Memetic search for the equitable coloring problem

Wen Sun [a], Jin-Kao Hao [b,c,*] Wenyu Wang [d], Qinghua Wu [e],

[a] *School of Cyber Science and Engineering, Southeast University, No. 2, Southeast University Road, 211189, Nanjing, China*

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

[c] *Institut Universitaire de France, 1 Rue Descartes, 75231 Paris, France*

[d] *Department of Industrial System Engineering and Management, National University of Singapore, Singapore*

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

Knowledge-Based Systems
https://doi.org/10.1016/j.knosys.2019.105000

Abstract

Given an undirected graph $G = (V, E)$ and a positive integer k, an equitable legal k-coloring of G is a partition of the vertex set V into k disjoint independent sets such that the cardinalities of any two independent sets differ by one at most. The equitable coloring problem is to find the smallest k for which an equitable legal k-coloring exists. The problem has a number of applications. However, it is known to be NP-hard and thus computationally challenging. In this work, we present the first population-based memetic algorithm for solving the problem. The proposed algorithm combines a backbone-based crossover operator (to generate promising offspring solutions), a 2-phase tabu search procedure (to seek high-quality local optima) as well as a quality-and-distance based pool updating strategy (to maintain a healthy population). The computational results on 73 benchmark instances demonstrate that the proposed algorithm competes favorably with the state-of-the-art algorithms in the literature. Specifically, our algorithm attains the optimal results for all 41 instances with known optima and discovers improved upper bounds for 9 out of the 32 instances whose optimal solutions are still unknown. We investigate the benefits of the 2-phase tabu search procedure and the crossover operator with the memetic framework.

Keywords: Memetic search; heuristics; equitable coloring; graph coloring.

1 Introduction

Given a simple undirected graph $G = (V, E)$ with vertex set $V = \{1, 2, \ldots, n\}$ and edge set $E \subset V \times V$, a legal k-coloring of G is a partition of the vertex set V into k color classes or disjoint independent sets, where an independent set is a set of vertices of V such that no pair of vertices are linked by an edge. Equivalently, a legal k-coloring can be defined as a mapping $c : V \to \{1, \ldots, k\}$ such that $c(i) \neq c(j)$ for all edges (i, j) in E (this is called the *coloring constraint*). The graph k-coloring problem (k-GCP) is to determine if a legal k-coloring of G exists for a given k. The classical graph coloring problem (GCP) is to find the minimum integer k (chromatic number $\chi(G)$) for which a legal k-coloring of G exists. k-GCP is known to be NP-complete while the optimization problem GCP is NP-hard [15].

An equitable k-coloring is a legal k-coloring verifying the condition that the sizes of any two color classes differ by at most one (this is called the *equity constraint* of the coloring). The equitable k-coloring problem (k-ECP) involves finding an equitable legal k-coloring of the given graph G while the equitable coloring problem (ECP) is to determine the smallest integer k (equitable chromatic number $\chi_e(G)$) for which an equitable k-colorable exists. Obviously, for any graph G, $\chi_e(G) \geq \chi(G)$, that is, $\chi(G)$ is a lower bound of $\chi_e(G)$.

As a variant of the conventional graph coloring problem, the decision problem of the ECP is NP-complete [11,27]. Polynomial algorithms are known only for split graphs, trees, outerplanar graphs [4,5], cubic graphs [6] and some corona graphs [12].

The notion of an equitable coloring was first introduced in [33] and motivated by scheduling problems with load balancing requirements. In such applications, a graph can be defined where a vertex represents a task and an edge linking two vertices indicates that the two underlying tasks cannot be performed at the same time. Then the number of colors required to color the vertices of the graph corresponds to the time steps needed to perform the given tasks. Moreover, the load balancing requirement asks that equal or nearly-equal numbers of tasks are performed in each time step. A coloring satisfying this additional balancing requirement is then an equitable coloring. Other practical applications arise from garbage collection [33,39], memory allocation in parallel systems [9], scheduling computer and manufacturing processes with load balancing [3], and timetabling [10,26]. In general, the equity constraint aims to ensure a

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

balanced occupation of the given resources and improve their utilization. As such, given that existing methods for the classical graph coloring problem do not guarantee the equity constraint, they cannot be used to solve the equitable coloring problem. Instead, they can be usefully applied to provide lower bounds of the equitable chromatic number of a graph.

Much effort has been devoted to theoretical studies of the ECP. For instance, Hajnal and Szemerédi [18] showed that $\chi_e(G) \leq \Delta(G) + 1$ for every graph, where $\Delta(G)$ is the maximal vertex degree of G. Meyer [33] conjectured that $\chi_e(G) \leq \Delta(G)$ for any connected graph except the complete graphs and the odd circuits. This conjecture has been verified to be true for many cases as listed in [11].

From a perspective of solution methods for the ECP in the general case, several exact algorithms have been proposed. Specifically, Méndez-Díaz et al. developed a cutting plane algorithm by devising an integer programming formulation based on the polyhedral structure of the ECP [31]. Bahiense et al. presented a branch-and-cut algorithm for the ECP based on two new integer programming formulations [1]. Méndez-Díaz et al. adopted the Dsatur coloring algorithm to the ECP [32] and presented computational results for a subset of benchmark instances from the DIMACS and COLOR competitions.

Given the computational challenge of the ECP, heuristic algorithms are often used to find sub-optimal solutions for problem instances that can not be solved exactly. Furmanczyk and Kubale presented two constructive heuristics called Naive and SubGraph that generate greedily an equitable coloring of a graph [13]. Méndez-Díaz et al. adapted for the first time the well-known TabuCol algorithm for the classical GCP [14,20] to the ECP [29]. Lai et al. improved TabuEqCol by combining a backtracking scheme and tabu search under the iterated local search framework [25]. Sun et al. presented a feasible and infeasible local search algorithm called FISA that is based on an extended penalty-based fitness function [37]. Wang et al. introduced another mixed approach that explores both feasible and infeasible solutions and integrates a novel cyclic exchange neighborhood [40].

We observe that the most effective heuristic algorithms are all based on the local search framework and explores both feasible and infeasible solutions. Meanwhile, it is known that for other graph coloring problems, population-based memetic algorithms are among the best performing methods [14,22,28,34]. Until now, the memetic approach remains unexplored for the ECP. In this work, we fill the gap and investigate for the first time the potential of the memetic search framework for solving the ECP. The contributions of the work are highlighted as follows.

First, we present the first memetic algorithm for the ECP (MAECP) that explores the synergy between a dedicated crossover operator for equitable colorings and a 2-phase infeasible tabu search. The algorithm also integrates an elite population initialization procedure and a quality-and-distance pool updating procedure.

Second, we show extensive computational results on 73 benchmark graphs from the DIMACS and COLOR competitions, which confirm the competitiveness of the proposed algorithm compared to the state-of-the-art results in the literature. Specifically, MAECP consistently reaches the optimal solutions for the 41 instances with known optima and finds 9 improved best solutions (new upper bounds) for the 32 remaining instances whose optimal solutions are still unknown. These new bounds are valuable for the assessment of new ECP algorithms.

Finally, given that the ECP is a convenient model for a number of practical applications, the proposed algorithm can help to better solve these real world problems.

The rest of the paper is organized as follows. Section 2 gives some basic definitions. Section 3 presents the proposed algorithm. Section 4 shows computational results and comparisons with state-of-the-art algorithms. Section 5 analyzes the impacts of key components of the proposed algorithm. Conclusions and future work are discussed in the last section.

2 Notations

We introduce the following definitions which are useful for the description of the proposed approach. Let $G = (V, E)$ be a given graph.

Definition 1 *A candidate coloring of G is any partition of the vertex set V into k subsets* $V_1, V_2, \ldots, V_k$*, where each* V_i *is called a color class. We use* $s = \{V_1, V_2, \cdots, V_k\}$ *to represent a candidate solution.*

Definition 2 *A legal coloring is a conflict-free coloring composed of independent sets, i.e., any pair of vertices of any color class are not linked by an edge in E. Otherwise, it is an illegal or conflicting coloring. A conflicting edge in an illegal coloring is an edge whose endpoints belong to the same color class. The endpoints of a conflicting edge are called conflicting vertices.*

Definition 3 *An equitable solution or equity-feasible solution is any candidate coloring satisfying the equity constraint, i.e., the cardinalities of any two color classes differ by at most one. Otherwise, it is an equity-infeasible solution.*

Definition 4 *Let $W^+ = \lceil |V|/k \rceil$ and $W^- = \lfloor |V|/k \rfloor$ represent respectively the theoretical cardinality of the largest and smallest color classes in an equitable k-coloring. We define the "equity penalty function" of a coloring $s = \{V_1, V_2, \cdots, V_k\}$ as the sum of the gaps between $|V_i|$ and the theoretical cardinalities.*

$$g(s) = \sum_{i=1}^{k} \rho_i, 1 \leq i \leq k \tag{1}$$

where the equity-infeasibility value ρ_i $(i = 1, \cdots, k)$ for each color class V_i of solution s is the gap between $|V_i|$ and the theoretical cardinalities W^+ and W^-.

$$\rho_i = \begin{cases} |V_i| - W^+, if\ |V_i| \geq W^+ \\ W^- - |V_i|, if\ |V_i| \leq W^- \end{cases} \tag{2}$$

Definition 5 *We define the "conflict penalty function" of a coloring $s = \{V_1, V_2, \cdots, V_k\}$ as the total number of conflicting edges induced by solution s.*

$$f(s) = \sum_{i=1}^{k} |C(V_i)|, 1 \leq i \leq k \tag{3}$$

where $C(V_i)$ is the set of conflicting edges in color class V_i.

3 Memetic algorithm for the ECP

Like [25,29,37], we handle the ECP by solving a series of k-ECP for decreasing k values. For a given k, if an equitable legal k-coloring is found, we continue to solve the new k-ECP problem by setting $k = k - 1$. This process is repeated until no equitable legal k-coloring can be found. The last k for which an equitable legal k-coloring is found corresponds to an upper bound of the equitable chromatic number of G ($k \geqslant \chi_e(G)$).

The MAECP algorithm is designed to solve the k-ECP. To fix the initial k, we use the binary search method proposed in [25] where we set $k = |V|$ to determine an appropriate k value that admits an equitable legal k-coloring.

Given that k corresponds to the number of available resources, it is easy to understand that finding an equitable k-coloring with decreasing k becomes more and more difficult. This is particularly true when k approaches the equitable chromatic number $\chi_e(G)$ or when k is set to a value smaller than the current well-known upper bound (i.e., the smallest k for which an equitable k-coloring has even been found).

3.1 General approach

Memetic algorithms (MAs) are hybrid search methods that combine the population-based search framework and local search framework [30]. Indeed, it is generally believed that population-based search offers more facilities for exploration while local search provides more capabilities for exploitation. A hybrid method mixing these two approaches is expected to take advantage of complementary search strategies offered by the composing approaches. Since their introduction, MAs have been applied to solve many problems [7,36], including graph coloring [14,23,28,34] and other graph optimization problems (e.g., [2,35,42]).

As a general optimization framework, MAs need to be carefully adapted to the given problem to achieve a high performance [19]. The required adaptation generally concerns the crossover operator and the local optimization procedure. The memetic algorithm for the ECP (MAECP) presented in this work adopts a backbone-based crossover operator, a 2-phase infeasible local search procedure based on tabu search [16] as well as a diversity preservation pool updating procedure. It is worth mentioning that MAECP also shares ideas from scatter search (SS) [17] in the sense that 1) both maintain a pool (called reference set in SS) of high-quality solutions from which new solutions are generated by recombination; 2) local optimization is used to improve each new solution; and 3) the pool management considers both quality and diversity. As a result, MAECP can also be considered as a simplified SS algorithm where the subset generation method of SS (parent selection in MAs) applies the simple random selection rule.

MAECP starts with an initial population P of p elite solutions generated by the procedure presented in Section 3.2. Then MAECP repeats a number of generations until a stopping condition is met. At each generation, two solutions are randomly selected and then recombined by the backbone-based crossover described in Section 3.3 to generate an offspring solution. The offspring is then further improved by the 2-phase tabu search procedure presented in Section 3.4. The improved solution is finally used to update the population by considering both its quality and its distance with respect to the solutions of P (see Section 3.5). The algorithm terminates when a predefined stopping condition (e.g., maximum number of generations, fixed cutoff time limit) is reached. In this work, the stopping condition corresponds to a time limit.

Algorithm 1 The MAECP algorithm for solving the k-ECP

```
1: Input: Graph $G = (V, E)$, number of colors $k$, population size $p$
2: Output: A legal and equitable $k$-coloring if found
3: $P \leftarrow Population\_initialization(G, k, p)$ /* Initial population generation, Section 3.2 */
4: $S^* \leftarrow best(P)$ /* $S^*$ records the best solution found so far */
5: while Stopping condition is not met do
6:     Choose randomly 2 solutions $S_i, S_j$ from $P$
7:     $S_0 \leftarrow Crossover(S_i, S_j)$ /* Offspring generation by backbone based crossover, Section 3.3 */
8:     $S_0 \leftarrow$ 2-$Phase\_Search(S_0)$ /* Offspring improvement by 2-phase tabu search, Section 3.4 */
9:     if $S_0$ is better than $S^*$ then
10:        $S^* \leftarrow S_0$
11:    end if
12:    $P \leftarrow Pool\_Update(S_0, P)$
13: end while
14: return $S^*$
```

3.2 Population initialization

The initial population is generated according to the following steps: 1) create an equitable, but conflicting k-partition S with the greedy algorithm of [25]; 2) improve S by using the first phase of the 2-phase search of Section 3.4 to obtain a new solution S^+; 3) insert the improved solution S^+ in the population; 4) randomly perturb S^+ and assign the perturbed solution to S; 5) repeat steps 2)-4) p times (p is the population size). To create an initial population of high-quality with a good diversity, step 4) adopts the following perturbation procedure [25]. With probability of 0.3, the procedure perturbs S^+ by exchanging the colors of $0.3 * N$ randomly selected vertices. With probability of 0.7, the procedure performs $5 * 10^3$ *One_move* or Swap moves according to the neighborhood exploration rule of Section 3.4.1. Thanks to the stochastic perturbation step and the subsequent solution improvement step, we obtain an initial population composed of diversified and high-quality solutions.

3.3 Backbone-based crossover

As an important component of evolutionary algorithms, the crossover operator should be designed with care in order to favor transmissions of useful information from parents to offspring [19]. The proposed crossover for the ECP follows the general backbone principle for designing meaningful recombination operators. The basic idea is to preserve common elements shared by parent solutions (backbone) in the offspring solution [2,41]. The proposed crossover consists of the following two steps (see Algorithm 2).

Algorithm 2 The backbone-based crossover for the k-ECP

```
1: Input: Two parent solutions $S_m = \{V_1^m, V_2^m, \dots, V_k^m\}$ and $S_n = \{V_1^n, V_2^n, \dots, V_k^n\}$.
2: Output: An offspring solution $S_0 = \{V_1^0, V_2^0, \dots, V_k^0\}$
   /* Match the color classes of the parents to find the shared common vertices */
3: Create a complete bipartite graph $H = (V', E')$ with $V' = \{V_1^m, V_2^m, \dots, V_k^m\} \cup \{V_1^n, V_2^n, \dots, V_k^n\}\}$ and $E' = \{V_1^m, V_2^m, \dots, V_k^m\} \times \{V_1^m, V_2^m, \dots, V_k^m\})$
4: Find a maximum matching in $H$ and put the matched color classes $(V_i^m, V_j^n)$ in $J$
   /* Create $k$ color classes of the offspring from the matched color classes */
5: for each $(V_i^m, V_j^n) \in J$ do
6:     Use the common vertices of $V_i^m$ and $V_j^n$ to create a color class of offspring $S_0$
7: end for
   /* Handle unassigned vertices */
8: for each unassigned vertex missing in $S_0$ do
9:     Assign the vertex to a color classe with the smallest conflicts
10: end for
11: return $S_0$
```

Let $S_m = \{V_1^m, V_2^m, \dots, V_k^m\}$ and $S_n = \{V_1^n, V_2^n, \dots, V_k^n\}$ be two parent solutions. The first step aims to identify for each color class the largest set of vertices shared by both parents. Due to the symmetry of colorings, it is common that the color classes from S_m and S_n that share the most common vertices have different class numberings. For example, color class V_i of S_m might correspond to a different color class V_j of S_n. Therefore, the first step of the crossover is to properly identify a color class matching with the largest number of common vertices between a class of S_m and a class of S_n. This task can conveniently be achieved by finding a maximum weight matching in an edge weight complete bipartite graph $H = (V', E')$ where $V' = \{\{V_1^m, V_2^m, \dots, V_k^m\} \cup \{V_1^n, V_2^n, \dots, V_k^n\}\}$ and $(V_i^m, V_j^n) \in E'$ for all $i, j = 1, 2, \dots, k$ with an edge weight $\omega_{V_i^m, V_j^n}$, which is the number of identical vertices in V_i^m of S_m and V_j^n of S_n. The maximum weight matching problem can be solved by the classical Hungarian algorithm [24]. However, this will be too expensive in our case since we need to find a maximum weight matching for each crossover application. Therefore, we

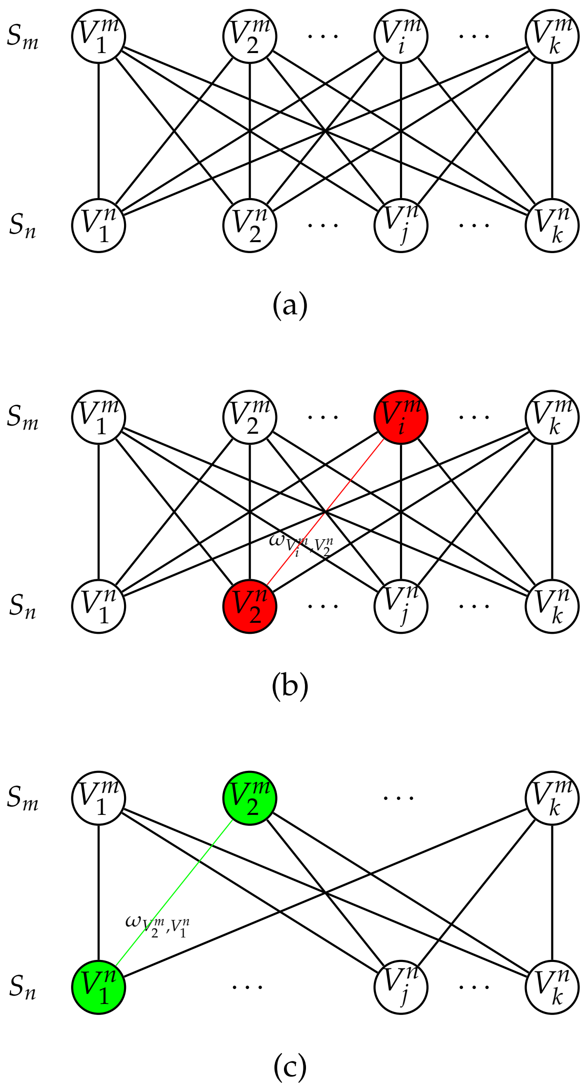


Fig. 1. Class matching via an edge weight complete bipartite graph H. (a) A complete bipartite graph H with edge weight ω. (b) Choosing an edge with the largest ω and deleting all edges incident to the endpoints of the edge. (c) Repeating step (b) until H becomes an empty.

apply a fast greedy algorithm to find a near-optimal weight matching, which iteratively chooses an edge $(V_i^m, V_j^n) \in H$ with the largest edge weight, and then deletes from H all edges linked to V_i^m and V_j^n. This procedure is repeated until H becomes empty, that is, when all color classes are matched. Figure 1 illustrates the color class matching process. From the edge weight complete bipartite graph (Figure 1 (a)) created from solutions S_m and S_n, we first identify the 'heaviest' edge having the highest weight (shown in red, Figure 1 (b)). This edge indicates that the ith color class of S_m and the 2nd color class of S_n share the largest number of vertices and thus are matched. A reduced graph (Figure 1 (c)) is obtained by deleting the two matched vertices together with all the edges adjacent to them. This matching process continues from the reduced graph to identify the next 'heaviest' edge (indicated in green, Figure 1 (c)), matching thus the 2nd color class of S_m and the 1st color class of S_n. This matching process is repeated until the graph becomes empty (the whole matching process repeats k times).

Second, from this matching, we create a new color class of the offspring S_0 with each of the k matched class. Then we greedily assign the unassigned vertices to a color class as follows. According to the decreasing order of the degrees, we assign each unassigned vertex to the color class such that the assignment leads to the smallest number of conflicts. At this point we obtain the offspring solution and submit it to the 2-phase tabu search described below for further improvement.

Figure 2 illustrates the crossover operator with a graph of 10 vertices and 3 colors (red, olive and azure). Parent solutions S_m and S_n are combined to generate the offspring S_0. The color classes of S_m and S_n are first matched: the olive class of S_m matches the red class of S_n, the red class of S_m matches the olive class of S_n, the two azure classes of S_m and S_n are matched. The shared vertices of each matched class are used to create a new color class of the partial offspring solution, leading to the partial coloring shown in the middle of Figure 2 with the three uncolored vertices (3,8,9) indicated in white. Finally, the uncolored vertices (3,8,9) are greedily assigned to obtain the complete offspring solution S_0 (bottom part of the figure).

3.4 The 2-phase tabu search

Local optimization is another critical component of a memetic algorithm and plays the key role of search intensification. In our case, we employ a 2-phase tabu search procedure. The first phase performs a large exploration of the search space by considering candidate solutions where both the coloring constraint and the equity constraint are relaxed, while the second phase makes a particular effort to satisfy the coloring constraint which is somewhat more difficult to resolve.

The relaxed search space: For a given k-ECP instance, the search space Ω_k explored by our algorithm is composed of all possible k-colorings which may or may not satisfy the coloring constraint and the equity constraint.

$$\Omega_k = \{\{V_1, V_2 : \cdots, V_k\}, \cup_{i=1}^{k} V_i = V, V_i \cap V_j = \emptyset\} \tag{4}$$

where $i \neq j, 1 \leq i, j \leq k$.

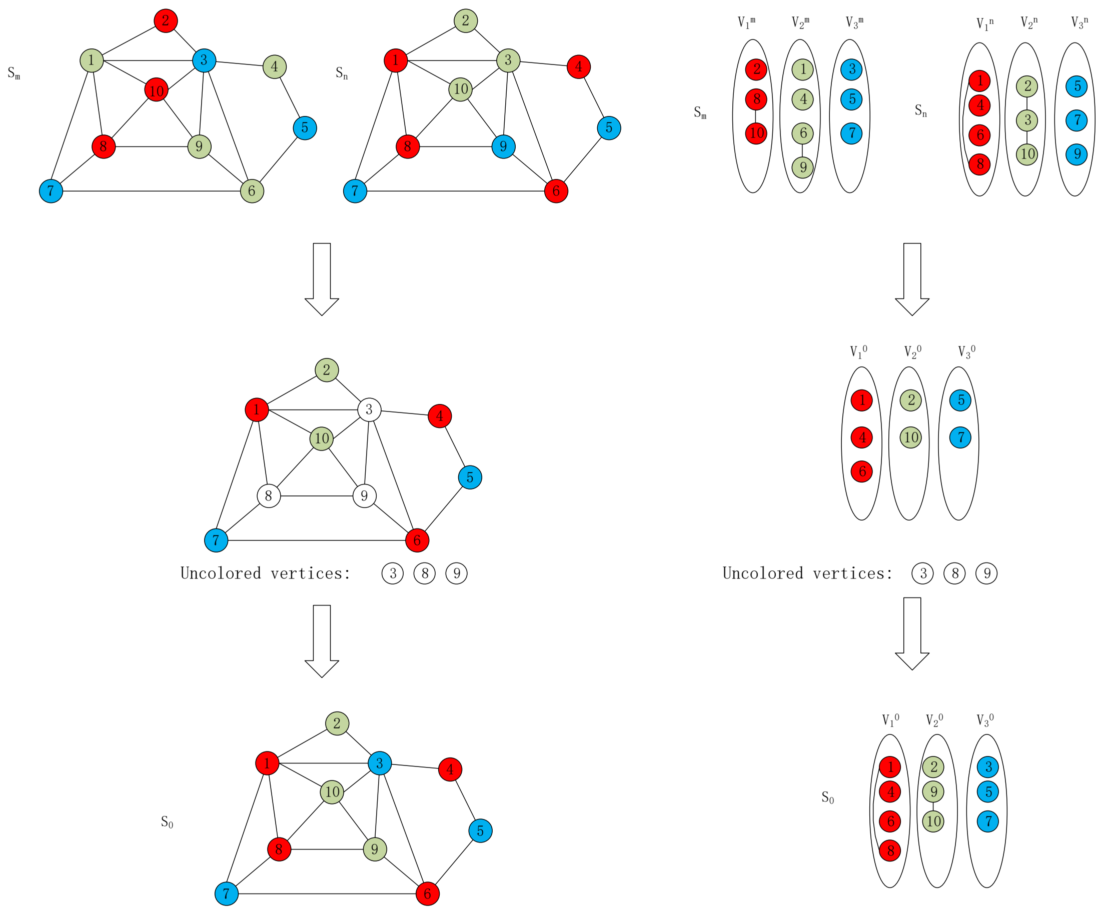


Fig. 2. The procedure of generating a new offspring solution from parent solutions S_m and S_n. For clarity reason, we only indicate the conflicting edges within the color classes: $\{8, 10\}$ and $\{6, 9\}$ in S_m; $\{1, 8\}$, $\{2, 3\}$ and $\{3, 10\}$ in S_n. Color classes of the parents S_m and S_n are first matched. The matched classes {1,4,6} {2,10} and {5,7} are then used to form the color classes of the offspring. The unassigned vertices (3,8,9) are finally allocated to color classes azure, red, and olive of the offspring.

3.4.1 *First phase*

Fitness function in the first phase: To explore the above search space, the first phase uses the following fitness function F (to be minimized).

$$F(s) = f(s) + g(s) \tag{5}$$

where $f(s)$ is the conflict penalty function and $g(s)$ is the equity penalty function defined by Equations (3) and (1) in Section 2.

Note that for a solution s, $F(s) = 0$ implies that s satisfies both the coloring and equity constraints and is thus a solution for the given k-ECP instance.

Move operators in the first phase: Following [37], we apply two move operators to generate neighbor solutions to explore the search space Ω_k. Let s be the incumbent solution and let OP be a move operator to transform s. We use $s \oplus OP$ to denote the neighbor solution obtained by applying OP to s.

(1) One_move operator: The $one\_move(v, V_i, V_j)$ operator displaces a *conflicting vertex* v from its color class V_i to another color class V_j. The neighborhood N_1 induced by this operator is given by

$$N_1(s) = \{s \oplus one\_move(v, V_i, V_j) \ : v \in \{V_i \cap C(s)\}, 1 \leq i, j \leq k, i \neq j\} \tag{6}$$

where $C(s)$ is the set of conflicting vertices of s.

Clearly N_1 is bounded by $O(|C(s)| \times k)$ in size. To effectively calculate the move gain that identifies the change in the fitness function F (Equation (5)), we adopt the fast incremental evaluation technique of [25]. We maintain a matrix A of size $n \times k$ with elements $A[v][i]$ recording the number of vertices adjacent to v in color class $V_i (1 \leq i \leq k)$. We maintain another $n \times k$ matrix B with elements $B[v][i]$ representing the equity-infeasibility value ρ_i (see Equation (2)) of vertex v assigned to color class V_i in the current solution. Then, the move gain of each one_move in terms of the variation of F can be conveniently computed by

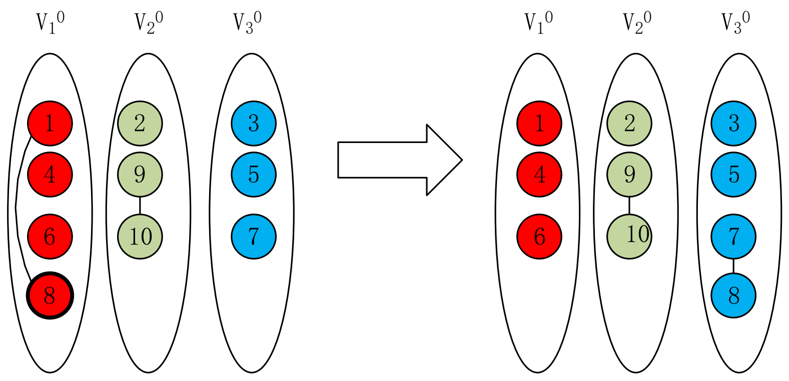
$$\Delta F = A[v][j] - A[v][i] + B[v][j] - B[v][i] \tag{7}$$

Each time an one move involving vertex v is performed, we just need to update a subset of values affected by this move as follows. For each vertex u adjacent to vertex v, $A[u][i] \leftarrow A[u][i] - 1$, and $A[u][j] \leftarrow A[u][j] + 1$. For any vertex w, $B[w][j] \leftarrow \sum_{i=1}^{k} \rho_i, 1 \leq j \leq k$. $B[w][j] = B[u][j]$, if w and u belong to the same color class.

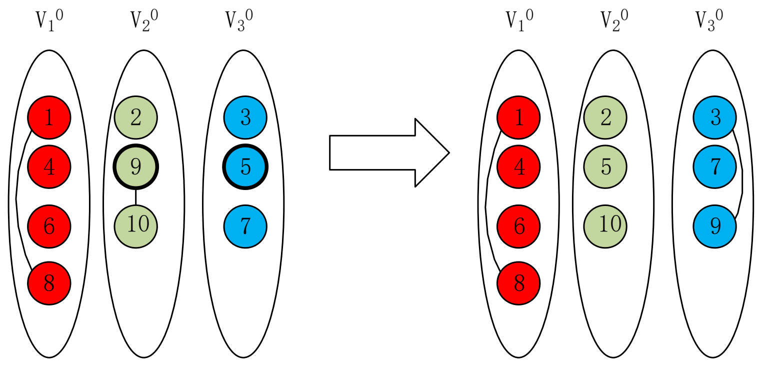
(2) Swap operator: The *swap(v,u)* operator exchanges a pair of vertices (u, v) from different color classes where at least one of them is a conflicting vertex. The resulting swap neighborhood N_2 is thus given as follows.

$$N_2(s) = \{s \oplus swap(v, u) : v \in V_i, u \in V_j, i \neq j, \{v, u\} \cap C(s) \neq \emptyset\} \tag{8}$$

It can be noted that the swap operation has no impact on the equity



(a) Case 1 One move operator



(b) Case 2 Swap operator

Fig. 3. Move operators in the first phase

constraint of the neighbor solution and can only change the number of conflicting edges. Then the move gain of a swap operation can be computed by,

$$\Delta F = A[u][i] - A[u][j] + A[v][j] - A[v][i] - 2e_{v,u} \qquad (9)$$

where $e_{v,u} = 1$ if v and u are adjacent vertices, otherwise $e_{v,u} = 0$.

Exploration of the neighborhoods in the first phase: The tabu search procedure selects, at each iteration, a best admissible neighbor solution in N_1 and N_2 with the smallest fitness gain ΔF and uses the solution to replace the current solution. A neighbor solution is admissible if it is not forbidden by the tabu list or is the best solution ever found. The underlying move $(one\_move(v, V_i, V_j)$, $swap(v, u))$ is recorded in the so-called tabu list in order to forbid the reverse move for a fixed number of next iterations. The procedure iteratively makes transitions between various candidate k-colorings while minimizing the function F. This process continues until one of the two following conditions is met. First, a legal and equitable k-coloring s with $F(s) = 0$ is found. In this case, the given k-ECP problem is solved for the current k and we continue to solve the next k-ECP problem with k decreased by one. Second, a fixed number of β_1 consecutive iterations (β_1 is a parameter called the search depth) have been performed without updating the best recorded solution S^*. In this case, we move to the second search phase described below.

3.4.2 Second phase

Fitness function in the second phase: The second phase aims to further explore the search space by focusing on minimizing the conflicting edges. For this, we use the conflict penalty function $f(s)$ (Equation (3) of Section 2) as our fitness function.

Move operators in the second phase: In order to enhance the search ability, the second phase jointly employs three neighborhoods: the one_move neighborhood (N_1) and the swap neighborhood (N_2) (both used in the first phase) as well as the constrained-three-cyclic-exchange neighborhood (N_3).

Constrained-three-cyclic-exchange neighborhood (N_3) [40]: This neighborhood is induced by the three-cyclic-exchange operator $cyclic(v, u, w)$ that displaces three vertices $u \in V_i$, $v \in V_j$ and $w \in V_h$ to another color classes V_j, V_h and V_i in a cyclic way. With no restriction, the three-cyclic-exchange operator leads to a very large neighborhood (whose size is bounded by O($|V|^3$)). In order to reduce the computational burden of this operator, we impose the following constraints: 1) vertex v is a conflicting vertex; 2) before and after moving the vertex v, the difference of the conflicting edges induced by the vertex v is no more than 2; for the movement of the vertex u and w, the sum of the changes of the conflicting edges induced by u and w is at most 2. The constrained-three-cyclic-exchange neighborhood N_3 is given as follows.

$$\begin{aligned} N_3(s) = \{s \oplus cyclic(v,u,w) : v \in V_i, u \in V_j, w \in V_h, i \neq j, i \neq h, j \neq h, v \cap C(s) \neq \emptyset, \\ A[v][j] - A[v][i] \leq 2, A[u][h] - A[u][j] + A[w][i] - A[w][h] \leq 2\} \end{aligned} \tag{10}$$

where A is a matrix of size $n \times k$ where each element $A[v][i]$ records the number of vertices adjacent to v in color class V_i $(1 \leq i \leq k)$.

The move gain of exchanging three vertices u, v and w (suppose $u \in V_i$, $v \in V_j$ and $w \in V_h$) can be calculated by

$$\Delta f = A[w][i] - A[w][h] + A[u][h] - A[u][j] + A[v][j] - A[v][i] - l\{v, u, w\} \tag{11}$$

where $l \in \{0, 1, 2, 3\}$ is the number of edges between vertices u, v and w.

Tabu search in the second phase: At each iteration, a best admissible solution with the smallest fitness gain Δf is taken among the neighbor solutions of N_1, N_2 and N_3 to replace the current solution. Then, the corresponding move ($one\_move(v, V_i, V_j)$, $swap(v, u)$ or $cyclic(v, u, w)$) is recorded in the tabu list. This process continues until a legal and equitable solution s is found (i.e., both $f(s) = 0$) and $g(s) = 0$ hold), or the best solution found so far cannot be improved during β_2 consecutive iterations (β_2 is the search depth of the second phase). To cope with the equity constraint during the second phase, a vertex from the largest color class is displaced to the smallest color class every α iterations (set to be 1000 in this work).

3.5 Pool updating strategy

To maintain a healthy diversity of the population, we adopt the quality-and-distance based pool updating method introduced in [28]. To decide whether a new offspring solution is added in the population, we consider its distance to the population and its quality relative to the solutions of the population. The interested reader is referred to [28] for a detailed description of this updating method. With this pool updating method, we ensure that not only is the population composed of high-quality solutions, but also the solutions of the population are well separated among them. This feature makes it possible to apply a simple random selection to choose the parent solutions for the crossover operator (as shown in Algorithm 1).

4 Experimental results and comparisons

We assess the performance of our proposed memetic approach on the set of 73 benchmark instances [1], [2] which are commonly used in the literature.

[1] http://www.dimacs.rutgers.edu/

[2] http://www.cs.hbg.psu.edu/txn131/graphcoloring.html/

For 41 instances, their equitable chromatic number is known, while the optimum is still unknown for the 32 remaining instances.

4.1 Setting for the computational studies

Our MAECP algorithm was coded in C++ and compiled by GNU g++ 4.1.2 with '-O3' flag. Our algorithm was run on a computer with an Intel Xeon E5-2670 processor (2.5 GHz and 2 GB RAM) running Ubuntu 12.04.

Parameters. The algorithm has three main parameters (Table 1): population size p and search depth β_1 of the first phase and search depth β_2 of the second phase. For p which is not a sensitive parameter, we follow the general practice of using memetic algorithms for solving combinatorial problems [14,23,28,34,42] and adopt a small value of $p = 20$. For the search depths which are more critical parameters, we fixed β_1 to 10^5 and β_2 to $2 * 10^5$ which generally lead to good results according to the experiment reported in Section 5.1. The parameter setting shown in Table 1 can be considered as the default setting while fine-tuning them (in particular β_1 and β_2) for a particular problem instance would lead to improved results (see the discussion of Section 4.2).

Table 1
Settings of important parameters

Parameters	Description	Value
p	Population size, Section 3.2	20
β_1	Search depth, Section 3.4.1	10^5
β_2	Search depth, Section 3.4.2	2×10^5

Stopping condition. Following [37,40], we present a first experiment where we ran our MAECP algorithm only once per instance with a cutoff time of 1 hour and a second experiment where we ran MAECP under a relaxed time condition. Specifically, the cutoff time was set to 2×10^4 seconds for the instances with up to 500 vertices and 4×10^4 seconds for larger instances. Given its stochastic nature, the MAECP algorithm was run 20 times with different random seeds to solve each instance. The use of a relaxed time condition aims to verify the ultimate search limit of MAECP beyond which it is hopeless to obtain still better results. Indeed, given that memetic algorithms like MAECP involves a population and additional components (e.g., crossover, pool update), memetic algorithms are known to be more computationally intensive compared to local optimization approaches.

It is worth noting that to solve a given instance, the above time budget is used to solve a series of k-ECP for decreasing k values (see Section 3 for the procedure used to determinate the initial k). When k decreases, the task of finding a solution becomes more and more difficult.

4.2 Computation results and comparison with state-of-the-art algorithms

In this section, we present the computational results of our MAECP algorithm on the set of 73 instances. For the purpose of comparison, we use the results of the two most recent and best performing algorithms (FISA [37] and HTS [40]) as our references. Table 2 reports the results of these three compared algorithms where the results of FISA and HTS are compiled from [37] and [40] respectively. The results of the reference algorithms have been obtained on an Intel Xeon E5-2670 processor (2.5 GHz and 2 GB RAM) for FISA and on an Intel Xeon E5440 CPU (2.83 GHz and 2 GB RAM), under both the short time condition (1 hour) and the long time condition (10^4 seconds for the instances with up to 500 vertices and 2×10^4 seconds for the instances with more than 500 vertices)[3] .

In Table 2, columns 1-2 give the name and the number of vertices of each instance. Columns 3 shows the chromatic number or its best lower bound ($\chi(G)$). Columns 4-5 present the current best lower bound (LB) and upper bound (UB) of the ECP reported in the literature [31,32]. The next 18 columns report detailed results of the reference algorithms (FISA and HTS) and our MAECP algorithm respectively. Specifically, in addition to the (best) result with the short time condition (k_1), we indicate for each algorithm under the respective relaxed long time condition, the best result k_{best}, the average result k_{avg}, the standard deviation k_{std}, the number of successful runs over 20 runs SR/20 to achieve k_{best} and the average computation time in seconds t(s) over the runs which attain k_{best}. Column Δ_1 indicates the difference between our best result (k_{best}) and the lower bound of the chromatic number of column $\chi(G)$, while column Δ_2 is the difference between our best result (k_{best}) and the lower bound of the equitable chromatic number of column LB. So the value of 0 for Δ_1 or Δ_2 indicates an optimal result (the equitable chromatic number). In the last two columns, we show the difference between our result (k_{best}) and the result of FISA (Δ_3) and HTS (Δ_4) (a negative value indicates an improved result). Finally, entries with "-" mean that the corresponding results are not available in the literature.

From Table 2, we can make the following comments. First, we observe that among the 73 tested benchmark instances, optimal solutions are achieved for 41 instances (indicated by $*$) since $\Delta_1 = 0$ or $\Delta_2 = 0$ holds. For these

[3] We verified that FISA and HTS cannot further improve their best results even if their long time budget is doubled.

instances, our MAECP algorithm is able to find the optimal solutions without exception (like FISA and HTS).

Table 2. Comparative results of MAECP with state-of-the-art algorithms on the 73 benchmark instances.

1

					FISA						HTS						MAECP									
Instance	$\|V\|$	$\chi(G)$	LB	UB	k_1	k_{best}	k_{avg}	k_{std}	SR	t(s)	k_1	k_{best}	k_{avg}	k_{std}	SR	t(s)	k_1	k_{best}	k_{avg}	k_{std}	SR	t(s)	Δ_1	Δ_2	Δ_3	Δ_4
DSJC125.1.col*	125	5	5	5	5	5	5	0	20/20	0.62	5	5	5	0	20/20	15.73	5	5	5	0	20/20	1.26	0	0	0	0
DSJC125.5.col*	125	17	9	18	17	17	17	0	20/20	428.28	17	17	17	0	20/20	563.44	17	17	17.05	0.22	19/20	2432.43	0	8	0	0
DSJC125.9.col*	125	44	43	45	44	44	44	0	20/20	0.09	44	44	44	0	20/20	0.3	44	44	44	0	20/20	2.80	0	1	0	0
DSJC250.1.col	250	6	5	8	8	8	8	0	20/20	3.62	8	8	8	0	20/20	426.5	8	8	8	0	20/20	4.81	2	3	0	0
DSJC250.5.col	250	26	12	32	29	29	29.35	0.49	13/20	5235.95	29	29	29	0	20/20	4584.46	29	29	29	0	20/20	1093.10	3	17	0	0
DSJC250.9.col*	250	72	63	83	72	72	72	0	20/20	892.24	72	72	72	0	20/20	1835.05	72	72	72.1	0.31	18/20	2539.71	0	9	0	0
DSJC500.1.col	500	7	5	13	13	13	13	0	20/20	3.57	13	13	13	0	20/20	147.54	13	13	13	0	20/20	112.67	6	8	0	0
DSJC500.5.col	500	17	13	62	53	52	53.25	0.55	1/20	8197.02	52	52	52	0	20/20	2098.2	53	51	51.95	0.22	1/20	20784.47	34	38	-1	-1
DSJC500.9.col	500	123	101	148	131	130	131	0.56	3/20	6269.63	130	129	129.65	0.48	7/20	8925.67	129	128	128.9	0.31	2/20	16170.75	5	27	-2	-1
DSJR500.1.col*	500	12	12	12	12	12	12	0	20/20	0.38	12	12	12	0	20/20	2.53	12	12	12	0	20/20	13.88	0	0	0	0
DSJR500.5.col	500	122	120	131	126	126	126.5	0.51	10/20	4459.50	125	125	125.65	0.48	7/20	8179.27	125	124	124.95	0.32	1/20	13266.53	2	4	-2	-1
DSJC1000.1.col	1000	7	5	22	21	21	21	0	20/20	1866.63	22	21	21.95	0.22	1/20	4809.68	21	21	21.15	0.37	17/20	1364.87	14	16	0	0
DSJC1000.5.col	1000	18	15	112	98	95	96.1	0.55	2/20	15698.60	97	95	95.9	0.7	6/20	13394.64	96	95	97.05	1.79	3/20	36321.49	77	80	0	0
DSJC1000.9.col	1000	215	126	268	253	252	252.2	0.41	16/20	2240.02	251	251	251	0	20/20	3563.76	253	251	251	0	20/20	963.55	36	125	-1 2	0
R125.1.col*	125	5	-	-	5	5	5	0	20/20	0	5	5	5	0	20/20	0.06	5	5	5	0	20/20	0.37	0	-	0	0
R125.5.col*	125	36	-	-	36	36	36	0	20/20	0.67	36	36	36	0	20/20	3.41	36	36	36	0	20/20	1.76	0	-	0	0
R250.1.col*	250	8	-	-	8	8	8	0	20/20	0	8	8	8	0	20/20	0.11	8	8	8	0	20/20	0.24	0	-	0	0
R250.5.col*	250	65	-	-	67	66	66.9	0.31	2/20	3041.22	66	65	65.9	0.32	2/20	9777.74	65	65	65.09	0.04	3/20	11291.38	0	-	-1	0
R1000.1.col*	1000	20	-	-	20	20	20	0	20/20	2.24	20	20	20	0	20/20	678.04	20	20	20	0	20/20	21.13	0	-	0	0
R1000.5.col	1000	234	-	-	251	250	250.45	0.51	11/20	11564.20	255	249	249.1	0.3	19/20	17816.84	251	247	247.65	0.59	8/20	41552.02	13	-	-3	-2
le450_5a.col*	450	5	5	5	5	5	5	0	20/20	30.20	5	5	5	0	20/20	332.37	5	5	5	0	20/20	38.40	0	0	0	0
le450_5b.col*	450	5	5	5	5	5	5	0	20/20	44.29	5	5	5	0	20/20	363.68	5	5	5	0	20/20	64.71	0	0	0	0
le450_5c.col*	450	5	-	-	5	5	5	0	20/20	16.39	5	5	5	0	20/20	143.03	5	5	5	0	20/20	17.52	0	-	0	0
le450_5d.col*	450	5	5	8	5	5	5	0	20/20	14.07	5	5	5	0	20/20	373.07	5	5	5	0	20/20	16.08	0	0	0	0
le450_15a.col*	450	15	15	15	15	15	15	0	20/20	2.99	15	15	15	0	20/20	17.18	15	15	15	0	20/20	7.45	0	0	0	0
le450_15b.col*	450	15	15	15	15	15	15	0	20/20	2.41	15	15	15	0	20/20	15.95	15	15	15	0	20/20	8.03	0	0	0	0
le450_15c.col*	450	15	-	-	15	15	15.2	0.41	16/20	553.79	15	15	15	0	20/20	95.72	15	15	15	0	20/20	1351.67	0	-	0	0
le450_15d.col*	450	15	15	16	15	15	15.85	0.37	3/20	638.13	15	15	15	0	20/20	58.14	15	15	15.35	0.49	13/20	4992.27	0	0	0	0
le450_25a.col*	450	25	25	25	25	25	25	0	20/20	0.41	25	25	25	0	20/20	4.76	25	25	25	0	20/20	20.22	0	0	0	0

continued on next page

continued from previous page

					FISA						HTS						MAECP									
Instance	$\|V\|$	$\chi(G)$	LB	UB	k_1	k_{best}	k_{avg}	k_{std}	SR	t(s)	k_1	k_{best}	k_{avg}	k_{std}	SR	t(s)	k_1	k_{best}	k_{avg}	k_{std}	SR	t(s)	Δ_1	Δ_2	Δ_3	Δ_4
le450_25b.col*	450	25	25	25	25	25	25	0	20/20	0.46	25	25	25	0	20/20	7.26	25	25	25	0	20/20	28.10	0	0	0	0
le450_25c.col	450	25	-	-	26	26	26	0	20/20	86.90	26	26	26	0	20/20	11.75	26	26	26	0	20/20	147.71	1	-	0	0
le450_25d.col	450	25	25	27	26	26	26	0	20/20	95.85	26	26	26	0	20/20	6.29	26	26	26	0	20/20	60.96	1	1	0	0
wap01a.col	2368	41	41	46	42	42	42.95	0.22	1/20	4544.77	42	42	42	0	20/20	700.41	42	42	42	0	20/20	10304.68	1	1	0	0
wap02a.col	2464	40	40	44	42	41	41.9	0.31	2/20	2538.33	41	41	41	0	20/20	2232.87	41	41	41	0	20/20	14295.51	1	1	0	0
wap03a.col	4730	40	40	50	46	45	45.7	0.47	6/20	20201.80	45	45	45.1	0.3	18/20	6150.34	44	44	45.7	0.86	2/20	34445.79	4	4	-1	-1
wap04a.col	5231	40	-	-	46	44	44.45	0.51	11/20	15614.20	45	44	44.4	0.49	12/20	12845.28	43	43	44.25	0.64	2/20	33286.35	3	-	-1	-1
wap05a.col*	905	50	-	-	50	50	50	0	20/20	99.26	50	50	50	0	20/20	0.18	50	50	50	0	20/20	10983.28	0	-	0	0
wap06a.col	947	40	-	-	42	41	41.9	0.26	1/20	9340.42	41	41	41	0	20/20	1206.07	41	41	41.05	0.22	19/20	13739.89	1	-	0	0
wap07a.col	1809	40	-	-	43	43	43.05	0.22	19/20	4077.71	42	42	42.4	0.49	12/20	2591.23	42	42	42.75	0.44	5/20	11304.96	2	-	-1	0
wap08a.col	1870	40	-	-	43	43	43.1	0.10	10/20	4872.74	42	42	42.7	0.46	6/20	3801.03	42	42	42.1	0.31	18/20	13821.39	2	-	-1	0
flat300_28_0.col	300	28	11	36	33	32	33.6	0.60	1/20	4910.48	33	33	33	0.71	20/20	3814.24	33	32	32.65	0.49	7/20	5209.56	4	21	0	-1
flat1000_50_0.col	1000	50	-	-	96	94	94.7	0.47	6/20	17321.40	97	92	93.3	0.64	1/20	18034.58	99	93	93.9	0.31	2/20	16779.12	43	-	-1	1
flat1000_60_0.col	1000	60	-	-	97	94	94.8	0.52	5/20	10488.80	100	94	94.7	0.64	8/20	16263.51	96	93	93.85	0.37	3/20	14715.85	33	-	-1	-1
flat1000_76_0.col	1000	76	14	112	98	94	95.15	0.59	2/20	15246.40	97	93	93.9	0.62	5/20	14306.6	93	93	94.1	0.55	2/20	24103.23	17	79	-1	0
latin_square_10.col	900	90	90	130	105	104	104.55	0.60	10/20	12666.20	112	107	108.5	1.07	3/20	18054.74	105	103	104.61	0.59	1/20	32403.96	13	13	-1	-4
C2000.5.col	2000	-	-	-	198	183	183.4	0.60	13/20	19702.30	201	188	190.35	1.24	1/20	19915.04	201	183	183	0	20/20	4555.56	-	-	0	-5
C2000.9.col	2000	-	-	-	503	493	495.21	0.89	2/20	21163.90	501	501	501	0	20/20	3952.28	504	468	477.4	2.23	1/20	36966.26	-	-	-25	-33
mulsol.i.1.col*	197	49	49	49	49	49	49	0	20/20	44.34	49	49	49	0	20/20	0.53	49	49	49	0	20/20	7.99	0	0	0	0
mulsol.i.2.col	188	31	34	39	36	36	36.95	0.39	2/20	1914.22	36	36	36	0	20/20	15.22	36	36	36	0	20/20	187.86	5	2	0	0
fpsol2.i.1.col*	496	65	65	65	65	65	65	0	20/20	1723.52	65	65	65	0	20/20	12.61	65	65	65	0	20/20	777.00	0	0	0	0
fpsol2.i.2.col*	451	30	47	47	47	47	47.2	0.52	17/20	2357.15	47	47	47	0	20/20	6.64	47	47	47	0	20/20	4983.91	-	0	0	0
fpsol2.i.3.col*	425	30	55	55	55	55	55	0	20/20	1310.01	55	55	55	0	20/20	3.77	55	55	55	0	20/20	746.01	25	0	0	0
inithx.i.1.col*	864	54	54	54	54	54	56.9	2.22	7/20	3356.31	54	54	54	0	20/20	47.31	54	54	54	0	20/20	1708.39	0	0	0	0
inithx.i.2.col	645	31	30	93	36	36	38.8	1.70	5/20	3275.50	35	35	35	0	20/20	207.41	35	35	35	0	20/20	4106.24	4	5	-1	0
inithx.i.3.col	621	31	-	-	38	37	39.85	1.53	4/20	2891.78	36	36	36	0	20/20	5256.11	36	36	36	0	20/20	6529.65	5	-	-1	0
zeroin.i.1.col*	211	49	49	49	49	49	49.6	0.50	8/20	1088.94	49	49	49	0	20/20	2.24	49	49	49.65	0.49	7/20	5749.18	0	0	0	0
zeroin.i.2.col*	211	30	36	36	36	36	36	0	20/20	123.88	36	36	36	0	20/20	0.7	36	36	36	0	20/20	66.78	6	0	0	0
zeroin.i.3.col*	206	30	36	36	36	36	36	0	20/20	129.45	36	36	36	0	20/20	1.14	36	36	36	0	20/20	165.69	6	0	0	0
myciel6.col*	95	7	7	7	7	7	7	0	20/20	0	7	7	7	0	20/20	0.01	7	7	7	0	20/20	0.56	0	0	0	0
myciel7.col*	191	8	8	8	8	8	8	0	20/20	0.02	8	8	8	0	20/20	0.17	8	8	8	0	20/20	1.81	0	0	0	0
4_FullIns_3.col*	114	7	7	7	7	7	7	0	20/20	0	7	7	7	0	20/20	0.01	7	7	7	0	20/20	0	0	0	0	0

continued on next page

continued from previous page

Instance	$\lvert V\rvert$	$\chi(G)$	LB	UB	FISA						HTS						MAECP									
					k_1	k_{best}	k_{avg}	k_{std}	SR	t(s)	k_1	k_{best}	k_{avg}	k_{std}	SR	t(s)	k_1	k_{best}	k_{avg}	k_{std}	SR	t(s)	Δ_1	Δ_2	Δ_3	Δ_4
4_FullIns_4.col*	690	8	6	8	8	8	8	0	20/20	0.12	8	8	8	0	20/20	3.98	8	8	8	0	20/20	2.36	0	2	0	0
4_FullIns_5.col*	4146	9	6	9	9	9	9	0	20/20	0.12	9	9	9.3	0.46	14/20	57.67	9	9	9	0	20/20	72.50	0	3	0	0
1_Insertions_6.col	607	4	3	7	7	7	7	0	20/20	0.17	7	7	7	0	20/20	0.38	7	7	7	0	20/20	4.61	3	4	0	0
2_Insertions_5.col	597	4	3	6	6	6	6	0	20/20	0.06	6	6	6	0	20/20	0.96	6	6	6	0	20/20	1.22	2	3	0	0
3_Insertions_5.col	1406	4	3	6	6	6	6	0	20/20	0.35	6	6	6	0	20/20	21.94	6	6	6	0	20/20	2.77	2	3	0	0
school1.col*	385	14	15	15	15	15	15	0	20/20	0.93	15	15	15	0	20/20	0.72	15	15	15	0	20/20	9.12	1	0	0	0
school1_nsh.col*	352	14	14	14	14	14	14	0	20/20	1.77	14	14	14	0	20/20	56.1	14	14	14	0	20/20	17.37	0	0	0	0
qg.order40.col*	1600	40	40	40	40	40	40	0	20/20	3.44	40	40	40	0	20/20	4291.18	40	40	40	0	20/20	162.05	0	0	0	0
qg.order60.col*	3600	60	60	60	60	60	60	0	20/20	14.53	60	60	60	0	20/20	64.34	60	60	60	0	20/20	1014.83	0	0	0	0
ash331GPIA.col*	662	4	4	4	4	4	4	0	20/20	0.78	4	4	4	0	20/20	5.39	4	4	4	0	20/20	43.85	0	0	0	0
ash608GPIA.col*	1216	4	3	4	4	4	4	0	20/20	0.25	4	4	4	0	20/20	854.45	4	4	4	0	20/20	0.44	0	1	0	0
ash958GPIA.col*	1916	4	3	4	4	4	4	0	20/20	10.89	4	4	4	0	20/20	11.95	4	4	4	0	20/20	165.93	0	1	0	0

Second, concerning the 32 instances for which optimal solutions are still unknown, MAECP performs very well under the short time condition. Specifically, MAECP finds 14 and 10 better results in terms of k_1 compared to FISA and HTS, while MAECP reports 3 and 4 worse results respectively. Under the long time condition, MAECP reaches a remarkable performance compared to the reference algorithms. MAECP dominates FISA by obtaining 17 better results (see negative entries in column Δ_3) and equal results for the remaining 16 instances. MAECP also performs better than HTS by reporting better results for 11 instances (see negative entries in column Δ_4), worse result for 1 instance, and equal results for other 21 instances. Especially, for the large graph C2000.9, MAECP significantly improves the best-known result by reducing the number of used colors by 25 units. Only for 1 instance ($flat1000\_50\_0.col$), MAECP reports a slightly worse result (using one more color) relative to the current best-known result (reported by HTS). Finally, when comparing with the upper bounds obtained by the exact algorithms (Column 5), we see that the bounds of MAECP (Column 19) are clearly much better.

Third, if we check the cases where MAECP achieved the same k_{best} as the reference algorithms (56 and 61 instances compared to FISA and HTS respectively), we can make the following comments. Compared to FISA, MAECP performs better in terms of k_{std} (11/56 cases vs 5/56 cases) and k_{avg} (11/56 vs 5/56), while the result of MAECP is worse in terms of $t(s)$ (10/56 cases vs 46/56 cases) and SR (4/61 vs 12/61). Compared to HTS, MAECP performs better in terms of k_{std} (14/61 cases vs 7/61 cases) and SR (8/61 vs 4/61). It performs worse in terms of $t(s)$ (10/61 cases vs 51/61 cases) and k_{avg} (4/61 vs 8/61).

Finally, to verify the statistical significance between the results of MAECP and FISA/HTS, we apply the non-parametric Wilcoxon signed-rank test to the k_{best} values. With a 95% level of confidence, the p-values of 1.46e-4 ($<< 0.05$) and 5.82e-3 ($<< 0.05$) for MAECP vs FISA and MAECP vs HTS confirm the dominance of MACEP over FISA and HTS in terms of k_{best}. The Wilcoxon test for the k_1 values leads to p-values of 4.61e-2 (< 0.05) and 1.53e-1 (> 0.05) for MAECP vs FISA and MAECP vs HTS respectively, indicating the dominance of MACEP is only marginally under the one hour time limit. Indeed, given that MACEP is a population algorithm, this cutoff limit is too short for MACEP to perform a sufficient search, when multiple k values need to be tested.

In summary, MAECP improves the best-known solutions for 9 instances (new upper bounds) among the 32 instances for which optimal solutions are still unknown (28%) and finds all optimal results for the 41 instances with known optima.

Finally, the above results have been achieved by running MACEP from $k = |V|$ for a given graph $G = (V, E)$ (see Section 3) and using consistently the default parameter setting of Table 1. By fine-tuning some parameters or just starting with an initial k value close to the best known value, better results can be found. For instance, by setting k to a value around the best known k value MACEP finds an equitable coloring with $k = 245$ (instead of 247 in Table 2) for R1000.5 in 27173 seconds. Similarly, MACEP also solves flat_1000_50_0.col with $k = 92$ (instead of 93 in Table 2) in 24311 seconds.

5 Analysis

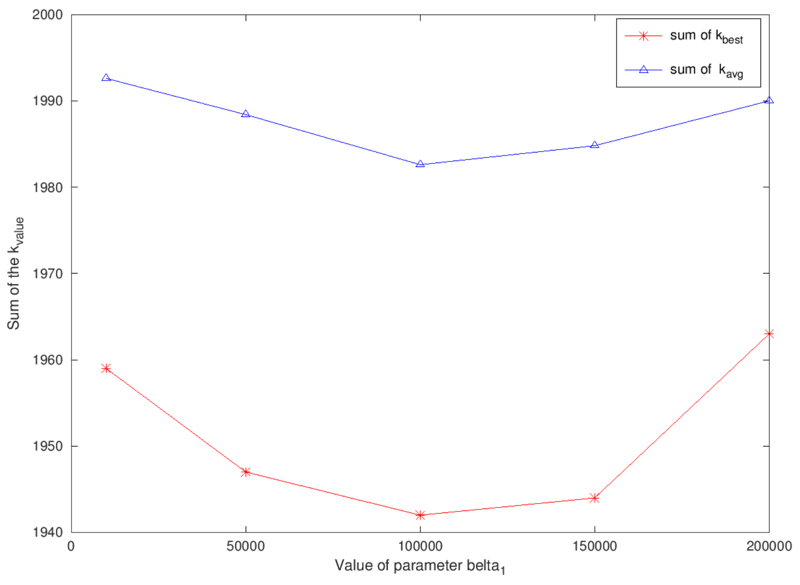
In this section, we carry out additional experiments to investigate the benefits of two important ingredients of the proposed MAECP algorithm: the backbone-based crossover and the 2-phase infeasible tabu search. These experiments were performed on a selection of 23 instances (shown in Tables 3 and 4) with unknown optimal solutions.

5.1 Parameter analysis

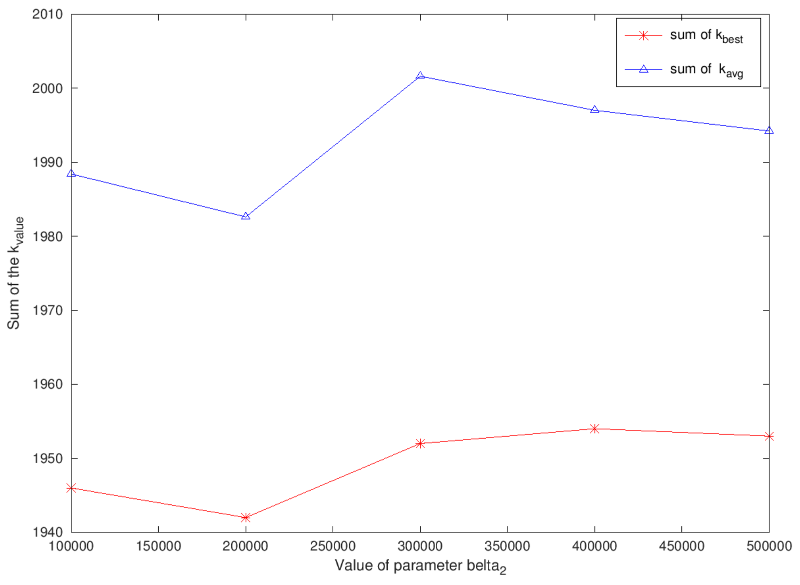
In this section, we show an analysis of the two main parameters β_1 and β_2. For this analysis, we vary one parameter value among a given range while fixing the other parameters to their default setting as shown in Table 1.

We use the following value ranges: $\beta_1 = \{10^4, 5 \times 10^4, 10^5, 5 \times 10^5, 10^6\}$ and $\beta_2 = \{10^5, 2 \times 10^5, 3 \times 10^5, 4 \times 10^5, 5 \times 10^5\}$. Figure 4 shows the behavior of MAECP with respect to each of the parameters under the short time condition (3600 seconds per run), where the X-axis indicates the values of each parameter and the Y-axis shows the sum of the k_{best} and k_{avg} over the 23 instances and 20 runs per instance.

Figure 4 shows that the performance of MAECP is significantly influenced by the value of the parameter β_1 and β_2. For β_1, the best performance is obtained when $\beta_1 = 10^5$, while for β_2, the value of 2×10^5 is the best choice. The unfitting values of β_1 (β_2) affect negatively the performance of MAECP since a too small value of β_1 (β_2) does not allow the local optimization component to make a sufficient examination of the current search zone, while a too large value of β_1 (β_2) implies a too long (and probably repetitive) local optimization. This study justifies the default parameter setting of Table 1.



(a) The influences of β_1



(b) The influences of β_2

Fig. 4. Analysis of the influences of the parameters (β_1 and β_2).

5.2 *Effectiveness of the 2-phase search*

As described in Section 3.4, the 2-phase tabu search is the local optimization procedure of our memetic algorithm. To assess its usefulness, we created two MAECP variants (called MA_HTS and MA_FISA) where we replace the 2-phase tabu search by the state-of-the-art algorithms HTS [40] and FISA [37] respectively. We ran these variants under the same long condition as specified in Section 4.1. That is, we ran each compared

Table 3
Analysis of the influence of the 2-phase tabu search on the performance of the MAECP algorithm.

	MAECP				*MA_HTS*				*MA_FISA*					
Instance	k_{best}	k_{avg}	SR	t(s)	k_{best}	k_{avg}	SR	t(s)	k_{best}	k_{avg}	SR	t(s)	Δ_1	Δ_2
DSJC250.5.col	29	29	20/20	1093.10	29	29	20/20	765.51	29	29.8	4/20	13323.03	0	0
DSJC500.5.col	51	51.95	1/20	20784.47	52	52	20/20	4645.58	53	53.05	19/20	5585.59	-1	-2
DSJC500.9.col	128	128.9	2/20	16170.75	128	128.8	4/20	12596.44	130	130.7	6/20	10919.48	0	-2
DSJR500.5.col	124	124.95	1/20	13266.53	125	125	20/20	7218.91	126	126	20/20	1948.05	-1	-2
DSJC1000.5.col	95	97.05	3/20	36321.49	95	96.45	6/20	29919.83	99	101.55	1/20	33887.1	0	-4
DSJC1000.9.col	251	251	20/20	963.55	251	251	20/20	10948.39	253	254	2/20	25818.87	0	-2
R250.5.col	65	65.09	3/20	11291.38	66	66	20/20	403.05	66	66.33	14/20	7652.724	-1	-1
R1000.5.col	247	247.65	8/20	41552.02	247	247.25	15/20	31595.44	249	249.95	19/20	7021.67	0	-2
wap01a.col	42	42	20/20	10304.68	43	43	20/20	1734.74	42	42.5	10/20	12329.4	-1	0
wap02a.col	41	41	20/20	14295.51	41	41.9	2/20	21858.33	41	41.8	4/20	15387.03	0	0
wap03a.col	44	45.7	2/20	34445.79	45	45.78	9/20	17666.15	45	45.05	19/20	16680.71	-1	-1
wap04a.col	43	44.25	2/20	33286.35	44	44.4	12/20	24259.4	44	44.05	19/20	18020.14	-1	-1
wap05a.col	50	50	20/20	10983.28	50	50	20/20	251.02	50	50	20/20	326.38	0	0
wap06a.col	41	41.05	19/20	13739.89	41	41.6	8/20	12376.91	41	41.3	14/20	10927.99	0	0
wap07a.col	42	42.75	5/20	11304.96	43	43.3	17/20	3123.80	43	43	20/20	59.33	-1	-1
wap08a.col	42	42.1	18/20	13821.39	43	43.1	18/20	4457.54	43	43	20/20	3202.44	-1	-1
flat300_28_0.col	32	32.65	7/20	5209.56	32	32.7	6/20	8243.73	33	33.8	4/20	9477.43	0	-1
flat1000_50_0.col	93	93.9	2/20	16779.12	93	93.45	11/20	20824.03	96	98.3	7/20	34202.33	0	-3
flat1000_60_0.col	93	93.85	3/20	14715.85	93	93.8	5/20	25844.12	96	97.7	9/20	33149.85	0	-3
flat1000_76_0.col	93	94.1	2/20	24103.23	93	94.2	1/20	25280.69	96	98.95	4/20	37222.5	0	-3
latin_square_10.col	103	104.61	1/20	32403.96	104	104.95	4/20	12202.16	104	104.25	15/20	11827.37	-1	-1
inithx.i.2.col	35	35	20/20	4106.24	37	39.7	10/20	8288.66	36	36.9	3/20	18516.37	-2	-1
inithx.i.3.col	36	36	20/20	6529.65	38	40.6	6/20	6896.93	37	38	2/20	26825.67	-2	-1
p-value					2.0e-3				1.6e-4					

algorithm 20 times to solve each instance under 2×10^4 seconds for the instances with up to 500 vertices and 4×10^4 seconds for larger instances.

We summarize in Table 3 the comparative results of MAECP against these two variants with the same information as in Table 2. Specifically, we report for each compared algorithm (MAECP, *MA_HTS*, *MA_FISA*), the best result k_{best}, the average result k_{avg}, the standard deviation k_{std}, the number of successful runs over 20 runs SR/20 to achieve k_{best} and the average computation time in seconds t(s) to attain k_{best}. The results show that MAECP substantially performs better than *MA_HTS* and *MA_FISA* in terms of best value (k_{best})and average value (k_{avg}). Specifically, for k_{best}, MAECP dominates *MA_HTS* and *MA_FISA* by obtaining 11 and 18 better results out of the 23 tested instances and reporting no worse result. Moreover, MAECP has 12 better average results (k_{avg}) compared to *MA_FISA* and *MA_HTS*. To verify the statistical significance of the differences between MAECP and the compared variant in terms of k_{best}, the last row indicates the p-values from the Wilcoxon signed-rank test with a 95% level of confidence. The p-values (2.0e-3 and 1.6e-4 which are both smaller than 0.05) confirm that the dominance of MAECP over the two compared variants is statistically significant. Since the compared methods lead to solutions of different quality, it is not meaningful to compare their respective success rates and computation times. On the other hand, for the 12 cases where MAECP and MA_HTS achieved the same k_{best} values, there is no clear dominance of one method over the other in terms of SR and t(s). Specifically, MAECP has 3 equal and 4 better SR and 5 better t(s). For the 5 cases where MAECP and MA_FISA reported the same k_{best}, MAECP has a better SR in 4 cases and 3 better t(s). Finally, one observes that both MAECP and MA_HTS dominate MA_FISA according to all indicators. Finally, according to the results reported for HTS [40], for FISA [37] and for the 2-phase tabu search (see Section 5.3), HTS and the 2-phase tabu search are more powerful than FISA. This experiment thus confirms the benefit of embedding a powerful local optimization procedure (in our case, the 2-phase tabu search) within a memetic algorithm to ensure an effective search intensification.

5.3 *Effectiveness of the crossover*

The comparative study of Section 4.2 shows the advantage of our population-based memetic algorithm compared to the two state-of-the-art local search algorithms FISA and HTS. In this section, we further assess the usefulness of the memetic framework via the crossover operator. For this purpose, we compare MAECP and its 2-phase tabu search alone by running them under the long time condition as specified in Section 4.1. To avoid penalizing the 2-phase tabu search, we applied the following re-start technique. When the 2-phase tabu search attains its end without finding a solution, it is re-started if the cutoff time is not reached. By this re-start technique, we make sure that the 2-phase tabu search consumes the given time budget like MAECP.

The comparative results of this experiment are presented in Table 4 with the same information as before. The last column also indicates the difference between the best results (k_{best}) of MAECP and 2-*phase_tabu_search*. Table 4 shows that MAECP dominates the 2-*phase_tabu_search* on the 23 instances tested, by obtaining 11 better results (see negative entries in column Δ_1), 1 worse result and the same results for the remaining instances. In terms of k_{best}, the small p-value (5.6e-3 $<$ 0.05) from the Wilcoxon signed-rank test with a 95% level of confidence confirms that MAECP with its crossover operator performs significantly better than the 2-phase tabu search alone. Incidentally, for the 11 cases both algorithms report the same k_{best}, MAECP is more robust with a better success rate SR in all but one case. In terms of computation time, MAECP performs similarly compared to the 2-phase tabu search with a shorter time for 6 cases against 5 cases in favor of the 2-phase tabu search. This experiment demonstrates that the crossover operator with the

Table 4
Analysis of the influence of the crossover on the performance of the MAECP algorithm.

	MAECP				2-*phase_tabu_search*				
Instance	k_{best}	k_{avg}	SR	t(s)	k_{best}	k_{avg}	SR	t(s)	Δ_1
DSJC250.5.col	29	29	20/20	1093.10	29	29.22	14/20	6673.11	0
DSJC500.5.col	51	51.95	1/20	20784.47	52	52.44	10/20	8658.09	-1
DSJC500.9.col	128	128.9	2/20	16170.75	129	129.67	6/20	15590.03	-1
DSJR500.5.col	124	124.95	1/20	13266.53	125	125.19	16/20	5709.86	-1
DSJC1000.5.col	95	97.05	3/20	36321.49	95	97.72	1/20	40134.7	0
DSJC1000.9.col	251	251	20/20	963.55	251	251.22	14/20	22198.38	0
R250.5.col	65	65.09	3/20	11291.38	65	65.94	1/20	11585.85	0
R1000.5.col	247	247.65	8/20	41552.02	248	248.78	8/20	32581.50	-1
wap01a.col	42	42	20/20	10304.68	42	42.72	5/20	4782.61	0
wap02a.col	41	41	20/20	14295.51	42	42	20/20	2408.73	-1
wap03a.col	44	45.7	2/20	34445.79	45	45.55	7/20	17761.70	-1
wap04a.col	43	44.25	2/20	33286.35	44	44.13	13/20	13997.34	-1
wap05a.col	50	50	20/20	10983.28	50	50	20/20	304.16	0
wap06a.col	41	41.05	19/20	13739.89	41	41.89	2/20	1955.88	0
wap07a.col	42	42.75	5/20	11304.96	43	43	20/20	2955.55	-1
wap08a.col	42	42.1	18/20	13821.39	42	42.89	3/20	19666.34	0
flat300_28_0.col	32	32.65	7/20	5209.56	32	33.17	1/20	2241.36	0
flat1000_50_0.col	93	93.9	2/20	16779.12	92	93.5	1/20	24311.86	1
flat1000_60_0.col	93	93.85	3/20	14715.85	93	94.39	2/20	37262.49	0
flat1000_76_0.col	93	94.1	2/20	24103.23	94	94.56	10/20	30406.57	-1
latin_square_10.col	103	104.61	1/20	32403.96	103	104.22	5/20	28201.09	0
inithx.i.2.col	35	35	20/20	4106.24	60	65.5	1/20	12.55	-25
inithx.i.3.col	36	36	20/20	6529.65	64	68	2/20	218.11	-28
p-value					5.6e-3				

population framework positively contributes to the performance of the MAECP algorithm in particularly in terms of search capacity.

6 Conclusions

We have presented the first population based memetic algorithm for the NP-hard equitable graph coloring problem. The algorithm relies on a backbone crossover to combine parent solutions and a dedicated 2-phase tabu search for solution improvement. We have evaluated the algorithm on a set of 73 popular benchmark instances in the literature and compared our results with those of the state-of-the-art algorithms. For the 41 instances with known optima, the proposed algorithm consistently achieves all the known optimal results. More significantly, among the 32 instances whose optima are still unknown, the proposed algorithm discovers 9 improved best results which correspond to new upper bounds of the equitable chromatic numbers, which can serve as new references for assessment of other ECP algorithms. More generally, this work advances the state-of-the-art of solving this challenging problem. Given that the ECP is able to formulate a number of practical applications in the real-world, the proposed algorithm could be usefully applied to better solve these real problems as well.

For future work, several directions could be followed. First, like other coloring algorithms, the proposed algorithm is computation intensive. It would be interesting to investigate specific techniques to speed up the local optimization component. For this purpose, implementations of the proposed algorithm on GPU are worthy of studies. Second, the combined fitness function used in the 2-phase search could be improved by introducing an self-adaptive technique to balance the two composing penalty terms like [8,38]. Finally, it would be interesting to investigate the proposed solving framework for solving other graph coloring problems.

Acknowledgment

We are grateful to the reviewers for their useful comments and suggestions which helped us to significantly improve the paper. Support for the first author from the Fundamental Research Funds for the Central Universities (Free Topic Selection Project for Talent Introduction and Supportive Project for Basic Scientific Research (2242019K40112)) is also acknowledged.

References

[1] Laura Bahiense, Yuri Frota, Thiago F. Noronha, and Celso C. Ribeiro. A branch-and-cut algorithm for the equitable coloring problem using a formulation by representatives. *Discrete Applied Mathematics*, 164:34–46, 2014.

[2] Una Benlic and Jin-Kao Hao. A multilevel memetic approach for improving graph k-partitions. *IEEE Transactions on Evolutionary Computation*, 15(5):624–642, 2011.

[3] Jacek Blazewicz, Klaus H. Ecker, Erwin Pesch, Günter Schmidt, and Jan Weglarz. Scheduling computer and manufacturing processes. *Journal of the Operational Research Society*, 48(6):659–659, 1997.

[4] Hans L. Bodlaender and Fedor V. Fomin. Equitable colorings of bounded treewidth graphs. In *International Symposium on Mathematical Foundations of Computer Science*, 180–190. Springer, 2004.

[5] Bor-Liang Chen, Ming-Tat Ko, and Ko-Wei Lih. Equitable and m-bounded coloring of split graphs. In *Combinatorics and Computer Science*, 1–5. Springer, 1996.

[6] Bor-Liang Chen, Ko-Wei Lih, and Pou-Lin Wu. Equitable coloring and the maximum degree. *European Journal of Combinatorics*, 15(5):443–447, 1994.

[7] Xianshun Chen, Yew-Soon Ong, Meng-Hiot Lim, and Kay Chen Tan. A multi-facet survey on memetic computation. *IEEE Transaction on Evolutionary Computation*, 15(5):591–607, 2011.

[8] Yuning Chen, Jin-Kao Hao, and Fred Glover. An evolutionary path relinking approach for the quadratic multiple knapsack problem. *Knowledge-Based Systems*, 92:23–34, 2016.

[9] Sajal K. Das, Irene Finocchi, and Rossella Petreschi. Conflict-free star-access in parallel memory systems. *Journal of Parallel and Distributed Computing*, 66(11):1431–1441, 2006.

[10] Hanna Furmańczyk. Equitable coloring of graph products. Opuscula Mathematica, 26(1): 31–44, 2006.

[11] Hanna Furmańczyk, Andrzej Jastrzebski, and Marek Kubale. Equitable coloring of graphs. Recent theoretical results and new practical algorithms. *Archives of Control Sciences*, 26(3):281–295, 2016.

[12] Hanna Furmańczyk, K. Kaliraj, Marek Kubale, and J. Vernold Vivin. Equitable coloring of corona products of graphs. *Advances and Applications in Discrete Mathematics*, 11(2):103–120, 2013.

[13] Hanna Furmańczyk and Marek Kubale. Equitable coloring of graphs. *Contemporary Mathematics*, 352:35–54, 2004.

[14] Philippe Galinier and Jin-Kao Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 1999.

[15] Michael R. Garey and David S. Johnson. Computers and intractability: a guide to the theory of np-completeness. 1979. *San Francisco, LA: Freeman*, 58, 1979.

[16] Fred Glover and Manuel Laguna. Tabu search. Kluwer Academic Publishers, Boston, 1997.

[17] Fred Glover, Manuel Laguna, Raphael Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3):652–684, 2000.

[18] A. Hajnal, E. Szemerédi, P. Erdős, A. Rényi, and VT Sós. Combinatorial theory and its applications. *Proof of a Conjecture of P. Erdös*, 2:601–623, 1970.

[19] Jin-Kao Hao. Memetic algorithms in discrete optimization. In F. Neri, C. Cotta, P. Moscato (Eds.), Handbook of Memetic Algorithms. Studies in Computational Intelligence. 379, Chapter 6, 73–94, 2012.

[20] Alain Hertz and Dominique de Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351, 1987.

[21] Sandy Irani and Vitus Leung. Scheduling with conflicts, and applications to traffic signal control. In *SODA*, Lecture Notes in Computer Science, 96, 85–94, 1996.

[22] Yan Jin and Jin-Kao Hao. Hybrid evolutionary search for the minimum sum coloring problem of graphs. *Information Sciences*, 352–353:15-34,2016

[23] Yan Jin and Jin-Kao Hao. Solving the Latin square completion problem by memetic graph coloring. *IEEE Transactions on Evolutionary Computation*, In press, available online 9 February 2019.

[24] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

[25] Xiangjing Lai, Jin-Kao Hao, and Fred Glover. Backtracking based iterated tabu search for equitable coloring. *Engineering Applications of Artificial Intelligence*, 46:269–278, 2015.

[26] Rhyd M.R. Lewis. A guide to graph colouring - algorithms and applications. Springer International Publishing Switzerland, 2016.

[27] Ko-Wei Lih. The Equitable Coloring of Graphs. In D.-Z. Du and P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, 3:543–566, Kluwer, Boston, 1998.

[28] Zhipeng Lü and Jin-Kao Hao. A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203(1):241–250, 2010.

[29] Isabel Méndez-Díaz, Graciela Nasini, and Daniel Severín. A tabu search heuristic for the equitable coloring problem. In *International Symposium on Combinatorial Optimization*, Lecture Notes in Computer Science 8596:347–358. Springer, 2014.

[30] Pablo Moscato. Memetic algorithms: a short introduction. In David Corne, Marco Dorigo, Fred Glover(Eds), New Ideas in Optimization, McGraw-Hill Ltd., Maidenhead, UK. pages 219–234, 1999.

[31] Isabel Méndez-Díaz, Graciela Nasini, and Daniel Severín. A polyhedral approach for the equitable coloring problem. *Discrete Applied Mathematics*, 164:413–426, 2014.

[32] Isabel Méndez-Díaz, Graciela Nasini, and Daniel Severín. A dsatur-based algorithm for the equitable coloring problem. *Computers & Operations Research*, 57:41–50, 2015.

[33] Walter Meyer. Equitable coloring. *The American Mathematical Monthly*, 80(8):920–922, 1973.

[34] Laurent Moalic and Alexandre Gondran: Variations on memetic algorithms for graph coloring problems. *Journal of Heuristics*, 24(1):1–24, 2018.

[35] Rumana Nazmul, Madhu Chetty, and Ahsan Raja Chowdhury. An improved memetic approach for protein structure prediction incorporating maximal hydrophobic core estimation concept. *Knowledge-Based Systems*, In press, available online 17 July 2018.

[36] Ferrante Neri, Carlos Cotta, and Pablo Moscato **(Eds.),** Handbook of Memetic Algorithms. Studies in Computational Intelligence 379, Springer, 2012.

[37] Wen Sun, Jin-Kao Hao, Xiangjing Lai, and Qinghua Wu. On feasible and infeasible search for equitable graph coloring. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 369–376. ACM, 2017.

[38] Wen Sun, Jin-Kao Hao, Xiangjing Lai, and Qinghua Wu. Adaptive feasible and infeasible tabu search for weighted vertex coloring. *Information Sciences*, 466:203–219, 2018

[39] Alan Tucker. Perfect graphs and an application to optimizing municipal services. *SIAM Review*, 15(3):585–590, 1973.

[40] Wenyu Wang, Jin-Kao Hao, and Qinghua Wu. Tabu search with feasible and infeasible searches for equitable coloring. *Engineering Applications of Artificial Intelligence*, 71:1–14, 2018.

[41] Qinghua Wu and Jin-Kao Hao. A hybrid metaheuristic method for the maximum diversity problem. *European Journal of Operational Research*, 231(2):452-464, 2013.

[42] Kai Wu, Jing Liu, and Dan Chen. Network reconstruction based on time series via memetic algorithm. *Knowledge-Based Systems*, 164:404–425, 2019.