Tabu search with feasible and infeasible searches for equitable coloring

Wenyu Wang [a], Jin-Kao Hao [b,c], Qinghua Wu [a,*]

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

[b] *LERIA, Université d'Angers, 2 bd Lavoisier, 49045 Angers, Cedex 01, France*

[c] *Institut Universitaire de France, 1 rue Descartes, 75231 Paris Cedex 05, France*

Engineering Applications of Artificial Intelligence, 2018
https://doi.org/10.1016/j.engappai.2018.01.012

Abstract

The equitable coloring problem is a variant of the classical graph coloring problem that arises from a number of real-life applications where the cardinality of color classes must be balanced. In this paper, we present a highly effective hybrid tabu search method for the problem. Based on three complementary neighborhoods, the algorithm alternates between a feasible local search phase where the search focuses on the most relevant feasible solutions and an infeasible local search phase where a controlled exploration of infeasible solutions is allowed by relaxing the equity constraint. A novel cyclic exchange neighborhood is also proposed in order to enhance the search ability of the hybrid tabu search algorithm. Experiments on a set of 73 benchmark instances in the literature indicate that the proposed algorithm is able to find improved best solutions for 15 instances (new upper bounds) and matches the best-known solutions for 57 instances. Additional analyses show the interest of the cyclic exchange neighborhood and the hybrid scheme combining both feasible and infeasible local searches.

Keywords: tabu search; heuristics; infeasible local search; equitable coloring; vertex coloring.

* Corresponding author.
Email addresses: Wang.Wen.Yu@foxmail.com (Wenyu Wang), jin-kao.hao@univ-angers.fr (Jin-Kao Hao), qinghuawu1005@gmail.com (Qinghua Wu).

1 Introduction

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. A subset I of V is an independent set if no two vertices in I are joined by an edge in E [29]. A k-coloring of G is any partition of V into k mutually disjoint subsets (also called color classes). A k-coloring is said legal if each of its color classes is an independent set. The graph k-coloring problem is to find a legal k-coloring of G for a given k. The classical graph coloring problem (GCP) is to determine a legal k-coloring with k minimum for a general graph G (this minimum k is called the chromatic number of G and is denoted by $\chi(G)$).

In this paper, we are interested in a related coloring problem known as the *equitable coloring problem* (ECP for short) [39], which adds an additional *equity constraint* to the GCP – the sizes of any two color classes differ in at most one unit. In other words, an equitable k-coloring of G (denoted by k-eqcoloring) is a legal k-coloring $\{V_1, V_2, ..., V_k\}$ satisfying the following equity constraint [38]: for any two color classes V_i and V_j $(i \neq j)$, $||V_i|-|V_j|| \leq 1$ where $|V_l|$ is the cardinality of color class V_l. The ECP is to determine the minimum integer k such that G admits an equitable k-coloring. This minimum k is called the equitable chromatic number of G and is denoted by $\chi_{eq}(G)$. Clearly, the chromatic number of a graph is a lower bound of the equitable chromatic number of the graph.

The ECP is known to be NP-hard in the general case [17]. In addition to its significance as a difficult combinatorial problem, the ECP is notable for its ability to formulate a number of practical problems. A simple example concerns the problem of assigning a set of workers to a set of tasks. Pairs of tasks may conflict each other and must not be assigned to the same worker. In addition, it is desirable that the number of tasks assigned to each worker be approximately the same for the consideration of fairness. The problem can be modeled by building a graph where a vertex corresponds to a task and an edge is created for each conflicting pair of tasks. Now if we represent the workers by colors, then this workforce assignment problem reduces to finding an equitable coloring in the corresponding graph. Other applications of the ECP arise from garbage collection [47], scheduling in communication systems [27] and load balancing in multiprocessor machines [17]. An introduction to the ECP and some basic results are provided in [16].

Despite its relevance, the ECP has been studied essentially from a theoretical point of view. For instance, Hajnal and Szemerédi [24] proved that a graph G has an equitable $(\Delta(G) + 1)$-coloring where $\Delta(G)$ is the maximum vertex degree of G. Meyer [39] conjectured that $\chi_{eq}(G) \leq \Delta(G)$ for all connected graphs except the complete graphs and the odd cycles. This conjecture has been confirmed to be true for some special cases such as connected bipartite

graphs [33], trees [11], outerplanar graphs [30,49] and graphs with $\Delta(G) \geq \frac{1}{2}|V|$ or $\Delta(G) \leq 3$ [12]. Kostochka and Nakprasit [31] showed that graphs with an average degree up to $\Delta(G)/5$ are equitably k-colorable for every $k \geq \Delta(G)$. In addition, some special cases such as graphs with bounded treewidth [5] have also been identified to admit efficient polynomial algorithms or approximation algorithms.

For the purpose of practical solving of the general ECP, several exact and heuristic algorithms were recently proposed in the literature. Exact approaches have theoretical advantage of guaranteeing the optimality of the solution found, but they need a time that grows exponentially with the problem size in the general case. Still, several effective mathematical formulations and algorithms have been proposed for exactly solving the ECP. For instance, based on the asymmetric representative formulation for the GCP described in [8], Bahiense et al. [4] investigated the integer linear programming approach and developed effective branch-and-cut algorithms for the ECP. Méndez-Díaz et al. [36] adapted to the ECP the formulation and techniques used in [35], studied its polyhedral structure and derived families of valid inequalities. Some of these inequalities were proven to be very effective for a cutting-plane algorithm. Very recently, Méndez-Díaz et al. [38] proposed a DSATUR-based exact algorithm that exploits arithmetical properties inherent in equitable colorings and combines them with the techniques originally developed by Brown [7] and Brélaz [6] for DSATUR, and improved by San Segundo [44] and Sewell [45].

On the other hand, given the intrinsic intractability of the ECP, practical heuristics have been devised to handle problem instances whose optimal solutions cannot be reached by exact approaches. Specifically, Furmańczyk and Kubale [16] proposed two constructive greedy heuristics to generate an equitable coloring of a graph. Méndez-Díaz et al. [37] adapted to the ECP the implementation due to Galinier and Hao [18] of the Tabucol algorithm initially proposed by Hertz and de Werra [26] for the GCP. Very recently, Lai et al. [32] proposed a backtracking based iterated tabu search (BITS) algorithm for the ECP, which produced highly competitive results on a set of commonly used benchmarks.

Compared with the GCP for which a huge number of heuristic algorithms have been investigated, the ECP is surprisingly much less studied – only two recent local search approaches are available in the literature [32,37]. In this work, we aim to partially fill the gap by introducing a new hybrid tabu search (HTS) method that incorporates a combined use of both feasible and infeasible local searches. Our proposed hybrid tabu search is mainly motivated by two considerations. First, from the perspective of solution methods, tabu search has shown to be the most effective local search algorithm for the classical GCP [19] as well as many of its variants such as sum coloring [28] and T-coloring [15]. Second, from the perspective of the problem under investigation, ECP is

a variant of the classical GCP where color class's equity is imposed as a strict constraint. As for strictly constrained problems, imposing solution feasibility within a neighborhood search algorithm often restricts the search process too much, while exploiting the infeasibility can help explore the search space more effectively [3,10,21,41]. Especially, methods that can tunnel through feasible and infeasible regions are particularly attractive as a means to solve these highly constrained problems [10,21,42]. Inspired by these two observations, we proposed a hybrid TS method combining both feasible and infeasible local searches to effectively explore the search space of the ECP. The main contributions of this work can be summarized as follows.

- From the algorithm perspective, the proposed HTS approach distinguishes itself by several original features. First, we investigate a hybrid scheme which integrates both feasible and infeasible local searches within the same search algorithm in the context of solving the ECP. By relaxing the equity constraint in a controlled manner, the search algorithm is allowed to tunnel through feasible and infeasible regions to reach global optima or high quality solutions which are difficult to attain by only visiting feasible solutions. Second, to ensure an effective local optimization, we introduce a novel constrained-three-cyclic-exchange neighborhood which complements two existing neighborhoods. Third, all these ingredients are integrated in the proposed hybrid tabu search algorithm which is able to explore the solution space effectively.
- From the computational perspective, the proposed approach shows a very competitive performance on the set of 73 commonly used benchmarks. The computational results indicate that our algorithm improves the best-known solutions for 15 instances (new upper bounds) and matches the best-known results for 57 instances. Only for one instance the algorithm misses the best-known result. In particular, for several well-known DIMACS instances, our approach can significantly improve the current best-known results.

The remainder of this paper is organized as follows. In section 2, we present the proposed algorithm for solving the ECP. Section 3 is dedicated to computational results and comparisons with state-of-the-art approaches. Section 4 investigates some important components of the HTS algorithm to understand their impacts on the performance of the algorithm. Concluding remarks are given in Section 5.

2 A hybrid tabu search algorithm for the ECP

Like the GCP [19], the ECP can be approximated by solving a series of equitable k-coloring problems (denoted by k-ECP). We start with an initial number of k colors ($k \leq |V|$) and solve the associated k-ECP. If an equitable

legal k-coloring (i.e., k-eqcoloring) is found, k is decreased by one and a new k-eqcoloring is sought. This process is repeated until no k-eqcoloring can be found. The last k-eqcoloring constitutes an approximation of the minimum equitable coloring of the graph. Consequently, the ECP comes down to the problem of finding k-eqcolorings for a given k. Our HTS algorithm follows the above solution procedure and is designed to find a conflict-free k-eqcoloring from an initially conflicting k-eqcoloring. To rapidly determine an appropriate initial number of k colors that admits an equitable k-coloring, we apply the same binary search method proposed in [32].

2.1 Main framework

Algorithm 1 Main scheme of the hybrid tabu search algorithm for the k-ECP

Require: Graph $G = (V, E)$, the number k of colors used
Ensure: A k-equitable coloring if found
1: $s_0 = greedy\_generate(G,k)$ /*Section 2.3 */
2: **while** stopping condition is not met **do**
3: /*the feasible local search phase*/
4: $(s_1, s_{local\_best}) \leftarrow feasible\_local\_search(s_0)$ /*Section 2.5*/
5: **if** $f(s_{local\_best}) = 0$ **then**
6: return(s_{local_best})
7: **end if**
8: /*the infeasible local search phase*/
9: $(s_0, s_{local\_best}) \leftarrow infeasible\_local\_search(s_1)$ /*Section 2.6*/
10: **if** s_{local_best} is a feasible solution and $f(s_{local\_best}) = 0$ **then**
11: return(s_{local_best})
12: **end if**
13: **end while**

Given the highly constrained feature of the k-ECP imposed by the equity constraint, one key issue to be considered is how to design search strategies to effectively explore the complex search space of the problem. Indeed, as observed in other studies [3,9,10,21,40,42], for strongly constrained combinatorial optimization problems, accounting for problem constraints in the definition of the search space can lead to one search space where the feasible region often consists of components which are separated from each other by infeasible regions [22]. In this case, imposing solution feasibility during the search often prevents the search from exploring new promising areas and sometimes may make the search process blocked. To unlock the situation, one attractive strategy is to allow a controlled exploration of infeasible solutions with the purpose of facilitating transitions between structurally different high-quality solutions [3,22]. Based on this idea, the proposed hybrid tabu search algorithm combines the exploitation ability of its feasible local search component with the exploration ability of its infeasible local search component to effectively explore the search space of the k-ECP.

The general HTS algorithm is described in Algorithm 1. Basically, the algorithm alternates between a feasible local search phase (FLS for short) where only solutions satisfying the equity constraint are examined and an infeasible local search phase (ILS for short) during which the equity constraint is relaxed in a controlled manner. These two search phases play different roles in our algorithm. FLS is used to ensure the exploitation capability of the algorithm by focusing on the most relevant feasible regions while ILS is responsible for search exploration by guiding the algorithm towards unexplored search regions. By alternating between these two complementary search phases, the algorithm can maintain a balance between the exploration and the exploitation of its search process and is expected to visist various zones of the search space. Additional analyses demonstrate that the combined use of these two search strategies within the HTS algorithm constitutes an effective hybridization and plays an important role in ensuring the performance of the algorithm (Section 4.2).

Starting from an initial feasible solution (Sections 2.2 and 2.3), the search first performs the feasible local search phase (Section 2.5). During the FLS phase, only feasible solutions satisfying the equity constraint are considered with the aim of finding a (feasible) conflict-free k-eqcoloring. For this purpose, it explores intensively the feasible search space through a joint use of three complementary neighborhoods induced by three move operators (Section 2.4). At each iteration of FLS, the algorithm explores the union of these neighborhoods and selects one overall best admissible (non-tabu or globally improving) neighbor solution which reduces the most the number of conflicting edges.

When FLS is judged to be trapped in a deep local optimum, the algorithm switches to the infeasible local search phase to bring more search freedom into the search process. The basic idea of the ILS procedure is to allow the search to move into the infeasible space by relaxing the equity constraint in a controlled manner with the purpose of first locating a conflict-free k-coloring near the boundary of the feasible region which still violates the equity constraint. Using this promising infeasible (relative to the equity constraint) solution as a starting point, our ILS algorithm seeks to induce the search to gradually converge to another promising feasible region, while simultaneously searching for a conflict-free feasible solution by optimizing both the number of conflicting edges and the infeasibility degree of the solutions. To achieve this, three basic types of move operations are applied and the directional move operation transferring vertices from color classes of larger cardinality to color classes of smaller cardinality is selected with a higher priority.

Finally, if the ILS phase fails to find a conflict-free k-coloring satisfying the equity constraint, the HTS algorithm switches back to the FLS phase for further improvement. The algorithm then iterates the above two search phases until a feasible conflict-free solution is found or a predefined stop condition

(typically a fixed timeout limit) is verified.

2.2 Search space and evaluation function

For a given k-ECP instance, the search space Ω explored by our algorithm is composed of all possible k-colorings which may or may not satisfy the equity constraint. A candidate solution in Ω (i.e., a k-coloring) is therefore any partition of the vertex set into k subsets V_1,...,V_k. For a given candidate solution $s = \{V_1, ..., V_k\}$, if for all $\{u, v\} \in E$, $u \in V_i$ and $v \in V_j$, $i \neq j$, then s is a *conflict-free* or *legal* k-coloring. Otherwise, s is a conflicting k-coloring. Furthermore, s is said *feasible* if it satisfies the equity constraint. Otherwise, it is said infeasible.

For a given infeasible solution $s = \{V_1, ..., V_k\}$ (i.e., the equity constraint is not satisfied), the degree of infeasibility of the solution s is measured by $DI(s) = \sum_1^k \lfloor ||V_i| - \frac{n}{k}| \rfloor$ where $\lfloor . \rfloor$ denotes the integer part of a positive real number and n is the number of vertices of G. Notice that a feasible solution is not necessarily a conflict-free solution. Then, the search space $\Omega_f \subset \Omega$ explored by the feasible local search procedure is composed of all possible k-colorings satisfying the equity constraint while the search space explored by the infeasible local search procedure includes both feasible and infeasible k-colorings (i.e., Ω).

Given a k-coloring $s = \{V_1, ..., V_k\}$ in Ω, which can be feasible or infeasible, the evaluation function $f(s)$ counts the total number of conflicting edges induced by s such that:

$$f(s) = \sum_{\{u,v\} \in E} \gamma_{uv} \tag{1}$$

where

$$\gamma_{uv} = \begin{cases} 1, & \text{if } u \in V_i,\ v \in V_j \text{ and } i = j, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, a candidate solution s with $f(s) = 0$ corresponds to a legal (i.e., conflict-free) k-coloring. It is a feasible solution if it additionally satisfies the equity constraint. Our HTS algorithm aims to find such a feasible conflict-free solution.

2.3 Generation of initial solutions for the k-ECP

The HTS algorithm begins with an initial solution $s \in \Omega$ which is a conflicting k-coloring that satisfies the equity constraint. The algorithm then tries to improve s by minimizing its number of conflicting edges. Specifically, we employ a greedy construction method similar to those applied in [32,37] to generate an initial k-eqcoloring $s = \{V_1, ..., V_k\}$. Let U denote the set of unassigned vertices. Initially, U is set to V and all color classes V_1,...,V_k of s are set to empty. Then we randomly pick k distinct vertices from U and use each of these k vertices to initialize a different color class. In each subsequent construction step, the k color classes V_1,...,V_k are considered in turn (see [32]) and the unassigned vertices are assigned to V_1,...,V_k one by one in a greedy way. Precisely, in step m, color class $V_{m\%k}$ is considered, and an unassigned vertex $v \in U$ that has the minimum number of adjacent vertices in $V_{m\%k}$ is identified and assigned to $V_{m\%k}$ (ties are broken randomly). This process is repeated until all vertices are assigned to a color class. Note that this construction procedure only ensures the equity constraint of an initial solution while the coloring constraint will be satisfied by the search algorithm.

2.4 Basic move operators and neighborhoods

One of the most critical features of local search is the definition of its neighborhood. Typically, a neighborhood is defined by a move operator that transforms a given solution s into a neighbor solution s_0. Our HTS algorithm jointly employs three basic move operators to generate its three neighborhoods which are explored during its two search phases: the directional-one-move operator defined by moving a vertex from a larger color class to a smaller color class, the two-exchange operator based on exchanging two vertices from different color classes, and the constrained-three-cyclic-exchange operator consisting in transferring vertices among three different color classes. The directional-one-move and two-exchange operators have been previously used in [37,32] while the novel constrained-three-cyclic-exchange operator is introduced for the first time in this work. In what follows, we introduce these three neighborhoods.

1) Two traditional neighborhoods: Given a solution $s = \{V_1, ..., V_k\}$, let $C(s)$ be the set of conflicting vertices in s, where a vertex is said to be conflicting if it belongs to the same color class as one of its adjacent vertices. The directional-one-move and two-exchange operators are described as follows:

- *Directional-one-move* (N_1): This move operator transfers a conflicting vertex $v \in C(s)$ from its current color class V_i to another color class V_j such that $i \neq j$ and $|V_i| > |V_j|$. In order to efficiently evaluate the move gain in-

duced by a directional-one-move move, which indicates how much a coloring is improved in terms of the number of conflicting edges after the move, we adopt the fast incremental evaluation technique first developed for the graph coloring problem [18]. The main idea is to maintain a $n \times k$ matrix Λ with elements $\Lambda_{[v][q]}$ recording the number of vertices adjacent to v in color class V_q ($1 \leq q \leq k$). With this memory, the move gain of a directional-one-move operation can be conveniently computed by:

$$\Delta_{v,V_i,V_j} = \Lambda_{[v][j]} - \Lambda_{[v][i]} \tag{2}$$

Once a directional-one-move operation is performed, one just needs to update a subset of values in Λ affected by this move as follows. For each vertex u adjacent to vertex v, $\Lambda_{[u][i]} \leftarrow \Lambda_{[u][i]} - 1$, and $\Lambda_{[u][j]} \leftarrow \Lambda_{[u][j]} + 1$. Clearly, updating Λ can be done in $O(n)$.

- *Two-exchange* (N_2): This move operator swaps a pair of vertices (u, v) from two different color classes where at least one of them is a conflicting vertex. Suppose $u \in V_{pu}$ and $v \in V_{pv}$, then the move gain of a two-exchange move can be efficiently computed by:

$$\Delta_{u,v} = \Lambda_{[u][pv]} - \Lambda_{[u][pu]} + \Lambda_{[v][pu]} - \Lambda_{[v][pv]} - 2e_{uv}$$

where $e_{uv} = 1$ if u and v are adjacent in the graph, otherwise $e_{uv} = 0$. Since a two-exchange move can be decomposed into two independent one vertex moves consisting in moving a vertex from its current color class to another color class, the update of the matrix Λ after a two-exchange move can also be realized in $O(n)$.

Note that for our FLS procedure, the directional-one-move operator is only applicable when n does not divide k, since this operator always moves vertices from a large color class to a small color class. Meanwhile, when n divides k, all color classes have the same size. For a feasible solution $s \in \Omega$, its feasibility can always be maintained under both the directional-one-move and two-exchange operators.

2) **New Constrained-Three-Cyclic-Exchange neighborhood:** To reinforce the search ability of our HTS algorithm, we introduce a novel neighborhood N_3 based on the constrained-three-cyclic-exchange move operator, which transfers vertices among three color classes under the equity constraint.

Constrained-three-cyclic-exchange (N_3): This move operator displaces three vertices a, b and c from three different color classes in a cyclic way (see Fig. 1 for an example). The move gain of exchanging three vertices a, b and c (suppose $a \in V_{pa}$, $b \in V_{pb}$ and $c \in V_{pc}$) can be calculated by

$$\Delta_{a,b,c} = \Lambda_{[a][pb]} - \Lambda_{[a][pa]} + \Lambda_{[b][pc]} - \Lambda_{[b][pb]} + \Lambda_{[c][pa]} - \Lambda_{[c][pc]} - l \tag{3}$$

where $l \in \{0, 1, 2, 3\}$ is the number of edges between vertices a, b and c.

Clearly, such an unconstrained three-cyclic-exchange move leads to a neighborhood whose size is bounded by $O(n^3)$, which is quite large compared to the neighborhoods induced by the directional-one-move and two-exchange operators and thus very expensive to explore by our tabu search procedure.

To increase the computational efficiency of the search procedure, we devise a constrained neighborhood which is both more focused and smaller-sized (see, e.g., [20]). The basic idea of our constrained neighborhood is that instead of permitting any vertex to take part in a three-cyclic-exchange operation, we use a vertex filtering technique to constraint the three exchange vertices a, b and c to be examined to come from three specifically identified subsets $A \subseteq V$, $B \subseteq V$ and $C \subseteq V$ such that the size of A, B and C is as small as possible, and the resulting neighborhood contains the most promising solutions of the full three-cyclic-exchange neighborhood.

When exchanging three vertices a, b and c in a cycle to generate a neighbor solution, the computation of the move gain (Equation 3) indicates that the quality of the neighbor solution mainly depends on the Λ value. Furthermore, as the HTS approach progresses, the solution quality in terms of conflicting edges becomes better and better, as for a highly refined solution, any vertex generally fits its color class well (i.e., $\Lambda_{[a][pa]}$, $\Lambda_{[b][pb]}$ and $\Lambda_{[c][pc]}$ generally have a very small value). Thus, the move gain crucially depends on the value of $\Lambda_{[a][pb]}$, $\Lambda_{[b][pc]}$ and $\Lambda_{[c][pa]}$. On the other hand, it's quite likely that moving a vertex v to a color class V_i with many vertices in V_i adjacent to v (i.e., $\Lambda_{[v][i]}$ takes a large value) tends to greatly deteriorate the quality of the current solution. Thus, to improve the computational efficiency, our constrained three-cyclic-exchange neighborhood tries to avoid examining three-cyclic-exchange moves involving the transfer of a vertex v to a color class V_i which have many vertices adjacent to v.

Based on the above observation, we use the following vertex filtering rule to determine the vertices that are included in the three special subsets A, B and

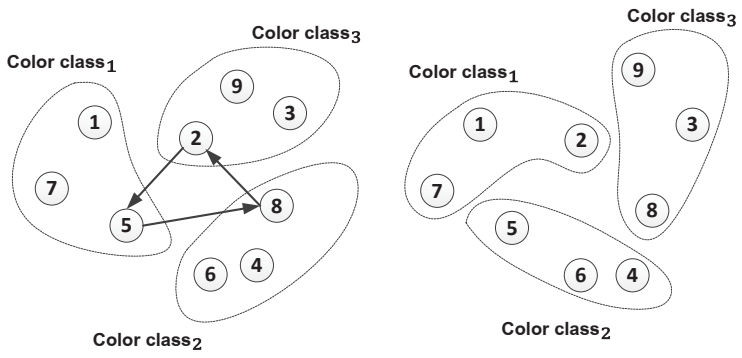

Fig. 1. Illustration of a three-cyclic-exchange move. A 3-eqcoloring before the three--cyclic-exchange move (left), the 3-eqcoloring after the move (right).

C for further examination. Subset A is set to include all conflicting vertices in the current solution s, then for each conflicting vertex $a \in A$, two vertex subsets B and C are constructed for exchange with a as follows: Each vertex $b \in V$ whose current color class is V_{pb}, such that $pb \neq pa$ and V_{pb} has at most T vertices adjacent to a (i.e., $\Lambda_{[a][pb]} \leq T$), is selected to be included in B, where T is a threshold parameter set to 2 in this work. On the other hand, any vertex $c \in V$ is selected to belong to C such that c is excluded from V_{pa} and has at most T adjacent vertices in V_{pa} (i.e., $\Lambda_{[c][pa]} \leq T$). Then in our constrained cyclic-exchange neighborhood, only vertices in the subset B and C are considered for exchange with the conflicting vertex a in order to obtain a neighbor solution. All the neighbors of s obtained with this scheme constitutes our constrained cyclic-exchange neighborhood.

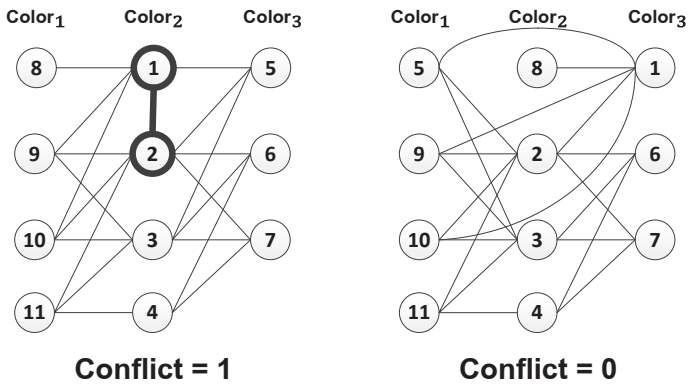

Fig. 2. An example for the constrained-three-cyclic-exchange neighborhood.

To illustrate the constrained-three-cyclic-exchange neighborhood, consider a simple 3-ECP instance shown in Figure 2. In this example, set A contains two conflicting vertices 1 and 2. For conflicting vertex 1, vertex subsets $B = \{5, 6, 7\}$ and $C = \{8\}$ are constructed. For conflicting vertex 2, vertex subsets $B = \emptyset$ and $C = \{8\}$ are constructed. Then the constrained-three-cyclic-exchange neighborhood includes only three possible neighbor solutions and the best neighbor solution provided by the constrained neighborhood is obtained by exchanging vertices 1, 5 and 8, which leads also to a feasible conflict-free solution.

Obviously, the size of our constrained cyclic-exchange neighborhood is typically smaller than the size of unconstrained neighborhood (n^3). It's noteworthy that the resulting constrained cyclic-exchange neighborhood contains almost all high-quality neighbor solutions of the whole unconstrained neighborhood, though it may occasionally miss the best solution in the unconstrained neighborhood. More importantly, it avoids the examination of most of the unpromising moves, and significantly improves the computational efficiency of our HTS algorithm. In Section 4.1, we will investigate the effectiveness of the constrained three-cyclic-exchange neighborhood and show its key role to the

overall performance of our algorithm.

Since a three-cyclic-exchange move can be decomposed into three independent one vertex moves, the update of the matrix Λ after a three-cyclic-exchange move can also be achieved in $O(n)$. When the constrained-three-cyclic-exchange operator is applied to a feasible solution, it always generates a feasible solution.

2.5 The feasible local search phase

Algorithm 2 Feasible local search for the k-ECP

Require: Graph $G = (V, E)$, an initial solution s_0, number of colors k, the search depth M_{FLS}

Ensure: The best equitable k-coloring s_{local_best} found so far, the end point s_1 of the feasible local search

$s \leftarrow s_0$
$s_{local\_best} \leftarrow s_0$ /*records the best solution found during the FLS phase*/
$Iteration \leftarrow 0$ /*counter of consecutive iterations during which $f(s)$ is not improved*/
Initialize *tabu_list*
while $Iteration < M_{FLS}$ **do**
  Construct neighborhoods N_1, N_2 and N_3 from s
  Choose an overall best admissible neighbor s' from $N_1 \cup N_2 \cup N_3$ with the minimum number of conflicting edges
  $s \leftarrow s'$
  if $f(s) = 0$ **then**
    Return(s) /*return the conflict-free k-eqcoloring found and the whole procedure stops*/
  end if
  Update *tabu_list*
  if $f(s) < f(s_{local\_best})$ **then**
    $s_{local\_best} \leftarrow s$
    $Iteration \leftarrow 0$
  else
    $Iteration \leftarrow Iteration + 1$
  end if
end while
Return(s,s_{local_best})

In our algorithm, FLS is used to examine intensively the feasible search regions so as to find a conflict-free k-coloring satisfying the equity constraint. For this purpose, FLS restricts its search only within the feasible search space Ω_f (satisfying the equity constraint) and explores feasible solutions through the combined use of the three complementary neighborhoods N_1 N_2 and N_3 introduced

in the last section. As indicated in Section 2.4, by applying the directional-one-move, two-exchange and constrained-three-cyclic-exchange moves to a feasible solution, solution feasibility is always maintained. Thus, by applying these three move operators, the search will stay in the feasible search region during the FLS phase.

The FLS procedure is based on the general tabu search framework and its main scheme is summarized in Algorithm 2. To make a more thorough examination in the feasible search space, our FLS procedure explores its three neighborhoods in a combined way, i.e., at each iteration of FLS, it examines all neighbor solutions generated by the directional-one-move, two-exchange and constrained-three-cyclic-exchange moves, and selects the overall best admissible (non-tabu or globally improving) solution with the minimum number of conflicting edges to generate the next solution. Such a combined neighborhood ensures an aggressive exploration of the most relevant feasible solutions and increases the chance of our FLS procedure to find a conflict-free solution.

To avoid short-term cycles, each time a vertex v is displaced (by the directional-one-move, two-exchange and constrained-three-cyclic-exchange operators) from its current color class V_i to another color class V_j, v is forbidden to move back to V_i for the next tt (tabu tenure) iterations. To tune the tabu tenure tt, we adopt the same hybrid tabu list management strategy proposed in [32]. To accompany this tabu rule, an aspiration criterion is also applied which allows a forbidden move to be selected if this move leads to a globally improving solution, i.e., a neighbor solution better than the best solution found so far.

When the best solution cannot be improved for M_{FLS} consecutive iterations during the FLS phase, the search is considered to be stagnating. To direct the search towards new regions of the search space, the search switches to the infeasible local search phase to diversify the search.

2.6 The infeasible local search phase

When restricting the search process only to feasible solutions, some beneficial vertex moves between color classes could be prohibited by the equity constraint and this confines the search process to a localized portion of the search space. In such cases, constraint relaxation is an attractive strategy. By visiting some intermediary infeasible solutions, despite violating the equity constraint, this strategy enables a much easier access from one promising search region to another and helps to enhance the exploration capability of the algorithm. Thus, when FLS is judged to be trapped in a local optimum, the algorithm switches to the infeasible local search phase to encourage the search process to explore new potential areas.

The ILS procedure (Algorithm 3) is composed of two stages: the divergent stage that encourages the search to diverge from the current search context by relaxing the equity constraint in a controlled manner, and the convergent stage that progressively directs the search towards a new feasible region by means of a tabu search procedure, which optimizes both the number of conflicting edges and the infeasibility degree of the solution with the aim of finding a conflict-free feasible k-coloring in the relaxed search space.

During the divergent stage, in order to drive the search to move away from the current search context, the algorithm relaxes the equity constraint as $\lfloor \frac{n}{k} \rfloor - t \leq |V_i| \leq \lceil \frac{n}{k} \rceil + t$, $\forall i \in \{1, 2, ..., k\}$, where t is an integer parameter used to control the amount of relaxation and initially set to 1. Then under the relaxed equity constraint, the algorithm aims to find a conflict-free k-coloring. If a conflict-free k-coloring satisfying the relaxed equity constraint is successfully located, the divergent stage stops. Otherwise, t is increased by one unit and a conflict-free k-coloring is sought under the more relaxed constraint. This process is repeated until a conflict-free k-coloring satisfying the relaxed equity constraint is found. To locate a conflict-free k-coloring satisfying the relaxed equity constraint, the divergent stage adopts a search procedure similar to the classic TabuCol procedure [26,19], with the only difference that our procedure accepts a neighbor solution under the relaxed equity constraint, while both procedures share the same neighborhood defined by moving a conflicting vertex from its original color class to another color class. The search for a conflict-free k-coloring under a given relaxed equity constraint stops when a conflict-free k-coloring is found or when a prefixed maximum number ($M_{iter} = 25000$) of iterations is attained, in which case t is increased by one and a new conflict-free k-coloring is sought.

The rationale of using a conflict-free k-coloring as the end point of the divergent stage can be explained as follows. There is a high chance that a feasible conflict-free k-coloring locates around an infeasible conflict-free k-coloring slightly violating the equity constraint in the search space. In most cases, it is very easy to locate a conflict-free k-coloring by slightly relaxing the equity constraint. One exception that rarely happens is that for very few k-ECP instances, when k is set to a sufficiently small value k^* (k^* is the best-known minimum number of colors reported in the literature for the classic graph coloring problem), it may be difficult to find a conflict-free k-coloring even under the totally relaxed equity constraint. In this case, the k-coloring with the minimum number of conflicting edges is returned as the output of the divergent procedure.

Starting from the infeasible k-coloring produced by the divergent stage, the convergent stage tries to find a feasible conflict-free k-coloring by optimizing both the number of conflicting edges and the degree of infeasibility of the solution. The proposed convergent procedure is also based on the tabu search

Algorithm 3 Infeasible local search for the k-ECP

Require: Graph $G = (V, E)$, a conflicting k-coloring s_1, the allowed maximum number of iterations M_{ILS}

Ensure: The best equitable k-coloring s_{local_best} found so far, the end point s_0 of the infeasible local search

/* Divergent stage */
$s_c \leftarrow Divergent\_Procedure(s_1)$
/* Convergent stage */
$s \leftarrow s_c$
$s_{local\_best} \leftarrow s_c$
$Iteration \leftarrow 0$
Initialize *tabu_list*
while $Iteration < M_{ILS}$ **do**
  Construct neighborhood N_1 from s
  if there exist admissible solutions in N_1 with objective function value not worse than $f(s)$ **then**
    Choose an admissible neighbor solution $s' \in N_1$ such that $f(s') \leq f(s)$ and s' reduces the most the infeasibility degree of the current solution
    $s \leftarrow s'$
  else
    Construct neighborhoods N_2 and N_3 from s
    Choose an overall best admissible neighbor s' from $N_1 \cup N_2 \cup N_3$ with the minimum number of conflicting edges
    $s \leftarrow s'$
  end if
  if $f(s) = 0$ and s is a feasible solution **then**
    Return(s) /*return the conflict-free k-eqcoloring found and the whole procedure stops*/
  end if
  Update *tabu_list*
  if $f(s) < f(s_{local\_best})$ **then**
    $s_{local\_best} \leftarrow s$
  end if
  $Iteration \leftarrow Iteration + 1$
end while
/* Repair procedure */
if s is an infeasible solution **then**
  $s_0 \leftarrow Repair\_Procedure(s)$
end if
Return(s_0,s_{local_best})

framework and uses the same three types of move operators introduced in Section 2.4 to explore the search space Ω composed of both feasible and infeasible solutions. Especially, among these three types of move operators, the convergent stage gives priority to the directional-one-move operator, since this move operator can be used not only to optimize the objective function value (the number of conflicting edges), but also to help reduce the infeasibility degree of the solution by transferring vertices from larger color classes to smaller ones. These three types of move operators are explored in the convergent procedure according to the following two rules:

- **Rule 1**: The convergent procedure first examines all directional-one-move moves, and identifies among them all moves with a move gain $\Delta \leq 0$ in terms of the objective function value (Eq. 1). If such moves exist, the convergent procedure selects among them the admissible move that mostly reduces the infeasibility degree of the solution to generate the next solution.
- **Rule 2**: If there is no admissible directional-one-move move satisfying Rule 1 (i.e., with $\Delta \leq 0$ and reducing the infeasibility degree of the solution), our convergent procedure then examines all the directional-one-move, two-exchange, and constrained-three-cyclic-exchange moves, and selects among them the best admissible (*non-tabu* or globally improving) move that reduces the conflicting edges most to generate the next solution.

By applying the selection rule that favors the directional-one-move transitions which reduce the infeasibility degree of the solution, the search is guided to progressively move towards feasible search regions and thus gradually increase the chance of the algorithm to find a feasible conflict-free k-coloring during the ILS phase. In addition, to prevent the convergent procedure from short-term cycling, each time a vertex v is moved from its original color class V_i (by the three types of move operators) to another color class V_j, it is forbidden to bring v back to color class V_i for the next tt iterations. The tabu tenure tt is tuned using the same method as described in Section 2.5.

The convergent procedure of ILS may visit both feasible and infeasible solutions. However, its ultimate objective is to find a feasible conflict-free k-coloring. To achieve this, at each iteration of the convergent procedure, we check if the current solution s is a feasible solution with $f(s) = 0$. This can be done easily in $O(k)$ by verifying that the cardinality of each color class of s satisfies the equity constraint (i.e., $\lfloor ||V_i| - \frac{n}{k}| \rfloor = 0,\ \forall i \in \{1, ..., k\}$, $\lfloor . \rfloor$ denotes the integer part of a positive real number). The convergent procedure stops when a feasible conflict-free k-coloring is found or when an allowed maximum number of iterations M_{ILS} is reached. In case the convergent procedure fails to find a feasible conflict-free k-coloring during the ILS phase, the final solution of the convergent procedure is a highly refined solution with few conflicting edges and should be very close to the boundaries of the feasible region relative to the equity constraint. This solution can then be used as a very promising

starting solution for the next round of the feasible local search procedure. Before the algorithm switches back to the FLS phase, a simple repair mechanism is invoked to transform this solution into a feasible solution (relative to the equity constraint). The repair mechanism removes vertices (selected randomly) from color classes for which $|V_i| > \lceil \frac{n}{k} \rceil$ and adds these vertices to color classes for which $|V_i| < \lfloor \frac{n}{k} \rfloor$. The process continues until the equity constraint is met.

3 Computational Experiments

This section is dedicated to an experimental assessment of the proposed HTS algorithm. For this purpose, we show computational results on a set of 73 benchmark instances commonly used in the literature and make comparisons with the current best performing heuristic approaches.

3.1 Problem instances and experimental protocol

The set of 73 commonly used benchmark instances were initially proposed for the conventional graph coloring problem. They were first collected for the "Second DIMACS Implementation Challenge on Maximum Clique, Graph Coloring, and Satisfiability" and later extended for the series of competitions on "Graph Coloring and its Generalizations" (COLOR02/03/04 competitions). These instances are available at http://mat.gsia.cmu.edu/COLOR/instances.html and http://www.cs.hbg.psu.edu/txn131/graphcoloring.html.

Our HTS algorithm was coded in C++ [1] and compiled by the g++ compiler with the '-O2' option. The experiments were conducted on a computer with an Intel Xeon E5440 processor (2.83 GHz CPU and 2 GB RAM) running the Linux operating system. Note that the HTS algorithm was run on the same computing platform as BITS [32].

To assess the performance of the algorithm, we report computational results using the same two stopping criteria used in [32]. The first stopping criterion is a short time limit of 3600 seconds per instance and per run, which was also previously used in [37]. Under this stopping criterion, we ran the HTS algorithm only one time to solve each instance in order to make a fair comparison with the results reported in [32,37]. The second stopping criterion is a long time limit of 10^4 seconds for the instances with $n \leq 500$ or 2×10^4 seconds for larger instances with $n > 500$. Such time limits were typically used in the

[1] The source code will be publicly available at: http://www.info.univ-angers.fr/ hao/eqcoloring.html.

Table 1
Settings of important parameters.

Parameters	Section	Description	Values
T	2.4	Threshold value for subset construction of the constrained cyclic exchange neighborhood	2
M_{FLS}	2.5	Iterations of feasible local search phase	20000
M_{ILS}	2.6	Iterations of infeasible local search phase	5000
tt	2.5, 2.6	Tabu tenure management factor	10
m	2	Search depth of the binary search procedure	4

literature for testing graph coloring algorithms and enable us to better observe the behavior of the HTS algorithm when more time is available. For the long time criterion, we ran our HTS algorithm 20 times for each instance like in [32].

3.2 Parameter settings

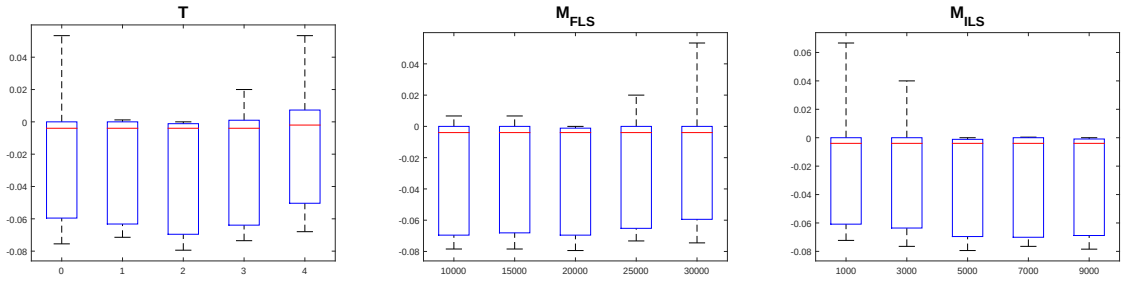

Fig. 3. Box and whisker plots corresponding to different values of T (left), M_{FLS} (middle) and M_{ILS} (right) in terms of solution quality. X-axis indicates the tested parameter values and Y-axis shows the performance.

Like other heuristic approaches for the GCP and the ECP, the proposed HTS algorithm has a number of parameters to be tuned, the parameter settings used in our experiments are summarized in Table 1. Three main parameters are the search depth of FLS (M_{FLS}), the allowed maximum number of iterations for ILS (M_{ILS}) and the threshold value for subset construction of the constrained cyclic exchange neighborhood. To determine these parameter values, we test for each parameter a set of potential values with the other parameters fixed to their default values from Table 1. We run our HTS algorithms 10 times under the long time limit on a selection of 21 instances. The average objective value over the 10 runs are considered for each instance and the corresponding parameter. Specifically, we tested values for T in the range [0..4], M_{FLS} in the range [10000..30000] and M_{ILS} in the range [1000..9000]. To compare the results in terms of solution quality, we use the popular box and whisker plots. Fig. 3 shows the box and whisker plots obtained with these three parameters. The left sub-figure shows the results for different T values while the middle sub-figure shows the results for different M_{FLS} values and the right sub-figure indicates the results for different M_{ILS} values. The X-axis in each sub-figure indicates the tested parameter values and the Y-axis shows the solution quality expressed as the percentage deviation of the obtained results from the best-

known results reported in the literature. From Fig. 3, we observe that the setting of $T = 2$, $M_{FLS} = 20000$ and $M_{ILS} = 5000$ globally leads to the best results. In addition to these three important parameters, HTS requires also several other parameters like the search depth of the binary search procedure and the tabu tenure for FLS and ILS. For these two parameters, we adopt the same values as recommended in [32].

3.3 Computational results and comparisons

Table 2 summarizes the computational statistics of our HTS algorithm obtained on the set of 73 benchmark instances. Columns 1–2 present the characteristics of the tested graphs, including the name of the graph and the number of vertices in the graph. Columns LB and UB give the current best-known lower and upper bounds of the ECP reported in the literature, which were achieved by some very recent algorithms or CPLEX. Column 5 indicates the current best-known results (k_{pre}) reported in the literature. The results of our HTS algorithm with short time condition (3600 seconds per run for each instance) are presented in column 6 (k_1). Columns 7–13 present detailed computational results of our HTS algorithm under the long time condition: the smallest number of colors (k_{best}) for which our HTS algorithm obtains a k-eqcoloring over the 20 independent runs, the worst results (k_{worst}), the average results (k_{avg}), the standard deviation (k_{std}), the success rate (SR), the average computation time in seconds (t(s)) over the runs which found a k-eqcoloring, and the difference (Diff) in number of colors between our best results (column k_{best}) and the current best-known results ever reported in the literature (column k_{pre}) (a negative value indicates an improved result).

From Table 2, we observe that the results obtained by our HTS algorithm (column k_{best}) are highly competitive compared to the current best-known results reported in the literature (column k_{pre}). HTS improves the best-known solutions for 15 instances (new upper bounds) and matches the best-known results for 57 instances. Only for 1 instance (DSJC500.9), HTS reports a slightly worse result (using one more color) relative to the current best-known result. Especially, for several classic DIMACS instances (DSJC1000.5, flat1000_50_0, flat1000_60_0, flat1000_76_0 and C2000.5), which are frequently used to test graph coloring algorithms, HTS can significantly improve the best-known results by reducing the number of used colors by more than 6 units. Furthermore, for all 26 instances with their best lower bounds equaling their best upper bounds, HTS can reach their optimum solutions. Finally, for 57 out of the 73 instances, our HTS algorithm can attain its results with a standard deviation of 0 while for the remaining 16 instances, it achieves a very small standard deviation, which further confirms the robustness of the algorithm.

Table 2
Performance of the proposed HTS algorithm on the 73 benchmark instances. A value in bold indicates an improved result obtained by HTS compared to the previous best upper bound.

					HTS							
Instance	N	LB	UB	k_{pre}	k_1	k_{best}	k_{worst}	k_{avg}	k_{std}	SR	t(s)	Diff
DSJC125.1	125	5	5	5	5	5	5	5.00	0.00	20/20	15.73	0
DSJC125.5	125	9	17	17	17	17	17	17.00	0.00	20/20	563.44	0
DSJC125.9	125	43	44	44	44	44	44	44.00	0.00	20/20	0.30	0
DSJC250.1	250	5	8	8	8	8	8	8.00	0.00	20/20	426.50	0
DSJC250.5	250	12	29	29	29	29	29	29.00	0.00	20/20	4584.46	0
DSJC250.9	250	63	72	72	72	72	72	72.00	0.00	20/20	1835.05	0
DSJC500.1	500	5	13	13	13	13	13	13.00	0.00	20/20	147.54	0
DSJC500.5	500	13	56	56	52	**52**	52	52.00	0.00	20/20	2098.20	-4
DSJC500.9	500	101	128	128	130	129	130	129.65	0.48	7/20	8925.67	1
DSJR500.1	500	12	12	12	12	12	12	12.00	0.00	20/20	2.53	0
DSJR500.5	500	120	126	126	125	**125**	126	125.65	0.48	7/20	8179.27	-1
DSJC1000.1	1000	5	21	21	22	21	22	21.95	0.22	1/20	4809.68	0
DSJC1000.5	1000	15	101	101	97	**95**	97	95.90	0.70	6/20	13394.64	-6
DSJC1000.9	1000	126	252	252	251	**251**	251	251.00	0.00	20/20	3563.76	-1
R125.1	125	-	5	5	5	5	5	5.00	0.00	20/20	0.06	0
R125.5	125	-	36	36	36	36	36	36.00	0.00	20/20	3.41	0
R250.1	250	-	8	8	8	8	8	8.00	0.00	20/20	0.11	0
R250.5	250	-	66	66	66	**65**	66	65.90	0.32	2/20	9777.74	-1
R1000.1	1000	-	20	20	20	20	20	20.00	0.00	20/20	678.04	0
R1000.5	1000	-	250	250	255	**249**	250	249.10	0.30	19/20	17816.84	-1
le450_5a	450	5	5	5	5	5	5	5.00	0.00	20/20	332.37	0
le450_5b	450	5	5	5	5	5	5	5.00	0.00	20/20	363.68	0
le450_5c	450	-	5	5	5	5	5	5.00	0.00	20/20	143.03	0
le450_5d	450	5	5	5	5	5	5	5.00	0.00	20/20	373.07	0
le450_15a	450	15	15	15	15	15	15	15.00	0.00	20/20	17.18	0
le450_15b	450	15	15	15	15	15	15	15.00	0.00	20/20	15.95	0
le450_15c	450	-	15	15	15	15	15	15.00	0.00	20/20	95.72	0
le450_15d	450	15	15	15	15	15	15	15.00	0.00	20/20	58.14	0
le450_25a	450	25	25	25	25	25	25	25.00	0.00	20/20	4.76	0
le450_25b	450	25	25	25	25	25	25	25.00	0.00	20/20	7.26	0
le450_25c	450	-	26	26	26	26	26	26.00	0.00	20/20	11.75	0
le450_25d	450	25	26	26	26	26	26	26.00	0.00	20/20	6.29	0
wap01a	2368	41	42	42	42	42	42	42.00	0.00	20/20	700.41	0
wap02a	2464	40	41	41	41	41	41	41.00	0.00	20/20	2232.87	0
wap03a	4730	40	45	45	45	45	46	45.10	0.30	18/20	6150.34	0
wap04a	5231	-	44	44	45	44	45	44.40	0.49	12/20	12845.28	0
wap05a	905	-	50	50	50	50	50	50.00	0.00	20/20	0.18	0
wap06a	947	-	41	41	41	41	41	41.00	0.00	20/20	1206.07	0
wap07a	1809	-	42	42	42	42	43	42.40	0.49	12/20	2591.23	0
wap08a	1870	-	42	42	42	42	43	42.70	0.46	6/20	3801.03	0
flat300_28_0	300	11	34	34	33	**33**	34	33.00	0.71	20/20	3814.24	-1
flat1000_50_0	1000	-	101	101	97	**92**	95	93.30	0.64	1/20	18034.58	-9
flat1000_60_0	1000	-	101	101	100	**94**	96	94.70	0.64	8/20	16263.51	-7
flat1000_76_0	1000	14	102	102	97	**93**	95	93.90	0.62	5/20	14306.60	-9
latin_square_10	900	90	113	113	112	**107**	111	108.50	1.07	3/20	18054.74	-6
C2000.5	2000	-	201	201	201	**188**	193	190.35	1.24	1/20	19915.04	-13
C2000.9	2000	-	502	502	501	**501**	501	501.00	0.00	20/20	3952.28	-1
mulsol.i.1	197	49	49	49	49	49	49	49.00	0.00	20/20	0.53	0
mulsol.i.2	188	34	36	36	36	36	36	36.00	0.00	20/20	15.22	0
fpsol2.i.1	496	65	65	65	65	65	65	65.00	0.00	20/20	12.61	0
fpsol2.i.2	451	47	47	47	47	47	47	47.00	0.00	20/20	6.64	0
fpsol2.i.3	425	55	55	55	55	55	55	55.00	0.00	20/20	3.77	0
inithx.i.1	864	54	54	54	54	54	54	54.00	0.00	20/20	47.31	0
inithx.i.2	645	30	36	36	35	**35**	35	35.00	0.00	20/20	207.41	-1
inithx.i.3	621	-	37	37	36	**36**	36	36.00	0.00	20/20	5256.11	-1
zeroin.i.1	211	49	49	49	49	49	49	49.00	0.00	20/20	2.24	0
zeroin.i.2	211	36	36	36	36	36	36	36.00	0.00	20/20	0.70	0
zeroin.i.3	206	36	36	36	36	36	36	36.00	0.00	20/20	1.14	0
myciel6	95	7	7	7	7	7	7	7.00	0.00	20/20	0.01	0
myciel7	191	8	8	8	8	8	8	8.00	0.00	20/20	0.17	0
4-FullIns_3	114	7	7	7	7	7	7	7.00	0.00	20/20	0.01	0
4-FullIns_4	690	6	8	8	8	8	8	8.00	0.00	20/20	3.98	0
4-FullIns_5	4146	6	9	9	9	9	10	9.30	0.46	14/20	57.67	0
1-Insertions_6	607	3	7	7	7	7	7	7.00	0.00	20/20	0.38	0
2-Insertions_5	597	3	6	6	6	6	6	6.00	0.00	20/20	0.96	0
3-Insertions_5	1406	3	6	6	6	6	6	6.00	0.00	20/20	21.94	0
school1	385	15	15	15	15	15	15	15.00	0.00	20/20	0.72	0
school1_nsh	352	14	14	14	14	14	14	14.00	0.00	20/20	56.10	0
qg.order40	1600	40	40	40	40	40	40	40.00	0.00	20/20	4291.18	0
qg.order60	3600	60	60	60	60	60	60	60.00	0.00	20/20	64.34	0
ash331GPIA	662	4	4	4	4	4	4	4.00	0.00	20/20	5.39	0
ash608GPIA	1216	3	4	4	4	4	4	4.00	0.00	20/20	854.45	0
ash958GPIA	1916	3	4	4	4	4	4	4.00	0.00	20/20	11.95	0

Table 3
Comparative results with two heuristic approaches

Instance	Short time criterion			Long time criterion							
	TabuEqCol	BITS	HTS	BITS				HTS			
				k_{best}	k_{avg}	SR	t(s)	k_{best}	k_{avg}	SR	t(s)
DSJC125.1	5	5	5	5	5.00	20/20	0.96	5	5.00	20/20	15.73
DSJC125.5	18	17	17	17	17.50	10/20	5169.38	17	17.00	20/20	563.44
DSJC125.9	45	44	44	44	44.00	20/20	0.16	44	44.00	20/20	0.30
DSJC250.1	8	8	8	8	8.00	20/20	5.50	8	8.00	20/20	426.50
DSJC250.5	32	32	**29**	30	31.90	1/20	3265.63	**29**	29.00	20/20	4584.46
DSJC250.9	83	72	72	72	72.00	20/20	1179.92	72	72.00	20/20	1835.05
DSJC500.1	13	13	13	13	13.00	20/20	6.96	13	13.00	20/20	147.54
DSJC500.5	63	57	**52**	56	56.95	1/20	484.60	**52**	52.00	20/20	2098.20
DSJC500.9	182	130	130	129	129.90	2/20	3556.53	129	129.65	7/20	8925.67
DSJR500.1	12	12	12	12	12.00	20/20	0.58	12	12.00	20/20	2.53
DSJR500.5	133	126	**125**	126	126.30	14/20	3947.61	**125**	125.65	7/20	8179.27
DSJC1000.1	22	22	22	21	21.95	1/20	3605.49	21	21.95	1/20	4809.68
DSJC1000.5	112	112	**97**	103	105.10	3/20	18078.94	**95**	95.9	6/20	13394.64
DSJC1000.9	329	254	**251**	252	253.30	1/20	4064.65	**251**	251.00	20/20	3563.76
R125.1	-	5	5	5	5.00	20/20	0.01	5	5.00	20/20	0.06
R125.5	-	36	36	36	36.00	20/20	0.39	36	36.00	20/20	3.41
R250.1	-	8	8	8	8.00	20/20	0.01	8	8.00	20/20	0.11
R250.5	-	67	**66**	66	66.65	7/20	6275.08	**65**	65.90	2/20	9777.74
R1000.1	-	20	20	20	20.00	20/20	3.09	20	20.00	20/20	678.04
R1000.5	-	269	**255**	250	250.40	12/20	10723.29	**249**	249.10	19/20	17816.84
le450_5a	-	5	5	5	5.00	20/20	45.86	5	5.00	20/20	332.37
le450_5b	7	5	5	5	5.00	20/20	74.43	5	5.00	20/20	363.68
le450_5c	-	5	5	5	5.00	20/20	1877.73	5	5.00	20/20	143.03
le450_5d	8	5	5	5	5.00	20/20	2231.59	5	5.00	20/20	373.07
le450_15a	-	15	15	15	15.00	20/20	4.44	15	15.00	20/20	17.18
le450_15b	15	15	15	15	15.00	20/20	4.16	15	15.00	20/20	15.95
le450_15c	-	15	15	15	15.10	18/20	410.35	15	15.00	20/20	95.72
le450_15d	16	15	15	15	15.70	6/20	629.83	15	15.00	20/20	58.14
le450_25a	-	25	25	25	25.00	20/20	0.72	25	25.00	20/20	4.76
le450_25b	25	25	25	25	25.00	20/20	0.78	25	25.00	20/20	7.26
le450_25c	-	26	26	26	26.00	20/20	16.50	26	26.00	20/20	11.75
le450_25d	27	26	26	26	26.00	20/20	14.08	26	26.00	20/20	6.29
wap01a	46	43	**42**	42	42.60	8/20	4183.29	42	42.00	20/20	700.41
wap02a	44	42	**41**	41	41.80	4/20	6829.03	41	41.00	20/20	2232.87
wap03a	50	46	**45**	45	45.05	19/20	11267.27	45	45.10	18/20	6150.34
wap04a	-	46	**45**	44	44.15	17/20	11345.30	44	44.40	12/20	12845.28
wap05a	-	50	50	50	50.00	20/20	8.46	50	50.00	20/20	0.18
wap06a	-	42	**41**	41	41.70	6/20	6892.09	41	41.00	20/20	1206.07
wap07a	-	43	**42**	43	43.05	19/20	718.25	**42**	42.40	12/20	2591.23
wap08a	-	43	**42**	43	43.05	19/20	951.85	**42**	42.70	6/20	3801.03
flat300_28_0	36	35	**33**	34	34.70	6/20	4407.62	**33**	33.00	20/20	3814.24
flat1000_50_0	-	112	**97**	101	102.80	1/20	9206.28	**92**	93.30	1/20	18034.58
flat1000_60_0	-	112	**100**	102	102.90	5/20	10201.53	**94**	94.70	8/20	16263.51
flat1000_76_0	112	112	**97**	102	103.40	3/20	13063.39	**93**	93.90	5/20	14306.60
latin_square_10	130	129	**112**	115	120.00	1/20	17859.13	**107**	108.50	3/20	18054.74
C2000.5	-	202	**201**	201	201.65	7/20	4808.96	**188**	190.35	1/20	19915.04
C2000.9	-	504	**501**	502	502.45	11/20	7772.04	**501**	501.00	20/20	3952.28
mulsol.i.1	50	49	49	49	49.00	20/20	14.82	49	49.00	20/20	0.53
mulsol.i.2	48	36	36	36	36.35	13/20	3633.61	36	36.00	20/20	15.22
fpsol2.i.1	78	65	65	65	65.00	20/20	830.30	65	65.00	20/20	12.61
fpsol2.i.2	60	47	47	47	47.00	20/20	976.07	47	47.00	20/20	6.64
fpsol2.i.3	79	55	55	55	55.00	20/20	729.47	55	55.00	20/20	3.77
inithx.i.1	66	54	54	54	54.00	20/20	1468.27	54	54.00	20/20	47.31
inithx.i.2	93	36	**35**	36	36.35	13/20	12412.83	**35**	35.00	20/20	207.41
inithx.i.3	-	38	**36**	37	37.45	11/20	9214.61	**36**	36.00	20/20	5256.11
zeroin.i.1	51	49	49	49	49.00	20/20	1367.14	49	49.00	20/20	2.24
zeroin.i.2	51	36	36	36	36.00	20/20	96.99	36	36.00	20/20	0.70
zeroin.i.3	49	36	36	36	36.00	20/20	109.11	36	36.00	20/20	1.14
myciel6	7	7	7	7	7.00	20/20	0.00	7	7.00	20/20	0.01
myciel7	8	8	8	8	8.00	20/20	0.02	8	8.00	20/20	0.17
4-FullIns_3	-	7	7	7	7.00	20/20	0.00	7	7.00	20/20	0.01
4-FullIns_4	8	8	8	8	8.00	20/20	0.23	8	8.00	20/20	3.98
4-FullIns_5	9	9	9	9	9.00	20/20	54.49	9	9.30	14/20	57.67
1-Insertions_6	7	7	7	7	7.00	20/20	0.34	7	7.00	20/20	0.38
2-Insertions_5	6	6	6	6	6.00	20/20	0.11	6	6.00	20/20	0.96
3-Insertions_5	6	6	6	6	6.00	20/20	0.57	6	6.00	20/20	21.94
school1	15	15	15	15	15.00	20/20	1.30	15	15.00	20/20	0.72
school1_nsh	14	14	14	14	14.00	20/20	2.63	14	14.00	20/20	56.10
qg.order40	40	40	40	40	40.00	20/20	4.73	40	40.00	20/20	4291.18
qg.order60	60	60	60	60	60.00	20/20	21.57	60	60.00	20/20	64.34
ash331GPIA	4	4	4	4	4.00	20/20	2.02	4	4.00	20/20	5.39
ash608GPIA	4	4	4	4	4.00	20/20	0.50	4	4.00	20/20	854.45
ash958GPIA	4	4	4	4	4.00	20/20	23.31	4	4.00	20/20	11.95

3.4 Comparison with state-of-the-art algorithms

In order to show the relative effectiveness of the proposed algorithm for the ECP, we further compare the algorithm with the two most recent state-of-

art algorithms from the literature: TabuEqCol [37] and BITS [32]. The main comparison criterion is the quality of the solutions found, i.e., the number of used colors for which an algorithm obtains a k-eqcoloring.

Table 3 summarizes the computational results of the proposed algorithm in comparison with the two reference algorithms under both the short time criterion and the long time criterion. The experimental platforms used by the reference algorithms are as follows.

- TabuEqCol was run on an Intel i5 CPU 750@2.67Ghz with Ubuntu Linux O.S. and Intel C++ Compiler. In [37], TabuEqCol only reported its results under the short time criterion.
- BITS was run on a Xeon E5440 with 2.83 GHz with 2 GB RAM and reported its results under both the short time criterion and the long time criterion. Note that both BITS and our HTS algorithm were run on the same computing platform.

When comparing these three algorithms under the short time criterion, we observe that our HTS algorithm competes very favorably with the TabuEqCol algorithm, indeed, for the 50 graphs where TabuEqCol reported its results, our HTS algorithm is able to deliver a better result for 29 instances, while achieving the same result for the remaining 21 instances. When comparing HTS with BITS, we can also observe that HTS is highly competitive with BITS under the short time criterion. Over all the 73 instances tested, the quality of solutions obtained by HTS matches or exceeds that of solutions obtained by BITS. In addition, HTS obtains solutions better than those found by BITS in 23 cases. When applying the statistical Wilcoxon test with a significance level of 0.05 for pairwise comparisons, the resulting p-values of $2.629e$-6 for HTS v.s. TabuEqCol, and $2.25e$-5 for HTS v.s. BITS show a clear dominance of HTS over the reference algorithms.

When comparing with the results of BITS under the long time criterion, one observes that the results are once again in favor of our HTS algorithm. For 18 out of the 73 instances, HTS finds better solutions than BITS. For the remaining 55 instances, both algorithms achieve the same minimal k_{best} value. Furthermore, for the 55 instances with the same minimal k_{best} value, our HTS algorithm is able to reach better average results than BITS for 8 instances while the reverse is true only for 3 instances. The comparative results between HTS and BITS also comply with the no free lunch (NFL) theorem [48] which states that no algorithm will always be the most effective considering all the possible problems. Finally, the Wilcoxon test outcomes indicate that the differences between HTS and BITS are statistically significant (p-value=0.000142) in terms of the best results achieved. Thus we conclude that our HTS algorithm dominates BITS.

To complement these comparisons, we also provide the convergence graphs to observe the running behavior of these three comparative algorithms. The convergence graph is given by the function: $i \rightarrow k$ where i is the number of iterations and k is the minimal number of colors achieved at iteration i. The convergence graph is a natural way to observe the evolution of the objective value during the search process. In Fig. 4 we show the convergence graphs of TabuEqCol, BITS and HTS on four selected instances. From Fig. 4, we first observe that HTS always attains the best performance by reaching a smaller k. Even if the minimal number of colors of these three algorithms decreases sharply at the beginning, TabuEqCol and BITS soon get trapped in local optima as they converge early for these four tested problem instances. On the other hand, HTS is able to continue its search and find improved solutions with the help of its feasible and infeasible local searches.

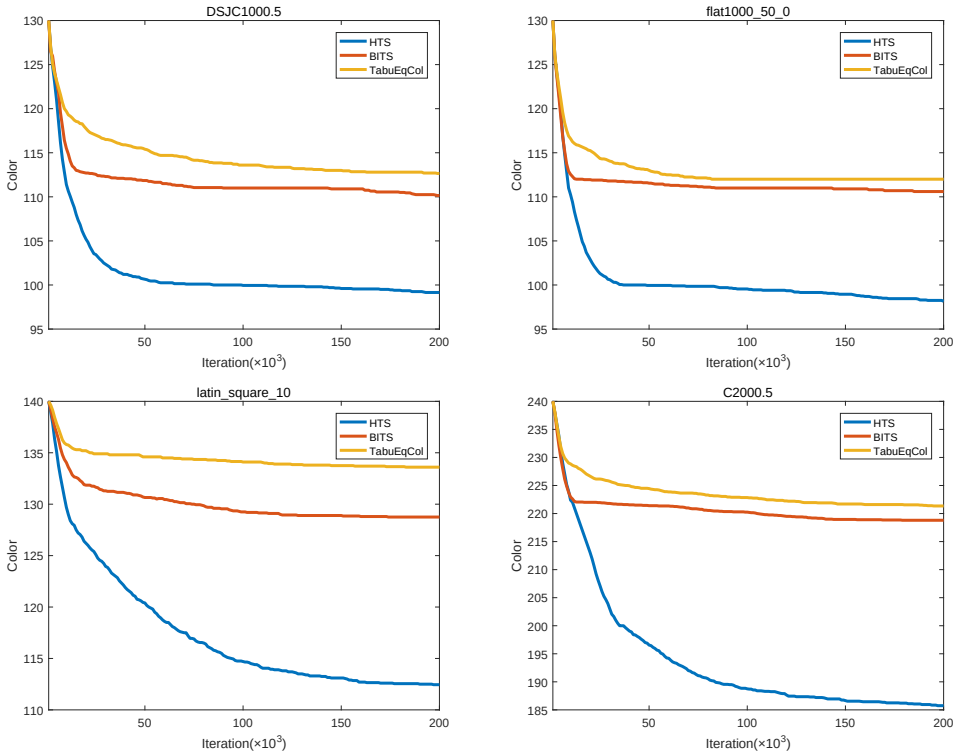

Fig. 4. Convergence graphs of TabuEqCol, BITS and HTS on four instances.

4 Analysis of HTS

In this section, we carry out additional analyses to gain a deeper understanding of the underlying mechanisms of the proposed algorithm.

Table 4
The influence of the constrained cyclic exchange neighborhood on 21 representative instances

Instance	FLS+CCEN						FLS-CCEN					
	k_{best}	k_{worst}	k_{avg}	k_{std}	SR	t(s)	k_{best}	k_{worst}	k_{avg}	k_{std}	SR	t(s)
DSJC500.5	52	52	52.00	0.00	20/20	1464.92	57	57	57.00	0.00	20/20	33.87
DSJR500.5	125	126	125.85	0.36	3/20	4407.02	126	127	126.95	0.22	1/20	108.64
DSJC1000.5	98	101	100.10	1.09	1/20	17967.18	112	112	112.00	0.00	20/20	27.70
DSJC1000.9	251	252	251.05	0.22	19/20	7371.64	254	255	254.35	0.48	13/20	333.24
R250.5	66	66	66.00	0.00	20/20	288.00	66	67	66.25	0.43	15/20	28.83
R1000.5	249	250	249.35	0.48	13/20	15019.77	252	252	252.00	0.00	20/20	210.40
le450_5a	5	7	6.20	0.98	8/20	1250.98	5	5	5.00	0.00	20/20	3.18
le450_5b	5	7	6.10	0.99	9/20	3418.52	5	5	5.00	0.00	20/20	79.97
le450_5c	7	7	7.00	0.00	20/20	61.31	7	7	7.00	0.00	20/20	0.48
le450_5d	7	7	7.00	0.00	20/20	59.00	7	7	7.00	0.00	20/20	1.12
le450_15a	15	15	15.00	0.00	20/20	25.81	15	15	15.00	0.00	20/20	18.18
le450_15b	15	15	15.00	0.00	20/20	29.75	15	15	15.00	0.00	20/20	12.70
le450_15c	15	15	15.00	0.00	20/20	369.59	15	15	15.00	0.00	20/20	274.32
le450_15d	15	16	15.75	0.43	5/20	588.27	15	16	15.25	0.43	15/20	606.97
flat300_28_0	33	34	33.95	0.22	1/20	1348.58	35	36	35.15	0.36	17/20	188.73
flat1000_50_0	93	94	93.85	0.36	3/20	12449.40	112	112	112.00	0.00	20/20	5.88
flat1000_60_0	94	96	94.70	0.71	9/20	11309.91	112	112	112.00	0.00	20/20	2.69
flat1000_76_0	94	96	94.40	0.58	13/20	10254.05	112	112	112.00	0.00	20/20	6.48
latin_square_10	109	113	111.05	1.12	1/20	14768.21	130	130	130.00	0.00	20/20	84.44
C2000.5	189	195	191.3	1.42	2/20	19428.78	202	202	202.00	0.00	20/20	154.40
C2000.9	501	501	501.00	0.00	20/20	2054.21	503	504	503.75	0.43	5/20	949.54

4.1 Effectiveness of the cyclic exchange neighborhood

The neighborhood is a critical element that affects the efficacy of a local search procedure. Our HTS algorithm relies on three dedicated neighborhoods which are induced by the directional-one-move, two-exchange and constrained-three-cyclic-exchange operators while previous algorithms like TabuEqCol and BITS only employ the directional-one-move and two-exchange operators to generate neighbor solutions. In this section, we investigate the critical role played by the constrained-three-cyclic-exchange neighborhood in boosting the overall performance of our algorithm. For this purpose, we compared two neighborhood exploring strategies, the first one jointly uses these three dedicated neighborhoods while the second strategy excludes the constrained-three-cyclic-exchange neighborhood. These two strategies were tested under the framework of our feasible local search procedure in order to highlight the role of the neighborhood. We summarize in Table 4 the comparative results between the two versions of our FLS procedure with and without the constrained-three-cyclic-exchange neighborhood (denoted by FLS+CCEN and FLS-CCEN respectively). We ran both algorithms under the long time stop criterion on a set of 21 representative instances, which were frequently used in the literature to test graph coloring algorithms.

Table 4 shows that removing the constrained-three-cyclic-exchange neighborhood significantly sacrifices the search power of FLS. Specifically, FLS+CCEN outperforms FLS-CCEN for 12 out of the 21 tested instances in terms of the best results, and matches FLS-CCEN for the remaining 9 instances. Especially, for the 9 instances with more than 900 vertices, FLS+CCEN can significantly improve on the results of FLS-CCEN by largely reducing the number of used colors. In terms of the average results and standard deviations, even if FLS-CCEN shows slightly better standard deviations, FLS+CCEN has a better average value than FLS-CCEN for 13 out of the 21 tested instances while the

reverse is true only for 4 instances. In addition, the Wilcoxon test outcomes in terms of the best solution values reveal a significant difference with the p-value = 0.002497 between these two versions of our FLS procedure, further confirming our conclusion that the cyclic exchange neighborhood makes a significant contribution to the overall performance of the algorithm.

4.2 Effectiveness of the combined use of feasible and infeasible local searches

Compared with previous approaches like TabuEqCol [37] and BITS [32] where the equity constraint is strictly maintained during the search process, one salient feature of our approach is its hybrid scheme combining feasible and infeasible local searches. To evaluate the merit of this hybrid scheme, we compare the performance of our HTS algorithm with its two underlying local search components, i.e., the feasible local search procedure and the infeasible local search procedure.

Table 5 summarizes the computational results of these 3 algorithms under the long time criterion on the 21 selected instances. From Table 5, it can be observed that HTS shows an overall better performance than its two underlying FLS and ILS local search procedures. Indeed, when comparing HTS against FLS, we notice that they reach the same minimal k value for 13 graphs. For the other 8 graphs, HTS finds better solutions than FLS. When it comes to comparing HTS and ILS, the results are once again in favor of HTS because HTS improves on the results of ILS for 17 out of 21 cases. Moreover, HTS usually achieves a significant better average value than FLS and ILS even if these tree algorithms have a comparable standard deviations.

When applying the Wilcoxon test to check the statistical differences between HTS and its two underlying algorithms FLS and ILS in terms of the best solution values, we obtain p-values of 0.01264 for HTS v.s. FLS, and 0.000316 for HTS v.s ILS, indicating that the differences between HTS and its two underlying local search methods are statistically significant. The above observation indicates that HTS reaches a better performance than its two underlying local search components which confirms the usefulness of the hybrid scheme integrating both the feasible and infeasible local searches into the search procedure. Finally, we also notice that even if FLS and ILS perform worse than HTS, their results remain competitive with respect to the other two reference approaches TabuEqCol [37] and BITS [32].

Table 5. Comparison of results among HTS, FLS and ILS

Instance	HTS						FLS						ILS					
	k_{best}	k_{worst}	k_{avg}	k_{std}	SR	t(s)	k_{best}	k_{worst}	k_{avg}	k_{std}	SR	t(s)	k_{best}	k_{worst}	k_{avg}	k_{std}	SR	t(s)
DSJC500.5	52	52	52.00	0.00	20/20	2098.20	52	52	52.00	0.00	20/20	1464.92	57	57	57.00	0.00	20/20	4.81
DSJR500.5	125	126	125.65	0.48	7/20	8179.27	125	126	125.85	0.36	3/20	4407.02	127	128	127.20	0.40	16/20	151.03
DSJC1000.5	95	97	95.90	0.70	6/20	13394.64	98	101	100.10	1.09	1/20	17967.18	104	105	104.05	0.22	19/20	101.71
DSJC1000.9	251	251	251.00	0.00	20/20	3563.76	251	252	251.05	0.22	19/20	7371.64	260	264	261.85	0.91	1/20	58.57
R250.5	65	66	65.90	0.32	2/20	9777.74	66	66	66.00	0.00	20/20	288.00	66	66	66.00	0.00	20/20	9.04
R1000.5	249	250	249.10	0.30	19/20	17816.84	249	250	249.35	0.48	13/20	15019.77	251	253	252.35	0.57	1/20	1162.09
le450_5a	5	5	5.00	0.00	20/20	332.37	5	7	6.20	0.98	8/20	1250.98	5	5	5.00	0.00	20/20	0.09
le450_5b	5	5	5.00	0.00	20/20	363.68	5	7	6.10	0.99	9/20	3418.52	5	5	5.00	0.00	20/20	0.10
le450_5c	5	5	5.00	0.00	20/20	143.03	7	7	7.00	0.00	20/20	61.31	5	5	5.00	0.00	20/20	0.16
le450_5d	5	5	5.00	0.00	20/20	373.07	7	7	7.00	0.00	20/20	59.00	5	5	5.00	0.00	20/20	0.14
le450_15a	15	15	15.00	0.00	20/20	17.18	15	15	15.00	0.00	20/20	25.81	16	16	16.00	0.00	20/20	0.20
le450_15b	15	15	15.00	0.00	20/20	15.95	15	15	15.00	0.00	20/20	29.75	16	16	16.00	0.00	20/20	0.12
le450_15c	15	15	15.00	0.00	20/20	95.72	15	15	15.00	0.00	20/20	369.59	22	23	22.60	0.49	8/20	0.26
le450_15d	15	15	15.00	0.00	20/20	58.14	15	16	15.75	0.43	5/20	588.27	22	23	22.40	0.49	12/20	0.23
flat300_28_0	33	34	33.00	0.71	20/20	3814.24	33	34	33.95	0.22	1/20	1348.58	35	35	35.00	0.00	20/20	0.45
flat1000_50_0	92	94	93.30	0.64	1/20	18034.58	93	94	93.85	0.36	3/20	12449.40	102	104	103.10	0.44	1/20	75.46
flat1000_60_0	94	96	94.70	0.64	8/20	16263.51	94	96	94.70	0.71	9/20	11309.91	103	104	103.30	0.46	14/20	68.43
flat1000_76_0	93	95	93.90	0.62	5/20	14306.60	94	96	94.40	0.58	13/20	10254.05	103	104	103.55	0.50	9/20	82.75
latin_square_10	107	111	108.50	1.07	3/20	18054.74	109	113	111.05	1.12	1/20	14768.21	118	130	121.20	3.46	1/20	91.57
C2000.5	188	193	190.35	1.24	1/20	19915.04	189	195	191.30	1.42	2/20	19428.78	201	203	202.05	0.38	1/20	208.15
C2000.9	501	501	501.00	0.00	20/20	3952.28	501	501	501.00	0.00	20/20	2054.21	511	517	513.80	1.83	3/20	55.86

Table 6
Comparison of HTS and PSO

Instance	HTS						PSO					
	k_{best}	k_{worst}	k_{avg}	k_{std}	SR	t(s)	k_{best}	k_{worst}	k_{avg}	k_{std}	SR	t(s)
DSJC500.5	52	52	52.00	0.00	20/20	2098.20	84	84	84.00	0.00	20/20	32.13
DSJR500.5	125	126	125.65	0.48	7/20	8179.27	141	141	141.00	0.00	20/20	652.06
DSJC1000.5	95	97	95.90	0.70	6/20	13394.64	127	127	127.00	0.00	20/20	6168.33
DSJC1000.9	251	251	251.00	0.00	20/20	3563.76	337	338	337.70	0.46	6/20	12651.42
R250.5	65	66	65.90	0.32	2/20	9777.74	86	87	86.70	0.46	6/20	8717.34
R1000.5	249	250	249.10	0.30	19/20	17816.84	283	284	283.3	0.46	14/20	9802.05
le450_5a	5	5	5.00	0.00	20/20	332.37	16	16	16.00	0.00	20/20	2324.66
le450_5b	5	5	5.00	0.00	20/20	363.68	16	16	16.00	0.00	20/20	3832.29
le450_5c	5	5	5.00	0.00	20/20	143.03	19	19	19.00	0.00	20/20	4942.30
le450_5d	5	5	5.00	0.00	20/20	373.07	19	26	20.40	2.80	16/20	5364.10
le450_15a	15	15	15.00	0.00	20/20	17.18	31	31	31.00	0.00	20/20	1119.89
le450_15b	15	15	15.00	0.00	20/20	15.95	31	31	31.00	0.00	20/20	188.40
le450_15c	15	15	15.00	0.00	20/20	95.72	35	35	35.00	0.00	20/20	4.90
le450_15d	15	15	15.00	0.00	20/20	58.14	35	35	35.00	0.00	20/20	10.97
flat300_28_0	33	34	33.00	0.71	20/20	3814.24	51	51	51.00	0.00	20/20	17.75
flat1000_50_0	92	94	93.30	0.64	1/20	18034.58	127	127	127.00	0.00	20/20	4266.42
flat1000_60_0	94	96	94.70	0.64	8/20	16263.51	127	127	127.00	0.00	20/20	4177.69
flat1000_76_0	93	95	93.90	0.62	5/20	14306.60	127	127	127.00	0.00	20/20	5319.39
latin_square_10	107	111	108.50	1.07	3/20	18054.74	183	183	183.00	0.00	20/20	11781.20
C2000.5	188	193	190.35	1.24	1/20	19915.04	287	287	287.00	0.00	20/20	1668.90
C2000.9	501	501	501.00	0.00	20/20	3952.28	514	514	514.00	0.00	20/20	5932.53

4.3 HTS v.s. particle swarm optimization

Recently, particle swarm optimization (PSO) has been successfully applied to solve various optimization problems (see e.g., [13,14,46,1,23,25]). One would wonder whether this approach could be effective for the ECP. So far we are unaware of any PSO algorithm proposed for the ECP in the literature. On the other hand, for the classic GCP and its variants, though various heuristics have been proposed, the most efficient approaches are almost based on local searches or hybrid algorithms that integrate local search within the memetic search framework [19,34,43]. Even if PSO is very popular, this approach is rarely applied to the GCP or its variants. To get some ideas about the capacity of PSO for solving the ECP, we adapted the PSO algorithm for the GCP presented in [2] to the ECP and tested it with the runtime condition specified in Section 3.1 on the 21 selected instances. The comparative results between HTS and PSO shown in Table 6 indicate a clear dominance of HTS over PSO on these instances. Indeed, for each tested instance, HTS performs significantly better than PSO in terms of best, worst and average values.

5 Conclusion and future research

The equitable coloring problem (ECP) is a useful model in practice, but represents a real computational challenge. Given the highly constrained feature of the ECP imposed by the equity constraint, we devised an effective tabu search method that integrates both feasible and infeasible local searches into a same search procedure. In order to reinforce the search ability of the proposed algorithm, we proposed a new scheme considering changes involving three color classes. Given that the new scheme is computationally expensive, we devised a constrained neighborhood strategy to exclude unpromising moves.

Computational assessments on 73 benchmark instances revealed that the proposed method competes favorably with two recent state-of-the-art approaches in the literature. For all 73 benchmark instances except one case, our algorithm is capable of matching or improving the best-known upper bounds. In particular, the proposed algorithm established 15 improved upper bounds which can serve as new references for evaluating new ECP algorithms.

We also compared HTS with its two underlying local search components to show the interest of the hybrid scheme of integrating both the feasible and infeasible local searches into a same search procedure. We carried additional experiments to illustrate the effectiveness of the proposed new neighborhood, which proves to be critical to the performance of the HTS algorithm.

For future work, we can consider several research directions. First, we can further improve the proposed HTS algorithm by investigating other types of neighborhoods and exploring other ways of combing the proposed neighborhoods within the framework of tabu search. Second, population based memetic algorithms are among the most powerful approaches for the classical graph coloring problem [18,34,43]. As a result, it would be interesting to investigate this approach for solving the ECP, by using the HTS algorithm as the key local search procedure of a memetic algorithm.

Finally, it is worth exploring the possibilities of using the proposed algorithm to solve practical problems related to the ECP. For instance, Furmańczyk and Kubale [17] mentions a university timetabling problem, namely assigning university courses to time slots in a way that avoids scheduling incompatible pairs of courses at the same time and spreads the courses evenly among the available time slots. This problem can be formulated as the equitable coloring problem, where each course is associated with a vertex, and there is an edge between each pair of courses that cannot be scheduled at the same time. In particular, courses that share a common teacher or a common classroom (or both) are linked by an edge. A feasible equitable coloring of the resulting graph corresponds then to a timetable. Given the effectiveness of the proposed HTS algorithm for the ECP, it can be applied to deal with such a problem.

Acknowledgements

We are grateful to the anonymous referees for valuable suggestions and comments which helped us improve the paper. This work is partially supported by the National Natural Science Foundation Program of China [Grant No. 71771099,71401059,71620107002,71531009], the Fundamental Research Funds for the Central Universities, HUST (Grant no. 2016AC055), and the fund from Huazhong University of Science and Technology (5001300001).

References

[1] Al-Dunainawi Y., Abbod M.F., Jizany A., A new MIMO ANFIS-PSO based NARMA-L2 controller for nonlinear dynamic systems, Engineering Applications of Artificial Intelligence, 2017, 62:265–275.

[2] Anh T.H., Giang T.T.T., Vinh T.L., A Novel Particle Swarm Optimization-Based Algorithm for the Graph Coloring Problem, International Conference on Information Engineering and Computer Science (ICIECS), 2009. DOI: 10.1109/ICIECS.2009.5365201

[3] Aringhieri R., Duma D., Grosso A., Hosteins P., Simple but effective heuristics for the 2-constraint bin packing problem, Accepted to Journal of Heuristics, DOI:10.1007/s10732-017-9326-0.

[4] Bahiense L., Frota Y., Noronha T.F., Ribeiro C.C., A branch-and-cut algorithm for the equitable coloring problem using a formulation by representatives, Discrete Applied Mathematics, 2014, 164(5):34–46.

[5] Bodlaender H.L., Fomin F.V., Equitable colorings of bounded treewidth graphs, Theoretical Computer Science, 2005, 349(1):22–30.

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

[7] Brown J.R., Chromatic scheduling and the chromatic number problem, Management Science, 1972, 19(4-part-1):456–463.

[8] Campêlo M., Corrêa R.C., Campos V.A., On the asymmetric representatives formulation for the vertex coloring problem, Discrete Applied Mathematics, 2008, 156(7):1097–1111.

[9] Chen Y.N., Hao J.K., Glover F., An evolutionary path relinking approach for the quadratic multiple knapsack problem, Knowledge-based Systems, 2016, 92:23–34.

[10] Chen Y.N., Hao J.K., Glover F., A hybrid metaheuristic approach for the capacitated arc routing problem, European Journal of Operational Research, 2016, 253(1): 25–39.

[11] Chen B.L., Lih K.W., Equitable coloring of trees, Journal of Combinatorial Theory, Series B, 1994, 61(1):83–87.

[12] Chen B.L., Lih K.W., Wu P.L., Equitable coloring and the maximum degree, European Journal Combinatorics, 1994, 15(5):443–447.

[13] De A., Mamanduru V.K.R., Gunasekaran A., Subramanian N., Tiwari M.K., Composite particle algorithm for sustainable integrated dynamic ship routing and scheduling optimization, Computers & Industrial Engineering, 2016, 96: 201–215.

[14] De A., Kumar S.K., Gunasekaran A., Tiwari M.K., Sustainable maritime inventory routing problem with time window constraints, Engineering Applications of Artificial Intelligence, 2017, 61:77–95.

[15] Dorne R., Hao J.K., Tabu search for graph coloring, T-coloring and set T-colorings, in Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, S. Voss, S. Martello, I.H. Osman and C. Roucairol (Eds.), Kluwer Academic Publishers, 1999, Chapter 6, pp. 77–92.

[16] Furmańczyk H., Kubale M., Equitable coloring of graphs, Contemporary Mathematics, American Mathematical Society, 2004, 352:35–54.

[17] Furmańczyk H., Kubale M., The complexity of equitable vertex coloring of graphs, Journal of Applied Computer Science, 2005, 13(2):95–106.

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

[19] Galinier P., Hertz A., A survey of local search methods for graph coloring, Computers & Operations Research, 2006, 33(9): 2547–2562.

[20] Glover F., Taillard E., A user's guide to tabu search, Annals of Operations Research, 1993, 41(1):1–28

[21] Glover F., Kochenberger G.A., (Eds.), Handbook of metaheuristics, Springer, 2006.

[22] Glover F., Hao J.K., The case for strategic oscillation, Annals of Operations Research, 2011, 183(1):163–173.

[23] Haddar B., Khemakhem M., Hanafi S., Wilbaut C., A hybrid quantum particle swarm optimization for the Multidimensional Knapsack Problem, Engineering Applications of Artificial Intelligence, 2016, 55:1–13.

[24] Hajnal A., Szemerédi E., Proof of a conjecture of P. Erdös, Erdös P., Rényi A.(Herausgeber):Combinatorial Theory and its Application, 1970, 4:601–623.

[25] Hein D., Hentschel A., Runkler T., Udluft S., Particle swarm optimization for generating interpretable fuzzy reinforcement learning policies, Engineering Applications of Artificial Intelligence, 2017, 65: 87–98.

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

[27] Irani S., Leung V., Scheduling with conflicts and applications to traffic signal control, SODA, 1996, 96:85–94.

[28] Jin Y., Hao J.K, Hamiez J.P., A memetic algorithm for the minimum sum coloring problem, Computers & Operations Research, 2014, 43(3): 318–327.

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

[30] Kostochka A.V., Equitable colorings of outerplanar graphs, Discrete Mathematics, 2002, 258(1):373–377.

[31] Kostochka A.V., Nakprasit K., On equitable Δ-coloring of graphs with low average degree, Theoretical Computer Science , 2005, 349(1):82–91.

[32] Lai X., Hao J.K., Glover F., Backtracking based iterated tabu search for equitable coloring, Engineering Applications of Artificial Intelligence, 2015, 46:269–278.

[33] Lih K.W., Wu P.L., On equitable coloring of bipartite graphs, Discrete Mathematics, 1996, 151(1):155–160.

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

[35] Méndez-Díaz I., Zabala P., A branch-and-cut algorithm for graph coloring, Discrete Applied Mathematics, 2006, 154(5):826–847.

[36] Méndez-Díaz I., Nasini G., Severín D., A polyhedral approach for the equitable coloring problem, Discrete Applied Mathematics, 2014, 164:413–426.

[37] Méndez-Díaz I., Nasini G., Severín D., A tabu search heuristic for the equitable coloring problem, International Symposium on Combinatorial Optimization, 2014, 347–358.

[38] Méndez-Díaz I., Nasini G., Severín D., A DSATUR-based algorithm for the equitable coloring problem, Computers & Operations Research, 2015, 57:41–50.

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

[40] Mezura-Montes E., Coello C.A.C., Useful infeasible solutions in engineering optimization with evolutionary algorithms, Mexican International Conference on Artificial Intelligence, 2005, 652–662.

[41] Moeini R., Soltani-nezhad M., Daei M., Constrained gravitational search algorithm for large scale reservoir operation optimization problem, Engineering Applications of Artificial Intelligence, 2017, 62:222–233.

[42] Qin J., Xu X., Wu Q., Cheng T.C.E., Hybridization of tabu search with feasible and infeasible local searches for the quadratic multiple knapsack problem, Computers & Operations Research, 2016, 66:199–214.

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

[44] San-Segundo P., A new DSATUR-based algorithm for exact vertex coloring, Computers & Operations Research, 2012, 39(7):1724–1733.

[45] Sewell E.C., An improved algorithm for exact graph coloring, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1996, 26:359–373.

[46] Tsai H.C., Unified particle swarm delivers high efficiency to particle swarm optimization, Applied Soft Computing, 2017, 55: 371–383.

[47] Tucker A., Perfect graphs and an application to optimizing municipal services, Siam Review, 1973, 15(3):585–590.

[48] Wolpert D.H., Macready W.G., No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation, 1997,1(1): 67–82.

[49] Yap H.P., Zhang Y., The equitable Δ-coloring conjecture holds for outerplanar graphs, Bulletin of the Institute of Mathematics Academia Sinica, 1997, 25:143–149.