorithm perspective, the HESA approach integrates several special features to ensure a high search efficiency. These include an original recombination mechanism to generate offspring solutions and an iterated double-phase tabu search procedure to ensure local optimization. The solution recombination mechanism combines a diversification-guided crossover operator and a grouping-guided crossover operator to create diversified and promising offspring solutions. The double-phase tabu search procedure is designed to handle both feasible and unfeasible solutions. A dedicated perturbation mechanism is also introduced to escape local optima. Finally, a population updating procedure is employed to maintain a high-quality population with a healthy diversity.
- From the computational perspective, we evaluate the HESA approach on 94 well-known DIMACS and COLOR 2002-2004 benchmark instances. The computational results show that HESA can achieve the best-known result for most of these benchmark instances established by several state-of-the-art algorithms. Moreover, HESA finds 51 improved best solutions (24 improved upper bounds and 27 improved lower bounds).

The rest of the paper is organized as follows. Section 2 presents the proposed algorithm for computing upper bounds of MSCP. Section 3 explains the adjustments of the proposed algorithm to compute lower bounds. Section 4 shows extensive computational results of HESA and comparisons with the state-of-the-art algorithms in the literature. Before concluding, Section 5 investigates and analyzes some key issues of the proposed algorithm.

2 A hybrid search algorithm

The proposed hybrid search algorithm follows the general memetic framework which combines population-based evolutionary search and local optimization [28,32]. In principle, the HESA algorithm repeatedly alternates between the double-crossover procedure that generates new offspring solutions (Section 2.3) and the iterated double-phase tabu procedure (IDTS) that optimizes the newly generated offspring solutions (Section 2.4). As soon as an offspring solution is improved by IDTS, the population is accordingly updated based on the solution quality and population diversity (Section 2.5).

The general scheme of HESA for MSCP is summarized in Algorithm 1. HESA starts with an initial population of solutions (line 3, see Sect. 2.2) and then repeats a number of generations until a stopping condition is met (lines 5–15, in our case, a time limit is used as stopping condition). At each generation, two solutions from the population are selected at random to serve as parent solutions (line 6). Then, the double-crossover recombination procedure is employed to create two offspring solutions (line 7) which are further improved

by the iterated double-phase tabu procedure (IDTS) (lines 9). Subsequently, the population updating rule decides whether the improved solution should be inserted into the population and which existing solution is to be replaced (lines 13). In the following subsections, we describe these basic components.

Algorithm 1 An overview of the HESA algorithm for MSCP

1: **Input**: A graph G, population size p
2: **Output**: The best sum coloring c_* found and its sum of colors f_*
3: Population_Initialization(P, p) /* Generate p initial solutions, Sect. 2.2 */
4: $f_* \leftarrow \min_{c \in P} f(c)$ /* f_* records the best objective value found so far */
5: **repeat**
6: $(P_1, P_2) \leftarrow$ Selection(P) /* Select at random parents for crossover */
7: $(o_1, o_2) \leftarrow$ Double-Crossover(P_1, P_2) /* Use two crossover operators to generate two offspring o_1 and o_2, Sect. 2.3 */
8: **for** each $i \in \{1, 2\}$ **do**
9: $o \leftarrow$ IDTS(o_i) /* Improve o_i with the IDTS procedure, Sect.2.4*/
10: **if** $f(o)$ is better than f_* **then**
11: $f_* \leftarrow f(o)$; $c_* \leftarrow o$
12: **end if**
13: Population_Updating(P, o) /* Use offspring o to update the population, Sect. 2.5 */
14: **end for**
15: **until** Stopping condition is met
16: **return** f_*, c_*

2.1 Search space and evaluation function

A proper k-coloring satisfies the coloring constraint which insures that any two adjacent vertices $\{u, v\} \in E$ belong to two different color classes. A k-coloring is improper if the coloring constraint is violated. The search space of HESA contains the set Ω of all possible partitions of V into at most $|V|$ color classes including both the proper and improper colorings. Given a proper coloring, its objective value is given by the function f defined by Eq. (1) (i.e., the sum of colors). For two proper coloring solutions $c_1 \in \Omega$ and $c_2 \in \Omega$, c_1 is better than c_2 if and only if $f(c_1) < f(c_2)$. We discuss the evaluation of improper colorings in Section 2.4.1.

2.2 Initial population

The initial population of HESA is composed of p (population size, $p = 20$ in this work) proper colorings. To obtain the initial colorings, any coloring

algorithm could be employed. In our case, we use the maximum independent set algorithm SBTS [18] to generate each initial coloring. SBTS builds in a step-by-step way k (k is not fixed) mutually disjoint independent sets (color classes). At each step, we apply SBTS to extract a maximal independent set V_i from the graph G and then remove from G the vertices of V_i and their incident edges. This process is repeated until the graph becomes empty. The resulting independent sets $\{V_1, \ldots, V_k\}$ form a proper k-coloring. Each new k-coloring is inserted into the population P if it does not duplicate any existing individual of the population. Otherwise, this k-coloring is discarded and another new coloring is generated. This initialization process is repeated until the population is filled up with p individuals (i.e., proper colorings). These individuals are generally of good quality and serve as inputs for the double-crossover recombination procedure.

The choice of SBTS for population initialization is motivated by two factors. First, SBTS is an effective algorithm for the maximum independent set problem. The resulting mutually disjoint independent sets form not only a proper coloring, but also a solution of high quality. Second, given that SBTS is a stochastic algorithm, each SBTS run generally leads to a different k-coloring. This feature favors the diversity of the initial population and prevents the search process from falling into local optima too quickly.

2.3 A double-crossover recombination procedure

Recombination is an important ingredient for a population-based memetic approach. For HESA, we propose a double-crossover recombination procedure which jointly uses two different operators to generate suitable offspring solutions: The diversification-guided crossover operator and the grouping-guided crossover operator. At each generation, HESA first randomly chooses two parents from the population which have not been selected to serve as parents in the previous generations, and then employs the two crossover operators to generate two offspring solutions respectively. Each offspring solution is finally submitted to the iterated double-phase tabu search procedure for quality improvement (minimization of the sum of colors). These dedicated crossover operators can be considered as being derived from a perspective that accords with the "structured combination" approach put forward in [9]. They also follow the general design principle of semantic recombination operators with respect to the optimization objective [13]. A fundamental notion of such a design is to explicitly select a problem-specific heuristic to construct offspring solutions, using semantic information contained in the parent solutions as a mechanism to generate and make choices presented by the decision rules of the heuristic.

2.3.1 Diversification-guided crossover

The diversification-guided crossover (DGX) aims to generate offspring solutions of reasonable quality and diversity. Given two parent solutions (i.e., two proper colorings) $P_1 = \{V_1^1, \ldots, V_{k_1}^1\}$ and $P_2 = \{V_1^2, \ldots, V_{k_2}^2\}$ (assume that $k_1 \geq k_2$). The offspring solution $o = \{V_1^o, \ldots, V_{k_o}^o\}$ is built as follows.

Step 1: We first transmit the vertices that share the same colors in both parents such that they keep their colors in the offspring. Formally, we set $V_i^o = V_i^1 \cap V_i^2, i = 1, \ldots, k_2$.

Step 2: Let $U = V \setminus \bigcup_{i=1}^{k_2} V_i^o$ be the set of unassigned vertices in o. We pick randomly m (see below) vertices from U to form U_m. Then for each vertex $v \in U_m$, v conserves its color of parent P_1 in o, i.e., if $v \in V_i^1$ $(i = 1, \ldots, k_1)$, v is added in V_i^o.

Step 3: Finally, for each remaining vertex u of $U \setminus U_m$, u conserves its color of parent P_2 in o, i.e., if $u \in V_j^2 (j = 1, \ldots, k_2)$, u is added in V_j^o.

The DGX operator uses the parameter m to control the relative importance of P_1 and P_2 with respect to the generated offspring solution o. In order to avoid a dominance of one parent over the other parent, we set $m = \lfloor \frac{1}{3}|U| \rfloor + rand(\lfloor \frac{1}{3}|U| \rfloor)$ where $|U|$ is the cardinality of U and $rand(\mathcal{N})$ gives a random number in $\{1, \ldots, \mathcal{N}\}$. This value for m ensures that o is separated from either parent by at least $\lfloor \frac{1}{3}|U| \rfloor$ vertices.

The above steps have a time complexity of $O(|V|)$. Since we need to maintain the order $|V_1| \geq \ldots \geq |V_k|$ of the color classes of the offspring, the total complexity for each crossover operation is $O(|V| \times k)$.

Figure 2 shows an illustrative example with 3 color classes and 10 vertices represented by $A, B, \ldots, J$. At step 1, the single vertex {A} shared by both parents is directly transmitted to the offspring solution. Then, the remaining vertices $\{B, C, D, E, F, G, H, I, J\}$ are collected in U and m is assumed to be 4. At step 2, we randomly choose $m = 4$ vertices from U (say $\{B, E, H, I\}$) and transfer them from parent P_1 in the offspring solution. Finally, the remaining unassigned vertices (i.e., $\{C, D, F, G, J\}$) are preserved from parent P_2 to complete the offspring solution.

One observes that the offspring solution generated by the DGX operator may be an improper k-coloring. If this happens, it is repaired by the iterated double-phase tabu search procedure described in Section 2.4.

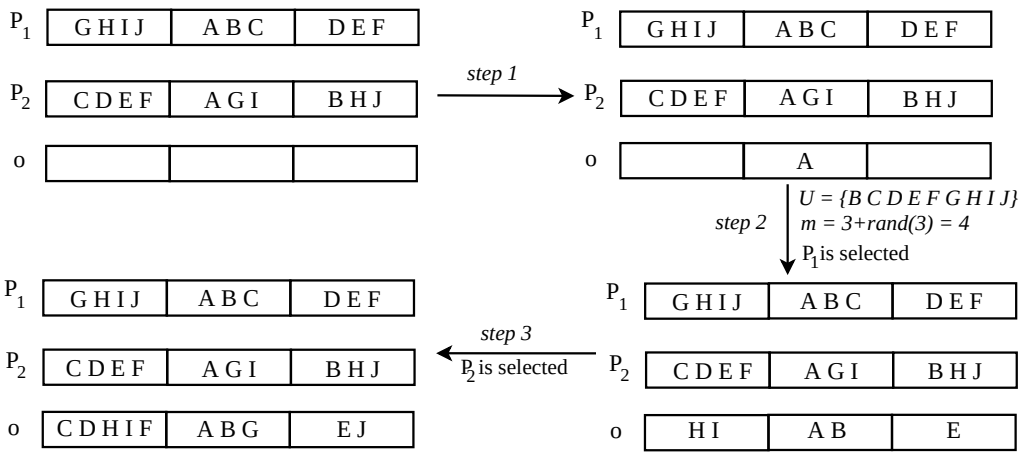

Fig. 2. An illustrative example of the DGX crossover.

2.3.2 Grouping-guided crossover

Unlike the previous DGX operator, the grouping-guided crossover (GGX) aims to transmit whole color classes from parents to the offspring solution. Given two parents (i.e., two proper colorings) $P_1 = \{V_1^1, \ldots, V_{k_1}^1\}$ and $P_2 = \{V_1^2, \ldots, V_{k_2}^2\}$ (assume that $k_1 \geq k_2$). The offspring solution $o = \{V_1^o, \ldots, V_{k_o}^o\}$ is constructed in two steps.

Step 1: We generate an integer $l = rand(\lceil \frac{2}{3} k_2 \rceil)$ and transmit the first l color classes from one parent (randomly selected, say P_2) to construct a partial offspring solution $o = \{V_1^o, \ldots, V_l^o\}$. We remove the vertices v ($v \in o$) from the other parent (P_1).

Step 2: We transmit the non-empty color classes of P_1 to form the $l+1, \ldots, k_o$ color classes of offspring o such that a complete offspring solution is constructed.

The choice of the value of l is based on the consideration that we wish to introduce some randomness when deciding the number of transmitted color classes while ensuring some distance between the offspring and each of its parent. An example of the GGX crossover is provided in Figure 3 where l takes the value of 1. The time complexity of the GGX crossover is $O(|V| \times k)$.

Contrary to the DGX crossover, the GGX operator ensures that offspring solutions are always proper colorings. By conserving pertinent properties (color classes) of parent solutions, the offspring colorings are generally of good quality. In the shown example, the offspring has a better quality than its parents although this is not generally the case.

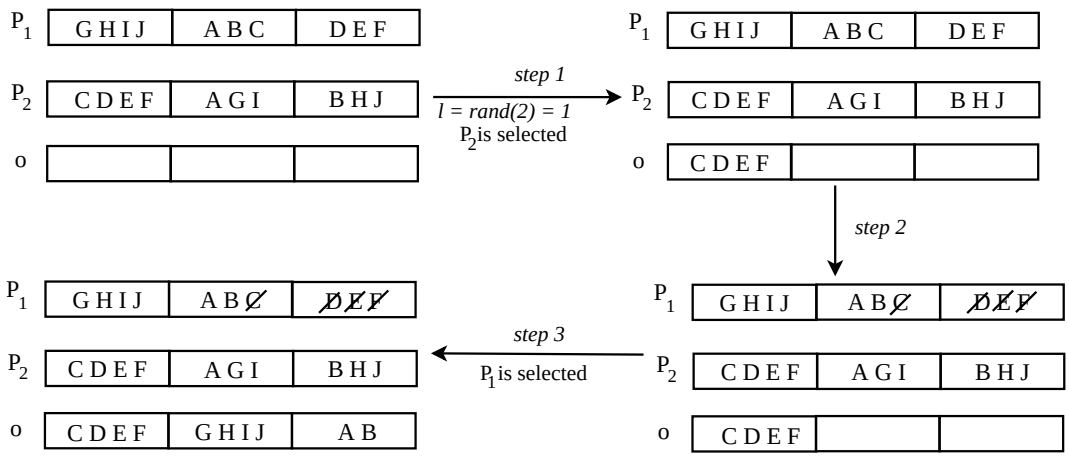

Fig. 3. An illustrative example of the GGX crossover.

2.4 An iterated double-phase tabu search procedure

Since the recombination procedure may lead to both feasible and unfeasible colorings, we devise an iterated double-phase tabu search (IDTS) able to repair unfeasible solutions while minimizing the sum of colors.

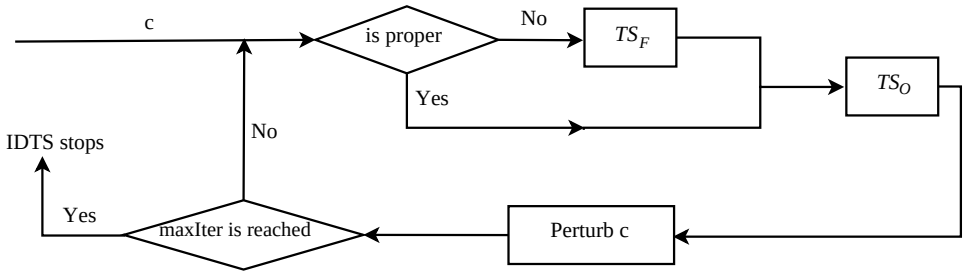

Fig. 4. An illustration for the IDTS procedure.

The overall IDTS procedure is illustrated in Figure 4. It uses a double-phase tabu search for intensified search and a perturbation mechanism for diversification. The intensification phase applies a tabu search procedure (denoted by TS_O) to improve the quality of a proper coloring according to the objective function and another tabu search procedure (denoted by TS_F) to reestablish the feasibility of an improper coloring. IDTS starts by checking whether the given solution c is a proper coloring. If this is the case, TS_O is called to improve its sum of colors. Otherwise, TS_F is applied for conflict-repairing to attain a proper coloring which is further improved by TS_O according to the objective function. Notice that, to repair an improper coloring, TS_F may increase the number of used colors k until a proper coloring is obtained. The perturbation mechanism is applied to escape local optima when TS_O stagnates, i.e., no improved solution is found after μ_O consecutive iterations. The perturbed solution is submitted to the next round of the double-phase tabu search process until a maximum number of iterations $maxIter$ is reached, where $maxIter$ is

a fixed parameter.

2.4.1 A double-phase tabu search

The two tabu search procedures TS_O and TS_F follow the general principle of the tabu search methodology [11]. Both procedures iteratively visit a series of solutions following the given neighborhood (see below). At each iteration, a best neighboring solution is chosen (ties are broken randomly) to replace the current solution, even if the selected solution does not improve the current solution. To avoid cycling, a tabu list is used to avoid a visited solution to be revisited during the next tt iterations (tt is called the tabu tenure). Nevertheless, a forbidden solution is always accepted if it is better than the best solution found so far (called aspiration criterion). The tabu search TS_F stops once a proper coloring is obtained and the TS_O procedure stops when the best solution cannot be improved within a given number of solution transitions.

Although both TS_F and TS_O employ the scheme of tabu search, they use different neighborhoods, evaluation functions and tabu tenures.

- **Neighborhood:** The neighborhood of the double-phase tabu search can be described by the "one-move" operator $mv(v, V_i, V_j)$ which displaces a vertex v from its original color class V_i to another color class V_j ($i \neq j$). Given a k-coloring $c = \{V_1, \ldots, V_k\}$, a vertex $v \in V_i$ is conflicting if v shares the same color class with at least one adjacent vertex v', i.e., $\exists v' \in V_i$ and $v' \neq v$, $\{v', v\} \in E$. The TS_F procedure (to repair improper colorings) operates with conflicting vertices. At each iteration, it displaces a single non-tabu conflicting vertex v. For a k-coloring c with $nb_{conf}(c)$ conflicting vertices, the size of this neighborhood is bounded by $O(nb_{conf}(c) \times k)$. The TS_O procedure (to optimize the sum of colors) applies $mv(v, V_i, V_j)$ to move a non-tabu vertex $v \in V_i$ to another color class V_j such that the resulting k-coloring remains proper ($\forall v' \in V_j, \{v', v\} \notin E$). Hence, the size of this neighborhood is bounded by $O(|V| \times k)$. In our implementation, we employ an incremental evaluation technique [7,8] to efficiently evaluate the whole neighborhood.
- **Evaluation function:** Both TS_F and TS_O scan their whole neighborhood to find a best neighboring solution to replace the current solution. The TS_F procedure evaluates all the neighbor solutions by considering both the variation in the number of conflicting vertices $\Delta conf(v, V_i, V_j)$ and the variation in the sum of colors $\Delta f(v, V_i, V_j)$ when applying the $mv(v, V_i, V_j)$ operator. The evaluation of each candidate move is given in Eq. (2) which is adopted from [3]. For two neighboring solutions s' and s'', s' is better than s'' if and only if $\Delta(s') < \Delta(s'')$.

$$\Delta = \Delta conf(v, V_i, V_j) \times \Delta f', \text{where}$$
$$\Delta f' = \begin{cases} abs(\Delta f(v, V_i, V_j)) + k + 1, \text{if } \Delta f(v, V_i, V_j) < 0 \\ k - \Delta f(v, V_i, V_j) + 1, \text{otherwise} \end{cases} \quad (2)$$

The TS_O procedure evaluates all the neighboring solutions by only considering the variation in the sum of colors $\Delta f(v, V_i, V_j)$ in terms of the objective function Eq. (1).

- **Tabu tenure:** Each time a $mv(v, V_i, V_j)$ move is performed to transform the incumbent coloring, vertex v is forbidden to join the color class V_i for the next tt iterations. To determine the tabu tenure t, we adopt the dynamic tuning technique introduced in [8] which is defined by the number of conflicting vertices adjusted by a random number. For TS_F, our tabu tenure tt_F takes the form $tt_F = \lfloor 0.6 * nb_{conf}(c) \rfloor + rand(10)$ where c is the incumbent coloring and $nb_{conf}(c)$ is the number of conflicting vertices in c. The tabu tenure tt_O for TS_O is similarly determined by $tt_O = \lfloor 0.1 * f(c) \rfloor + rand(10)$.
- **Tabu list implementation:** To implement the tabu list conveniently, we use a two-dimensional table $T[|V|, k]$ where the first and the second element of T corresponds to the number of vertices and colors (In our case, k is set to be the maximum number of colors that can be used (i.e., $|V|$). Each entry $T[v, i]$ is initialized to 0. When a move $mv(v, V_i, V_j)$ is performed at iteration $Iter$, $T[v, i]$ is set to $tt + Iter$ ($tt = tt_F$ or tt_O) where tt is the tabu tenure of the performed move. Now to know if a move $mv(v, V_i, V_j)$ is forbidden by the tabu list at a subsequent iteration $Iter_{cur}$, it suffices to compare $Iter_{cur}$ with $T[v, i]$: If $Iter_{cur} \leq T[v, i]$, the move is forbidden; otherwise the move is eligible.

2.4.2 Perturbation strategy

The above double-phase tabu search is generally able to attain solutions of good quality but can get stuck in local optima occasionally. To help the procedure to continue its search, we apply a perturbation mechanism to bring the search out of the problematic local optima. The perturbation makes a $(1, r)$-swap ($r = 0, 1, 2, \ldots$) move if the current iterations $iter_{cur}\%100 \neq 0$; otherwise, it replaces the current solution by the local optimal solution. The $(1, r)$-swap procedure operates as follows. First, we randomly choose a vertex v ($v \in V_i$) and a color class V_j ($V_j \neq V_i$). Then, we identify all the vertices v' in V_j which are adjacent to v ($\{v', v\} \in E$). Finally, we move v from V_i to V_j and move all the vertices v' from V_j to V_i. These vertices are forbidden to move back to their original color classes for the next tt_P iterations ($tt_P = rand(10) + \lfloor 0.1 * f(c) \rfloor$, where c is the current solution). This perturbation mechanism may introduce conflicts in the current solution and enables the search to transit between feasible and infeasible regions. From this perturbed solution, the double-phase tabu search procedure is relaunched.

2.5 The population updating procedure

In order to prevent the search process from premature convergence, the population updating procedure is critical for our hybrid search algorithm. The population updating rule decides whether and how an offspring solution, which is optimized by the IDTS procedure, should replace an existing individual of the population. Basically, our updating rule is based on both solution quality and distance between solutions in the population [26,33,35].

Definition 1. Distance between two individuals D_{ij} [17] : Given two individuals $P_i = \{V_1^i, \ldots, V_{k_i}^i\}$ and $P_j = \{V_1^j, \ldots, V_{k_j}^j\}$, the distance between P_i and P_j is defined as the total number of vertices whose corresponding colors are different in the two individuals, $D_{ij} = |\{v \in V : v \in V_{k_i}^i, v \in V_{k_j}^j, k_i \neq k_j\}|$.

Algorithm 2 The population updating procedure

1: **Input**: Population $P = \{P_1, \ldots, P_p\}$ and offspring P_o
2: **Output**: The updated population P
3: $D_{min} \leftarrow +\infty$
4: **for** $i \in \{1, \ldots, p\}$ **do**
5: Calculate the distance D_{oi} between P_o and P_i
6: /*Identify the closest individual P_d with the minimum distance D_{min} to P_o*/
7: **if** $D_{oi} < D_{min}$ **then**
8: $D_{min} \leftarrow D_{oi}$
9: $P_d \leftarrow P_i$
10: **end if**
11: **end for**
12: Identify the worst individual P_w with the largest objective value in P
13: **if** $D_{min} \geq d_\alpha$ and $f(P_o) \leq f(P_w)$ **then**
14: Replace P_w with P_o: $P \leftarrow P \bigcup \{P_o\} \backslash \{P_w\}$
15: **else**
16: **if** $f(P_o) \leq f(P_d)$ **then**
17: Replace P_d with P_o: $P \leftarrow P \bigcup \{P_o\} \backslash \{P_d\}$
18: **else**
19: **if** $rand(0, 1) \leq 0.1$ **then**
20: Replace P_w with P_o: $P \leftarrow P \bigcup \{P_o\} \backslash \{P_w\}$
21: **end if**
22: **end if**
23: **end if**

The general scheme of our population updating strategy is described in Algorithm 2. In order to update the population, we calculate the distance D_{oi} between the offspring P_o and any existing solution of the population $P_i \in P$ $(i = 1, \ldots, p)$, record the minimum distance D_{min}, and identify the closest individual P_d with respect to P_o (lines 3–11). Meanwhile, the worst individual P_w with the largest objective value ($P_w \in P$) is also identified (line 12).

Now if P_o is sufficiently separated from any solution of P, indicated by $D_{min} \geq d_\alpha$, where $d_\alpha = 0.1 \times |V|$ and P_o is no worse than P_w, then the offspring P_o is inserted into the population and replaces the worst individual P_w (lines 13–14). Otherwise, if P_o is no worse than the closest individual P_d, then P_o replaces P_d (lines 16–17). Otherwise, the worst individual P_w is replaced by P_o with a small probability of 0.1 (lines 19–20). This last case corresponds to the situation where P_o is close to P_d ($D_{min} < d_\alpha$) and worse than P_d ($f(P_o) > f(P_d)$), but P_d is different from P_w. In this case, there is no reason to replace P_d by P_o given that P_o is worse than P_d and it is more reasonable to use P_o to replace the worst individual P_w. On the other hand, replacing P_w by P_o would decrease the population diversity (since P will contain two close individuals P_o and P_d). For this reason, we only authorize this replacement occasionally, controlled with a small probability (line 19). The computational complexity of the population updating procedure is $O(p \times |V|)$.

2.6 Discussions

In this section, we discuss the relation between HESA and two previous algorithms (MASC) [17] and (MA) [30] for MSCP. Indeed, all these algorithms follow the general memetic framework which combines population-based search with local optimization. However, HESA distinguishes itself from these algorithms by its key components.

First, HESA employs a maximum independent set algorithm to generate proper initial solutions of high quality while MASC and MA use respectively the TabuCol procedure [15] and a greedy coloring heuristic [25] for this purpose. Second, for solution recombination, HESA uses two different crossover operators which can generate both feasible and infeasible solutions while MASC and MA allow only feasible solutions. Third, HESA applies an iterated two-phase tabu search procedure to make transitions between feasible and infeasible regions while MASC and MA only explore feasible solutions. Finally, HESA employs a more elaborated pool updating rule to decide whether an offspring solution should replace the worst (or closest) individual in the population. In MASC, this is achieved by a "scoring" function combining solution quality and distance while in MA only solution quality is considered.

As shown in Section 4, the proposed HESA algorithm equipped with its particular features attains a highly competitive performance for computing the upper and lower bounds of MSCP.

3 The lower bounds of MSCP

Given an undirected graph $G = (V, E)$ and its partial graph $G' = (V, E')$ $(E' \subset E)$, it is evident that any proper coloring of G must be a proper coloring of G' and that the chromatic sum of G' is a lower bound for the chromatic sum of G. Hence, we could try to find a partial graph of the original graph to maximize this lower bound. In the literature, several studies were devoted to decompose the original graph into partial graphs like trees, paths and cliques [14,21,29]. Moukrim *et al.* showed that clique decomposition provides better lower bounds than tree and path decompositions [29]. Let $c = \{S_1, S_2, \ldots, S_k\}$ be a clique decomposition of G, then the following quantity gives a lower bound for MSCP.

$$f_{LB}(c) = \sum_{l=1}^{k} \frac{|S_l|(|S_l|+1)}{2} \tag{3}$$

Nevertheless, as shown in [30], clique decomposition for the lower bounds of MSCP is itself a NP-hard problem. To obtain a clique decomposition, one popular approach is to find a proper coloring of the complementary graph $\bar{G}$ of G [14,30,38] since each color class of $\bar{G}$ is a clique of G.

In this work, we apply HESA to color the complementary graph $\bar{G}$ in order to obtain lower bounds. For this purpose, we need to make the following adjustments to the HESA algorithm.

- To evaluate the colorings, we use the objective function defined by Eq. (3) instead of sum of colors. For the purpose of computing lower bounds, this objective function is to be maximized. For two proper colorings c_1 and c_2 (of $\bar{G}$), c_1 is better than c_2 if and only if $f_{LB}(c_1) > f_{LB}(c_2)$.
- The evaluation function used in the double-phase tabu search needs to be adjusted. The TS_F procedure (for conflict repairing) applies the evaluation function Eq. (4) to evaluate a neighboring coloring.

$$\begin{aligned} &\Delta = \Delta conf(v, S_i, S_j) \times \Delta f'_{LB}, \text{where} \\ &\Delta f'_{LB} = \begin{cases} \Delta f_{LB}(v, S_i, S_j) + k + 1, \text{if } \Delta conf(v, S_i, S_j) < 0 \\ k - \Delta f_{LB}(v, S_i, S_j) + 1, \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

  where $\Delta conf(v, S_i, S_j)$ is the variation in the number of conflicting vertices and $\Delta f_{LB}(v, S_i, S_j)$ is the variation in the objective value of Eq. (3). For two neighboring colorings s' and s'', s' is better than s'' if and only if $\Delta(s') < \Delta(s'')$. The TS_O procedure evaluates the neighboring colorings by only considering the variation in terms of the objective function of Eq. (3).
- For a k-coloring $c = \{S_1, \ldots, S_k\}$, it is no more necessary to sort its color classes S_i $(i = 1, 2, \ldots, k)$.

4 Experimental results

4.1 Benchmark instances

To evaluate the efficiency of our proposed HESA algorithm, we carried out experiments on two sets of 94 benchmark instances: 58 COLOR 2002-2004 instances and 36 DIMACS instances [1]. These instances were initially collected for the "Second DIMACS Implementation Challenge on Maximum Clique, Graph Coloring, and Satisfiability" (`http://dimacs.rutgers.edu/Challenges/`) and extended for the series of competition on "Graph Coloring and its Generalizations" (`http://mat.gsia.cmu.edu/COLOR04/`). In particular, the well-known DIMACS instances refer to graphs of different topologies and densities:

- Twelve classic random graphs (DSJCχ.γ, χ = 125, 250, 500 and 1000, γ = 0.1, 0.5 and 0.9)
- Three geometric graphs (DSJRχ.γ, χ = 500, $\gamma = 0.1c$, 0.1 and 0.5)
- Six flat graphs (flat300_χ_0, flat1000_γ_0, χ = 20, 26 and 28, γ = 50, 60 and 76)
- Twelve Leighton graphs (le450_15χ, le450_25χ, $\chi = a$, b, c and d)
- One latin square graph (latin_sqr_10)
- Two very large random graphs (C2000.5 and C4000.5)

4.2 Experimental protocol

The proposed HESA algorithm was coded in C++ and compiled using g++ with the '-O3' option on a cluster running Linux with a 2.83 GHz processor and 8 GB RAM. After running the DIMACS machine benchmark program [2] with g++ on our machine, we obtained the following results: 0.20 CPU seconds for graph r300.5, 1.23 CPU seconds for r400.5 and 4.68 CPU seconds for r500.5.

To obtain our computational results, each instance was solved 30 times independently with different random seeds. Each run was stopped when the processing time reaches a fixed timeout limit which was set to be 2 hours in this paper (This cutoff time was frequently used in the literature, see below). All the computational results were obtained with the parameter setting given in Table 1.

In order to identify an appropriate value for a given parameter, we varied its values within a reasonable range ($\mu_O \in [10, 10^2, 10^3, 10^4, 10^5]$, $maxIter \in$

[1] Available from `http://mat.gsia.cmu.edu/COLOR/instances.html`
[2] `ftp://dimacs.rutgers.edu/pub/dsj/clique/`

$[10^2, 10^3, 10^4, 10^5, 10^6]$) and compared their performances, while keeping the other parameter with its default values (as those shown in Table 1). To compare the results in terms of both solution quality and computation time, we used a sample of 20 graphs taken (without bias) from the 94 graphs.

Table 1
Parameters settings

Parameter	Sect.	Description	Value
μ_O	2.4	Maximum number of non-improving moves for TS_O	10
$maxIter$	2.4	Maximum number of iters for perturbation	10^4

4.3 Computational results

This section is dedicated to an evaluation of HESA's performance based on quality of upper and lower bounds of MSCP on the 94 benchmark instances. Columns 1–3 in Table 2 present the characteristics of the tested graphs and columns f^b_{UB} and f^b_{LB} give the current best-known upper and lower bounds of MSCP reported in the literature. Columns 6–8 present detailed computational results of HESA for the upper bounds of MSCP: the best upper bounds f^*_{UB}, the average upper bounds $Avg.$ and the average running time t to reach the best value across each of the 30 runs (in minutes). Columns 9–11 show the computational results for the lower bounds: the best lower bounds f^*_{LB}, the average lower bounds $Avg.$ and the average running time t to reach the best value (in minutes).

Table 2. Detailed computational results of HESA on the set of 58 COLOR 2002-2004 instances and 36 DIMACS instances

Characteristics of the graphs					HESA (upper bounds)			HESA (lower bounds)		
Name	$\|V\|$	$\|E\|$	f^b_{UB}	f^b_{LB}	f^*_{UB}	$Avg.$	t	f^*_{LB}	$Avg.$	t
myciel3	11	20	21	16	21	21.0	0.0	16	16.0	0.0
myciel4	23	71	45	34	45	45.0	0.0	34	34.0	0.0
myciel5	47	236	93	70	93	93.0	0.0	70	70.0	0.0
myciel6	95	755	189	142	189	189.0	0.0	142	142.0	0.3
myciel7	191	2 360	381	286	381	381.0	0.0	286	286.0	2.4
anna	138	493	276	273	276	276.0	0.2	273	273.0	0.4
david	87	406	237	234	237	237.0	0.1	234	234.0	0.1
huck	74	301	243	243	243	243.0	0.0	243	243.0	0.0
jean	80	254	217	216	217	217.0	0.0	216	216.0	0.0
homer	561	1 628	1 155	1 129	**1 150**	1 151.8	47.8	1 129	1 129.0	16.6
queen5.5	25	160	75	75	75	75.0	0.0	75	75.0	0.0
queen6.6	36	290	138	126	138	138.0	0.0	126	126.0	0.0
queen7.7	49	476	196	196	196	196.0	0.0	196	196.0	0.0
queen8.8	64	728	291	288	291	291.0	0.1	288	288.0	0.0
queen8.12	96	1 368	624	624	624	624.0	0.0	624	624.0	0.0
queen9.9	81	1 056	409	405	409	409.0	0.5	405	405.0	0.0

Continued on next page

Continued from previous page

Characteristics of the graphs					HESA (upper bounds)			HESA (lower bounds)		
Name	$\|V\|$	$\|E\|$	f^b_{UB}	f^b_{LB}	f^*_{UB}	*Avg.*	t	f^*_{LB}	*Avg.*	t
queen10.10	100	1 470	553	550	553	553.6	29.6	550	550.0	0.0
queen11.11	121	1 980	733	726	733	734.4	30.1	726	726.0	0.0
queen12.12	144	2 596	944	936	**943**	947.0	41.1	936	936.0	0.0
queen13.13	169	3 328	1 192	1 183	**1 191**	1195.4	29.3	1 183	1183.0	0.0
queen14.14	196	4 186	1 482	1 470	1 482	1487.3	21.6	1 470	1 470.0	0.0
queen15.15	225	5 180	1 814	1 800	1 814	1820.1	25.3	1 800	1 800.0	0.0
queen16.16	256	6 320	2 197	2 176	**2 193**	2199.4	28.1	2 176	2 176.0	0.0
school1	385	19 095	2 674	2 345	2 674	2674.0	0.1	**2 439**	2 418.9	70.6
school1-nsh	352	14 612	2 392	2 106	2 392	2392.0	0.3	**2 176**	2 169.4	61.5
games120	120	638	443	442	443	443.0	0.3	442	442.0	0.0
miles250	128	387	325	318	325	325.0	1.4	318	318.0	0.2
miles500	128	1 170	705	686	705	705.8	20.3	686	686.0	0.0
miles750	128	2 113	1 173	1 145	1 173	1173.6	16.1	1 145	1 145.0	0.0
miles1000	128	3 216	1 679	1 623	**1 666**	1670.5	28.6	1 623	1 623.0	0.1
miles1500	128	5 198	3 354	3 239	3 354	3354.0	0.5	3 239	3 239.0	0.0
fpsol2.i.1	496	11 654	3 403	3 403	3 403	3 403.0	8.5	3 403	3 403.0	13.1
fpsol2.i.2	451	8 691	1 668	1 668	1 668	1 668.0	0.8	1 668	1 668.0	8.9
fpsol2.i.3	425	8 688	1 636	1 636	1 636	1 636.0	1.2	1 636	1 636.0	7.2
mug88_1	88	146	178	164	178	178.0	0.0	164	164.0	0.1
mug88_25	88	146	178	162	178	178.0	0.0	162	162.0	0.1
mug100_1	100	166	202	188	202	202.0	0.0	188	188.0	0.2
mug100_25	100	166	202	186	202	202.0	0.0	186	186.0	0.2
2-Insert_3	37	72	62	55	62	62.0	0.0	55	55.0	0.0
3-Insert_3	56	110	92	84	92	92.0	0.0	84	84.0	0.1
inithx.i.1	864	18 707	3 676	3 676	3 676	3 676.0	1.3	3 676	3 675.3	82.1
inithx.i.2	645	13 979	2 050	2 050	2 050	2 050.0	1.3	2 050	2 050.0	21.5
inithx.i.3	621	13 969	1 986	1 986	1 986	1 986.0	0.0	1 986	1 986.0	18.8
mulsol.i.1	197	3 925	1 957	1 957	1 957	1 957.0	0.2	1 957	1 957.0	0.5
mulsol.i.2	188	3 885	1 191	1 191	1 191	1 191.0	0.0	1 191	1 191.0	0.6
mulsol.i.3	184	3 916	1 187	1 187	1 187	1 187.0	0.0	1 187	1 187.0	0.5
mulsol.i.4	185	3 946	1 189	1 189	1 189	1 189.0	0.0	1 189	1 189.0	0.6
mulsol.i.5	186	3 973	1 160	1 160	1 160	1 160.0	0.0	1 160	1 160.0	0.2
zeroin.i.1	211	4 100	1 822	1 822	1 822	1 822.0	0.0	1 822	1 822.0	0.5
zeroin.i.2	211	3 541	1 004	1 004	1 004	1 004.0	0.0	1 004	1 004.0	1.1
zeroin.i.3	206	3 540	998	998	998	998.0	0.0	998	998.0	0.5
wap05	905	43 081	13 669	12 428	13 887	13 962.7	43.8	**12 449**	12 438.9	57.9
					13656	13 677.8	1 872.5			
wap06	947	43 571	13 776	12 393	14 028	14 090.6	46.0	**12 454**	12 431.6	53.2
					13773	13 777.6	621.3			
wap07	1 809	103 368	28 617	24 339	29 154	29261.1	4.4	**24 800**	24 783.6	72.6
wap08	1 870	104 176	28 885	24 791	29 460	29542.3	3.0	**25 283**	25 263.4	65.6
qg.order30	900	26 100	13 950	13 950	13 950	13 950.0	0.0	13 950	13 950.0	0.1
qg.order40	1 600	62 400	32 800	32 800	32 800	32 800.0	0.0	32 800	32 800.0	0.3
qg.order60	3 600	212 400	109 800	109 800	109 800	109 800.0	0.2	109 800	109 800.0	2.7
DSJC125.1	125	736	326	247	326	326.1	5.2	247	247.0	0.4
DSJC125.5	125	3 891	1 012	549	1 012	1012.2	10.1	549	548.5	34.0
DSJC125.9	125	6 961	2 503	1 689	2 503	2 503.0	0.3	**1 691**	1 691.0	18.8
DSJC250.1	250	3 218	973	569	**970**	980.4	30.7	**570**	569.2	49.0
DSJC250.5	250	15 668	3 214	1 280	**3 210**	3 235.6	47.1	**1 287**	1 271.6	65.6
DSJC250.9	250	27 897	8 277	4 279	8 277	8 277.2	24.6	**4 311**	4 279.4	58.1

Continued on next page

Continued from previous page

Characteristics of the graphs					HESA (upper bounds)			HESA (lower bounds)		
Name	$\|V\|$	$\|E\|$	f^b_{UB}	f^b_{LB}	f^*_{UB}	*Avg.*	t	f^*_{LB}	*Avg.*	t
DSJC500.1	500	12 458	2 841	1 250	2 848	2 867.1	82.6	1 250	1 243.4	62.0
					2836	28 36.0	1 997.9			
DSJC500.5	500	62 624	10 897	2 921	10 992	11 063.2	97.0	**2 923**	2 896.0	65.6
					10 886	10 891.5	4 919.3			
DSJC500.9	500	112 437	29 896	10 881	**29 886**	29910.4	95.4	**11 053**	10 950.1	68.6
					29 862	29 874.3	5 513.3			
DSJC1000.1	1 000	49 629	8 995	2 762	9 182	9 237.2	101.6	2 719	2 707.6	66.0
					8 991	8 996.5	5 604.4			
DSJC1000.5	1 000	249 826	37 594	6 708	38 520	37 597.6	33.5	6 582	6 541.3	44.6
					37 575	37 594.7	3 090.3			
DSJC1000.9	1 000	449 449	103 464	26 557	104 483	105 221.3	103.1	26 296	26 150.3	51.8
					103 445	103 463.3	211.2			
DSJR500.1	500	3 555	2 173	2 061	**2 156**	2 170.7	99.8	**2 069**	2 069.0	4.2
DSJR500.1c	500	121 275	16 311	15 025	**16 286**	16 286.0	21.7	**15 398**	15 212.4	65.0
DSJR500.5	500	58 862	25 630	22 728	**25 440**	25 684.1	97.6	**22 974**	22 656.7	32.0
flat300_20_0	300	21 375	3 150	1 524	3 150	3 150.0	0.0	**1 531**	1 518.2	75.1
flat300_26_0	300	21 633	3 966	1 536	3 966	3 966.0	0.4	**1 548**	1 530.3	70.2
flat300_28_0	300	21 695	4 238	1 541	4 260	4 290.0	49.7	**1 547**	1 536.5	62.5
flat1000_50_0	1 000	245 000	25 500	6 601	25 500	25 500.0	0.3	6 476	6 452.1	51.5
flat1000_60_0	1 000	245 830	30 100	6 640	30 100	30 100.0	2.7	6 491	6 466.5	46.2
flat1000_76_0	1 000	246 708	37 167	6 632	38 089	38 313.5	36.8	6 509	6 482.8	34.1
					37 164	37 165.9	2 237.0			
le450_5a	450	5 714	1 350	1 190	1 350	1 350.0	0.1	**1 193**	1 191.5	67.4
le450_5b	450	5 734	1 350	1 186	1 350	1 350.1	0.8	**1 189**	1 185.0	67.0
le450_5c	450	9 803	1 350	1 272	1 350	1 350.0	0.9	**1 278**	1 270.4	66.8
le450_5d	450	9 757	1 350	1 269	1 350	1 350.0	0.3	**1 282**	1 274.2	71.6
le450_15a	450	8 168	2 632	2 329	2 634	2 648.4	91.5	**2 331**	2 331.0	23.3
le450_15b	450	8 169	2 642	2 348	**2 632**	2 656.5	89.9	2 348	2 348.0	4.8
le450_15c	450	16 680	3 491	2 593	**3 487**	3 792.4	86.7	**2 610**	2 606.6	57.3
le450_15d	450	16 750	3 506	2 622	**3 505**	3 883.1	82.7	**2 628**	2 627.1	54.9
le450_25a	450	8 260	3 153	3 003	3 157	3 166.7	88.5	3 003	3 003.0	1.2
le450_25b	450	8 263	3 366	3 305	**3 365**	3 375.2	88.6	3 305	3 305.0	1.0
le450_25c	450	17 343	4 515	3 638	4 553	4 583.8	84.8	**3 657**	3 656.9	41.7
le450_25d	450	17 425	4 544	3 697	4 569	4 607.6	92.4	**3 698**	3 698.0	8.3
latin_sqr_10	900	307 350	41 444	40 950	41 492	41 672.8	98.3	40 950	40 950.0	0.0
C2000.5	2 000	999 836	132 515	15 091	139 141	139 676.0	21.4	14 498	14 442.9	24.2
					132 483	132 513.9	161.8			
C4000.5	4 000	4 000 268	473 234	33 033	513 457	514 639.0	75.3	31 525	31 413.3	66.5

From Table 2, one observes that HESA is able to improve a number of best upper and lower bounds reported in the literature (indicated in bold) within a time limit of 2 hours. Specifically, for the upper bounds, HESA improves the best result for 15 instances and matches the previous best values for 61 instances. For the lower bounds, HESA improves the previous best known result for 27 instances and matches the previous best result for 59 instances.

When we compare the upper bounds and the lower bounds, we observe large gaps for the DIMACS instances. However, there are 21 instances where the upper bounds are identical to the lower bounds (underlined). Hence, the optimality of these instances is proven.

On the other hand, we observe that the HESA algorithm performs less well on some large DIMACS instances which are known to be very difficult for most MSCP algorithms. In order to see if HESA can improve its results on these instances, we carried out another experiment focusing on 14 large graphs with at least 500 vertices as follows. We used the solution of EXSCOL [37] as one of the 20 initial solutions of the population and reran HESA 30 times on each of the 14 graphs under the same test condition as before. Interestingly, the results of this experiment showed that HESA can find improved best known upper bounds of 10 out of 14 graphs (bold italic entries in Table 2).

To conclude, these outcomes provide evidence for the efficacy for the HESA algorithm.

4.4 Comparisons with four state-of-the-art algorithms for the upper bounds

In order to further evaluate the proposed HESA algorithm, we compare its upper and lower bounds with those reported by some of the best performing algorithms in the literature.

Table 3 summarizes the upper bounds of HESA in comparison with four very recent state-of-the-art algorithms, which cover the best known results for all the tested graphs. These algorithms are respectively named EXSCOL [37], BLS [3], MASC [17] and MA [30]. The experimental platforms used by the reference algorithms are as follows.

- EXSCOL was run on a 2.8 GHz computer with 2GB RAM and iteratively extracts maximum independent sets until the graph becomes empty.
- BLS was run on a Xeon E5440 with 2.83 GHz with 2GB RAM and used a timeout limit of 2 hours as the stopping condition.
- MASC was run on a 2.7 GHz PC with 4GB RAM and used 10^4 maximum iterations of its TS procedure and 50 maximum generations.
- MA was run on an Intel Core 2 Duo T5450–1.66 GHz with 2 GB RAM and used a timeout limit of 2 hours as the stopping condition.

Table 3. Comparisons of HESA with four state-of-the-art sum coloring algorithms for the upper bounds of MSCP on 94 graphs

Graph		EXSCOL [37]		BLS [3]		MASC [17]		MA [30]		HESA	
Name	f^b_{UB}	f_*	*Avg.*	f_*	*Avg.*	f_*	*Avg.*	f_*	*Avg.*	f_*	*Avg.*
myciel3	21	21	21.0	21	21.0	21	21.0	21	21.0	21	21.0
myciel4	45	45	45.0	45	45.0	45	45.0	45	45.0	45	45.0
myciel5	93	93	93.0	93	93.0	93	93.0	93	93.0	93	93.0
myciel6	189	189	189.0	189	196.6	189	189.0	189	189.0	189	189.0
myciel7	381	381	381.0	381	393.8	381	381.0	381	381.0	381	381.0

Continued on next page

Continued from previous page

Graph		EXSCOL [37]		BLS [3]		MASC [17]		MA [30]		HESA	
Name	f^b_{UB}	f^*_{UB}	Avg.	f^*_{UB}	Avg.	f^*_{UB}	Avg.	f^*_{UB}	Avg.	f^*_{UB}	Avg.
anna	276	*283*	283.2	276	276.0	276	276.0	276	276.0	276	276.0
david	237	237	238.1	237	237.0	237	237.0	237	237.0	237	237.0
huck	243	243	243.8	243	243.0	243	243.0	243	243.0	243	243.0
jean	217	217	217.3	217	217.0	217	217.0	217	217.0	217	217.0
homer	1 155	–	–	–	–	1 155	1 158.5	*1 157*	1 481.9	1 150	1 151.8
queen5.5	75	75	75.0	75	75.0	75	75.0	75	75.0	75	75.0
queen6.6	138	*150*	150.0	138	138.0	138	138.0	138	138.0	138	138.0
queen7.7	196	196	196.0	196	196.0	196	196.0	196	196.0	196	196.0
queen8.8	291	291	291.0	291	291.0	291	291.0	291	291.0	291	291.0
queen8.12	624	–	–	–	–	624	624.0	624	624.0	624	624.0
queen9.9	409	–	–	–	–	409	410.5	409	411.9	409	409.0
queen10.10	553	–	–	–	–	–	–	553	555.2	553	553.6
queen11.11	733	–	–	–	–	–	–	733	735.4	733	734.4
queen12.12	944	–	–	–	–	–	–	944	948.7	943	947.0
queen13.13	1 192	–	–	–	–	–	–	1 192	1 197.0	1 191	1 195.4
queen14.14	1 482	–	–	–	–	–	–	1 482	1 490.8	1 482	1 487.3
queen15.15	1 814	–	–	–	–	–	–	1 814	1 823.0	1 814	1 820.1
queen16.16	2 197	–	–	–	–	–	–	2 197	2 205.9	2 193	2 199.4
school1	2 674	–	–	–	–	–	–	2 674	2 766.8	2 674	2 674.0
school1-nsh	2 392	–	–	–	–	–	–	2 392	2 477.1	2 392	2 392.0
games120	443	443	447.9	443	443.0	443	443.0	443	443.0	443	443.0
miles250	325	*328*	333.0	*327*	328.8	325	325.0	325	325.4	325	325.0
miles500	705	*709*	714.5	*710*	713.3	705	705.0	*708*	711.2	705	705.8
miles750	1 173	–	–	–	–	–	–	1 173	1 183.9	1 173	1 173.6
miles1000	1 679	–	–	–	–	–	–	1 679	1 697.3	1 666	1 670.5
miles1500	3 354	–	–	–	–	–	–	3 354	3 357.2	3 354	3 354.0
fpsol2.i.1	3 403	–	–	–	–	3 403	3 403.0	3 403	3 403.0	3 403	3 403.0
fpsol2.i.2	1 668	–	–	–	–	1 668	1 668.0	1 668	1 668.0	1 668	1 668.0
fpsol2.i.3	1 636	–	–	–	–	1 636	1 636.0	1 636	1 636.0	1 636	1 636.0
mug88_1	178	–	–	–	–	178	178.0	–	–	178	178.0
mug88_25	178	–	–	–	–	178	178.0	–	–	178	178.0
mug100_1	202	–	–	–	–	202	202.0	–	–	202	202.0
mug100_25	202	–	–	–	–	202	202.0	–	–	202	202.0
2-Insert_3	62	–	–	–	–	62	62.0	–	–	62	62.0
3-Insert_3	92	–	–	–	–	92	92.0	–	–	92	92.0
inithx.i.1	3 676	–	–	–	–	3 676	3 676.0	3 676	3 679.6	3 676	3 676.0
inithx.i.2	2 050	–	–	–	–	2 050	2 050.0	2 050	2 053.7	2 050	2 050.0
inithx.i.3	1 986	–	–	–	–	1 986	1 986.0	1 986	1 986.0	1 986	1 986.0
mulsol.i.1	1 957	–	–	–	–	1 957	1 957.0	1 957	1 957.0	1 957	1 957.0
mulsol.i.2	1 191	–	–	–	–	1 191	1 191.0	1 191	1 191.0	1 191	1 191.0
mulsol.i.3	1 187	–	–	–	–	1 187	1 187.0	1 187	1 187.0	1 187	1 187.0
mulsol.i.4	1 189	–	–	–	–	1 189	1 189.0	1 189	1 189.0	1 189	1 189.0
mulsol.i.5	1 160	–	–	–	–	1 160	1 160.0	1 160	1 160.0	1 160	1 160.0
zeroin.i.1	1 822	–	–	–	–	1 822	1 822.0	1 822	1 822.0	1 822	1 822.0
zeroin.i.2	1 004	–	–	–	–	1 004	1 004.0	1 004	1 004.0	1 004	1 004.0
zeroin.i.3	998	–	–	–	–	998	998.0	998	998.0	998	998.0
wap05	13 669	*13 680*	13 718.4	–	–	13 669	13 677.8	–	–	13 656	13 677.8
wap06	13 776	*13 778*	13 830.9	–	–	13 776	13 777.8	–	–	13 773	13 777.6
wap07	28 617	*28 629*	28 663.8	–	–	28 617	28 624.7	–	–	*29 154*	29 261.1
wap08	28 885	*28 896*	28 946.0	–	–	28 885	28 890.9	–	–	*29 460*	29 542.3

Continued on next page

Continued from previous page

Graph		EXSCOL [37]		BLS [3]		MASC [17]		MA [30]		HESA	
Name	f^b_{UB}	f^*_{UB}	Avg.	f^*_{UB}	Avg.	f^*_{UB}	Avg.	f^*_{UB}	Avg.	f^*_{UB}	Avg.
qg.order30	13 950	13 950	13 950.0	–	–	13 950	13 950.0	13 950	13 950.0	13 950	13 950.0
qg.order40	32 800	32 800	32 800.0	–	–	32 800	32 800.0	32 800	32 800.0	32 800	32 800.0
qg.order60	10 9800	*110 925*	110 993.0	–	–	10 9800	10 9800.0	10 9800	10 9800.0	10 9800	10 9800.0
DSJC125.1	326	326	326.7	326	326.9	326	326.6	326	327.3	326	326.1
DSJC125.5	1 012	*1 017*	1 019.7	1 012	1 012.9	1 012	1 020.0	*1 013*	1 018.5	1 012	1 012.2
DSJC125.9	2 503	*2 512*	2 512.0	2 503	2 503.0	2 503	2 508.0	2 503	2 519.0	2 503	2 503.0
DSJC250.1	973	*985*	985.0	973	982.5	*974*	990.5	*983*	995.8	970	980.4
DSJC250.5	3 214	*3 246*	3 253.9	*3 219*	3 248.5	*3 230*	3 253.7	3 214	3 285.5	3 210	3 235.6
DSJC250.9	8 277	*8 286*	8 288.8	*8 290*	8 316.0	*8 280*	8 322.7	8 277	8 348.8	8 277	8 277.2
DSJC500.1	2 841	*2 850*	2 857.4	*2 882*	2 942.9	2 841	2 844.1	*2 897*	2 990.5	2 836	2 836.0
DSJC500.5	10 897	*10 910*	10 918.2	*11 187*	11 326.3	10 897	10 905.8	*11 082*	11 398.3	10 886	10 891.5
DSJC500.9	29 896	*29 912*	29 936.2	*30 097*	30 259.2	29 896	29 907.8	*29 995*	30 361.9	29 862	29 874.3
DSJC1000.1	8 995	*9 003*	9 017.9	*9 520*	9 630.1	8 995	9 000.5	*9 188*	9 667.1	8 991	8 996.5
DSJC1000.5	37 594	*37 598*	37 673.8	*40 661*	41 002.6	37 594	37 597.6	*38 421*	40 260.9	37 575	37 594.7
DSJC1000.9	103 464	103 464	103 531.0	–	–	103 464	103 464.0	*105 234*	107 349.0	103 445	103 463.3
DSJR500.1	2 173	–	–	–	–	–	–	2 173	2 253.1	2 156	2 170.7
DSJR500.1c	16 311	–	–	–	–	–	–	16 311	16 408.5	16 286	16 286.0
DSJR500.5	25 630	–	–	–	–	–	–	25 630	26 978.0	25 440	25 684.1
flat300_20_0	3 150	3 150	3 150.0	–	–	3 150	3 150.0	3 150	3 150.0	3 150	3 150.0
flat300_26_0	3 966	3 966	3 966.0	–	–	3 966	3 966.0	3 966	3 966.0	3 966	3 966.0
flat300_28_0	4 238	*4 282*	4 286.1	–	–	4 238	4 313.4	*4 261*	4 389.4	*4 260*	4 290.0
flat1000_50_0	25 500	25 500	25 500.0	–	–	25 500	25 500.0	25 500	25 500.0	25 500	25 500.0
flat1000_60_0	30 100	30 100	30 100.0	–	–	30 100	30 100.0	30 100	30 100.0	30 100	30 100.0
flat1000_76_0	37 167	37 167	37 213.2	–	–	37 167	37 167.0	*38 213*	39 722.7	37 164	37 165.9
le450_5a	1 350	–	–	–	–	1 350	1 350.0	1 350	1 350.0	1 350	1 350.0
le450_5b	1 350	–	–	–	–	1 350	1 350.0	1 350	1 350.0	1 350	1 350.1
le450_5c	1 350	–	–	–	–	1 350	1 350.0	1 350	1 350.0	1 350	1 350.0
le450_5d	1 350	–	–	–	–	1 350	1 350.0	1 350	1 350.0	1 350	1 350.0
le450_15a	2 632	2 632	2 641.9	–	–	*2 706*	2 742.6	*2 681*	2 733.1	*2 634*	2 648.4
le450_15b	2 642	2 642	2 643.4	–	–	*2 724*	2 756.2	*2 690*	2 730.6	2 632	2 656.5
le450_15c	3 491	*3 866*	3 868.9	–	–	3 491	3 491.0	*3 943*	4 048.4	3 487	3 792.4
le450_15d	3 506	*3 921*	3 928.5	–	–	3 506	3 511.8	*3 926*	4 032.4	3 505	3 883.1
le450_25a	3 153	3 153	3 159.4	–	–	*3 166*	3 176.8	*3 178*	3 204.3	*3 157*	3 166.7
le450_25b	3 366	3 366	3 371.9	–	–	3 366	3 375.1	*3 379*	3 416.2	3 365	3 375.2
le450_25c	4 515	4 515	4 525.4	–	–	*4 700*	4 773.3	*4 648*	4 700.7	*4 553*	4 583.8
le450_25d	4 544	4 544	4 550.0	–	–	*4 722*	4 805.7	*4 696*	4 740.3	*4 569*	4 607.6
latin_sqr_10	41 444	*42 223*	42 392.7	–	–	41 444	41 481.5	–	–	*41 492*	41 672.8
C2000.5	132 515	132 515	132 682.0	–	–	–	–	–	–	132 483	132 513.9
C4000.5	473 234	473 234	473 211.0	–	–	–	–	–	–	*513 457*	514 639.0
Suc#/Total#		29/52		18/27		69/77		61/81		85/94	

Columns 1–2 in Table 3 present the tested graphs and the best known upper bounds f^b_{UB}, the next 10 columns indicate the best results f_* and the average results *Avg.* for the four reference algorithms and HESA respectively. line 490-492: The dash "-" symbol indicates that the given reference algorithm did not report results for the given test graph. The italic entries in the table mean that the reference algorithms fail to attain the best known results on the tested graphs. The last row presents the number of cases where an algorithm can achieve the best known result (*Suc#*) over the total number of the tested graphs (*Total#*). Given the difference in programming languages, compiler options and computers, we focus on solution quality and indicate computing

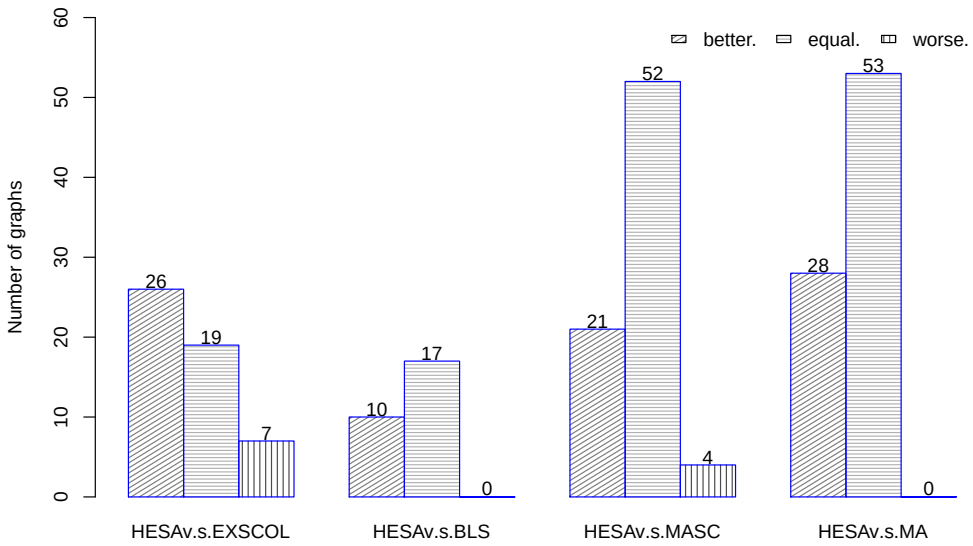

Fig. 5. Comparisons of HESA and four reference algorithms for the upper bounds.

times only for indicative purposes. It is worth mentioning that the timeout limit (2hours) for HESA is the same as for MA and BLS, and similar to the time limits of MASC and EXSCOL on the small instances. However, the time limit for HESA is shorter than that used for MASC and EXCOL on the large graphs. We mention that EXSCOL, BLS, MASC, MA and the HESA algorithm were tested on 52, 27, 77, 81 and 94 graphs respectively.

From Table 3, we observe that EXSCOL, BLS, MASC, MA and our HESA algorithm can match the best known results for 29, 18, 69, 61 and 85 graphs, but fail to reach the best results for 23, 9, 8, 20 and 9 instances respectively. In particular, our HESA algorithm can improve the best known results for 24 graphs.

Since each reference algorithm only reports results on a subset of the considered 94 graphs, we compare the performances between HESA and the four reference algorithms one by one and summarize the comparisons of the upper bounds of MSCP in Figure 5. The height of each bar in the figure represents the number of graphs. Three different bars of each comparison (HESA vs a reference algorithm) indicate the number of cases for which the results obtained with HESA are better than, equal to and worse than the reference algorithm. From Figure 5, we can observe that HESA obtains better results for 26, 10, 21 and 28 graphs, equal results for 19, 17, 52 and 53 graphs and worse results for 7, 0, 4 and 0 results compared to EXSCOL, BLS, MASC and MA. Furthermore, HESA never produces a result that is worse than that reported with BLS and MA, and reports a worse result than EXCOL and MASC only in

several cases. This comparison study clearly shows that HESA competes very favorably with the reference algorithms to computer upper bounds of MSCP.

4.5 Comparisons with four state-of-the-art algorithms for the lower bounds

Table 4 provides a computational comparison of the lower bounds obtained with HESA and four state-of-art algorithms which cover the best known lower bounds for all the tested graphs. The reference algorithms are respectively named RMDS(n) [29], MDS(5)+LS [14], EXCLIQUE [38] and MA [30]. Notice that the results of RMDS(n) were extracted from [14]. The experimental platforms used by the reference algorithms are as follows.

- RMDS(n) was run on an Intel Core i7 processor 2.93 GHz with 4 GB RAM and used five heuristics.
- MDS(5)+LS was run on an Intel Core i7 processor 2.93 GHz with 4 GB RAM and used a cutoff time limit of 1 hour as the stopping condition.
- EXCLIQUE was run on a 2.8 GHz computer with 2GB RAM and iteratively extracts maximum independent sets until the graph becomes empty.
- MA was run on an Intel Core 2 Duo T5450–1.66 GHz with 2 GB RAM and used a cutoff time limit of 2 hours as the stopping condition.

Table 4. Comparisons of HESA with four state-of-the-art sum coloring algorithms for the lower bounds of MSCP on 94 graphs

Graph		RMDS(n) [29]		MDS(5)+LS [14]		EXCLIQUE [38]		MA [30]		HESA	
Name	f^b_{LB}	f^*_{LB}	Avg.	f^*_{LB}	Avg.	f^*_{LB}	Avg.	f^*_{LB}	Avg.	f^*_{LB}	Avg.
myciel3	16	16	–	16	–	16	16.0	16	16.0	16	16.0
myciel4	34	34	–	34	–	34	34.0	34	34.0	34	34.0
myciel5	70	70	–	70	–	70	70.0	70	70.0	70	70.0
myciel6	142	142	–	142	–	142	142.0	142	139.5	142	142.0
myciel7	286	286	–	286	–	286	286.0	286	277.5	286	286.0
anna	273	*272*	–	273	–	273	273.0	273	273.0	273	273.0
david	234	234	–	234	–	*229*	229.0	234	234.0	234	234.0
huck	243	243	–	243	–	243	243.0	243	243.0	243	243.0
jean	216	216	–	216	–	216	216.0	216	216.0	216	216.0
homer	1 129	–	–	–	–	–	–	1 129	1 129.0	1 129	1 129.0
queen5.5	75	75	–	75	–	75	75.0	75	75.0	75	75.0
queen6.6	126	126	–	126	–	126	126.0	126	126.0	126	126.0
queen7.7	196	196	–	196	–	196	196.0	196	196.0	196	196.0
queen8.8	288	288	–	288	–	288	288.0	288	288.0	288	288.0
queen8.12	624	–	–	–	–	–	–	624	624.0	624	624.0
queen9.9	405	–	–	–	–	–	–	405	405.0	405	405.0
queen10.10	550	–	–	–	–	–	–	550	550.0	550	550.0
queen11.11	726	–	–	–	–	–	–	726	726.0	726	726.0
queen12.12	936	–	–	–	–	–	–	936	936.0	936	936.0
queen13.13	1 183	–	–	–	–	–	–	1 183	1 183.0	1 183	1 183.0
queen14.14	1 470	–	–	–	–	–	–	1 470	1 470.0	1 470	1 470.0

Continued on next page

Continued from previous page

Graph		RMDS(n) [29]		MDS(5)+LS [14]		EXCLIQUE [38]		MA [30]		HESA	
Name	f^b_{LB}	f_*	Avg.	f_*	Avg.	f_*	Avg.	f_*	Avg.	f_*	Avg.
queen15.15	1 800	–	–	–	–	–	–	1 800	1 800.0	1 800	1 800.0
queen16.16	2 176	–	–	–	–	–	–	2 176	2 176.0	2 176	2 176.0
school1	2 345	–	–	–	–	–	–	2 345	2 283.3	2 439	2 418.9
school1-nsh	2 106	–	–	–	–	–	–	2 106	2 064.6	2 176	2 169.4
games120	442	442	–	442	–	442	441.4	442	442.0	442	442.0
miles250	318	*316*	–	318	–	318	316.2	318	318.0	318	318.0
miles500	686	*677*	–	686	–	*677*	671.4	686	686.0	686	686.0
miles750	1 145	–	–	–	–	–	–	1 145	1 145.0	1 145	1 145.0
miles1000	1 623	–	–	–	–	–	–	1 623	1 623.0	1 623	1 623.0
miles1500	3 239	–	–	–	–	–	–	3 239	3 239.0	3 239	3 239.0
fpsol2.i.1	3 403	*3 402*	–	*3 151*	–	3 403	3 403.0	3 403	3 403.0	3 403	3 403.0
fpsol2.i.2	1 668	–	–	–	–	–	–	1 668	1 668.0	1 668	1 668.0
fpsol2.i.3	1 636	–	–	–	–	–	–	1 636	1 636.0	1 636	1 636.0
mug88_1	164	*163*	–	164	–	164	162.3	–	–	164	164.0
mug88_25	162	*161*	–	162	–	162	160.3	–	–	162	162.0
mug100_1	188	*186*	–	188	–	188	188.0	–	–	188	188.0
mug100_25	186	*183*	–	186	–	186	183.4	–	–	186	186.0
2-Insert_3	55	55	–	55	–	55	55.0	–	–	55	55.0
3-Insert_3	84	84	–	84	–	84	82.8	–	–	84	84.0
inithx.i.1	3 676	*3 581*	–	*3 486*	–	3 676	3 676.0	3 676	3 616.0	3 676	3 675.3
inithx.i.2	2 050	–	–	–	–	–	–	2 050	1 989.2	2 050	2 050.0
inithx.i.3	1 986	–	–	–	–	–	–	1 986	1 961.8	1 986	1 986.0
mulsol.i.1	1 957	–	–	–	–	–	–	1 957	1 957.0	1 957	1 957.0
mulsol.i.2	1 191	–	–	–	–	–	–	1 191	1 191.0	1 191	1 191.0
mulsol.i.3	1 187	–	–	–	–	–	–	1 187	1 187.0	1 187	1 187.0
mulsol.i.4	1 189	–	–	–	–	–	–	1 189	1 189.0	1 189	1 189.0
mulsol.i.5	1 160	–	–	–	–	–	–	1 160	1 160.0	1 160	1 160.0
zeroin.i.1	1 822	–	–	–	–	–	–	1 822	1 822.0	1 822	1 822.0
zeroin.i.2	1 004	1 004	–	1 004	–	1 004	1 004.0	1 004	1 002.1	1 004	1 004.0
zeroin.i.3	998	998	–	998	–	998	998.0	998	998.0	998	998.0
wap05	12 428	–	–	–	–	12 428	12 339.3	–	–	12 449	12 438.9
wap06	12 393	–	–	–	–	12 393	12 348.8	–	–	12 454	12 431.6
wap07	24 339	–	–	–	–	24 339	24 263.8	–	–	24 800	24 783.6
wap08	24 791	–	–	–	–	24 791	24 681.1	–	–	25 283	25 263.4
qg.order30	13 950	–	–	–	–	13 950	13 950.0	13 950	13 950.0	13 950	13 950.0
qg.order40	32 800	–	–	–	–	32 800	32 800.0	32 800	32 800.0	32 800	32 800.0
qg.order60	109 800	–	–	–	–	109 800	109 800.0	109 800	109 800.0	109 800	109 800.0
DSJC125.1	247	*238*	–	*238*	–	*246*	244.1	247	244.6	247	247.0
DSJC125.5	549	*504*	–	*493*	–	*536*	522.4	549	541.0	549	548.5
DSJC125.9	1 689	*1 600*	–	*1 621*	–	*1 664*	1 592.5	1 689	1 677.7	1 691	1 691.0
DSJC250.1	569	*537*	–	*521*	–	*567*	562.0	569	558.4	570	569.2
DSJC250.5	1 280	*1 150*	–	*1 128*	–	*1 270*	1 258.8	1 280	1 249.4	1 287	1 271.6
DSJC250.9	4 279	*3 972*	–	*3 779*	–	*4 179*	4 082.4	4 279	4 160.9	4 311	4 279.4
DSJC500.1	1 250	*1 163*	–	*1 143*	–	1 250	1 246.6	*1 241*	1 214.9	1 250	1 243.4
DSJC500.5	2 921	*2 616*	–	*2 565*	–	2 921	2 902.6	*2 868*	2 797.7	2 923	2 896.0
DSJC500.9	10 881	*10 074*	–	*9 731*	–	10 881	10 734.5	*10 759*	10 443.8	11 053	10 950.1
DSJC1000.1	2 762	*2 499*	–	*2 456*	–	2 762	2 758.6	*2 707*	2 651.2	*2 719*	2 707.6
DSJC1000.5	6 708	*5 787*	–	*5 660*	–	6 708	6 665.9	*6 534*	6 182.5	*6 582*	6 541.3
DSJC1000.9	26 557	*23 863*	–	*23 208*	–	26 557	26 300.3	*26 157*	24 572.0	*26 296*	26 150.3
DSJR500.1	2 061	–	–	–	–	–	–	2 061	2 052.9	2 069	2 069.0
DSJR500.1c	15 025	–	–	–	–	–	–	15 025	14 443.9	15 398	15 212.4
DSJR500.5	22 728	–	–	–	–	–	–	22 728	22 075.0	22 974	22 656.7
flat300_20_0	1 524	–	–	–	–	1 524	1 505.7	*1 515*	1 479.3	1 531	1 518.2

Continued on next page

Continued from previous page

Graph		RMDS(n) [29]		MDS(5)+LS [14]		EXCLIQUE [38]		MA [30]		HESA	
Name	f^b_{LB}	f_*	Avg.	f_*	Avg.	f_*	Avg.	f_*	Avg.	f_*	Avg.
flat300_26_0	1 536	–	–	–	–	*1 525*	1 511.4	1 536	1 501.6	1 548	1 530.3
flat300_28_0	1 541	–	–	–	–	*1 532*	1 515.3	1 541	1 503.9	1 547	1 536.5
flat1000_50_0	6 601	–	–	–	–	6 601	6 571.8	*6 433*	6 121.5	*6 476*	6 452.1
flat1000_60_0	6 640	–	–	–	–	6 640	6 600.5	*6 402*	6 047.7	*6 491*	6 466.5
flat1000_76_0	6 632	–	–	–	–	6 632	6 583.2	*6 330*	6 074.6	*6 509*	6 482.8
le450_5a	1 190	–	–	–	–	–	–	1 190	1 171.5	1 193	1 191.5
le450_5b	1 186	–	–	–	–	–	–	1 186	1 166.5	1 189	1 185.0
le450_5c	1 272	–	–	–	–	–	–	1 272	1 242.3	1 278	1 270.4
le450_5d	1 269	–	–	–	–	–	–	1 269	1 245.2	1 282	1 274.2
le450_15a	2 329	–	–	–	–	2 329	2 313.7	2 329	2 324.3	2 331	2 331.0
le450_15b	2 348	–	–	–	–	*2 343*	2 315.7	2 348	2 335.0	2 348	2 348.0
le450_15c	2 593	–	–	–	–	*2 591*	2 545.3	2 593	2 569.1	2 610	2 606.6
le450_15d	2 622	–	–	–	–	*2 610*	2 572.4	2 622	2 587.2	2 628	2 627.1
le450_25a	3 003	–	–	–	–	*2 997*	2 964.4	3 003	3 000.4	3 003	3 003.0
le450_25b	3 305	–	–	–	–	3 305	3 304.1	3 305	3 304.1	3 305	3 305.0
le450_25c	3 638	–	–	–	–	*3 619*	3 597.1	3 638	3 617.0	3 657	3 656.9
le450_25d	3 697	–	–	–	–	*3 684*	3 627.4	3 697	3 683.2	3 698	3 698.0
latin_sqr_10	40 950	–	–	–	–	40 950	40 950.0	–	–	40 950	40 950.0
C2000.5	15 091	–	–	–	–	15 091	15 077.6	–	–	*14 498*	14 442.9
C4000.5	33 033	–	–	–	–	33 033	33 018.8	–	–	*31 525*	31 413.3
Suc#/Total#		17/38		24/38		46/62		71/81		86/94	

Table 4 summarizes the computational results (lower bounds) of the five compared algorithms with the same information as in Table 3. Once again, we focus on solution quality and indicate computing times for indicative purposes. We mention that the time limit used for HESA (2 hours) is the same as that used for MA, similar to that used for EXCLIQUE on small instances. This limit is shorter than that adopted for EXCLIQUE on large graphs, and longer than those reported for RMDS(n) and MDS(5)+LS.

Table 4 discloses that RMDS(n), MDS(5)+LS, EXCLIQUE, MA and HESA can match the best known results for 17/38 (i.e., 17 over 38 tested graphs), 24/38, 46/62, 71/81 and 86/94 graphs respectively. In particular, HESA can improve 27 best known lower bounds.

Like for the upper bounds, we compare HESA with each of the four reference algorithms and summarize the comparisons of lower bounds in Figure 6 in the same way as in Figure 5. From Figure 6, we can observe that HESA obtains improved lower bounds for 21, 14, 24 and 30 graphs, equal results for 17, 24, 30 and 51 graphs and worse results for 0, 0, 8 and 0 graphs compared to RMDS(n), MDS(5)+LS, EXCLIQUE and MA respectively.

Finally, since MDS(5)+LS used a time limit of 1 hour, we reran HESA under this reduced time condition. We observe that HESA still obtains better lower bounds for 14 instances, equal lower bounds for 24 instances and no worse results compared to the MDS(5)+LS algorithm.

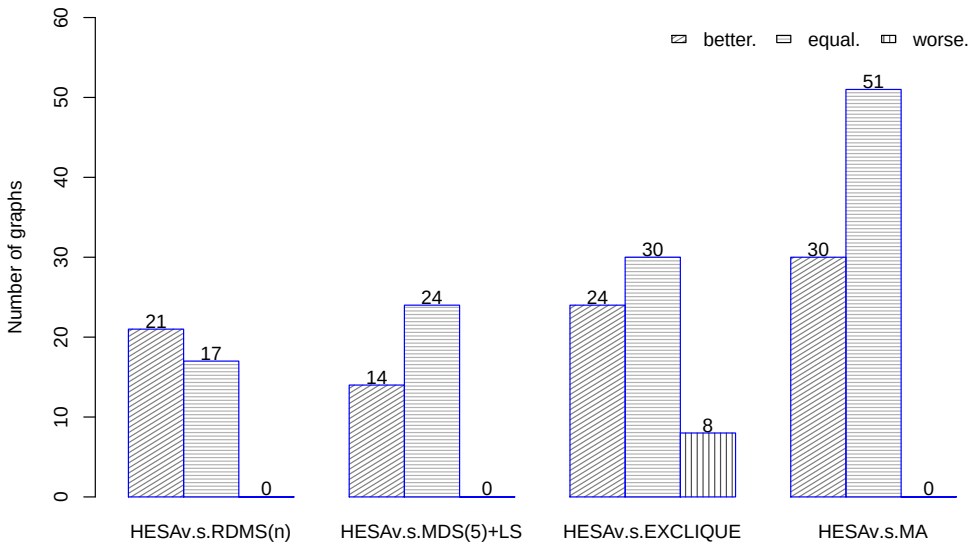

Fig. 6. Comparisons of HESA and four reference algorithms for the lower bounds.

5 Analysis of HESA

In this section, we first study the impact of the joint use of two crossover operators on the performance of the proposed HESA algorithm. Moreover, we perform a fitness distance analysis (FDA) [19] in order to obtain some insight into the hardness of some benchmark instances, which may help understand the behavior of our HESA algorithm.

5.1 Analysis on the double-crossover operator

As indicated in Section 2.3, HESA employs two crossover operators (GGX and DGX) to generate offspring solutions. In order to investigate the positive role of this mechanism, we compared HESA with its two variants: HESA$_{GGX}$ uses only the GGX crossover while HESA$_{DGX}$ uses only the DGX crossover. We carried out additional experiments on 20 selected graphs and run HESA, HESA$_{GGX}$ and HESA$_{DGX}$ for 30 times on each graph to computer the upper and lower bounds of MSCP. These three algorithms used the same parameter settings given in Table 1 and the same timeout limit (2 hours).

Table 5 summarizes the computational results of HESA, HESA$_{GGX}$ and HESA$_{DGX}$. Column 1–3 recall the best known upper and lower bounds for the 20 graphs. Columns 4–15 present the best upper bounds, the average upper bounds, the

Table 5
HESA with its two crossover operators (GGX and DGX) compared with two variants where only one crossover operator was used

Graph			HESA				$HESA_{GGX}$				$HESA_{DGX}$			
Name	f^b_{UB}	f^b_{LB}	f^*_{UB}	avg_{UB}	f^*_{LB}	avg_{LB}	f^*_{UB}	avg_{UB}	f^*_{LB}	avg_{LB}	f^*_{UB}	avg_{UB}	f^*_{LB}	avg_{LB}
homer	1155	1129	**1150**	1151.8	1129	1129.0	**1150**	1151.9	1129	1129.0	**1151**	1152.2	1129	1129.0
queen11.11	733	726	733	734.4	726	726.0	733	736.2	726	726.0	733	735.2	726	726.0
queen12.12	944	936	**943**	947.0	936	936.0	945	948.5	936	936.0	**942**	947.5	936	936.0
queen13.13	1192	1183	**1191**	1195.4	1183	1183.0	1194	1197.3	1183	1183.0	1192	1197.3	1183	1183.0
miles250	325	318	325	325.0	318	318.0	325	325.0	318	318.0	325	325.0	318	318.0
miles500	705	686	705	705.8	686	686.0	705	706.2	686	686.0	705	705.9	686	686.0
DSJC250.1	973	569	**970**	980.4	**570**	569.2	977	981.0	**570**	569.5	**972**	980.9	**570**	568.8
DSJC250.5	3214	1280	**3210**	3235.6	**1287**	1271.6	3222	3243.7	**1285**	1275.1	**3210**	3235.7	1270	1261.6
DSJC250.9	8277	4279	8277	8277.2	**4305**	4268.3	8277	8279.4	**4294**	4269.4	8277	8277.0	**4312**	4273.1
DSJC500.1	2841	1250	2848	2867.1	1250	1243.4	2850	2870.4	1248	1244.4	2848	2870.4	1245	1241.2
DSJC500.5	10897	2921	10992	11063.2	**2923**	2896.0	10969	11066.2	2918	2898.6	11012	11094.0	2894	2876.1
DSJC500.9	29896	10881	**29886**	29910.4	**11053**	10950.1	**29869**	29900.0	**11049**	10952.0	29900	29927.0	**10982**	10853.3
DSJR500.1	2173	2061	**2156**	2170.7	**2069**	2069.0	**2154**	2167.1	**2069**	2069.0	**2159**	2172.6	**2069**	2069.0
DSJR500.1c	16311	15025	**16286**	16286.0	**15246**	15132.5	**16286**	16286.0	**15313**	15185.6	**16286**	16286.0	**15118**	15006.4
DSJR500.5	25630	22728	**25440**	25684.1	**22974**	22656.7	**25439**	25565.9	22641	22634.7	25935	26029.2	**22999**	22643.0
flat300_28_0	4238	1541	4260	4290.0	**1547**	1536.5	4270	4296.5	**1544**	1535.9	4261	4289.4	1533	1519.3
le450_15a	2632	2329	2634	2648.4	**2331**	2331.0	2637	2649.5	**2331**	2330.9	2642	2658.5	**2331**	2331.0
le450_15b	2642	2348	**2632**	2656.5	2348	2348.0	2644	2656.3	2348	2348.0	**2641**	2659.9	2348	2348.0
le450_15c	3491	2593	**3487**	3792.4	**2610**	2606.6	**3490**	3853.5	**2610**	2607.4	3491	3814.7	**2610**	2606.6
le450_15d	3506	2622	**3505**	3883.1	**2628**	2627.1	3829	3913.8	**2628**	2626.7	**3504**	3774.9	**2628**	2627.2
suc#			16		20		10		17		14		16	

best lower bounds and the average lower bounds achieved by HESA, $HESA_{GGX}$ and $HESA_{DGX}$ respectively. The last row gives the number of times an algorithm finds a better or equal result compared to the best known result.

From Table 5, we can make the following observations. First, HESA with its two crossover operators can reach 36 best known (upper and lower) bounds out of the 40 cases. $HESA_{GGX}$ and $HESA_{DGX}$ can only reach 27 and 30 best known results respectively. Second, HESA is able to improve 24 best known results (bold) while $HESA_{GGX}$ and $HESA_{DGX}$ improve 16 and 17 best results respectively. Third, among the three compared algorithms, HESA holds 11 best results (underlined) while both $HESA_{GGX}$ and $HESA_{DGX}$ hold 4 best results. Besides, $HESA_{GGX}$ and $HESA_{DGX}$ complements each other on some graphs (e.g., on DSJC500.9 and DSJR500.9). In summary, the joint use of DGX and GGX crossovers allows HESA to reach a better performance than when these crossovers are used separately. This is particularly useful when handling different graphs.

Table 6
FDC analysis on 20 selected graphs for the upper and lower bounds of MSCP

Graph	Upper bounds of MSCP				Lower bounds of MSCP			
	$\#lo$	$avg\ d_{lo}$	$avg\ d_{go}$	ρ	$\#lo$	$avg\ d_{lo}$	$avg\ d_{go}$	ρ
homer	1188	166.388	96.875	0.772	1011	540.925	0.542	-0.999
queen11_11	1188	105.468	45.633	0.666	730	108.392	0.006	-0.999
queen12_12	1191	119.169	31.407	0.874	784	131.224	0.084	-0.975
queen13_13	1191	147.310	67.307	0.688	829	154.965	0.000	-0.999
miles250	1137	36.563	3.991	0.865	1200	123.578	0.000	-0.999
miles500	1191	69.946	28.412	0.729	1200	119.666	0.123	-0.704
DSJC250.1	706	87.201	94.048	0.462	1200	246.501	243.506	-0.126
DSJC250.5	1183	205.100	213.692	0.122	1198	236.674	222.969	-0.292
DSJC250.9	1199	238.464	86.344	0.637	1198	214.859	168.488	-0.472
DSJC500.1	942	354.208	184.689	0.675	1200	494.570	491.143	-0.100
DSJC500.5	1200	484.804	482.820	0.150	1200	489.638	488.290	-0.094
DSJC500.9	1200	491.331	490.818	0.056	1200	461.389	460.501	-0.133
DSJR500.1	1185	279.052	226.710	0.646	1200	481.245	286.637	-0.272
DSJR500.1c	1198	421.765	212.482	0.304	1200	440.222	450.543	-0.127
DSJR500.5	1200	491.142	489.561	0.105	769	289.422	385.740	-0.020
flat300_28_0	1199	274.961	262.538	0.341	1200	287.138	286.731	-0.139
le450_15a	1159	273.573	299.082	0.161	1200	442.403	389.603	-0.151
le450_15b	1115	234.293	202.203	0.527	1200	443.224	329.997	-0.130
le450_15c	1108	352.946	230.713	0.921	1200	441.681	426.002	-0.217
le450_15d	1086	350.593	418.864	0.122	1200	441.665	436.336	-0.056

5.2 Landscape analysis

The fitness-distance correlation (FDC) coefficient ρ measures the correlation between the quality (fitness or objective function value) of local optima and their distances to the global optimum of a given problem instance [19]. If the solution quality increases with the diminution of distance to the optimum, then there is a path to the optimum via solutions with increasing (better) fitness. Even if FDC alone cannot fully characterize the hardness of a problem, it could provide useful information about the landscape of the problem. On the other hand, FDC has some known shortcomings and limits [1] and consequently, the analysis shown in this section should be interpreted with caution.

For a minimization problem, a ρ value close to 1 (the largest possible value) indicates a strong fitness-distance correlation while a ρ value close to -1 (the smallest possible value) means the absence of any correlation. The reverse is true for a maximization problem. In this section, we present the first FDC analysis of MSCP both for the problems of computing upper bounds (minimization) and lower bounds (maximization).

Based on the 20 selected graphs, we studied the cases of calculating upper and lower bounds of MSCP. For each case and each graph, we used the iterated double-phase tabu search procedure presented in Section 2.4 to collect 1200

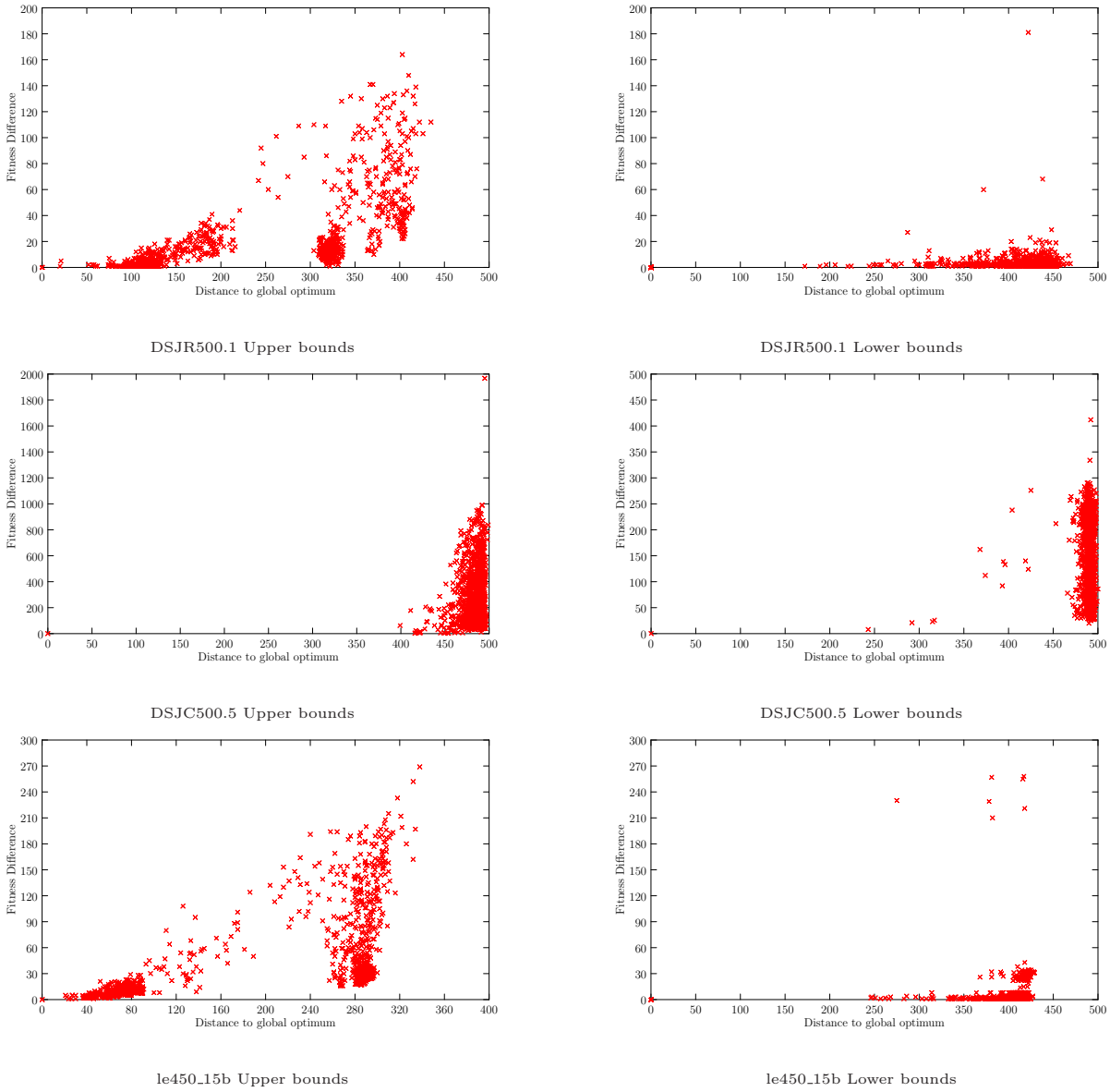

Fig. 7. FD correlation plots on 4 graphs for upper and lower bounds of MSCP

high quality solutions (one solution per IDTS run). Moreover, since optimal solutions are unknown for the selected graphs, we used the best solutions found by HESA as an approximation of the optima to calculate the FDC values. Table 6 presents the results of the FDC analysis on these graphs. For each graph, we identified the number of distinct local optima among these 1200 collected solutions ($\#lo$), the average distance between local optima ($avg\ d_{lo}$), the average distance between a local optimum and the closest best known local optimum ($avg\ d_{go}$) and the FDC coefficient (ρ).

From Table 6, one notices that for the minimization problem of upper bounds, the ρ values of COLOR 2002-2004 instances (close to 1) are larger than the ρ values of most DIMACS instances (close to 0). For the maximization problem

of lower bounds, the ρ values of COLOR 2002-2004 instances are close to -1 while the ρ values of most DIMACS instances are close to 0. These observations indicate that most DIMACS instances would be more difficult to solve compared to the COLOR 2002-2004 instances. This is indeed coherent with the experimental results of Section 4.3. In order to investigate the landscape in a visual way, we provide the FDC plots in Figure 7 with respect to the fitness difference and the distance between a local optimum and the nearest global optimum on three difficult DIMACS graphs (for the problems of upper and lower bounds). We can clearly see that there is no correlation for DSJC500.5 (for both problems of upper bounds and lower bounds), for DSJR500.1 and for le450_15b (the lower bounds). Finally, we insist that FDC alone cannot fully characterize the hardness of a problem instance. As such, when HESA does not perform well on a particular instance, this may be due to the real difficulty of the instance as measured by the FDC value. Or it may as well imply that the search operators of HESA, especially the crossover operators are not strong enough to escape from deep optima independent of the FDC value of the instance at hand.

6 Conclusion

In this paper, we presented an efficient hybrid search algorithm (HESA) for the upper and lower bounds of the minimum sum coloring problem (MSCP). HESA combines a double-crossover recombination method, a dedicated iterated double-phase tabu search (IDTS) procedure and a quality and diversity based population updating method. The recombination method jointly applies a diversification-guided crossover (DGX) and a grouping-guided crossover (GGX) to generate promising offspring solutions. The IDTS applies specific strategies to make transitions between feasible and infeasible solutions and a perturbation mechanism to escape local optima traps.

Experimental evaluations on 94 benchmark instances showed that the proposed HESA algorithm is highly competitive in comparison with the state-of-the-art algorithms for MSCP. HESA can match most of the current best known upper and lower bounds. In particular, it is able to improve the best known upper bound for 24 graphs and the best known lower bound for 27 graphs.

Additionally, we carried out experiments to verify the merit of the double-crossover recombination method. Moreover, we showed the first landscape analysis on a number of selected instances for the upper and lower bounds of MSCP, which allows us to understand why some instances are more difficult than others.

The existing studies showed that crossover is a very useful search operator for MSCP. For future work, it would be interesting to investigate other graph coloring crossover operators like those introduced in [31] and explore other ways of recombining solutions as it is done with the so-called path relinking method [10,12]. It would also be worthy of testing multi-parent variants of the DGX and GGX crossovers presented in Section 2.3.1 and studying adaptations of multi-parent coloring crossover operators like those proposed in [26,33].

Acknowledgments

We are grateful to the anonymous referees and Dr. Una Benlic for valuable suggestions and comments which have helped us to improve the paper. The work is partially supported by the LigeRo project (2009-2013) from the Region of Pays de la Loire (France) and the PGMO (2014-0024H) project from the Jacques Hadamard Mathematical Foundation. Support for Yan Jin from the China Scholarship Council is also acknowledged.

References

[1] L. Altenberg, 1997. Fitness distance correlation analysis: an instructive counterexample. In Proceedings of the 7th International Conference on Genetic Algorithms (ICGA97), pages 57–64, Morgan Kaufmann, San Francisco, CA.

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

[3] U. Benlic, J.K. Hao, 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.

[4] F. Bonomo, G. Durán, J. Marenco, M. Valencia-Pabon, 2011. Minimum sum set coloring of trees and line graphs of trees. Discrete Applied Mathematics 159(5): 288–294.

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

[6] D. Brélaz, 1979. New methods to color the vertices of a graph. Communications of the ACM 22(4): 251–256.

[7] C. Fleurent, J.A. Ferland, 1996. Genetic and hybrid algorithms for graph coloring. Annals of Operations Research 63(3): 437–461.

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

[9] F. Glover, 1994. Tabu Search for nonlinear and parametric optimization (with links to genetic algorithms). Discrete Applied Mathematics 49: 231–255.

[10] F. Glover, 1997. A template for scatter search and path relinking. Lecture Notes in Computer Science 1363: 1–51.

[11] F. Glover, M. Laguna, 1997. Tabu search. Kluwer Academic Publishers, Dordrecht, The Netherlands.

[12] F. Glover, M. Laguna, and R. Martí, 2000. Fundamentals of scatter search and path relinking. Control Cybernetics 39: 653–684.

[13] J.K. Hao, 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.

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

[15] A. Hertz, D. de Werra, 1987. Using tabu search techniques for graph coloring. Computing 39(4): 345–351.

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

[17] Y. Jin, J.K. Hao, J.P. Hamiez, 2014. A memetic algorithm for the minimum sum coloring problem. Computers & Operations Research 43: 318–327.

[18] Y. Jin, J.K. Hao, 2015. General swap-based multiple neighborhood tabu search for finding maximum independent set. Engineering Applications of Artificial Intelligence 37: 20–33.

[19] T. Jones, S. Forrest, 1995. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. Proceedings of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann, 184–192.

[20] Z. Kokosiński, K. Kwarciany, 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.

[21] L.G. Kroon, A. Sen, H. Deng, A. Roy, 1996. The optimal cost chromatic partition problem for trees and interval graphs. Lecture Notes in Computer Science 1197: 279–292.

[22] E. Kubicka, A.J. Schwenk, 1989. An introduction to chromatic sums. In Proceedings of the 17th ACM Annual Computer Science Conference, pages 39–45, New York, NY, USA.

[23] E. Kubicka, G. Kubicki, D. Kountanis, 1991. Approximation algorithms for the chromatic sum. Proceedings of the First Great Lakes Computer Science Conference. Lecture Notes in Computer Science 507: 15–21.

[24] F.T. Leighton, 1979. A graph coloring algorithm for large scheduling problems. Journal of research of the national bureau of standards 84(6): 489–506.

[25] Y. Li, C. Lucet, A. Moukrim, K. Sghiouer, 2009. Greedy algorithms for the minimum sum coloring problem. In: International Workshop on Logistics and Transports, Sousse, Tunisia, pp.LT–027.

[26] Z.P. Lü, J.K. Hao, 2010. A memetic algorithm for graph coloring. European Journal of Operational Research 203(1): 241–250.

[27] M. Malafiejski, 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.

[28] P. Moscato, C. Cotta, 2003. A gentle introduction to memetic algorithms. In F. Glover and G. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer, Norwell, Massachusetts, USA.

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

[30] A. Moukrim, K. Sghiouer, C. Lucet, Y. Li, 2014. 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*

[31] P.B. Myszkowski, 2008. Solving scheduling problems by evolutionary algorithms for graph coloring problems. In F. Xhafa and A. Abraham (Eds.): Metaheuristics for scheduling in industrial and manufacturing applications. Studies in Computational Intelligence, Vol. 128, pages 145–167.

[32] F. Neri, C. Cotta, P. Moscato (Eds.), 2012. Handbook of Memetic Algorithms. Studies in Computational Intelligence 379, Springer.

[33] D. Porumbel, J.K. Hao, P. Kuntz, 2010. An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. Computers & Operations Research 37(10): 1822–1832.

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

[35] K. Sörensen, M. Sevaux, 2006. Ma|pm memetic algorithms with population management. Computers and Operations Research 33(5):1214–1225.

[36] K.J. Supowit, 1987. Finding a maximum planar subset of a set of nets in a channel. IEEE transactions on computer-aided design of integrated circuits and systems 6(1): 93–94.

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

[38] Q. Wu, J.K. Hao, 2013. Improved lower bounds for sum coloring via clique decomposition. CoRR abs/1303.6761.