A memetic algorithm for the minimum sum coloring problem

Yan Jin, Jin-Kao Hao *, Jean-Philippe Hamiez

LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France
***Computers & Operations Research** 43(3): 318-327, 2014*

Abstract

Given an undirected graph G, the Minimum Sum Coloring Problem (MSCP) is to find a legal assignment of colors (represented by natural numbers) to each vertex of G such that the total sum of the colors assigned to the vertices is minimized. This paper presents a memetic algorithm for MSCP based on a tabu search procedure with two neighborhoods and a multi-parent crossover operator. Experiments on a set of 77 well-known DIMACS and COLOR 2002-2004 benchmark instances show that the proposed algorithm achieves highly competitive results in comparison with five state-of-the-art algorithms. In particular, the proposed algorithm can improve the best known results for 15 instances.

Keywords: Sum coloring, memetic algorithm, heuristics, combinatorial optimization

1 Introduction

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 $c(v_i) \neq c(v_j),\ \forall \{v_i, v_j\} \in E$. A legal or proper k-coloring can also be defined as a partition of V into k independent sets or stables $V_1, \ldots, V_k$ such that $\forall u, v \in V_i\ (i = 1, \ldots, k), \{u, v\} \notin E$. The classical Graph Coloring Problem (GCP) aims at finding a proper k-coloring with k minimum.

This paper is dedicated to the NP-hard Minimum Sum Coloring Problem (MSCP) [10,11], which is closely related to GCP. The objective of MSCP is to find a proper k-coloring which minimizes the sum of the colors assigned

* Corresponding author.
Email address: hao@info.univ-angers.fr (Jin-Kao Hao).

to the vertices. This minimum is the chromatic sum $\Sigma(G)$ of G: $\Sigma(G) = \min_{c \in \mathcal{C}} f(c)$, with $\mathcal{C}$ the set of all proper k-colorings of G (for all possible k values) and $f(c) = \sum_{i=1}^{n} c(v_i)$, or $f(c) = \sum_{l=1}^{k} l|V_l|$ equivalently (where $|V_l|$ is the cardinality of V_l), the "coloring sum" of the proper k-coloring c. MSCP applications include VLSI design, scheduling, and resource allocation [15].

Considering the theoretical intractability of MSCP, a number of heuristic algorithms have been proposed to find suboptimal solutions, such as a parallel genetic algorithm [9], two greedy heuristics [12], a tabu search metaheuristic [2], a memetic algorithm (MA) [17], an independent set extraction heuristic (EXSCOL) [21], a local search heuristic (MDS5) [7], and a breakout local search (BLS) [1]. To our knowledge, EXSCOL, MDS5, MA and BLS are the state-of-the-art algorithms in the literature. EXSCOL is based on extracting large disjoint independent sets and is particular effective for handling large graphs (with at least 500 vertices). MDS5 is based on variable neighborhood search and iterated local search. MA uses a local search procedure within a genetic algorithm. BLS is an iterated local search algorithm with an adaptive perturbation mechanism.

This paper introduces another Memetic Algorithm for the minimum Sum Coloring problem (MASC), which relies on three key components. First, a double-neighborhood tabu search procedure is especially designed for MSCP (DNTS). DNTS is based on a token-ring application of two complementary neighborhoods to explore the search space and a perturbation strategy to escape from local optima. Second, a multi-parent crossover operator is used for solution recombination. Basically, it tries to transmit large color classes from the parents to the offspring. Finally, a population updating mechanism is devised to determine how the offspring solution is inserted into the population.

We evaluate the performance of MASC on 77 well-known graphs from DIMACS and COLOR 2002-2004 graph coloring competitions. The computational results show that MASC can frequently match the best known results in the literature for most cases. In particular, it improves the previous best solution for 15 graphs for which an upper bound is known.

The paper is organized as follows. Next section describes the general framework and the components of our MASC memetic algorithm, including the population initialization, the crossover operator and the double-neighborhood tabu search procedure. Detailed computational results and comparisons with five state-of-the-art algorithms are presented in Section 3. Before concluding, Section 4 investigates and analyzes three key issues of the proposed memetic algorithm.

2 MASC: A Memetic Algorithm for minimum Sum Coloring

A memetic algorithm is a population-based approach where the traditional mutation operator is replaced by a local search procedure [18,19]. Memetic algorithms are among the most powerful paradigms for solving NP-hard combinatorial optimization problems. In particular, they have been successfully applied to the tightly related GCP [4,13,16,20].

Our MASC algorithm is summarized in Algorithm 1. After population initialization, MASC repeats a series of generations (limited to $MaxGeneration$) to explore the search space which is defined by the set of all proper k-colorings (k is not a fixed value, Section 2.1). At each generation, two or more parents are selected at random (line 6) and used by the dedicated crossover operator to generate an offspring solution (line 7, Section 2.3). The offspring solution is then improved by a double neighborhood tabu search (line 8, Section 2.4). If the improved offspring has a better sum of colors, it is then used to update the current best solution found so far (lines 9-10). Finally, the population updating criterion decides whether the improved offspring will replace one existing individual of the population or not (line 12, Section 2.5).

Algorithm 1 An overview of the MASC memetic algorithm for MSCP

```
1: input: A graph G
2: output: The minimum sum coloring c_* found and its objectif f(c_*)
3: Population_Initialization(P, p) /* Population P has p solutions, Sect. 2.2 */
4: f_* ← min_{c∈P} f(c) /* f_* records the best objective value found so far */
5: for i ← 1 to MaxGeneration do
6:    P' ← Selection(P) /* Select 2 or more parents at random for crossover */
7:    o ← Crossover(P') /* Crossover to get an offspring solution, Sect. 2.3 */
8:    o ← DNTS(o) /* Improve o with the DNTS procedure, Sect. 2.4*/
9:    if f(o) < f_* then
10:      f_* ← f(o); c_* ← o
11:   end if
12:   Population_Updating(P, o) /* Sect. 2.5 */
13: end for
14: return f_*, c_*
```

As we observe in the rest of this section, compared to the algorithm of [17] which is also based on the memetic framework, our memetic algorithm uses quite different design for the key components (initial population, crossover, local search, population management) of our MASC algorithm.

2.1 Search Space and Evaluation Function

The search space explored by MASC is the set $\mathcal{C}$ of all *proper* k-colorings of G (k is not fixed). For a given proper k-coloring c, its quality is directly assessed by the sum of colors $f(c) = \sum_{v \in V} c(v) = \sum_{l=1}^{k} l|V_l|$.

2.2 Initial Population

Our algorithm begins with a population P of p feasible colorings. This population can be obtained by any graph coloring algorithm that is able to generate different proper colorings for a graph. In our case, we employ the well-known TABUCOL [8], more precisely its improved version introduced in [4]. For a given graph G, TABUCOL tries to find a proper k-coloring where k is the best known result for GCP, i.e., the smallest k for which a k-coloring is known in the literature. If TABUCOL cannot reach a proper k-coloring for the current k value, TABUCOL is restarted with k increased by 1 (this makes the task of finding a legal coloring easier). This process is repeated until a proper k-coloring is obtained. Each resulting k-coloring is then submitted to the dedicated DNTS procedure to improve its coloring sum (see Sect. 2.4). Each improved k-coloring is finally inserted into P if the coloring is not already present in P (discarded otherwise). This process is repeated until P is filled with p different k-colorings. Notice that the solutions generated by TABUCOL may take different k values due to the stochastic nature of TABUCOL. Also in Section 4.3, we provide a comparative study to show to which extent the solutions can be improved by the proposed MASC approach.

2.3 Crossover Operator

The crossover operator is an important component in a population-based algorithm. It is used to generate one or more new offspring individuals to discover new promising search areas.

MASC uses a multi-parent crossover operator, called MGPX, which is similar to the one introduced in [6] as a variant of the well-known GPX crossover first proposed in [4] for GCP (restricted to two parents). MGPX generates only one offspring solution o from α parents randomly chosen from P, where α varies from 2 to 4 according to $|V|$ and the best k-coloring found for GCP (see Eq. 1).

$$\alpha = \begin{cases} 2, \text{ if } |V|/k < 5 \\ 3, \text{ if } 5 \leq |V|/k \leq 15 \\ 4, \text{ otherwise} \end{cases} \quad (1)$$

Motivations for these α values can be found in [20]. The dense graphs obviously need more colors k such that the average color class sizes become very low ($|V|/k < 5$, i.e., the classes become very small). In this case, it is better to use 2 parents in order to avoid excessive disruptions when blending the color classes. Inversely, the sparse graphs need fewer colors such that the average class sizes are very high ($|V|/k > 15$). In this case, we use more parents (4 in our case) in order to increase the probability of selecting and inheriting good classes from different parents. Besides, we choose 3 parents for the graphs between these two extreme situations.

Algorithm 2 Pseudo-code of the MGPX combination operator

1: **input**: A set P' of α parents randomly chosen from P
2: **output**: An offspring o
3: $\nu \leftarrow 0$ /* Counts the number of colored vertices in o */
4: $\kappa \leftarrow 0$ /* Counts the number of colors used in o */
5: $\omega \leftarrow 0$ /* Counts the iterations */
6: Set forbidden length for each parent: $\tau(P_i) = 0$, $i = 1, ..., \alpha$
7: **while** $\nu < |V|$ **do**
8: $\quad \kappa \leftarrow \kappa + 1$
9: $\quad$ Indicate the non-forbidden parents: P_s=$\{P_i | \tau(P_i) \leqslant \omega\}$
10: $\quad$ Find the maximum cardinality class V_*^j ($V_*^j \in P_j$) from P_s
11: $\quad \nu \leftarrow \nu + |V_*^j|$
12: $\quad$ **for all** $v \in V_*^j$ **do**
13: $\quad\quad o(v) \leftarrow \kappa$
14: $\quad\quad$ Remove v from α parents P'
15: $\quad$ **end for**
16: $\quad \tau(P_j) \leftarrow \omega + \lfloor \alpha/2 \rfloor$ /* Forbid parent P_j for $\lfloor \alpha/2 \rfloor$ steps */
17: $\quad \omega \leftarrow \omega + 1$
18: **end while**
19: **return** o

MGPX is summarized in Algorithm 2. It builds the color classes of the o offspring one by one, transmitting as many vertices as possible from the parents at each step (for quality purpose) (lines 8-15). Once a parent has been used for transmitting an entire color class to o, the parent is not considered for $\lfloor \alpha/2 \rfloor$ steps with the purpose of varying the origins of the color classes of o (line 16). This strategy avoids transmitting always from the same parent and introduces some diversity in o [13]. Note that the offspring solution is always a proper k coloring while the number of colors used by the offspring can be

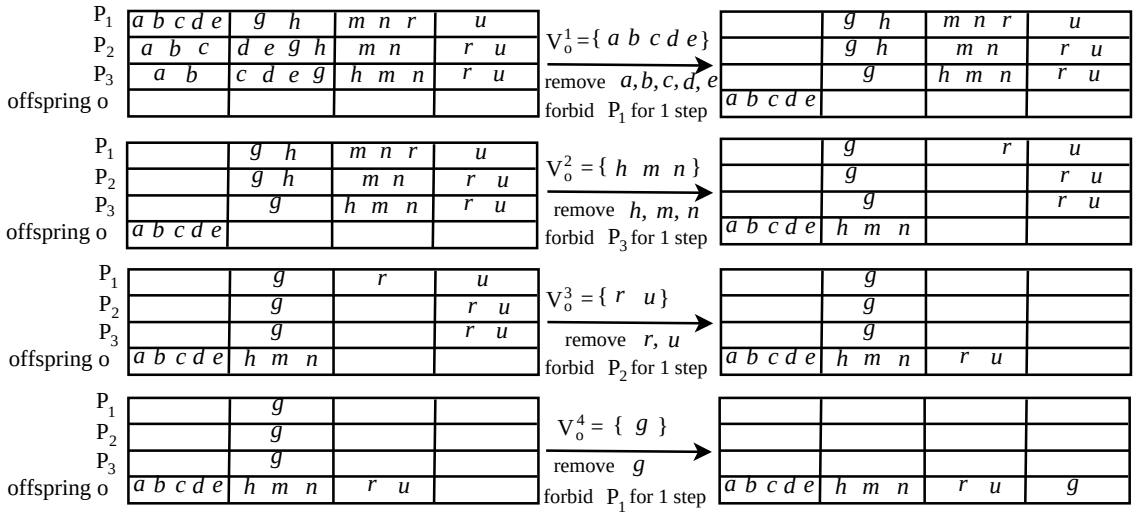


Fig. 1. An illustrative example of the MGPX crossover

higher than those of the considered parents.

Figure 1 illustrates the detailed operations of our MGPX crossover. In the example, there are 3 parents ($\alpha = 3$) with $k = 4$ colors, and 11 vertices $a, b, ..., u$. The forbidden length for each parent is initially set to 0 ($\tau(P_i) = 0$) which means all three parents can be selected at the beginning of the MGPX operator. At the first step, the largest color class $\{a, b, c, d, e\}$ in parent P_1 is chosen to become the first class V_o^1 of the offspring o. Then vertices a, b, c, d, e are removed from all three parents. And parent P_1 is forbidden for 1 step ($\lfloor \alpha/2 \rfloor = \lfloor 3/2 \rfloor = 1$). Similarly, we build the color class $V_o^2 = \{h, m, n\}$ from parent P_3, $V_o^3 = \{r, u\}$ from parent P_2 and $V_o^4 = \{g\}$ from parent P_1 respectively. After four steps, all the vertices are assigned such that a complete offspring is constructed. One notices that in this example, the sum of colors in the offspring is better than or equal with its parents.

2.4 A Double-Neighborhood Tabu Search for Sum Coloring

Local optimization is another important element within a memetic algorithm. In our case, its role is to improve as far as possible the quality (i.e., the sum of colors) of a given solution returned by the MGPX crossover operator. This is achieved by a Double-Neighborhood Tabu Search (DNTS) procedure specifically designed for MSCP (see Algorithm 3).

DNTS is based on tabu search [5] and uses two different and complementary neighborhoods N_1 and N_2 which are applied in a token-ring way [3,14] to find good local optima (intensification) (lines 2-14). More precisely, we start our search with one neighborhood (lines 6-9) and when the search ends with

its best local optimum, we switch to the other neighborhood to continue the search while using the last local optimum as the starting point (lines 10-13). When this second search terminates, we switch again to the first neighborhood and so on. DNTS continues the exploration of each neighborhood N_i $(i = 1, 2)$ until μ_i $(i = 1, 2)$ consecutive iterations fail to update the best solution found.

This neighborhood-based intensification phase terminates if the best local optimum is not updated for μ_ρ consecutive iterations (line 14). At this point, we enter into a diversification phase by triggering a perturbation to escape from the local optimum (line 15, Section 2.4.4). The DNTS procedure stops when a maximum number of iterations $MaxIters$ is met. We explain below the two neighborhoods, the tabu list management and the perturbation mechanism.

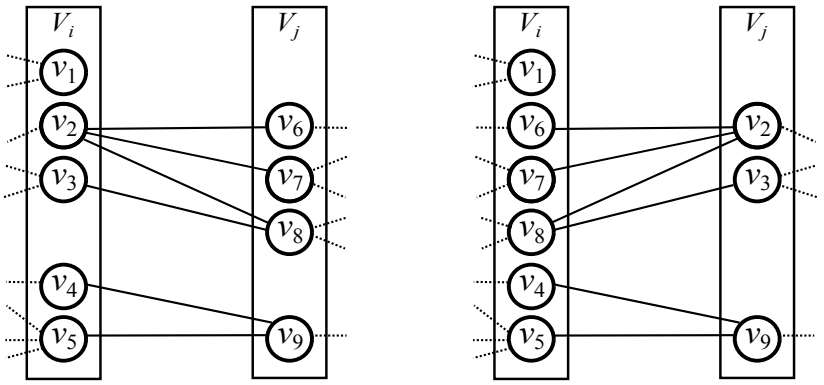
Algorithm 3 Pseudo-code of double-neighborhood tabu search for MSCP

```
1: Input: Graph G, a k-coloring c
2: Output: the best improved k-coloring
3: c* ← c
4: while a stop condition is not met do
5:    repeat
6:       c ← TS(N1, c, μ1)  /* Tabu search with neighborhood N1, Sect. 2.4.1 */
7:       if f(c) < f(c*) then
8:          c* ← c
9:       end if
10:      c ← TS(N2, c, μ2)    /* Tabu search with neighborhood N2, Sect. 2.4.2 */
11:      if f(c) < f(c*) then
12:         c* ← c
13:      end if
14:   until c* not improved for μρ consecutive iterations
15:   c ← Perturbation(c*)    /* Search is stagnating, generate a new starting
      solution by perturbing the best k-coloring found so far, Sect. 2.4.4 */
16: end while
```

2.4.1 N_1: A Neighborhood Based on Connected Components

The first neighborhood N_1 can be described by the operator $Exchange(i, j)$. Given a proper k-coloring $c = \{V_1, \ldots, V_k\}$, operator $Exchange(i, j), (1 \leq i \neq j \leq k)$ swaps some vertices of a color class V_i against some connected vertices of another color class V_j. Formally, let $G_{i,j}(c)$ be the set of all connected components of more than one vertex in the subgraph of G induced by color classes V_i and V_j in a proper k-coloring c. For a k-coloring c, the size of neighborhood N_1 is bounded by $O(\frac{n}{2} \times \frac{k(k-1)}{2})$ (n is the number of vertices). In Figure 2 (left) for instance, $G_{i,j}(c)$ is composed of two graphs (say g_1 and g_2): g_1 is the subgraph induced by $\{v_2, v_3, v_6, v_7, v_8\}$ and g_2 is induced by $\{v_4, v_5, v_9\}$.

Neighborhood $N_1(c)$ is composed of the set $\mathcal{G}(c)$ of all possible elements in all the $G_{i,j}(c)$ sets: $\mathcal{G}(c) = \cup_{1 \leq i < j \leq k} G_{i,j}(c)$. In other words, $N_1(c)$ includes all



A proper k-coloring c A possible $c' \in N_1(c)$

Fig. 2. N_1: An illustrative example with two partial colorings (c and c' are restricted here to two V_i and V_j color classes)

the proper k-colorings that can be obtained from the current k-coloring c by exchanging the vertices of a connected component induced by color classes V_i and V_j. Figure 2 shows an example where by exchanging the two sets of vertices $\{v_2, v_3\}$ and $\{v_6, v_7, v_8\}$ of connected component g_1 of the current k-coloring c (left drawing), we obtain a neighboring k-coloring c' (right drawing).

2.4.2 N_2: A Neighborhood Based on One-Vertex-Move

The second neighborhood N_2 is conventional and is simpler than N_1. N_2 can be described by the operator $OneMove(v, i, j)$. Given a proper k-coloring $c = \{V_1, \ldots, V_k\}$, operator $OneMove(v, i, j), (1 \leq i \neq j \leq k)$ displaces one single vertex v of a color class V_i to another color class V_j such that the resulting k-coloring remains proper. For instance, from the current coloring c of the left drawing of Figure 2, moving vertex v_1 from V_i to V_j gives a neighboring solution. Neighborhood $N_2(c)$ is composed of all the possible proper k-colorings by applying $OneMove(v, i, j)$ to the current k-coloring c. Like neighborhood N_1, the solutions of this second neighborhood are also proper k-colorings. The size of the neighborhood N_2 is bounded by $O(n \times k)$. Moreover, the number of colors of the neighboring solutions remains the same as that of the current coloring.

2.4.3 Neighborhood Examination and Tabu List

DNTS applies these two neighborhoods N_1 and N_2 in a token-ring way [3,14]. The alternation between N_1 and N_2 is triggered when the current neighborhood is exhausted, i.e., when the current best solution cannot be further improved for a fixed number of consecutive iterations.

As shown in Algorithm 3, at each iteration of our DNTS, a best neighboring

solution is selected among all the allowed solutions (from N_1 or N_2) to replace the current solution. Precisely, for the neighborhood N_1 defined by the operator $Exchange(i, j)$, we first identify all the connected components in each pair of color classes for the current k-coloring. Theoretically, this step has a worst time complexity of $O(\frac{n^2}{2} \times \frac{k(k-1)}{2})$. But in practice, the time consuming is much lower since this operator is related to the density of the graph. Then we select the best connected components for exchange according to the objective function $f(c)$ (ties are broken at random). When a set of vertices of a color class V_i are exchanged with a set of vertices of another color class V_j, exchanges between V_i and V_j are forbidden for the next TT iterations (called tabu tenure). Finally, we only need to update the connected components in the pairs of color classes which contain class V_i or V_j. For the neighborhood N_2 defined by the $OneMove(v, i, j)$ operator, we go through all legal moves (there are $O(n \times k)$ of them) and select a best move for the $OneMove(v, i, j)$ operation. When a vertex v of a color class V_i is displaced to another color class V_j, the vertex v is forbidden to go back to V_i for the next TT iterations.

The tabu list is introduced to avoid short-term cycles [5] and is updated after each iteration. The tabu tenure TT is determined simply by taking a random number from $\{0, \ldots k-1\}$. Moreover, a forbidden $Exchange$ or $OneMove$ operation is always accepted if it leads to a neighboring solution better than the best solution found so far (this is called aspiration according to the tabu search terminology).

2.4.4 The Perturbation Mechanism

In addition to the basic diversification mechanism of the tabu list, our DNTS algorithm applies a stronger diversification strategy based on perturbations to escape deep local optima. The perturbation is triggered when the current intensification phase cannot update the recorded best solution c_* for μ_ρ consecutive iterations (see line 15, Algorithm 3). In this case, the search is considered to be trapped in a deep local optimum and a strong diversification is needed to bring the search to a new search region. To achieve this, we apply the following perturbation technique to modify the recorded best solution c_* and then use this perturbed solution to initialize DNTS. Suppose c_* is composed of k different color classes and let V_l be the largest color class. We introduce an additional color class V_{k+1} and then move randomly one third of the vertices of V_l into V_{k+1}. In order to prevent the subsequent search from coming back to c_*, V_l and V_{k+1} are classified tabu and cannot take part of an $Exchange$ or a $OneMove$ operation for the next TT iterations (see Section 2.4.3).

2.5 Population Updating

The management of the population usually controls and balances two important factors in population-based heuristics: Quality and diversity.

Quality can naturally be measured here using the coloring sum function (f). The proper k-coloring c_i is better than c_j if $f(c_i) < f(c_j)$.

We use the following distance H to estimate the diversity. Given two coloring c_i and c_j, $H_{i,j}$ is the number of vertices in c_i and c_j which have different colors: $H_{i,j} = |\{v \in V : c_i(v) \neq c_j(v)\}|$. A small $H_{i,j}$ value indicates a high similarity between c_i and c_j. H is also employed to measure how much diversity $H_{i,P}$ a particular k-coloring c_i contributes to the entire population P: $H_{i,P} = \min_{j \neq i} H_{i,j}$. Again, a small (large) $H_{i,P}$ value indicates that c_i adds a low (high) diversity to P.

In MASC, f and H are combined in a s "score" function which is used to decide whether an offspring solution o replaces an individual in the population P or not: $s(c_i) = f(c_i) + e^{0.08|V|/H_{i,P}}\ \forall c_i \in P$. Precisely we first add o into P and compute all $s(c_i)$. We then identify the worse configuration c_w (i.e., $s(c_w)$ is maximum). The replacement strategy applies the following rules:

Case 1 (c_w is not o): Remove c_w from P;
Case 2 (c_w is o): Remove the second worse individual from P with probability 0.2, and discard o otherwise.

Notice that unlike partition based distances [20], the distance used here does not consider explicitly the symmetry of solutions. We adopted this simpler distance for two practical reasons. First, given that the solutions are all improved by DNTS (tabu search), the population has generally a certain level of diversity. So the diversity control mechanism has a limited role. Second, the computation of a partition distance is much more expensive. The experimental results show that in the context of this work, the above distance seems sufficient for our MASC algorithm.

3 Experimental Results

Our MASC approach was tested on a benchmark composed of 77 well-known graphs commonly used to report computational results for MSCP: 39 are part of the COLOR 2002–2004 competitions and the 38 others are known as "DIMACS" instances. Most of these graphs are available on-line from

http://mat.gsia.cmu.edu/COLOR04 [1], except the 6 "flat" instances that can be retrieved from http://mat.gsia.cmu.edu/COLOR/instances.html. The main characteristics of each graph appear in Tables 2 and 5, see columns 1–4 (COLOR 2002–2004 instances are at the top of Table 2 and DIMACS instances at the bottom): Name of the graph, order ($|V|$), size ($|E|$), and chromatic sum f_b.

MASC is programmed in C++ and compiled using GNU gcc on a PC with 2.7 GHz CPU and 4 Gb RAM. Like many memetic algorithms, we use a small population of 10 individuals. The values of the other parameters were determined empirically, see Table 1. Notice that $MaxGenerations = 50$ is the stop condition that determines the running time of the algorithm. Given its stochastic nature, MASC is run 30 times with different seeds. The best results of our MASC algorithm are available at http://www.info.univ-angers.fr/pub/hao/masumcol.html.

Table 1
Settings of parameters

Parameter	Sect.	Description	Value
μ_1	2.4	Maximum number of non-improving moves for TS using N_1	500
μ_2	2.4	Maximum number of non-improving moves for TS using N_2	1 000
μ_ρ	2.4	Maximum number of non-improving moves of TS for perturbation	4 000
$MaxIters$	2.4	Maximum iterations of TS procedure	10 000
$MaxGenerations$	2.5	Maximum number of generations	50

3.1 Computational Results

Columns 6–10 in Table 2 present detailed computational results of our MASC algorithm: Best result obtained (f_*) with the number of required colors (k_*), success rate (SR, percentage of runs such that the sum of colors f_* of MASC is better than the current best known value f_b, i.e., $f_* \leq f_b$), average coloring sum (Avg.), standard deviation (σ) and average running time to reach f_* (t, in minutes). Column k shows the chromatic number or its best upper bound (i.e., the smallest number of colors for which a k-coloring is ever reported). The reported values are based on 30 independent runs (i.e., with different random seeds).

From Table 2, one observes that for the 39 COLOR 2002–2004 instances with known upper bounds (see top part of the table), MASC improves the best known upper bound for two instances (miles500 and homer) and equals the best known results for the other 37 graphs. Furthermore, MASC achieves robust results here since SR = 30/30 and $\sigma = 0.0$ for these graphs except two

[1] The homer instance available from http://mat.gsia.cmu.edu/COLOR04/ contains some invalid entries for MSCP (e.g., "e 95 95" appears twice), hence we remove these two edges before running our algorithm.

Table 2
Detailed computational results of MASC on the set of 39 COLOR 2002-2004 instances (upper part) and 24 DIMACS instances (bottom part)

Characteristics of the graphs				MASC					
Name	$\|V\|$	$\|E\|$	f_b	k	$f_*(k_*)$	SR	Avg.	σ	t
myciel3	11	20	21	4	21(4)	30/30	21.0	0.0	0.0
myciel4	23	71	45	5	45(5)	30/30	45.0	0.0	0.0
myciel5	47	236	93	6	93(6)	30/30	93.0	0.0	0.0
myciel6	95	755	189	7	189(7)	30/30	189.0	0.0	0.1
myciel7	191	2 360	381	8	381(8)	30/30	381.0	0.0	1.1
anna	138	493	276	11	276(11)	30/30	276.0	0.0	0.1
david	87	406	237	11	237(11)	30/30	237.0	0.0	0.1
huck	74	301	243	11	243(11)	30/30	243.0	0.0	0.0
jean	80	254	217	10	217(10)	30/30	217.0	0.0	0.0
homer	561	1 628	1 157	13	**1 155(13)**	1/30	1 158.5	1.7	63.9
queen5.5	25	160	75	5	75(5)	30/30	75.0	0.0	0.0
queen6.6	36	290	138	7	138(8)	30/30	138.0	0.0	1.1
queen7.7	49	476	196	7	196(7)	30/30	196.0	0.0	0.0
queen8.8	64	728	291	9	291(9)	30/30	291.0	0.0	12.8
queen9.9	81	1 056	409	10	409(10)	9/30	410.5	1.2	1.2
queen8.12	96	1 368	624	12	624(12)	30/30	624.0	0.0	0.0
games120	120	638	443	9	443(9)	30/30	443.0	0.0	0.5
miles250	128	387	325	8	325(8)	30/30	325.0	0.0	0.4
miles500	128	1 170	$\leq$ 708	20	**705(20)**	30/30	705.0	0.0	1.0
fpsol2.i.1	496	11 654	3 403	65	3 403(65)	30/30	3 403.0	0.0	8.7
fpsol2.i.2	451	8 691	1 668	30	1 668(30)	30/30	1 668.0	0.0	5.7
fpsol2.i.3	425	8 688	1 636	30	1 636(30)	30/30	1 636.0	0.0	7.0
mug88_1	88	146	178	4	178(4)	30/30	178.0	0.0	0.1
mug88_25	88	146	178	4	178(4)	30/30	178.0	0.0	0.2
mug100_1	100	166	202	4	202(4)	30/30	202.0	0.0	0.2
mug100_25	100	166	202	4	202(4)	30/30	202.0	0.0	0.3
2-Insertions_3	37	72	62	4	62(4)	30/30	62.0	0.0	0.0
3-Insertions_3	56	110	92	4	92(4)	30/30	92.0	0.0	0.0
inithx.i.1	864	18 707	3 676	54	3 676(54)	30/30	3 676.0	0.0	7.6
inithx.i.2	645	13 979	2 050	31	2 050(31)	30/30	2 050.0	0.0	4.4
inithx.i.3	621	13 969	1 986	31	1 986(31)	30/30	1 986.0	0.0	1.8
mulsol.i.1	197	3 925	1 957	49	1 957(49)	30/30	1 957.0	0.0	0.1
mulsol.i.2	188	3 885	1 191	31	1 191(31)	30/30	1 191.0	0.0	0.2
mulsol.i.3	184	3 916	1 187	31	1 187(31)	30/30	1 187.0	0.0	0.2
mulsol.i.4	185	3 946	1 189	31	1 189(31)	30/30	1 189.0	0.0	0.2
mulsol.i.5	186	3 973	1 160	31	1 160(31)	30/30	1 160.0	0.0	0.2
zeroin.i.1	211	4 100	1 822	49	1 822(49)	30/30	1 822.0	0.0	0.2
zeroin.i.2	211	3 541	1 004	30	1 004(30)	30/30	1 004.0	0.0	0.1
zeroin.i.3	206	3 540	998	30	998(30)	30/30	998.0	0.0	0.1
DSJC125.1	125	736	326	5	326(7)	20/30	326.6	0.9	4.4
DSJC125.5	125	3 891	1 012	17	1 012(18)	2/30	1 020.0	3.9	3.5
DSJC125.9	125	6 961	2 503	44	2 503(44)	12/30	2 508.0	5.6	1.9
DSJC250.1	250	3 218	973	8	974(9)	0/30	990.5	8.3	17.3
DSJC250.5	250	15 668	3 214	28	3 230(31)	0/30	3 253.7	14.3	23.1
DSJC250.9	250	27 897	8 277	72	8 280(74)	0/30	8 322.7	22.3	5.6
DSJC500.1	500	12 458	2850	12	2 940(14)	0/30	3 013.4	28.3	50.4
DSJC500.5	500	62 624	10 910	48	11 101(53)	0/30	11 303.5	73.9	202.5
DSJC500.9	500	112 437	29 912	126	29 994(126)	0/30	30059.1	31.6	90.9
flat300_20_0	300	21 375	3 150	20	3 150(20)	30/30	3 150.0	0.0	0.0
flat300_26_0	300	21 633	3 966	26	3 966(26)	30/30	3 966.0	0.0	0.8
flat300_28_0	300	21 695	$\leq$ 4 261	28	**4 238(30)**	1/30	4 313.4	22.3	309.7
le450_5a	450	5 714	1 350	5	1 350(5)	30/30	1 350.0	0.0	0.7
le450_5b	450	5 734	1 350	5	1 350(5)	30/30	1 350.0	0.0	0.4
le450_5c	450	9 803	1 350	5	1 350(5)	30/30	1 350.0	0.0	0.2
le450_5d	450	9 757	1 350	5	1 350(5)	30/30	1 350.0	0.0	0.5
le450_15a	450	8 168	2 632	15	2 706(19)	0/30	2 742.6	13.8	41.3
le450_15b	450	8 169	2 642	15	2 724(19)	0/30	2 756.2	14.8	40.3
le450_15c	450	16 680	$\leq$ 3 866	15	**3 491(16)**	30/30	3 491.0	0.0	45.3
le450_15d	450	16 750	$\leq$ 3 921	15	**3 506(17)**	30/30	3 511.8	3.6	59.8
le450_25a	450	8 260	3 153	25	3 166(27)	0/30	3 176.8	4.4	39.2
le450_25b	450	8 263	3 366	25	3 366(26)	1/30	3 375.1	3.4	40.3
le450_25c	450	17 343	4 515	25	4 700(31)	0/30	4 773.3	25.2	75.3
le450_25d	450	17 425	4 544	25	4 722(29)	0/30	4 805.7	27.4	63.4

instances (homer and queen9.9). The average running time of MASC ranges from less than one second to about 13 minutes except for the homer instance.

For the set of 24 DIMACS instances (bottom part), the MASC algorithm improves the best known upper bound for 3 graphs (flat300_28_0, le450_15c, and le450_15d) and equals the best known results for 10 instances. Unfortunately, MASC was unable to reach the best known results for the other 11 graphs (see lines where SR = 0/30). The average running time is less than 76 minutes except for the DSJC500.5 and flat300_28_0 instances. Finally, we notice that the number of colors needed to ensure the best sum coloring (k_*) can be larger than the chromatic number or its best upper bound (k).

3.2 Comparisons With State-of-the-art Algorithms

Table 3 compares MASC with 5 recent effective algorithms that cover the best known results for the considered benchmark: EXSCOL [21], BLS [1], MA [17], MDS5 [7] and MRLF [12]. No averaged value appears in the table for MA, MDS5 and MRLF since this information is not given in [7,12,17]. Furthermore, "–" marks signal that some instances were not tested by some approaches.

Since most reference algorithms give only results for a (small) subset of the considered benchmark, it is difficult to analyze the performance of these algorithms by statistical tests. Hence, we compare the performance between MASC and these reference algorithms one by one and summarize the comparisons in Table 4. The first column of Table 4 indicates the name of the reference heuristics, followed by the number $\#G$ of graphs tested by each algorithm and shown in Table 3. The last three columns give the number of times MASC reports a better, equal, or worse result compared to each reference algorithm.

From Table 4, it can be observed that MASC obtains absolutely no worse results than MDS5 and MRLF (see the last three lines). Furthermore, MASC gets better results than these algorithms for 9 and 16 instances respectively. Our algorithm is also quite competitive with EXSCOL, BLS and MA which are the most recent and effective methods since it obtains better or equivalent results for 28, 22 and 49 graphs respectively. MASC reaches worse results than EXSCOL, BLS and MA only for 8, 3 and 8 graphs respectively.

3.3 Experiment on Large Graphs

We turn now our attention to the performance of our MASC algorithm to color large graphs with at least 500 vertices. These large graphs are known to be quite difficult for almost all the existing sum coloring approaches except

Table 3
Comparisons of MASC with five state-of-the-art sum coloring algorithms

Graph		EXSCOL [21]		BLS [1]		MA [17]	MDS5 [7]	MRLF [12]	MASC	
Name	f_b	f_*	Avg.	f_*	Avg.	f_*	f_*	f_*	f_*	Avg.
myciel3	21	21	21.0	21	21.0	21	21	21	21	21.0
myciel4	45	45	45.0	45	45.0	45	45	45	45	45.0
myciel5	93	93	93.0	93	93.0	93	93	93	93	93.0
myciel6	189	189	189.0	189	196.6	189	189	189	189	189.0
myciel7	381	381	381.0	381	393.8	381	381	381	381	381.0
anna	276	283	283.2	276	276.0	276	276	277	276	276.0
david	237	237	238.1	237	237.0	237	237	241	237	237.0
huck	243	243	243.8	243	243.0	243	243	244	243	243.0
jean	217	217	217.3	217	217.0	217	217	217	217	217.0
homer	1 157	–	–	–	–	1 157	–	–	**1 155**	1 158.5
queen5.5	75	75	75.0	75	75.0	75	75	75	75	75.0
queen6.6	138	150	150.0	138	138.0	138	138	138	138	138.0
queen7.7	196	196	196.0	196	196.0	196	196	196	196	196.0
queen8.8	291	291	291.0	291	291.0	291	291	303	291	291.0
queen9.9	409	–	–	–	–	409	–	–	409	410.5
queen8.12	624	–	–	–	–	624	–	–	624	624.0
games120	443	443	447.9	443	443.0	443	443	446	443	443.0
miles250	325	328	333.0	327	328.8	325	325	334	325	325.0
miles500	$\leq$ 708	709	714.5	710	713.3	708	712	715	**705**	705.0
fpsol2.i.1	3 403	–	–	–	–	3 403	3 403	–	3 403	3 403.0
fpsol2.i.2	1 668	–	–	–	–	1 668	–	–	1 668	1 668.0
fpsol2.i.3	1 636	–	–	–	–	1 636	–	–	1 636	1 636.0
mug88_1	178	–	–	–	–	–	178	–	178	178.0
mug88_25	178	–	–	–	–	–	178	–	178	178.0
mug100_1	202	–	–	–	–	–	202	–	202	202.0
mug100_25	202	–	–	–	–	–	202	–	202	202.0
2-Insertions_3	62	–	–	–	–	–	62	–	62	62.0
3-Insertions_3	92	–	–	–	–	–	92	–	92	92.0
inithx.i.1	3 676	–	–	–	–	3 676	–	–	3 676	3 676.0
inithx.i.2	2 050	–	–	–	–	2 050	–	–	2 050	2 050.0
inithx.i.3	1 986	–	–	–	–	1 986	–	–	1 986	1 986.0
mulsol.i.1	1 957	–	–	–	–	1 957	–	–	1 957	1 957.0
mulsol.i.2	1 191	–	–	–	–	1 191	–	–	1 191	1 191.0
mulsol.i.3	1 187	–	–	–	–	1 187	–	–	1 187	1 187.0
mulsol.i.4	1 189	–	–	–	–	1 189	–	–	1 189	1 189.0
mulsol.i.5	1 160	–	–	–	–	1 160	–	–	1 160	1 160.0
zeroin.i.1	1 822	–	–	–	–	1 822	–	–	1 822	1 822.0
zeroin.i.2	1 004	–	–	–	–	1 004	1 004	–	1 004	1 004.0
zeroin.i.3	998	–	–	–	–	998	998	–	998	998.0
DSJC125.1	326	326	326.7	326	326.9	326	326	352	326	326.6
DSJC125.5	1 012	1 017	1 019.7	1 012	1 012.9	1 013	1 015	1 141	1 012	1 020.0
DSJC125.9	2 503	2 512	2 512.0	2 503	2 503.0	2 503	2 511	2 653	2 503	2 508.0
DSJC250.1	973	985	985.0	973	982.5	983	977	1 068	974	990.5
DSJC250.5	3 214	3 246	3 253.9	3 219	3 248.5	3 214	3 281	3 658	3 230	3 253.7
DSJC250.9	8 277	8 286	8 288.8	8 290	8 316.0	8 277	8 412	8 942	8 280	8 322.7
DSJC500.1	2 850	2 850	2 857.4	2 882	2 942.9	2 897	2 951	3 229	2 940	3 013.4
DSJC500.5	10 910	10 910	10 918.2	11 187	11 326.3	11 082	11 717	12 717	11 101	11 303.5
DSJC500.9	29 912	29 912	29 936.2	30 097	30 259.2	29 995	30 872	32 703	29 994	30 059.1
flat300_20_0	3 150	3 150	3 150.0	–	–	3 150	–	–	3 150	3 150.0
flat300_26_0	3 966	3 966	3 966.0	–	–	3 966	–	–	3 966	3 966.0
flat300_28_0	$\leq$ 4 261	4 282	4 286.1	–	–	4 261	–	–	**4 238**	4 313.4
le450_5a	1 350	–	–	–	–	1 350	–	–	1 350	1 350.0
le450_5b	1 350	–	–	–	–	1 350	–	–	1 350	1 350.0
le450_5c	1 350	–	–	–	–	1 350	–	–	1 350	1 350.0
le450_5d	1 350	–	–	–	–	1 350	–	–	1 350	1 350.0
le450_15a	2 632	2 632	2 641.9	–	–	2 681	–	–	2 706	2 742.6
le450_15b	2 642	2 642	2 643.4	–	–	2 690	–	–	2 724	2 756.2
le450_15c	$\leq$ 3 866	3 866	3 868.9	–	–	3 943	–	–	**3 491**	3 491.0
le450_15d	$\leq$ 3 921	3 921	3 928.5	–	–	3 926	–	–	**3 506**	3 511.8
le450_25a	3 153	3 153	3 159.4	–	–	3 178	–	–	3 166	3 176.8
le450_25b	3 366	3 366	3 371.9	–	–	3 379	–	–	3 366	3 375.1
le450_25c	4 515	4 515	4 525.4	–	–	4 648	–	–	4 700	4 773.3
le450_25d	4 544	4 544	4 550.0	–	–	4 696	–	–	4 722	4 805.7

Table 4
MASC vs. five state-of-the-art sum coloring algorithms

Competitor	#G	Results of MASC (f_*)		
		Better	Equal	Worse
EXSCOL [21]	36	12	16	8
BLS [1]	25	5	17	3
MA [17]	57	10	39	8
MDS5 [7]	34	9	25	0
MRLF [12]	25	16	9	0

Table 5
Results of MASC on 17 large graphs with at least 500 vertices

Characteristics of the graphs				EXSCOL		MASC				
Name	$\|V\|$	$\|E\|$	f_b	f_*	Avg.	k	$f_*(k_*)$	Avg.	σ	t
DSJC500.1	500	12 458	2 850	2 850	2 857.4	12	**2 841(14)**	2 844.1	3.2	28.9
DSJC500.5	500	62 624	10 910	10 910	10 918.2	48	**10 897(51)**	10 905.8	4.6	73.3
DSJC500.9	500	112 437	29 912	29 912	29 936.2	126	**29 896(131)**	29 907.8	5.8	59.0
DSJC1000.1	1 000	49 629	9 003	9 003	9 017.9	20	**8 995(22)**	9 000.5	3.0	70.7
DSJC1000.5	1 000	249 826	37 598	37 598	37 673.8	83	**37 594(87)**	37 597.6	1.2	200.4
DSJC1000.9	1 000	449 449	103 464	103 464	103 531.0	223	103 464(231)	103 464.0	0.0	125.9
flat1000_50_0	1 000	245 000	25 500	25 500	25 500.0	50	25 500(50)	25 500.0	0.0	0.1
flat1000_60_0	1 000	245 830	30 100	30 100	30 100.0	60	30 100(60)	30 100.0	0.0	114.6
flat1000_76_0	1 000	246 708	37 167	37 167	37 213.2	82	37 167 (85)	37 167.0	0.0	1.1
latin_sqr_10	900	307 350	42 223	42 223	42 392.7	98	**41 444(100)**	41 481.5	19.1	101.2
wap05	905	43 081	13 680	13 680	13 718.4	50	**13 669(51)**	13 677.8	3.7	3.3
wap06	947	43 571	13 778	13 778	13 830.9	46	**13 776(48)**	13 777.8	0.6	4.1
wap07	1 809	103 368	28 629	28 629	28 663.8	46	**28 617(50)**	28 624.7	3.8	12.4
wap08	1 870	104 176	28 896	28 896	28 946.0	45	**28 885(50)**	28 890.9	3.2	15.1
qg.order30	900	26 100	13 950	13 950	13 950.0	30	13 950(30)	13 950.0	0.0	3.8
qg.order40	1 600	62 400	32 800	32 800	32 800.0	40	32 800(40)	32 800.0	0.0	11.8
qg.order60	3 600	212 400	109 800	110 925	110 993.0	60	109 800(60)	109 800.0	0.0	290.6

EXSCOL which dominates the other heuristics particularly on large graphs. We show a new experiment with MASC applied to color 17 large graphs. In this experiment, we run MASC 10 times on each graph under exactly the same condition as in Section 3.1. The only difference is that we use the solution of EXSCOL [2] as one of MASC's 10 initial solutions while the 9 other initial solutions are generated according to the procedure described in Section 2.2. With this experiment, we aimed to investigate two interesting questions. Is it possible for MASC to improve the results of the powerful EXSCOL algorithm? Does the initial population influence the performance of MASC? The computational outcomes of this experiment are provided in Table 5.

In Table 5, column 4 presents the best known result (f_b) in the literature, columns 5–6 present the best result (f_*) and the average coloring sum (Avg.)

[2] Available at `http://www.info.univ-angers.fr/pub/hao/exscol.html`

of EXSCOL and columns 7–11 present detailed computational results of our MASC algorithm: Best result obtained (f_*) with the number of required colors (k_*), average coloring sum (Avg.), standard deviation (σ), and average running time to reach f_* (t, in minutes). One notices that the values of columns 4 and 5 are identical except the *qg.order60* instance.

Table 5 shows that with the help of its search mechanism, our MASC algorithm is able to further improve the best known results of 10 instances (entries in bold). This is remarkable given that very few existing approaches can even equal the previous best known results. Moreover, if we contrast the results of the three DSJC500.x graphs ($x = 1, 5, 9$) reported in Tables 2 and 5, it is clear that the initial population impacts directly MASC's outcomes. This indicates that the performance of MASC could be further improved by using a more powerful coloring algorithm to generate the initial solutions of its population.

4 Analysis of MASC

In this section, we investigate the influence of three important ingredients of the proposed memetic algorithm, i.e., the multi-parent crossover operator, the combined neighborhood and the improvement of MASC over the initial population. Experiments were based on 16 selected graphs of different types, for which some reference algorithms cannot achieve the best known results. Hence, these selected instances can be considered to be difficult and representative.

4.1 Influence of the Multi-parent Crossover Operator

For our memetic algorithm, it is relevant to evaluate the effectiveness of its crossover operator. To verify this, we carry out experiments on the 16 selected graphs and run both MASC (using the MGPX crossover) and DNTS (without MGPX) for 30 times (with the same parameter μ_1, μ_2 and μ_ρ settings as defined in Table 1). The DNTS (without MGPX) starts with a single solution which is generated for MASC. DNTS stops when a maximum number of 5×10^5 iterations in order to make sure that MASC and DNTS are given the same search effort. The results are given in Table 6.

From Table 6, one notices that DNTS equals and improves respectively 5 and 3 best known results while MASC equals and improves respectively 5 and 11 best known results. Furthermore, the last column *tt* (t-test) indicates whether the observed difference between MASC and DNTS is statistically significant when a 95% confidence t-test is performed in terms of the best result obtained (f_*). If MASC and DNTS achieve always the same results, *tt* column is marked

Table 6
Comparative results of MASC and DNTS

Graph		MASC		DNTS		*tt*
Name	f_b	f_*	Avg.	f_*	Avg.	
anna	276	*276*	276.0	*276*	276.0	-
queen6.6	138	*138*	138.0	*138*	138.0	-
miles250	325	*325*	325.0	*325*	325.0	-
miles500	$\leq$ 709	**705**	705.0	**705**	705.6	Y
DSJC125.1	326	*326*	326.6	*326*	328.6	Y
DSJC125.5	1 012	**1 012**	1 020.0	1 016	1 029.8	Y
DSJC125.9	2 503	**2 503**	2 508.0	2 506	2 530.1	Y
DSJC250.1	973	**974**	990.5	981	997.7	Y
DSJC250.5	3 219	**3 230**	3 253.7	3 234	3 301.7	Y
DSJC250.9	$\leq$ 8 286	**8 280**	8 322.7	8 321	8 381.9	Y
flat300_26_0	3 966	*3 966*	3 966.0	*3 966*	3 966.0	-
flat300_28_0	$\leq$ 4 282	**4 238**	4 313.4	4 303	4 406.3	Y
le450_15c	$\leq$ 3 866	**3 491**	3 491.0	**3 491**	3 492.1	Y
le450_15d	$\leq$ 3 921	**3 506**	3 511.8	**3 506**	3 515.0	Y
le450_25c	4 515	**4 700**	4 773.3	4 749	4 803.9	Y
le450_25d	4 544	**4 722**	4 805.7	4 784	4 835.3	Y

by '-'. The t-test indicates that MASC is statistically better than DNTS for 12 out of 16 cases except for the instances where DNTS can achieve the best known results (f_b). These comparative results provide clear evidences that the MGPX crossover operator plays an important role in the MASC algorithm.

4.2 Influence of the Neighborhood Combination

The neighborhood is an important element that influences the local search procedure. Our proposed algorithm relies on two different neighborhoods: N_1 (neighborhood based on connected components) and N_2 (neighborhood based on one-vertex-move) which are explored in a token-ring way (see Section 2.4). In this section, we investigate the interest of this combined use of the two neighborhoods. For this purpose, we carried out experiments on the 16 selected graphs to compare the original Double-Neighborhood Tabu Search (DNTS) with two variants which uses only one neighborhood N_1 or N_2. We use below TS_{N1} and TS_{N2} to denote these two variants. These three TS procedures (DNTS, TS_{N1} and TS_{N2}) are run under the same stop condition, i.e. limited to 5×10^5 iterations.

We run 30 times these TS procedures to solve each of the 16 selected graphs and report the computational outcomes (the best and average results) in Table 7. One easily observes that DNTS obtains better or equal results compared to TS_{N1} and TS_{N2} for all the instances in terms of the best known result (f_*)

and the average result (Avg.). The t-test tt_i ($i = 1, 2$) in the last two columns confirms that with a 95% confidence level DNTS is slightly or significantly better than TS_{N1} and TS_{N2}. This experiment demonstrates thus the advantage of the token-ring combination of the two neighborhoods compared to each individual neighborhood.

Table 7
Comparative results of the tabu search improvement method according to the neighborhood employed

Graph		DNTS		TS_{N2}		TS_{N1}		tt_2	tt_1
Name	f_b	f_*	Avg.	f_*	Avg.	f_*	Avg.		
anna	276	276	276.0	282	285.8	276	276.0	Y	-
queen6.6	138	138	138.0	138	138.4	138	138.0	Y	-
miles250	325	**325**	325.0	346	361.6	335	340.7	Y	Y
miles500	$\leq$ 709	**705**	705.6	722	736.0	719	730.9	Y	Y
DSJC125.1	326	**326**	328.6	334	340.8	329	334.0	Y	Y
DSJC125.5	1 012	**1016**	1029.8	1 031	1 045.1	1 020	1 031.8	Y	N
DSJC125.9	2 503	**2 506**	2 530.1	2 514	2 557.6	2 512	2 538.3	Y	N
DSJC250.1	973	**981**	997.7	1 004	1 021.3	1 022	1 039.9	Y	Y
DSJC250.5	3 219	**3 234**	3 301.7	3 271	3 323.9	3 260	3 306.5	Y	N
DSJC250.9	$\leq$ 8 286	**8 321**	8 381.9	8 347	8 405.6	8 318	8 387.5	Y	N
flat300_26_0	3 966	3 966	3 966.0	3 966	3 966.0	3 966	3 966.0	-	-
flat300_28_0	$\leq$ 4 282	**4 303**	4 406.3	4 347	4 427.8	4 332	4 435.5	N	Y
le450_15c	$\leq$ 3 866	**3 491**	3 492.5	3 503	3 517.2	3 508	3 551.8	Y	Y
le450_15d	$\leq$ 3 921	**3 506**	3 515.0	3 528	3 538.5	3 526	3 568.2	Y	Y
le450_25c	4 515	**4 749**	4 803.9	4 828	4 893.9	5 005	5 067.4	Y	Y
le450_25d	4 544	**4 784**	4 835.3	4 848	4 907.0	5 035	5 119.1	Y	Y

4.3 Improvements of MASC over TABUCOL

Recall that the initial population is generated by the well-known graph coloring procedure TABUCOL. It is interesting to know to which extent our MASC procedure (which is specially designed for the Minimum Sum Coloring Problem) can improve the quality of solutions generated by TABUCOL in terms of sum of colors. For this purpose, we re-run 30 times our MASC procedure on the set of 16 selected graphs. Like for the previous experiments, we report in Table 8 the best and average objective value f_* both for TABUCOL (initial population) and MASC (final population). Given the stochastic nature of TABUCOL and MASC, some results reported in this experiment may be slightly different from those reported in Table 2.

From Table 8, one easily observes that MSCP improves significantly the initial results generated by TABUCOL. Indeed, the best and average sums of colors achieved by MSCP are systematically smaller (better) than those of TABU-

Table 8
Comparative results of MASC and TABUCOL

Graph		MASC		TABUCOL		tt
Name	f_b	f_*	Avg.	f_*	Avg.	
anna	276	276	276.0	371	374.4	Y
queen6.6	138	138	138.0	141	141.0	-
miles250	325	325	325.0	429	436.6	Y
miles500	$\leq$ 709	705	705.0	1 040	1 066.0	Y
DSJC125.1	326	326	327.0	337	348.2	Y
DSJC125.5	1 012	1 012	1 021.5	1 028	1 047.0	Y
DSJC125.9	2 503	2 503	2 509.6	2 523	2 563.9	Y
DSJC250.1	973	985	993.8	1 022	1 059.1	Y
DSJC250.5	3 219	3 230	3 263.2	3 279	3 312.4	Y
DSJC250.9	$\leq$ 8 286	8 290	8 319.5	8 338	8 373.8	Y
flat300_26_0	3 966	3 966	3 966.0	*3 966*	3 966.0	-
flat300_28_0	$\leq$ 4 282	4 238	4 313.4	4 363	4 430.4	Y
le450_15c	$\leq$ 3 866	3 491	3 491.0	3 532	3 584.7	Y
le450_15d	$\leq$ 3 921	3 506	3 512.6	3 567	3 591.8	Y
le450_25c	4 515	4 743	4 798.3	5 384	5 448.6	Y
le450_25d	4 544	4 750	4 833.4	5 226	5 464.4	Y

COL for all the graphs except one case (flat300_26_0) for which TABUCOL alone achieves already the best known result. Furthermore, the last column tt (t-test) confirms with a 95% confidence level the significance of the improvements of MASC over the solutions provided by TABUCOL.

5 Conclusion

In this paper, we presented a memetic algorithm (MASC) to deal with the Minimum Sum Coloring Problem (MSCP). The proposed algorithm employs an effective tabu search procedure with a combination of two neighborhoods, a multi-parent crossover operator and a population updating mechanism to balance intensification and diversification.

We assessed the performance of MASC on 77 well-known graphs from the DIMACS and COLOR 2002-2004competitions. MASC can improve 15 best known upper bounds including 10 large and very hard graphs with at least 500 vertices while equaling 54 previous best results. Compared with five recent and effective algorithms which cover the best known results for the tested instances, our MASC algorithm remains quite competitive.

Furthermore, we investigated two important components of the proposed algorithm. The experiments demonstrate the relevance of the multi-parent crossover operator and the combined neighborhood for the overall performance of MASC.

Finally, we showed the proposed MASC approach significantly improves the classical tabu search graph coloring approach for the Minimum Sum Coloring Problem.

Acknowledgment

We would like to thank the anonymous referees for their helpful comments and suggestions which helped us to improve the paper. The work is partially supported by the RaDaPop (2009-2013) and LigeRo projects (2009-2013) from the Region of Pays de la Loire (France). Support for Yan Jin from the China Scholarship Council is also acknowledged.

References

[1] U. Benlic, J.K. Hao. 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, 2012, pp. 128–137.

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

[3] L, Di Gaspero, A. Schaerf. Neighborhood portfolio approach for local search applied to timetabling problems. Journal of Mathematical Modeling and Algorithms 5(1) (2006) 65–89.

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

[5] F. Glover, M. Laguna. Tabu Search. Kluwer Academic Publishers, Dordrecht, The Nederlands, 1997.

[6] J.-P. Hamiez, J.K. Hao. Scatter search for graph coloring. In: P. Collet, E. Lutton, M. Schoenauer, C. Fonlupt, J.K. Hao (eds.), Artificial Evolution, vol. 2310 of Lecture Notes in Computer Science, Springer, Berlin / Heidelberg, Germany, 2002, pp. 168–179.

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

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

[9] Z. Kokosiński, K. Kwarciany. 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, 2007, pp. 211–219.

[10] E. Kubicka. The chromatic sum of graphs. Ph.D. thesis, Western Michigan University, USA (1989).

[11] E. Kubicka, A. Schwenk. An introduction to chromatic sums. In: Proceedings of the 17th ACM Annual Computer Science Conference, ACM Press, New York (NY) USA, 1989, pp. 39–45.

[12] Y. Li, C. Lucet, A. Moukrim, K. Sghiouer. Greedy algorithms for the minimum sum coloring problem. In: Logistique et Transports Conference, 2009.

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

[14] Z. Lü, J.K. Hao, F. Glover. Neighborhood analysis: a case study on curriculum-based course timetabling. Journal of Heuristics 17(2) (2011) 97–118.

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

[16] E. Malaguti, M. Monaci, P. Toth. A metaheuristic approach for the vertex coloring problem. INFORMS Journal on Computing 20 (2) (2008) 302–316.

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

[18] P. Moscato, C. Cotta. A gentle introduction to memetic algorithms. In: F. Glover, G. Kochenberger (eds.), Handbook of Metaheuristics, vol. 57 of International Series in Operations Research and Management Science, chap. 5, Kluwer Academic Publishers, Dordrecht, The Nederlands, 2003, pp. 105–144.

[19] F. Neri, C. Cotta, P. Moscato (eds.). Handbook of Memetic Algorithms. Vol. 379 of Studies in Computational Intelligence, Springer, Berlin / Heidelberg, Germany, 2012.

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

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