Engineering Applications of Artificial Intelligence 52 (2016) 81–91

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

A learning-based path relinking algorithm for the bandwidth coloring problem

Xiangjing Lai [a], Jin-Kao Hao [a,b,*], Zhipeng Lü [c], Fred Glover [d]

[a] LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France
[b] Institut Universitaire de France, Paris, France
[c] SMART, School of Computer Science and Technology, Huazhong University of Science and Technology, 430074 Wuhan, PR China
[d] Engineering & Science, University of Colorado, Boulder, CO 80309, USA

ARTICLE INFO

Article history:
Received 6 October 2015
Received in revised form
21 January 2016
Accepted 10 February 2016
Available online 3 March 2016

Keywords:
Path relinking and tabu search
Learning mechanism
Bandwidth and graph coloring problems
Population-based computing

ABSTRACT

This paper proposes a learning-based path relinking algorithm (LPR) for solving the bandwidth coloring problem and the bandwidth multicoloring problem. Based on the population path-relinking framework, the proposed algorithm integrates a learning-driven tabu optimization procedure and a path-relinking operator. LPR is assessed on two sets of 66 common benchmark instances, and achieves highly competitive results in terms of both solution quality and computational efficiency compared to the state-of-the-art algorithms in the literature. Specifically, the algorithm establishes 7 new upper bounds while matching the best known results for 56 cases. The impacts of the learning mechanism and the path relinking operators are investigated, confirming their critical role to the success of the proposed algorithm.

1. Introduction

Given an undirected graph $G=(V,E)$ with a vertex set $V=\{v_1, v_2, \dots, v_n\}$ and an edge set $E \subset V \times V$, a bandwidth k-coloring is an assignment of a color $c(v_i)$ $(1 \le c(v_i) \le k)$ to each vertex v_i $(1 \le i \le |V|)$ such that for each edge $e(v_i, v_j) \in E$ the difference between the colors of vertices v_i and v_j must be larger than or equal to the associated edge weight $d(i,j)$, i.e., $|c(v_i)-c(v_j)| \ge d(i,j)$. The bandwidth coloring problem (BCP) is to find the smallest number k of colors such that a bandwidth k-coloring exists. It is obvious that BCP is a generalization of the classical vertex coloring problem when $d(i,j)=1$ for all the edges of the graph. On the other hand, the bandwidth multicoloring problem (BMCP) is a generalization of BCP, where each vertex v_i is assigned $y(i) \ge 1$ colors. In addition to the distance constraint between two colors assigned to two adjacent vertices (i.e., $|c(v_i)-c(v_j)| \ge d(i,j)$), two colors $c(v_i)$ and $c'(v_i)$ assigned to the same vertex v_i must be at least distanced by $d(i,i)$, i.e., $|c(v_i)-c'(v_i)| \ge d(i,i)$. Like BCP, the objective of BMCP is to minimize the number k of the colors. BCP and BMCP correspond to special cases of the T-coloring and set T-coloring problems (Dorne and Hao, 1998; Hale, 1980) that are particularly relevant to model frequency assignment problems in telecommunications networks.

BMCP can be converted into BCP by splitting each vertex v_i into a clique with $y(i)$ vertices, where each edge of the clique has an edge weight $d(i,i)$ of the original graph (see Dorne and Hao, 1998 for an example) (Lim et al., 2005; Malaguti and Toth, 2008). The resulting new graph has $\sum_{i=1}^{n} y(i)$ vertices. Thus, any approach for BCP can be applied to BMCP. For this reason, we focus on solving BCP in this paper. Notice that generalized graph coloring problems are attracting increasing attention in the last decades, as exemplified by the series of COLOR02/03/04 Challenges dedicated to Graph Coloring and its Generalizations (http://mat.gsia.cmu.edu/COLOR03/).

A large number of solution approaches exist in the literature for these generalized coloring problems. In 1993, Costa applied simulated anneaning and tabu search to the T-coloring problem (Costa, 1993) while in 1998, Dorne and Hao introduced a general tabu search algorithm for both T-coloring and set T-coloring (Dorne and Hao, 1998). In 2002, Phan and Skiena proposed a general heuristic called Discropt for BCP (Phan and Skiena, 2002). In 2002, Prestwich presented a hybrid approach combining local search and constraint programming for BCP and BMCP which was extended in Prestwich (2008). In 2005, Lim et al. introduced a hybrid algorithm for BCP and BMCP (Lim et al., 2005). In 2007, Chiarandini et al. investigated several stochastic local search algorithms for the set T-coloring problem (Chiarandini and Stützle, 2007). In 2008, Malaguti and Toth reported an effective

* Corresponding author at: LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France.
E-mail addresses: laixiangjing@gmail.com (X. Lai), hao@info.univ-angers.fr (J.-K. Hao), zhipeng.lui@gmail.com (Z. Lü), glover@opttek.com (F. Glover).

http://dx.doi.org/10.1016/j.engappai.2016.02.008
0952-1976/© 2016 Elsevier Ltd. All rights reserved.

evolutionary approach for BCP and BMCP (Malaguti and Toth, 2008). In 2010, Martí et al. developed several heuristics for BCP using memory structures in both constructive and improvement methods (Martí et al., 2010). In 2013, Lai and Lü developed a multistart iterated tabu search algorithm for BCP (Lai and Lü, 2013). In 2015, Jin and Hao proposed an effective learning-based hybrid search approach for BCP, and reported improved results for a number of benchmarks (Jin and Hao, 2015). Very recently, Matić et al. (2015) developed a variable neighborhood search (VNS) method for BCP, and reported improved results for two instances. Finally, various heuristic algorithms were also proposed in the literature to solve the tightly related frequency assignment problem (Hao et al., 1998; Lai and Hao, 2015; Montemanni and Smith, 2010; Smith et al., 1998).

The general path relinking (PR) framework (Glover, 1997; Glover et al., 2000) has recently shown outstanding performances in solving a number of difficult combinatorial optimization problems, such as capacitated clustering (Deng and Bard, 2011), the far from most string problem (Gallardo and Cotta, 2015), vehicle routing problem with route balancing (Lacomme et al., 2015), multi-depot periodic vehicle routing (Rahimi-Vahed et al., 2013), and unconstrained binary quadratic optimization (Wang et al., 2012). In this paper, we introduce an effective path relinking algorithm for solving BCP which relies on both a solution relinking procedure and a local search procedure.

The following identifies the main contributions of the present work:

- This paper presents a path relinking approach for the BCP and BMCP problems. The proposed LPR approach introduces a new learning-based penalty function to provide an informative guidance to the search procedure. LPR explores two path relinking operators for new solution generation and uses a two-phase Tabu Search procedure for local improvement which relies among other things on the learning-based function as well as a fine technique for tabu list management.
- When it is assessed on two sets of 66 common benchmark instances in the literature, the proposed algorithm discovers improved best known results (new upper bounds) for 7 instances and matches the best known results for 56 additional cases. Compared to the state-of-the-art algorithms in the literature, such a performance remains highly competitive both in terms of solution quality and computational efficiency.
- The basic principle of the proposed LPR algorithm, i.e., embedding a learning-based two-phase local optimization procedure into an evolutionary framework, is quite general. It could be used to design effective algorithms for other constrained search problems.

The rest of this paper is organized as follows. In Section 2, we describe in detail the proposed LPR algorithm. In Section 3, we show our computational results and comparisons with the current best performing algorithms in the literature. In Section 4, we investigate the contributions of some key ingredients of the LPR algorithm, before concluding the paper in Section 5.

2. Learning-based path relinking algorithm

The learning-based PR (LPR) algorithm presented in this paper is based on the general path relinking method (Glover, 1997; Glover et al., 2000) and extends our first algorithm introduced in Lai et al. (2014) (unpublished paper). From the point of view of algorithmic design, the proposed algorithm is also inspired by the work presented in Lai and Hao (2015) on the fixed spectrum frequency assignment problem (FS-FAP). Given that both algorithms employ the PR framework, they naturally share some similarities in their algorithmic outlines and the way of generating initial solutions and relinking paths. On the other hand, given that BCP and FS-FAP have different objectives (FS-FAP aims to minimize a given penalty function (Lai and Hao, 2015)), dedicated techniques need to be developed for each case in order to attain satisfactory solutions. The proposed LPR algorithm for BCP incorporates several distinguishing features. First, LPR employs a new learning-based evaluation function to provide an informative guidance to the search procedure. Second, LPR uses a two-phase Tabu Search procedure for local improvement which relies among other things on the learning-based evaluation function as well as a fine technique for tabu list management. Third, LPR implements two path relinking methods which are tailored to the BCP problem. As we show in our computational experiments, the proposed LPR algorithm achieves remarkable results when it is tested on the BCP benchmark instances. We describe in detail the proposed LPR algorithm in the following subsections.

2.1. Search space and evaluation function

BCP can be considered from the point of view of constraint satisfaction by solving a series of k-BCP problems, each k-BCP aiming to find a k-coloring (k being fixed) that satisfies all the edge constraints. Starting from a large enough initial k, our algorithm seeks to solve the k-BCP problem, i.e., to find a legal k-coloring such that all edge constraints of BCP are satisfied. Once such a k-coloring is found with the current k, we set k to $k-1$ and solve again the new k-BCP problem. The process is repeated until no legal k-coloring can be found. The last k for which a legal k-coloring was found is returned as the output of the algorithm.

For the k-BCP problem, the search space Ω explored by our LPR algorithm is composed of all possible k-colorings, including both legal and illegal k-colorings. The size of the search space Ω is bounded by $O(k^n)$, where n is the number of the vertices in the graph.

To evaluate the quality of a candidate k-coloring $s \in \Omega$, we adopt an evaluation function which is defined as the summation $f(s)$ of all constraint violations induced by s:

$$f(s) = \sum_{e(v_i,v_j) \in E} \max\{0, d(i,j) - |c(v_i) - c(v_j)|\} \tag{1}$$

where $d(i,j)$ is the weight of edge $e(v_i,v_j)$, and $c(v_i)$ and $c(v_j)$ respectively represent the color of vertices v_i and v_j in s.

Given two solutions s' and s'', s' is better than s'' if $f(s') < f(s'')$. In other words, a better solution has fewer constraint violations. The purpose of the LPR algorithm is to minimize the function f to find a solution s with $f(s) = 0$, which corresponds to a legal k-coloring.

2.2. Main framework

Let P denote a population of p candidate solutions (k-colorings). Let s^* and s^w respectively represent the best solution among all solutions visited so far and the worst solution among the solutions in P (in terms of the evaluation function of Eq. (1)). Let *PairSet* be a set of solution pairs (s^i, s^j) initially composed of all possible pairs in P. Then the proposed LPR algorithm can be described as follows (See Algorithm 1).

Algorithm 1. Pseudo-code of our LPR algorithm for BCP.

```
1:  Input: Problem instance I, the number of colors (k), the size
    of population (p)
2:  Output: the best k-coloring s* found
3:  Initialize the penalty function g(s), i.e., w(i,j) ← 0, (i ≤ j)   /*
    Section 2.4.2*/
4:  repeat
```

```
5:     $P=\{s^1,\dots,s^p\} \leftarrow$ Population_Initialization$(I,k)$   /* Section 2.3 */
6:     if it is not in the first loop then
7:         $s^w \leftarrow \arg\max\{f(s^i): i=1,\dots,p\}$
8:         $P \leftarrow P \cup \{s^*\} \setminus \{s^w\}$
9:     end if
10:    $s^* \leftarrow \arg\min\{f(s^i): i=1,\dots,p\}$
11:    $PairSet \leftarrow \{(s^i,s^j): 1 \le i < j \le p\}$
12:    while $PairSet \neq \emptyset$ do
13:        Randomly pick a solution pair $(s^i,s^j) \in PairSet$
14:        $PairSet \leftarrow PairSet \setminus \{(s^i,s^j)\}$
15:        $s^a \leftarrow PathRelinking(s^i,s^j)$, $s^b \leftarrow PathRelinking(s^j,s^i)$   /* Section 2.5 */
16:        Improvement_and_Updating $(s^a,P,PairSet,s^*,g(s))$   /* Algorithm 2 */
17:        Improvement_and_Updating $(s^b,P,PairSet,s^*,g(s))$   /* Algorithm 2 */
18:    end while
19: until a stopping criterion is met
```

LPR starts with an initial population $P=\{s^1,\dots,s^p\}$ (line 5), where each solution s^i is randomly generated and improved by the basic tabu search procedure described in Section 2.4.1. The search enters then into a while loop which forms the main body of the LPR algorithm (lines 12–18). At each iteration, the following operations are realized. First, from a solution pair (s^i,s^j) taken at random from *PairSet* (the pair is removed from *PairSet* after selection), LPR creates, with a relinking operator (line 15, Section 2.5), two solution paths, one from s^i to s^j and one from s^j to s^i. Then one particular solution is selected from each path for further improvement by the two-phase tabu search procedure (see Section 2.4.2 and Algorithm 3) and the improved solution is finally used to update the population P (see Section 2.6), the best solution found s^*, *PairSet* and the penalty function $g(s)$ (Eq. (3) of Section 2.4.2). The procedure of local optimization and update operations (lines 16–17, Algorithm 1) is described in Algorithm 2. The while loop continues until *PairSet* becomes empty. Then the population P is recreated (lines 5–9) and the above while loop is repeated until a legal solution (k-coloring) is found or a maximum allowed number of path relinking (PR) rounds (denoted by N_{pr}) is reached, where one "while" loop (i.e., lines 12–18) of Algorithm 1 is a PR round.

Algorithm 2. Local improvement and information updating.

```
1:  Function: Improvement_and_Updating $(s,P,PairSet,s^*,g(s))$
2:  Input: Solution generated by relinking operator $(s)$, the population $P$, the best solution $s^*$ found so far, the set of solution pairs available ($PairSet$), and the learning-based penalty function $(g(s))$
3:  Output: $P$, $Pairset$, $s^*$, $g(s)$
4:  $s \leftarrow$ Two−Phase Tabu Search $(f(s),g(s),s)$   /* Section 2.4.2* /
5:  Update learning-based penalty function $g(s)$ according to $s$   /* Section 2.4.2* /
6:  if $f(s) < f(s^*)$ then
7:      $s^* \leftarrow s$
8:  end if
9:  $s^w \leftarrow \arg\max\{f(s^i): s^i \in P\}$   /* $Distance(s,P)$ represents the Hamming distance between $s$ and the closest solution from it in $P$ * /
10: if $(f(s) < f(s^w)) \wedge (Distance(s,P) > 0.1n)$ then
11:     $P \leftarrow P \cup \{s\} \setminus \{s^w\}$
12:     $PairSet \leftarrow PairSet \setminus \{(s^w,s^k): s^k \in P\}$
13:     $PairSet \leftarrow PairSet \cup \{(s,s^k): s^k \in P\}$
14: end if   /* Section 2.6 * /
15: return $\{P, PairSet, s^*, g(s)\}$
```

2.3. Population initialization

From the scratch, an initial population is constructed as follows. We first generate $3p$ random solutions, where each variable (or vertex) of each solution is randomly assigned a color from 1 to k. Then, for each generated solution, the basic tabu search method with the evaluation function $f(s)$ (see Section 2.4.1) is used to optimize it to a local optimum. Finally, we choose the p best solutions to form the initial population.

2.4. Local refinement methods

In our LPR algorithm, we employ two tabu search procedures for local optimization: a basic tabu search (TS) procedure and an advanced two-phase tabu search (TP-TS) procedure.

2.4.1. Basic tabu search

The basic tabu search (TS) procedure operates in the search space Ω defined in Section 2.1. From a given solution (i.e., an illegal k-coloring), it iteratively improves the current solution by following the "critical one-move" neighborhood (Dorne and Hao, 1998), in which a neighboring solution of the current k-coloring is obtained by changing the color of a conflicting vertex u from its original color c_i to another color c_j ($c_i \neq c_j$) (denoted by the move $\langle u,c_i,c_j\rangle$). Here, a vertex is considered to be conflicting if at least one of the distance constraints associated with this vertex is violated. Thus, for a k-coloring s, the size of this neighborhood is bounded by $O(f(s) \times k)$.

At each iteration of TS, the best solution among the eligible neighboring solutions is first chosen to replace the current solution. If there are more than one best neighboring solutions, one of them is taken at random. Then the corresponding move $\langle u,c_i,c_j\rangle$ of the chosen solution is recorded in the tabu list to prevent the reverse move from being selected during the next tt iterations, where tt is the tabu tenure. The TS method stops when the best solution found so far has not been improved during α consecutive iterations, where α is called the depth of TS (Lü and Hao, 2010). Moreover, a simple aspiration criterion is applied to accept a forbidden neighboring solution as long as it is better than the best solution found so far. A neighboring solution is said eligible if it is not forbidden by the tabu list or satisfies the aspiration criterion.

In the TS method, we use a specific function to dynamically tune the tabu tenure tt, according to a technique proposed in Galinier et al. (2011) where the tabu tenure is defined by a periodic step function. If the current iteration is x, then the tabu tenure of a move is denoted by $tt(x)$. Specifically, our tabu tenure function is defined, for each period, by a sequence of values $(a_1,a_2,\dots,a_{p+1})$ and a sequence of interval margins $(x_1,x_2,\dots,x_{p+1})$ such that for each x in $[x_i, x_{i+1}-1]$, $tt(x)=a_i+rand(2)$, where $rand(2)$ denotes a random integer in [0,2]. In our case, p is fixed to 15, and $(a)_{i=1,\dots,15}=\frac{T_{max}}{8}(1,2,1,4,1,2,1,8,1,2,1,4,1,2,1)$, where T_{max} is a parameter which represents the maximum tabu tenure. Finally, the interval margins are defined by $x_1=1$, $x_{i+1}=x_i+4a_i$, $(i \le 15)$. Thus, this function varies periodically and for each period, 15 tabu tenures are used dynamically, each being kept for a number of consecutive iterations.

This basic tabu search procedure is employed to generate an initial population and also served as a key component of the two-phase tabu search procedure explained in the next section.

2.4.2. Two-phase tabu search with an augmented evaluation function

Algorithm 3. Pseudo-code of updating penalty function used in TP-TS.

```
1:  Input: Penalty matrix $[w(i,j)]_{n\times n}$, distance matrix $[d(i,j)]_{n\times n}$,
    weight updating parameters $\theta$ and $\rho$, local minimum
    solution $s=(c(v_1),c(v_2),\ldots,c(v_n))$.
2:  Output: Updated penalty matrix $[w(i,j)]_{n\times n}$.
3:  for each $e(v_i,v_j)\in E$ do
4:    if $|c(v_i)-c(v_j)|<d(i,j)$ then
5:      $w(i,j)\leftarrow w(i,j)+1$
6:    end if
7:  end for
8:  $w_{max}\leftarrow \max_{i<j} w(i,j)$
9:  if $w_{max}>\theta$ then
10:     foreach $e(v_i,v_j)\in E$ do
11:     $w(i,j)\leftarrow \lfloor \rho\cdot w(i,j)\rfloor$
12:   end for
13: end if
```

In order to make local optimization well-informed, we devise a two-phase tabu search (TP-TS) method. TP-TS is composed of two applications of the basic TS procedure described in Section 2.4.1. The first phase applies the TS procedure with an augmented evaluation function F (instead of f) to guide the search to explore promising regions of the search space, while the second phase switches to the TS procedure with the original evaluation function f to seek better solutions around the solution returned by the first phase. To distinguish these two TS applications, we will call them TS_F and TS_f respectively. Hence, the key component of the TP-TS concerns the augmented evaluation function F which takes the following form:

$$F(s)=f(s)+g(s) \tag{2}$$

where $f(s)$ is the original function defined in Section 2.1, and $g(s)$ is an augmented *penalty function* which is given by:

$$g(s)=\sum_{e(v_i,v_j)\in E;|c(v_i)-c(v_j)|<d(i,j)} w(i,j) \tag{3}$$

The penalty function $g(s)$ and its updating mechanism are essential parts of our LPR algorithm, where the terms $w(i,j)$ are the additional weights of the edge $e(v_i,v_j)$, $(e(v_i,v_j)\in E)$, which are initialized to 0 at the beginning of the LPR algorithm and updated dynamically during the search process.

More specifically, during the search of LPR, each time TP-TS returns a local optimum s', the edge weights $w(i,j)$, $(e(v_i,v_j)\in E)$ are updated according to the returned s' as follows. First, for any edge $e(v_i,v_j)$, $w(i,j)$ is increased by one if the distance constraint ($|c(v_i)-c(v_j)|\geq d(i,j)$) is violated in s'. Then, if the maximum edge weight ($\max_{e\in E} w(e)$) exceeds a predetermined parameter value θ, all the edge weights $w(e)$ $(e\in E)$ are decreased by multiplying a constant factor ρ, i.e., $w(e)=\lfloor\rho\times w(e)\rfloor$, $(e\in E,\rho\in(0,1))$. Algorithm 3 describes this weight updating mechanism.

2.4.3. Discussions on the learning based penalty function

In the TP-TS procedure, an important element is the definition of edge weight $w(i,j)$ employed in the penalty function $g(s)$. This weight approximately represents the number of the recently encountered local optima where the distance constraint for the edge $e(v_i,v_j)$ is not satisfied. $w(i,j)$ is thus an indicator to know whether the distance constraint associated to $e(v_i,v_j)$ is easy or difficult to satisfy (i.e., a large $w(i,j)$ value indicates the constraint $|c(v_i)-c(v_j)|\geq d(i,j)$ is difficult to satisfy and vice versa).

With the introduced penalty function $g(s)$, if the constraint $|c(v_i)-c(v_j)|\geq d(i,j)$ is repetitively violated in (high quality) local optima, the associated weight $w(i,j)$ will increase gradually. Since the first phase of TP-TS aims to minimize F, the conflicting vertices involved in edges with a heavy weight $w(i,j)$ will be selected with preference for color changes to satisfy constraint violation. Consequently, this will favor the satisfaction of those constraints that are identified as difficult to satisfy.

Notice that the weight updating mechanism of $g(s)$ uses a simple forgetting technique. When the maximum weight (i.e., $\max_{0<i<j\leq n} w(i,j)$) in $g(s)$ exceeds a predetermined value θ, all the weights $w(i,j)$ $(i<j)$ are decreased by a constant factor ρ, with the goal of periodically decreasing the edge weights in order to forget too old information. A similar technique is used in a heuristic algorithm for the vertex cover problem (Cai et al., 2013). Finally, we can relate the learning mechanism used in TP-TS to learning methods developed in other contexts like local search methods for continuous optimization (Lai et al., 2011; Locatelli and Schoen, 2003), guided local search (Voudouris and Tsang, 1999), and learning-based hybrid search (Jin and Hao, 2015). Nevertheless, our TP-TS procedure is different from these approaches in the way of utilizing information gathered during the search process.

2.5. Path relinking operators

A path relinking operator aims to generate a path of intermediate solutions connecting two high-quality (parent) solutions. In this work, we adopt two *general* path relinking operators: the greedy path relinking (GPR) and the mixed path relinking (MPR) (Lai and Hao, 2015) which are tailored to the BCP problem. The general procedures of these two path relinking operators are respectively given in Algorithms 4 and 5.

Algorithm 4. Pseudo-code of constructing a path for the greedy PR operator.

```
1:  Input: A pair of solutions $(s',s'')$
2:  Output: Path solutions $s(0),s(1),\ldots,s(r)$ from $s'$ to $s''$
3:  $NC\leftarrow\{l: s'_l\neq s''_l, l=1,2,\ldots,n\}$ /* $s'_l$ denotes the value of $l$th
    variable of $s'$ */
4:  $s(0)\leftarrow s'$, $s\leftarrow s'$, $m\leftarrow 1$, $r\leftarrow |NC|$
5:  while $|NC|>0$ do
6:    $l^*\leftarrow \arg\min\{f(s\otimes l): l\in NC\}$   /* $s\otimes l$ means that the
    value of $l$th variable of $s$ is changed to the value of $l$th
    variable of the guiding solution $s''$ */
7:    $NC\leftarrow NC\setminus\{l^*\}$
8:    $s\leftarrow s\otimes l^*$
9:    $s(m)\leftarrow s$ , $m\leftarrow m+1$
10: end while
```

Algorithm 5. Pseudo-code of the mixed PR operator.

```
1:  Input: A pair of solutions $(s',s'')$
2:  Output: Reference solution $s(r)$
3:  $NC\leftarrow\{l: s'_l\neq s''_l, l=1,2,\ldots,n\}$ /* $s'_l$ denotes the value of $l$th
    variable of $s'$ * / /* generate alternately two sub-paths
    whose guiding solutions are respectively $s'$ and $s''$ */
4:  $s(0)\leftarrow s'$, $s(1)\leftarrow s''$, $m\leftarrow 2$, $r\leftarrow |NC|$
5:  while $|NC|>0$ do
6:    $s\leftarrow s(m-2)$
7:    if $m$ is odd then
8:    $s^g\leftarrow s'$   /* $s^g$ represents the current guiding solution */
9:    else
```

```
10:         $s^g \leftarrow s''$
11:     end if
12:     $l^* \leftarrow \arg\min\{f(s \otimes l) : l \in NC\}$ /* $s \otimes l$ means that the
        value of $l$th variable of $s$ is changed to the value of $l$th
        variable of the current guiding solution $s^g$ */
13:     $NC \leftarrow NC \setminus \{l^*\}$
14:     $s \leftarrow s \otimes l^*$
15:     $s(m) \leftarrow s$
16:     $m \leftarrow m+1$
17: end while
18: return $s(r)$
```

- *Greedy path relinking*: As shown in Algorithm 4, GPR generates a solution sequence (i.e., a path of length $HD+1$) $< s(0), s(1), s(2) \ldots s(HD) >$ in a step by step way by starting from $s(0)$, where the first and last solutions are respectively called 'initiating solution' (denoted by s') and 'guiding solution' (denoted by s''), and HD is the Hamming distance between s' and s''. Moreover, a path exhibits the following property: two consecutive solutions in the path differ only by the value of one single variable. To build the path, GPR changes, at each relinking step, the value of one variable. To make the decision, GPR uses the following greedy criterion. Let NC denote the set of variables (vertices) whose values (colors) are different between the current solution $s(i)$ $(i = 1, \ldots, HD-1)$ and the guiding solution s'', then GPR examines the vertices of NC and seeks a new variable $l \in NC$ such that changing the value of the variable l of the current solution s to the value of the variable l of guiding solution s'' leads to the greatest improvement of the function f defined by Eq. (1). We use $s \otimes l$ to denote the new solution generated by changing the value of variable l of solution s to the value of variable l of guiding solution s''. The above process repeats until NC becomes empty.

From a solution path, one particular solution (called the reference solution) is picked for further improvement. Following Lai and Hao (2015) and Wang et al. (2012), we select as reference solution a high-quality solution that is distanced from both the initiating and guiding solutions. For this purpose, we first construct a candidate solution list (CSL) composed of the intermediate solutions having a distance of at least $\xi \cdot HD$ (where ξ is a predetermined parameter value between 0 and 0.5, which is empirically set) from the initiating and guiding solutions. Then, the solution having the best value in terms of function f defined by Eq. (1) in CSL is elected as the reference solution and sent to the two-phase tabu search procedure (Section 2.4.2) for further improvement.

- *Mixed path relinking*: Given a pair of solutions s' and s'', this relinking operator (described in Algorithm 5) generates alternately a solution sequence (i.e., a path connecting s' and s''). Indeed, the generated path is composed of two sub-paths which can be denoted by $\langle s(0), s(2), s(4), \ldots, s(r) \rangle$ and $\langle s(1), s(3), s(5), \ldots, s(r) \rangle$, respectively. Specifically, these two sub-paths have the following features. First, their directions are inverse: the direction of the first one is from $s'(=s(0))$ to s'' and that of another one is from $s''(=s(1))$ to s'. For the sub-path $\langle s(0), s(2), s(4), \ldots, s(r) \rangle$, the initiating solution is $s'(=s(0))$ and the guiding solution is s''. As for the sub-path $\langle s(1), s(3), s(5), \ldots, s(r) \rangle$, the initiating solution and the guiding solution are respectively $s''(=s(1))$ and s'. Second, these two sub-paths are alternately generated starting from $s(0)$ (i.e., s') (lines 4–16). Third, these two sub-paths intersect at the middle (i,e., $s(r)$) of the path connecting s' and s''.

In addition, like the GPR operator, at each relinking step the MPR operator always performs a best move which leads to the greatest improvement of the function f defined by Eq. (1). Finally, the last generated intermediate solution (i.e., $s(r)$) is chosen as the reference solution for further improvement with the two-phase tabu search procedure.

These PR operators share the general procedures of two relinking operators presented in Lai and Hao (2015) designed for the fixed spectrum frequency assignment problem. Yet, given the difference between the two problems, our relinking operators for the BCP problem are based on different implementations. Specifically, our relinking operators maintain one (or two) $n \times k$ matrix M to speed up the relinking procedure, where n is the number of vertices in the graph, k is the number of colors used, and the item $M[i][q]$ $(1 \le i \le n, 1 \le q \le k)$ is defined as:

$$M[i][q] = \sum_{e(v_i, v_j) \in E} \max\{0, d(i,j) - |q - c(v_j)|\} \tag{4}$$

where $d(i,j)$ is the weight of edge $e(v_i, v_j)$, and $c(v_j)$ represents the color of vertex v_j in the current intermediate solution. Clearly, $M[i][q]$ measures the constraint violation degree of vertex i if it receives color q for the current solution. With the help of this memory structure, at each path relinking step, if the value of a variable i is changed from its current value c_i to a different value c_j, the Δ value (i.e, the change of evaluation function value) of the move can be determined rapidly as $\Delta_f = M[i][c_j] - M[i][c_i]$, and then the matrix M can accordingly updated in $O(n)$.

However, for the FS-FAP problem defined on a double weight undirected graph, the implementation of the path relinking operators described above must maintain different information according to the different nature of its objective function. In summary, the procedures in Algorithms 4 and 5 are general, but their implementation depends largely on the problem to be optimized.

Finally, our LPR algorithm employs the MPR operator as its path relinking operator, and the GPR operator is designed just for the purpose of comparison and analysis (see Section 4.2 for a comparison between the GPR and MPR operators).

2.6. Updating population and PairSet

As illustrated in Algorithm 2, the population P and the *PairSet* structure are updated with each offspring solution generated by the path relinking operator and improved by the TP-TS procedure. First, for each improved offspring solution s, if it is better than the worst solution s^w in P and the distance between s and the population P (denoted by $Distance(s, P)$) is larger than $0.1n$, the offspring solution replaces the worst solution s^w in P. Here, n is the number of vertices in the graph, and $Distance(s, P)$ is defined as $Distance(s, P) = \min\{distance(s, s^i) : s^i \in P\}$ in which $distance(s, s^i)$ represents the Hamming distance between solutions s and s^i. Otherwise, the offspring solution is discarded. If the population is updated, then

Table 1
Settings of important parameters.

Parameters	Section	Description	Values
p	2.2	Population size for LPR	20
α	2.4.1	Depth of TS	$\{2 \times 10^3, 10^4\}$
ξ	2.5	Distance parameter used in the GPR operator	0.4
T_{max}	2.4.1	Maximum tabu tenure in TS	$\{50, 100\}$
N_{pr}	2.2	Maximum path relinking rounds	5000
ρ	2.4.2	Scaling factor for the weighting mechanism	0.4
θ	2.4.2	The maximum penalty value for the weighting mechanism	$\{30, 50\}$

Table 2
Comparison of the LPR algorithm with other algorithms on BCP instances.

Instance	k^*	Phan and Skiena (2002)	Lim et al. (2003)	Prestwich (2008)	EA (Malaguti and Toth, 2008)		MITS (Lai and Lü, 2013)		VNS (Matić et al., 2015)		LPR		
		k	k	k	k	$t(s)$	k	$t(s)$	k	$t(s)$	k_{best}	SR	$t(s)$
GEOM20	20	20	21	21	21	0.0	21	0.0	21	0.0	21	20/20	0.0
GEOM20a	20	20	22	20	20	0.0	20	0.0	20	0.0	20	20/20	0.0
GEOM20b	13	13	14	13	13	0.0	13	0.0	13	0.0	13	20/20	0.0
GEOM30	27	27	29	28	28	0.0	28	0.0	28	0.0	28	20/20	0.0
GEOM30a	27	27	32	27	27	0.0	27	0.0	27	0.0	27	20/20	0.0
GEOM30b	26	26	26	26	26	0.0	26	0.0	26	0.0	26	20/20	0.0
GEOM40	27	27	28	28	28	0.0	28	0.0	28	0.0	28	20/20	0.0
GEOM40a	37	38	38	37	37	0.0	37	0.0	37	1.2	37	20/20	0.0
GEOM40b	33	36	34	33	33	0.0	33	0.0	33	2.9	33	20/20	0.0
GEOM50	28	29	28	28	28	0.0	28	0.0	28	0.0	28	20/20	0.0
GEOM50a	50	54	52	50	50	0.0	50	0.0	50	4.3	50	20/20	0.0
GEOM50b	35	40	38	35	35	0.0	35	3.0	35	716.4	35	20/20	0.4
GEOM60	33	34	34	33	33	0.0	33	0.0	33	0.2	33	20/20	0.0
GEOM60a	50	54	53	50	50	0.0	50	1.0	50	23.6	50	20/20	0.0
GEOM60b	41	47	46	43	41	29	41	277	61	6430.7	41	20/20	12.1
GEOM70	38	40	38	38	38	0.0	38	0.0	38	0.3	38	20/20	0.0
GEOM70a	61	64	63	62	61	12	61	45	61	1513.1	61	20/20	7.2
GEOM70b	47	54	54	48	48	52.0	47	8685.0	48	7702.6	47	20/20	104.0
GEOM80	41	44	42	41	41	0.0	41	0.0	41	2.6	41	20/20	0.0
GEOM80a	63	69	66	63	63	150.0	63	21.0	63	3487.4	63	20/20	5.7
GEOM80b	60	70	65	61	60	145.0	60	322.0	60	7851.8	60	20/20	13.4
GEOM90	46	48	46	46	46	0.0	46	0.0	46	1.6	46	20/20	0.0
GEOM90a	63	74	69	64	63	150.0	63	230.0	63	8082.1	63	20/20	20.8
GEOM90b	69	83	77	72	70	1031.0	69	20 144.0	71	7627.3	69	5/20	605.1
GEOM100	50	55	51	50	50	2.0	50	2.0	50	320.9	50	20/20	0.4
GEOM100a	67	84	76	68	68	273.0	67	11 407.0	68	8227.3	67	19/20	292.9
GEOM100b	71	87	83	73	73	597.0	72	24 561.0	73	7945.9	71	8/20	301.6
GEOM110	50	59	53	50	50	3.0	50	2.0	50	943.8	50	20/20	0.3
GEOM110a	≤ 71	88	82	73	72	171.0	72	1529.0	71	9152.9	**70**	12/20	319.5
GEOM110b	77	87	88	79	78	676.0	78	24 416.0	79	8829.1	77	2/20	733.5
GEOM120	59	67	62	60	59	0.0	59	1.0	59	615.2	59	20/20	0.3
GEOM120a	82	101	92	84	84	614.0	82	34 176.0	82	9112.6	82	16/20	596.3
GEOM120b	84	103	98	86	84	857.0	84	–	86	9159.9	84	8/20	344.1

PairSet is updated accordingly: all solution pairs containing s^w are removed from *PairSet* and all the solution pairs that can be generated by combining s with other solutions in P are added into *PairSet*.

3. Experimental results and comparisons

3.1. Benchmark instances

Our LPR algorithm was assessed on two sets of benchmark instances, where the first set (denoted by BCP) is composed of 33 instances and is available in Trick (1998), and the other set (denoted by BMCP) is composed of 33 larger instances transformed from the BMCP instances. These instances were created for the COLOR02/03/04 Competitions on graph coloring and its generalizations, and were widely used to evaluate the performance of BCP and BMCP algorithms like (Chiarandini and Stützle, 2007; Jin and Hao, 2015; Lai and Lü, 2013; Lim et al., 2005, 2003; Malaguti and Toth, 2008; Matić et al., 2015; Phan and Skiena, 2002; Prestwich, 2008).

3.2. Parameter setting and experimental protocol

Our LPR algorithm was programmed in C++ and compiled by g++ with "-O2" option,[1] and all computational experiments were carried out on a computer with an Intel Xeon E5440 processor (2.83 GHz CPU and 2 Gb RAM). Following the DIMACS machine benchmark procedure,[2] our machine requires 0.44, 2.63 and 9.85 s for graphs r300.5, r400.5, and r500.5, respectively.

Table 1 gives the descriptions and settings of the parameters adopted in our LPR algorithm. For parameter α (the depth of TS), 2×10^3 and 10^4 were used for the first phase and the second phase of the TP-TS procedure. For T_{max} which represents the maximum tabu tenure of TS, 50 was used for instances with $n \leq 150$, and 100 otherwise. For θ, 30 was used for instances with $n < 150$ and 50 otherwise. These values were determined via a preliminary experiment and all the experiments were carried out without additional parameter tunings.

As explained in Section 2.1, our LPR algorithm is designed for k-BCP. Therefore, given the stochastic nature of our LPR algorithm, each k-BCP instance considered was independently solved 20 times. For each instance, the value of k was set to the one close to the best known result in the literature.

3.3. Computational results on the BCP instances

Our first experiment aims to evaluate our LPR algorithm on the set of 33 BCP instances with up to 120 vertices. Table 2 summarizes the computational statistics and lists the results of 6 reference algorithms from the literature. Column 2 gives the previous best known results (k^*). Columns 3–11 respectively show the results of the 6 reference algorithms, namely the Discropt

[1] The source code of our LPR algorithm and the best solutions found in this work will be available online at http://www.info.univ-angers.fr/pub/hao/bcp.html upon publication of the paper.

[2] dfmax: ftp://dimacs.rutgers.edu/pub/dsj/clique/, benchmark procedure was compiled by gcc compiler with default option.

heuristic (Phan and Skiena, 2002), a hybrid algorithm (Lim et al., 2003), the FCNS algorithm (Prestwich, 2008), an evolutionary algorithm (EA) (Malaguti and Toth, 2008), the MITS algorithm (Lai and Lü, 2013), and a variable neighborhood search (VNS) algorithm (Matić et al., 2015). These results are directly extracted from the literature. The computational results of our LPR algorithm are reported in columns 12–14, including the best results achieved (k_{best}), the success rate (SR) of finding a legal k_{best}-coloring, the average computing time per successful run in seconds ($t(s)$), where the improved results are indicated in bold, the symbol '–' means that the corresponding algorithm fails to find a legal k-coloring, and the symbol '$\leq$' in the second column means that an improved result is found. Note that for the k-BCP which is a constraint satisfaction problem, the success rate is usually reported together with the corresponding averaged time (Lai and Lü, 2013; Malaguti and Toth, 2008; Matić et al., 2015).

Table 2 discloses that our LPR algorithm matches the previous best-known results for 29 out of 33 instances. Moreover, LPR is able to improve the best known result for one instance (i.e., GEOM110a) with an average computing time of 319.5 s and a success rate of 12/20. One can also observe that the success rate of our LPR algorithm is higher than or equal to 12/20 except for 4 instances, demonstrating the robustness of our algorithm.

When comparing with the best results of the first four reference algorithms, one finds that the LPR algorithm is able to obtain better results on at least 6 instances. In comparison with the MITS algorithm, our LPR algorithm is able to find better results in 4 instances. In addition, compared with the recent VNS algorithm, our LPR algorithm finds the better results in 6 instances. These outcomes show that the LPR algorithm has a strong search capability compared to these reference algorithms.

Note that in Table 2 we focus on the computational results rather than the computational times, since it is difficult to make a fair comparison between the above algorithms in terms of computational time due to the differences among programming languages, compilers, computers and stopping criteria used by these algorithms. For example, for the hybrid algorithm of Lim et al. (2003), the time limits were set to 1 min and 3 h respectively for the BCP and BMCP instances, whereas for MITS (Lai and Lü, 2013) the time limits were set to 1 and 2 h for the BCP and BMCP instances. The EA algorithm of Malaguti and Toth (2008) used 500 and 3000 seconds for each tested value of k to solve the BCP and BMCP instances respectively, and the recent VNS (Matić et al., 2015) was run till 2 and 4 h respectively. The computational times are thus listed only for indicative purposes.

In addition to the algorithms mentioned above, the learning-based hybrid search (LHS) algorithm published very recently (in 2015) is shown to be the best performing algorithm for BCP (Jin and Hao, 2015). Fortunately, the LHS algorithm and our LPR algorithm were run on the same computing environment, including programming language, compiler and computer, which allows us to make a fair comparison between these two algorithms. We conducted the experiment based on 39 k-BCP instances, where the LPR algorithm was performed 20 times for each tested k-BCP instance, as the LHS algorithm did in Jin and Hao (2015). Computational results are summarized in Table 3, together with the results of the LHS algorithm. The rows *Better*, *Equal*, and *Worse* at the bottom respectively show the number of k-BCP instances for which the associated algorithm obtains better, equal, and worse results compared to another one, and the row *Total* shows the total number of k-BCP instances.

Table 3 discloses that our LPR algorithm has a better and worse success rate (the number of times the best solution is found) than the LHS algorithm on 4 and 7 instances respectively. For the remaining 28 cases, the results of our LPR algorithm match those of the LHS algorithm. Regarding the average computing time, our LPR algorithm is superior to or equals the LHS algorithm for all instances with only two exceptions. Moreover, our LPR algorithm improves the best known result (k^*) for one instance (GEOM110a), whereas the LHS algorithm fails to find the current best result. Thus, these outcomes indicate that our LPR algorithm is highly competitive compared to the LHS algorithm on the tested instances.

Table 3
Comparison of LPR algorithm with the best performing algorithm (i.e., LHS) in the literature on the BCP instances based on the same experimental platform. Better results between two algorithms are indicated in bold.

Graph		LHS (Jin and Hao, 2015)			LPR		
Name	k^*	k	SR	$t(s)$	k	SR	$t(s)$
GEOM20	20	21	20/20	0.0	21	20/20	0.0
GEOM20a	20	20	20/20	0.0	20	20/20	0.0
GEOM20b	13	13	20/20	0.0	13	20/20	0.0
GEOM30	27	28	20/20	0.0	28	20/20	0.0
GEOM30a	27	27	20/20	0.0	27	20/20	0.0
GEOM30b	26	26	20/20	0.0	26	20/20	0.0
GEOM40	27	28	20/20	0.0	28	20/20	0.0
GEOM40a	37	37	20/20	0.0	37	20/20	0.0
GEOM40b	33	33	20/20	0.0	33	20/20	0.0
GEOM50	28	28	20/20	0.0	28	20/20	0.0
GEOM50a	50	50	20/20	0.1	50	20/20	**0.0**
GEOM50b	35	35	20/20	1.2	35	20/20	**0.4**
GEOM60	33	33	20/20	0.0	33	20/20	0.0
GEOM60a	50	50	20/20	0.1	50	20/20	**0.0**
GEOM60b	41	41	20/20	214.7	41	20/20	**12.1**
GEOM70	38	38	20/20	0.0	38	20/20	0.0
GEOM70a	61	61	20/20	23.7	61	20/20	**7.2**
GEOM70b	47	47	20/20	665.4	47	20/20	**104.0**
GEOM80	41	41	20/20	0.1	41	20/20	**0.0**
GEOM80a	63	63	20/20	6.6	63	20/20	**5.7**
GEOM80b	60	60	20/20	19.9	60	20/20	**13.4**
GEOM90	46	46	20/20	0.0	46	20/20	0.0
GEOM90a	63	63	20/20	23.8	63	20/20	**20.8**
GEOM90b	69	69	**20/20**	779.2	69	5/20	**605.1**
GEOM100	50	50	20/20	1.0	50	20/20	**0.4**
GEOM100a	67	67	8/20	1557.4	67	**19/20**	**292.9**
		68	20/20	189.8	68	20/20	**52.8**
GEOM100b	71	71	**12/20**	2038.6	71	8/20	**301.6**
		72	**20/20**	759.6	72	12/20	**374.9**
GEOM110	50	50	20/20	1.3	50	20/20	**0.3**
GEOM110a	71	70	0/20	–	70	**12/20**	**319.5**
		71	19/20	2218.7	71	**20/20**	**129.4**
		72	20/20	324.2	72	20/20	**44.0**
GEOM110b	77	77	**10/20**	2598.7	77	2/20	**733.54**
		78	**20/20**	**94.3**	78	16/20	267.0
GEOM120	59	59	20/20	0.5	59	20/20	**0.3**
GEOM120a	82	82	**20/20**	**171.1**	82	16/20	596.3
GEOM120b	84	84	2/20	3568.1	84	**8/20**	**344.1**
		85	**20/20**	1829.7	85	15/20	**472.6**
#Better			7	2		4	24
#Equal			28	13		28	13
#Worse			4	24		7	2
#Total			39	39		39	39

3.4. Computational results on the BMCP instances

In this Section, we evaluate our LPR algorithm by comparing it with the top performing algorithms in the literature for the BMCP instances. We carried out experiments on the set of 33 large BCP instances transformed from the BMCP instances. The results of the experiment are summarized in Table 4, together with the results of 7 reference algorithms, including two hybrid algorithms (Lim et al., 2005, 2003), the FCNS algorithm (Prestwich, 2008), a stochastic local search method named OF-SW (Chiarandini and Stützle, 2007), an evolutionary algorithm (EA) (Malaguti and Toth, 2008), the MITS algorithm (Lai and Lü, 2013), and the very recent VNS algorithm (Matić et al., 2015). The results of the reference

Table 4
Comparison of the LPR algorithm with the reference algorithms on BMCP instances.

Instance	k^*	Lim et al. (2003)	Lim et al. (2005)	Prestwich (2008)	OF-SW (Chiarandini and Stützle, 2007)		EA (Malaguti and Toth, 2008)		MITS (Lai and Lü, 2013)		VNS (Matić et al., 2015)		LPR		
		k	k	k	k	t_{max}	k	$t(s)$	k	$t(s)$	k	$t(s)$	k_{best}	SR	$t(s)$
GEOM20	149	149	149	149	–	–	149	18.0	149	2.0	149	54.3	149	20/20	0.0
GEOM20a	169	169	169	170	–	–	169	9.0	169	15.0	169	3289.4	169	20/20	0.1
GEOM20b	44	44	44	44	44	30.0	44	5.0	44	0.0	44	0.0	44	20/20	0.0
GEOM30	160	160	160	160	–	–	160	1.0	160	0.0	160	5.7	160	20/20	0.0
GEOM30a	209	211	209	214	209	380.0	210	954.0	209	10.0	209	4123.7	209	20/20	1.4
GEOM30b	77	77	77	77	77	80.0	77	0.0	77	0.0	77	1.3	77	20/20	0.0
GEOM40	167	167	167	167	–	–	167	20.0	167	0.0	167	2107.1	167	20/20	0.0
GEOM40a	213	214	213	217	214	500.0	214	393.0	213	328.0	213	13 192.2	213	20/20	6.2
GEOM40b	74	76	74	74	74	140.0	74	1.0	74	2.0	74	1821.9	74	20/20	0.1
GEOM50	224	224	224	224	–	–	224	1197.0	224	8.0	224	1671.3	224	20/20	0.1
GEOM50a	311	326	318	323	315	1080.0	316	4675.0	314	40 373.0	311	21 685.9	311	16/20	3560.0
GEOM50b	83	87	87	86	84	200.0	83	197.0	83	1200.0	83	14260.7	83	20/20	16.8
GEOM60	258	258	258	258	258	710.0	258	139.0	258	19.0	258	5780.5	258	20/20	0.1
GEOM60a	353	368	358	373	356	1420.0	357	8706.0	356	38 570.0	353	23 989.3	353	13/20	3818.6
GEOM60b	113	119	116	116	117	300.0	115	460.0	113	104 711.0	114	16 381.0	113	9/20	710.3
GEOM70	266	279	273	277	267	1060.0	272	1413.0	270	7602.0	267	12 384.3	266	20/20	1569.1
GEOM70a	463	478	469	482	478	1470.0	473	988.0	467	38 759.0	463	20 686.4	465	2/20	9789.7
GEOM70b	115	124	121	119	120	380.0	117	897.0	116	213 545.0	116	16 783.2	115	11/20	764.6
GEOM80	379	394	383	398	382	1490.0	388	132.0	381	212 213.0	379	16 538.8	379	20/20	1983.3
GEOM80a	≤ 355	379	379	380	360	1510.0	363	8583.0	361	41 235.0	355	29 208.9	**352**	2/20	7879.5
GEOM80b	138	145	141	141	139	490.0	141	1856.0	139	255.0	138	13 704.1	138	20/20	51.8
GEOM90	328	335	332	339	334	1810.0	332	4160.0	330	4022.0	329	18 760.9	328	20/20	2168.3
GEOM90a	372	382	377	382	377	1910.0	382	5334.0	375	10 427.0	373	24 087.2	372	5/20	3911.9
GEOM90b	≤ 142	157	157	147	147	590.0	144	1750.0	144	211 366.0	142	19 996.9	**140**	5/20	2142.4
GEOM100	404	413	404	424	404	2170.0	410	3283.0	404	40 121.0	404	14 816.9	404	20/20	64.8
GEOM100a	≤ 429	462	459	461	437	2500.0	444	12 526.0	442	381.0	429	35 663.4	**426**	8/20	13 059.5
GEOM100b	≤ 153	172	170	159	159	690.0	156	3699.0	156	213 949.0	156	24 776.4	**151**	1/20	3178.9
GEOM110	375	389	383	392	378	2510.0	383	2344.0	381	183.0	375	22 997.4	375	20/20	3510.2
GEOM110a	478	501	494	500	490	3120.0	490	2318.0	488	926.0	480	46 955.0	478	13/20	10 979.0
GEOM110b	201	210	206	208	208	790.0	206	480.0	204	944.0	202	22 076.9	201	7/20	1920.0
GEOM120	396	409	402	417	397	2730.0	396	2867.0	396	inf	396	17 919.3	396	20/20	999.2
GEOM120a	≤ 536	564	556	565	549	3690.0	559	3873.0	554	1018.0	539	59 717.4	**531**	1/20	24 880.4
GEOM120b	187	201	199	196	191	910.0	191	3292.0	189	213 989.0	190	22 835.9	187	3/20	1837.4

algorithms are directly extracted from the associated references. To report our results, we show the same statistics as in Table 2, where the improved results are indicated in bold, '–' means that the corresponding result is not available, and the symbol '$\leq$' in the second column means that an improved result is found. The stopping conditions used by the main reference algorithms were given in Section 3.3.

Table 4 shows that our LPR algorithm is able to improve the best known results for 5 out of 33 instances, and match the best known results for the remaining instances, with only one exception (i.e., GEOM70a). Our LPR algorithm attains a success rate greater than or equal to 11/20 except for 10 instances. Moreover, compared to the best results yielded by these 7 reference algorithms, our LPR algorithm is able to find better results for 12 out of 33 instances, which indicates clearly that our LPR algorithm dominates these 7 reference algorithms on the BMCP instances. Finally, we do not insist on computing times since it is meaningless to make a comparison with different k for a given graph (k^*-BCP is generally much more difficult to solve than (k^*+1)-BCP).

To make a fair comparison between our LPR algorithm and the best performing LHS algorithm (Jin and Hao, 2015), we ran our LPR algorithm 20 times for each of the 58 k-BCP instances shown in Table 5 based on the platform employed by the LHS algorithm, and the computational results are summarized together with the results of the LHS algorithm in Table 5, where the symbols have the same meanings as in Table 3.

First, our LPR algorithm is able to find a better result than the LHS algorithm for 5 instances and to match the best results of the LHS algorithm for the remaining instances, indicating that our LPR algorithm is superior to the LHS algorithm in terms of search capability. Second, in terms of the average computing time, our LPR algorithm is better and worse than the LHS algorithm for 43 and 13 instances, respectively. Specifically, our LPR algorithm is about 20 times faster than the LHS algorithm for some instances, such as the instance GEOM90b with $k=143$. Third, our LPR algorithm achieves a higher and lower success rate than the LHS algorithm for 16 and 7 instances, respectively. These outcomes indicate that our LPR algorithm performs globally better than the LHS algorithm on these large instances.

4. Analysis and discussions

We now turn our attention to analyze two important elements of the proposed LPR algorithm, i.e., the learning based penalty function and the path relinking operator. To show the effect of each of these elements on the performance of the LPR algorithm, we carried out several experiments on a selection of some hard instances (8 BCP instances and 23 BMCP instances). The remaining instances were not investigated in this section, because they are very easy to be solved by means of our LPR algorithm according to the results in Tables 2 and 4.

4.1. Importance of the learning mechanism

To highlight that the learning mechanism is a key ingredient of our LPR algorithm, we carried out the following two experiments. In the first experiment, we used the GPR operator as our path relinking operator, and denote the associated PR algorithms with and without the learning based function g respectively by LPR-GPR

Table 5
Comparison of LPR algorithm with the best performing algorithm (i.e., LHS) in the literature on the BMCP instances based on the same experimental platform. Better results between two algorithms are indicated in bold.

Graph		LHS (Jin and Hao, 2015)			LPR		
Name	k^*	k	SR	$t(s)$	k	SR	$t(s)$
GEOM20	149	149	20/20	1.8	149	20/20	**0.0**
GEOM20a	169	169	20/20	0.5	169	20/20	**0.1**
GEOM20b	44	44	20/20	0.0	44	20/20	0.0
GEOM30	160	160	20/20	0.1	160	20/20	**0.0**
GEOM30a	209	209	20/20	16.2	209	20/20	**1.4**
GEOM30b	77	77	20/20	0.0	77	20/20	0.0
GEOM40	167	167	20/20	0.2	167	20/20	**0.0**
GEOM40a	213	213	20/20	9.0	213	20/20	**6.2**
GEOM40b	74	74	20/20	1.5	74	20/20	**0.1**
GEOM50	224	224	20/20	0.3	224	20/20	**0.1**
GEOM50a	311	311	**20/20**	**1452.6**	311	16/20	3560.0
		312	20/20	**307.5**	312	20/20	1623.2
GEOM50b	83	83	20/20	72.1	83	20/20	**16.8**
GEOM60	258	258	20/20	1.3	258	20/20	**0.1**
GEOM60a	353	353	2/20	9007.1	353	**13/20**	**3818.6**
		354	20/20	4996.0	354	20/20	**1844.1**
GEOM60b	113	113	**20/20**	910.7	113	9/20	**710.3**
GEOM70	266	266	20/20	2534.0	266	20/20	**1569.1**
GEOM70a	463	465	**6/20**	36 604.9	465	2/20	**9789.7**
		466	**20/20**	12 622.1	466	3/20	**1172.1**
GEOM70b	115	115	3/20	3640.7	115	**11/20**	**764.6**
		116	20/20	1844.7	116	20/20	**213.2**
GEOM80	379	379	20/20	**357.8**	379	20/20	1983.3
		380	20/20	**164.0**	380	20/20	573.0
GEOM80a	355	352	0/20	–	352	**2/20**	**7879.5**
		357	2/20	43 403.0	357	**20/20**	**2583.8**
		358	7/20	25 852.0	358	**20/20**	**2493.2**
		360	20/20	13 302.9	360	20/20	**1560.1**
GEOM80b	138	138	20/20	**46.5**	138	20/20	58.1
GEOM90	328	328	20/20	**162.2**	328	20/20	2168.3
GEOM90a	372	372	3/20	16 782.1	372	**5/20**	**3911.9**
		373	20/20	**3164.5**	373	20/20	3218.8
GEOM90b	142	140	0/20	–	140	**5/20**	**2142.4**
		142	7/20	7680.8	142	**19/20**	**437.7**
		143	20/20	4858.5	143	20/20	**196.5**
		144	20/20	1331.8	144	20/20	**161.0**
GEOM100	404	404	20/20	64.9	404	20/20	**64.8**
GEOM100a	429	426	0/20	–	426	**8/20**	**13 059.5**
		429	1/20	78 363.1	429	**20/20**	**5843.9**
		434	20/20	10 767.1	434	20/20	**2629.6**
		436	20/20	3147.6	436	20/20	**2035.6**
GEOM100b	153	151	0/20	–	151	**1/20**	**3178.9**
		153	1/20	10840.1	153	**13/20**	**1892.3**
		155	20/20	3147.6	155	20/20	**557.4**
		156	20/20	726.3	156	20/20	**337.1**
GEOM110	375	375	20/20	**1598.8**	375	20/20	3510.2
GEOM110a	478	478	1/20	49457.1	478	**13/20**	**10 979.0**
		480	20/20	**2921.5**	480	20/20	4160.9
		482	20/20	**375.3**	482	20/20	2045.4
GEOM110b	201	201	3/20	5388.4	201	**7/20**	**1920.0**
		203	20/20	**303.1**	203	20/20	392.0
GEOM120	396	396	20/20	**626.1**	396	20/20	999.2
GEOM120a	536	531	0/20	–	531	**1/20**	**24 880.4**
		536	2/20	69 518.6	536	**19/20**	**12 372.8**
		539	20/20	20 286.1	539	20/20	**7720.9**
GEOM120b	187	187	**8/20**	8025.8	187	3/20	**1837.4**
		188	**20/20**	1642.0	188	12/20	**1455.3**
		189	**20/20**	**315.2**	189	16/20	834.8
#Better			7	13		16	43
#Equal			35	2		35	2
#Worse			16	43		7	13
#Total			58	58		58	58

and PR-GPR. In other words, within PR-GPR, the two-phase tabu search procedure was simply replaced by the basic tabu search and the other ingredients were kept unchanged. Then, we independently solved each instance 20 times by using LPR-GPR and PR-GPR. The computational results are summarized in Table 6,

Table 6
Comparison of the PR algorithms with and without learning mechanism, where the GPR operator is used in the PR algorithms. Better results between two algorithms are indicated in bold, and the improved results are indicated in italic.

Graph	Type	k_{best}	k	PR-GPR		LPR-GPR	
				SR	$t(s)$	SR	$t(s)$
GEOM70b	BCP	47	47	17/20	137.8	**19/20**	**127.0**
GEOM90b	BCP	69	69	4/20	**165.6**	**11/20**	233.4
GEOM100a	BCP	67	67	8/20	**278.9**	**13/20**	370.2
GEOM100b	BCP	71	71	3/20	502.5	**4/20**	**339.8**
GEOM110a	BCP	*70*	*70*	6/20	509.1	**13/20**	**437.5**
GEOM110b	BCP	77	77	0/20	–	0/20	–
GEOM120a	BCP	82	82	8/20	**558.5**	**14/20**	606.1
GEOM120b	BCP	84	84	0/20	–	**4/20**	**338.7**
GEOM50a	BMCP	311	311	1/20	**2130.2**	**11/20**	2185.8
GEOM50b	BMCP	83	83	20/20	19.7	20/20	**17.8**
GEOM60	BMCP	258	258	20/20	0.1	20/20	0.1
GEOM60a	BMCP	353	353	13/20	**2088.5**	**15/20**	3101.4
GEOM60b	BMCP	113	113	1/20	**148.6**	**5/20**	444.7
GEOM70	BMCP	266	266	16/20	1619.2	**20/20**	**1029.6**
GEOM70a	BMCP	*462*	*462*	0/20	–	**1/20**	**2707.0**
GEOM70b	BMCP	115	115	0/20	–	**3/20**	**848.4**
GEOM80	BMCP	379	379	18/20	1310.4	**20/20**	**1062.8**
GEOM80a	BMCP	*352*	357	4/20	**2222.1**	**11/20**	3387.9
GEOM80b	BMCP	138	138	20/20	**44.8**	20/20	50.0
GEOM90	BMCP	328	328	15/20	**1639.1**	**19/20**	2398.3
GEOM90a	BMCP	372	372	2/20	**2498.4**	**3/20**	4746.2
GEOM90b	BMCP	*140*	*140*	0/20	–	**1/20**	**626.8**
GEOM100	BMCP	404	404	20/20	49.3	20/20	**33.7**
GEOM100a	BMCP	*426*	*427*	0/20	–	**4/20**	**4137.5**
GEOM100b	BMCP	*151*	153	0/20	–	**1/20**	**1475.7**
GEOM110	BMCP	375	375	15/20	**1946.4**	**17/20**	2517.3
GEOM110a	BMCP	478	478	4/20	**10 123.7**	**14/20**	13 244.7
GEOM110b	BMCP	201	201	3/20	1712.9	**7/20**	**1132.8**
GEOM120	BMCP	396	396	20/20	681.9	20/20	**672.9**
GEOM120a	BMCP	*531*	*533*	0/20	–	**4/20**	**23 530.6**
GEOM120b	BMCP	187	187	0/20	–	**3/20**	**1144.0**
#Better				0	12	25	17
#Equal				6	2	6	2
#Worse				25	17	0	12
#Total				31	31	31	31

including the current best known results (k_{best}), the number of colors used (k), the success rate of finding a legal k-coloring (SR), and the average computing time ($t(s)$) per successful run, where better results between two algorithms are indicated in bold, and the improved results are indicated in italic. The rows *Better, Equal*, and *Worse* at the bottom of the Table respectively show the number of instances for which the associated algorithm obtains better, equal, and worse results compared to another one, and the row *Total* shows the total number of instances.

In the second experiment, we simply replaced the GPR operator by the MPR operator for both PR algorithms with and without the learning based function g, and the resulting algorithms are respectively denoted by LPR-MPR and PR-MPR. Then, the above experiment was performed by means of LPR-MPR and PR-MPR. Computational results are summarized in Table 7.

Table 6 shows that when the GPR operator is used in our PR algorithms, the PR algorithm with the learning mechanism (i.e., LPR-GPR) outperforms the one without the learning mechanism (i.e., PR-GPR) in terms of both the success rate and the average computing time. Specifically, in terms of the success rate, LPR-GPR is superior to PR-GPR for 25 out of 31 instances, while matching the results of PR-GPR for the 6 remaining instances. As to the average computing time, LPR-GPR obtains better and worse results respectively for 17 and 12 instances compared to PR-GPR. Moreover, it can be observed that in terms of the number of colors LPR-GPR yields better results compared to the PR-GPR in 8 cases.

Table 7
Comparison of the PR algorithms with and without learning mechanism, where the MPR operator is used in the PR algorithms. Better results between two algorithms are indicated in bold, and the improved results are indicated in italic.

Graph	Type	k_{best}	k	PR-MPR		LPR-MPR	
				SR	*t*(*s*)	*SR*	*t*(*s*)
GEOM70b	BCP	47	47	19/20	**80.8**	**20/20**	104.0
GEOM90b	BCP	69	69	**8/20**	**409.0**	5/20	605.1
GEOM100a	BCP	67	67	9/20	362.0	**19/20**	**292.9**
GEOM100b	BCP	71	71	3/20	399.9	**8/20**	**301.6**
GEOM110a	BCP	*70*	*70*	7/20	755.7	**12/20**	**319.5**
GEOM110b	BCP	77	77	0/20	–	**2/20**	**733.54**
GEOM120a	BCP	82	82	7/20	**473.9**	**16/20**	596.3
GEOM120b	BCP	84	84	1/20	746.9	**8/20**	**344.1**
GEOM50a	BMCP	311	311	6/20	**3411.9**	**16/20**	3560.0
GEOM50b	BMCP	83	83	20/20	26.8	20/20	**16.8**
GEOM60	BMCP	258	258	20/20	0.1	20/20	0.1
GEOM60a	BMCP	353	353	10/20	**2608.9**	**13/20**	3818.6
GEOM60b	BMCP	113	113	3/20	**286.0**	**9/20**	710.3
GEOM70	BMCP	266	266	17/20	**1531.9**	**20/20**	1569.1
GEOM70a	BMCP	*462*	465	**7/20**	**7279.4**	2/20	9789.7
GEOM70b	BMCP	115	115	2/20	964.9	**11/20**	**764.6**
GEOM80	BMCP	379	379	20/20	**1020.1**	20/20	1983.3
GEOM80a	BMCP	*352*	*352*	0/20	–	**2/20**	**7879.5**
GEOM80b	BMCP	138	138	20/20	**52.1**	20/20	58.1
GEOM90	BMCP	328	328	19/20	**2145.9**	19/20	2168.3
GEOM90a	BMCP	372	372	1/20	**3045**	**5/20**	3911.9
GEOM90b	BMCP	*140*	*140*	0/20	–	**5/20**	**2142.4**
GEOM100	BMCP	404	404	20/20	**42.4**	20/20	64.8
GEOM100a	BMCP	*426*	*426*	0/20	–	**8/20**	**13 059.5**
GEOM100b	BMCP	151	153	0/20	–	**13/20**	**1892.3**
GEOM110	BMCP	375	375	19/20	4074.2	**20/20**	**3510.2**
GEOM110a	BMCP	478	478	6/20	16487.9	**13/20**	**10 979.0**
GEOM110b	BMCP	201	201	3/20	**1123.4**	**7/20**	1920.0
GEOM120	BMCP	396	396	20/20	1052.0	20/20	**999.2**
GEOM120a	BMCP	*531*	*533*	0/20	–	**7/20**	**23 214.7**
GEOM120b	BMCP	187	187	0/20	–	**3/20**	**1837.4**
#Better				2	14	22	16
#Equal				7	1	7	1
#Worse				22	16	2	14
#Total				31	31	31	31

Table 8
Comparison between the path relinking operators within the LPR algorithm. Better results between two path relinking operators are indicated in bold, and the improved results are indicated in italic.

Graph	Type	k_{best}	k	LPR-GPR		LPR-MPR	
				SR	*t*(*s*)	*SR*	*t*(*s*)
GEOM70b	BCP	47	47	19/20	127.0	**20/20**	**104.0**
GEOM90b	BCP	69	69	**11/20**	**223.4**	5/20	605.1
GEOM100a	BCP	67	67	13/20	370.2	**19/20**	**292.9**
GEOM100b	BCP	71	71	4/20	339.8	**8/20**	**301.6**
GEOM110a	BCP	*70*	*70*	**13/20**	437.5	12/20	**319.5**
GEOM110b	BCP	77	77	0/20	–	**2/20**	**733.54**
GEOM120a	BCP	82	82	14/20	606.1	**16/20**	**596.3**
GEOM120b	BCP	84	84	4/20	**338.7**	**8/20**	344.1
GEOM50a	BMCP	311	311	11/20	**2185.8**	**16/20**	3560.0
GEOM50b	BMCP	83	83	20/20	17.8	20/20	**16.8**
GEOM60	BMCP	258	258	20/20	0.1	20/20	0.1
GEOM60a	BMCP	353	353	**15/20**	**3101.4**	13/20	3818.6
GEOM60b	BMCP	113	113	5/20	**444.7**	**9/20**	710.3
GEOM70	BMCP	266	266	20/20	**1029.6**	20/20	1569.1
GEOM70a	BMCP	*462*	*463*	**7/20**	**3381.9**	0/20	–
GEOM70b	BMCP	115	115	3/20	848.4	**11/20**	**764.6**
GEOM80	BMCP	379	379	20/20	**1026.7**	20/20	1983.3
GEOM80a	BMCP	*352*	*352*	0/20	–	**2/20**	**7879.5**
GEOM80b	BMCP	138	138	20/20	**50.0**	20/20	58.1
GEOM90	BMCP	328	328	19/20	2398.3	19/20	**2168.3**
GEOM90a	BMCP	372	372	3/20	4746.2	**5/20**	**3911.9**
GEOM90b	BMCP	*140*	*140*	1/20	626.8	**5/20**	**2142.4**
GEOM100	BMCP	404	404	20/20	**33.7**	20/20	64.8
GEOM100a	BMCP	*426*	*426*	0/20	–	**8/20**	**13 059.5**
GEOM100b	BMCP	151	151	0/20	–	**1/20**	**3178.9**
GEOM110	BMCP	375	375	17/20	**2517.2**	**20/20**	3510.2
GEOM110a	BMCP	478	478	**14/20**	13 244.7	13/20	**10 979.0**
GEOM110b	BMCP	201	201	7/20	**1132.8**	7/20	1920.0
GEOM120	BMCP	396	396	20/20	**672.9**	20/20	999.2
GEOM120a	BMCP	*531*	*533*	4/20	23 530.6	**7/20**	**23 214.7**
GEOM120b	BMCP	187	187	3/20	**1144.0**	3/20	1837.4
#Better				5	14	16	16
#Equal				10	1	10	1
#Worse				16	16	5	14
#Total				31	31	31	31

From Table 7, one also finds that when the MPR operator is employed, the PR algorithm with the learning mechanism (i.e., LPR-MPR) outperforms the variant without the learning mechanism (i.e., PR-MPR) on the overall performance. At first, in terms of the success rate LPR-MPR obtains better or equal results for all tested instances with only two exceptions. Second, as for the average computing time, LPR-MPR is comparable with PR-MPR. Specifically, LPR-MPR takes respectively a shorter and longer average time to reach a legal k-coloring on 16 and 14 instances compared to PR-MPR. Third, LPR-MPR yields better results for 7 cases in terms of the number of colors compared to PR-MPR, and matches the results of PR-MPR for the remaining instances, which indicates that LPR-MPR has a stronger search capability compared to PR-MPR.

In sum, the above results indicate that the learning mechanism proposed in this work plays an essential role within the learning-based PR algorithms.

4.2. Comparison between the path relinking operators

The relinking operator is another fundamental ingredient of our LPR algorithm. In order to analyze the influence of the relinking operator on the performance of the LPR algorithm, we compared our GPR and MPR relinking operators, and the corresponding LPR algorithms are respectively denoted by LPR-GPR and LPR-MPR. The experiment was carried out on the same set of benchmarks as used in Section 4.1, where each instance with the given k was solved 20 times by both algorithms. The computational results are summarized in Table 8. The first two columns give the name of the graph and the type of instance, columns 3 and 4 give the current best known results k_{best} and the value of k used. The computational results of LPR-GPR are listed in columns 5 and 6, including the success rate (SR) and the average computing time per successful run ($t(s)$ in seconds), and the results of LPR-MPR are listed in the last two columns.

Compared to LPR-GPR, LPR-MPR obtains respectively better and worse results on 16 and 5 instances in terms of the success rate, which implies that the MPR operator is more robust than the GPR operator. On the other hand, the average computing time of LPR-MPR is comparable to that of LPR-GPR, since LPR-GPR and LPR-MPR require a shorter computing time for 14 and 16 instances, respectively. As for the search capability, no algorithm dominates the other one. For example, for instance GEOM70a of BMCP, LPR-GPR is able to find a legal k-coloring (k=463) with a success rate of 7/20, while LPR-MPR fails to find a legal k-coloring (k=463) over 20 runs. On the contrary, for instance GEOM100a of BMCP, LPR-GPR fails to find a legal k-coloring (k=426), while LPR-MPR obtains this result with a success rate of 8/20. Therefore, it can be concluded that the search capability of the relinking operators depends strongly on the structure of the given instances.

5. Conclusions

The proposed learning-based path relinking (LPR) algorithm for solving the bandwidth coloring problem and the bandwidth multicoloring problem incorporates a learning-based two-phase tabu search method with a path relinking operator. We tested the proposed LPR algorithm on 66 benchmark instances commonly used in the literature. Computational results showed that our algorithm was highly effective compared to the state-of-the-art algorithms in the literature. Specifically, the proposed algorithm improves the best known results (new upper bounds) for 7 out of 66 instances, while matching the best known results in 56 cases. Compared to the best performing algorithms in the literature, our LPR algorithm shows a highly competitive performance.

We studied some essential ingredients of the proposed algorithm which allowed to shed light on the following points. First, the learning mechanism introduced in this work plays a crucial role in the high performance of the LPR algorithm. Second, the mixed path relinking (MPR) operator is generally better than the greedy path relinking (GPR) operator for BCP. However, the efficacy of the path relinking operator is closely related to the structure of the problem instances, and no path relinking operator fully dominates another one for all instances.

One notices that the basic principle of embedding a learning-based two-phase local tabu search method into an evolutionary framework, is independent of the BCP problem. It would be interesting to investigate such an application in other settings. Also, the idea of the penalty based learning function could be applied to solve other constraint problems.

Acknowledgments

We are grateful to the reviewers for their useful comments which helped us to improve the paper. This work was partially supported by the National Natural Science Foundation of China (Grant nos. 61370183 and 61100144), a post-doc grant (for X.J. Lai) from the Region of Pays de la Loire (France) and the PGMO project (2014–2015, Jacques Hadamard Mathematical Foundation, Paris).

References

Cai, S.W., Su, K.L., Luo, C., Sattar, A., 2013. NuMVC: an efficient local search algorithm for minimum vertex cover. J. Artif. Intell. Res. 46 (1), 687–716.
Chiarandini, M., Stützle, T., 2007. Stochastic local search algorithms for graph set T-colouring and frequency assignment. Constraints 12 (3), 317–403.
Costa, D., 1993. On the use of some known methods for T-coloring of graphs. Ann. Oper. Res. 41 (4), 343–358.
Deng, Y., Bard, J.F., 2011. A reactive GRASP with path relinking for capacitated clustering. J. Heuristics 17 (2), 119–152.
Dorne, R., Hao, J.K., 1998. Tabu search for graph coloring, T-colorings and set T-colorings. In: Voss, S., Martello, S., Osman, I.H., Roucairol, C. (Eds.), Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academic Publishers, New York, pp. 77–92 (Chapter 6).
Galinier, P., Boujbel, Z., Fernandes, M.C., 2011. An efficient memetic algorithm for the graph partitioning problem. Ann. Oper. Res. 191 (1), 1–22.
Gallardo, J.E., Cotta, C., 2015. A GRASP-based memetic algorithm with path relinking for the far from most string problem. Eng. Appl. Artif. Intell. 41, 183–194.
Glover, F., Laguna, M., Martí, R., 2000. Fundamentals of scatter search and path relinking. Control Cybern. 39, 653–684.
Glover, F., 1997. A template for scatter search and path relinking. Lecture Notes in Computer Science, vol. 1363, pp. 1–51.
Hale, W.K., 1980. Frequency assignment: theory and applications. Proc. IEEE 68 (12), 1497–1514.
Hao, J.K., Dorne, R., Galinier, P., 1998. Tabu search for frequency assignment in mobile radio networks. J. Heuristics 4 (1), 47–62.
Jin, Y., Hao, J.K., 2015. Effective learning-based hybrid search for bandwidth coloring. IEEE Trans. Syst. Man Cybern.: Syst. 45 (4), 624–635.
Lü, Z.P., Hao, J.K., 2010. A memetic algorithm for graph coloring. Eur. J. Oper. Res. 203 (1), 241–250.
Lacomme, P., Prins, C., Prodhon, C., Ren, L., 2015. A multi-start split based path relinking (MSSPR) approach for the vehicle routing problem with route balancing. Eng. Appl. Artif. Intell. 38, 237–251.
Lai, X.J., Hao, J.K., 2015. Path relinking for the fixed spectrum frequency assignment problem. Expert Syst. Appl. 42 (10), 4755–4767.
Lai, X.J., Lü, Z.P., 2013. Multistart iterated tabu search for bandwidth coloring problem. Comput. Oper. Res. 40 (5), 1401–1409.
Lai, X.J., Xu, R.C., Huang, W.Q., 2011. Prediction of the lowest energy conuration for Lennard–Jones clusters. Sci. China Chem. 54 (6), 985–991.
Lai, X.J., Lü, Z.P., Hao, J.K., Glover, F., Xu, L.P., 2013. Path Relinking for Bandwidth Coloring Problem. Unpublished Working Paper, November 2013. Posted on CoRR September 3, 2014. arXiv:1409.0973.
Lim, A., Zhang, X., Zhu, Y., 2003. A hybrid method for the graph coloring and its related problems. In: Proceedings of MIC2003: The Fifth Metaheuristic International Conference, Kyoto, Japan.
Lim, A., Zhu, Y., Lou, Q., Rodrigues, B., 2005. Heuristic methods for graph coloring problems. In: Proceedings of the 2005 ACM Symposium on Applied Computing, Santa Fe, New Mexico, pp. 933–939.
Locatelli, M., Schoen, F., 2003. Efficient algorithms for large scale global optimization: Lennard–Jones clusters. Comput. Optim. Appl. 26 (2), 173–190.
Malaguti, E., Toth, P., 2008. An evolutionary approach for bandwidth multicoloring problems. Eur. J. Oper. Res. 189 (3), 638–651.
Martí, R., Gortazar, F., Duarte, A., 2010. Heuristics for bandwidth coloring problem. Int. J. Metaheuristics 1 (1), 11–29.
Matić, D., Kratica, J., Filipović, V., 2015. Variable neighborhood search for solving bandwidth coloring problem. Posted on CoRR 22 May 2015. arXiv:1505.06032.
Montemanni, R., Smith, D.H., 2010. Heuristic manipulation, tabu search and frequency assignment. Comput. Oper. Res. 37 (3), 543–551.
Phan, V., Skiena, S., 2002. Coloring graphs with a general heuristic search engine. In: Proceedings of Computational Symposium on Graph coloring and its generalization, New York, pp. 92–99.
Prestwich, S., 2008. Generalized graph colouring by a hybrid of local search and constraint programming. Discrete Appl. Math. 156 (2), 148–158.
Rahimi-Vahed, A., Crainic, T.G., Gendreau, M., Rei, W., 2013. A path relinking algorithm for a multi-depot periodic vehicle routing problem. J. Heuristics 19 (3), 497–524.
Smith, D.H., Hurley, S., Thiel, S.U., 1998. Improving heuristics for the frequency assignment problem. Eur. J. Oper. Res. 107 (1), 76–86.
Trick, M.A., 2002. Computational symposium: graph coloring and its generalization. ⟨http://mat.gsia.cmu.edu/COLOR03/⟩.
Voudouris, C., Tsang, E., 1999. Guided local search and its application to the traveling salesman problem. Eur. J. Oper. Res. 113 (2), 469–499.
Wang, Y., Lü, Z.P., Glover, F., Hao, J.K., 2012. Path relinking for unconstrained binary quadratic programming. Eur. J. Oper. Res. 223 (3), 595–604.