Effective learning-based hybrid search for bandwidth coloring

Yan Jin and Jin-Kao Hao

***Abstract*—The Bandwidth Coloring Problem (BCP) and the Bandwidth MultiColoring Problem (BMCP) are two important generalizations of the classical vertex coloring problem. This paper presents Learning-based Hybrid Search (LHS) for BCP and BMCP. LHS combines a construction phase to progressively build feasible (partial) colorings and a local search phase to re-establish feasibility when an illegal partial solution is encountered. The construction phase relies on a learning-based guiding function to determine the next vertex for color assignment while the local search phase uses a tabu search repair procedure to resolve coloring conflicts. Experiments on a set of 33 well-known benchmarks for BCP and a set of 33 benchmarks for BMCP demonstrate that the proposed LHS approach can match the best known solution for most benchmarks. In particular, LHS finds an improved best solution for 14 instances.**

***Index Terms*—Bandwidth coloring, learning-based heuristics, tabu search, combinatorial optimization.**

I. Introduction

Given an undirected graph $G = (V, E)$ with vertex set $V = \{v_1, \ldots, v_n\}$, edge set $E \subset V \times V$ and edge weight $d(i, j)$ for each edge $\{v_i, v_j\} \in E$ ($d(i, j)$ can also be considered as a distance between two adjacent vertices v_i and v_j), a legal bandwidth coloring is a function $c : V \rightarrow \{1, 2, ..., k\}$ such that the absolute difference between $c(v_i)$ and $c(v_j)$ of an edge $\{v_i, v_j\} \in E$ is at least $d(i, j)$, i.e.,

$$|c(v_i) - c(v_j)| \geqslant d(i, j),\ \forall \{v_i, v_j\} \in E \tag{1}$$

The Bandwidth Coloring Problem (BCP, also known as the restricted T-coloring problem [13], [22]) is to find a legal bandwidth coloring of G with k minimum. The problem of k-BCP corresponds to BCP with k being fixed where one seeks a legal bandwidth coloring with k colors. It is easy to see that the classic NP-hard vertex graph coloring problem [11] is a special case of BCP when $d(i, j) = 1$ for each edge of the graph.

BCP can be generalized as the Bandwidth MultiColoring Problem (BMCP, also called the restricted set T-coloring problem [13], [22]) where each vertex v_i receives a subset $S(i) \subset \{1, 2, ..., k\}$ of $p(i)$ different colors. A legal bandwidth multicoloring must satisfy two distance constraints: the absolute difference between any member of $S(i)$ and $S(j)$ is at least $d(i, j)$ for each edge $\{v_i, v_j\} \in E$ and the absolute difference between two distinct values in $S(i)$ is at least $d(i, i)$ for each vertex ($d(i, i)$ is the color separation distance for vertex v_i), i.e.,

$$\begin{aligned} |x - y| \geqslant d(i, j), \forall \{v_i, v_j\} \in E,\ \forall x \in S(i), y \in S(j) \\ |a - b| \geqslant d(i, i), \forall v_i \in V, \forall a, b \in S(i) \end{aligned} \tag{2}$$

Yan Jin and Jin-Kao Hao (Corresponding author) are with the LERIA, University of Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01, France (e-mail: {jin,hao}@info.univ-angers.fr)

BMCP is to find a legal bandwidth multicoloring of G with k minimum.

BCP and BMCP are notable for their applicability to a number of important applications in particular in the area of frequency assignment in mobile networks [1], [2], [3], [10], [13], [14], [23], [24], [25], [26]. However, from a computational point of view, both BCP and BMCP are comparable to the well-known NP-hard vertex graph coloring problem [11] since in their simplest form, they reduce to the vertex graph coloring problem.

Over the past decades, much effort has been devoted to BCP and BMCP, including COLOR02/03/04, a series of computational challenges dedicated to solution methods for graph coloring and its generalizations [5]. Given the computational complexity of these problems, a number of heuristic and metaheuristic algorithms have been proposed in the literature such as simulated annealing [6], tabu search algorithms [7], [18], evolutionary approach [19], [20] and hybrid algorithms [21]. More methods can be found in the 2008 special issue of Discrete Applied Mathematics [17] dedicated to computational methods for graph coloring and its generalizations.

In this work, we propose a new and highly effective heuristic approach, called Learning-based Hybrid Search (LHS) for BCP and BMCP. LHS is based on the cooperation of an informed construction procedure and a local search repair procedure and integrates several distinguishing features. The main contributions of the work can be summarized as follows.

- From the algorithm perspective, the proposed LHS approach establishes an original cooperative framework between an informed construction approach and a local search approach. The construction procedure progressively builds feasible (partial) solutions while relying on 1) a dynamic learning-based guiding function to determine the order of vertices to be colored and 2) a forward checking technique to reduce the available colors of the considered vertex. In particular, the guiding function takes into account both static information of the problem instance under consideration and dynamic information learned during the construction and the repair processes. When the construction procedure runs into a *dead-end* (i.e., the partial solution under construction can not be extended any more without violating some problem constraints), the search switches to the repair procedure to try to unlock the dead-end situation in order to switch back to the coloring construction process. The local search repair procedure is based on the tabu search metaheuristic reinforced by a simple perturbation strategy. To our knowledge, this is the first hybrid algorithm of this kind proposed for BCP and BMCP. Moreover, the

underlying cooperative framework could be useful and adapted to other search problems.

- From the computational perspective, we evaluate the LHS approach on two sets of 66 commonly used BCP and BMCP benchmark instances in the literature. The computational results show that our LHS algorithm can achieve the best-known result for most of these benchmark instances established by several existing algorithms. Moreover, LHS finds an improved best solution for 14 instances (2 BCP instances and 12 BMCP instances). To our knowledge, no existing algorithm from the literature can attain such a performance.

The rest of the paper is organized as follows. Next section presents the Learning-based Hybrid Search (LHS) for BCP and BMCP. Section III shows computational results on the benchmark instances and comparisons with some best performing algorithms. Before concluding, Section IV shows an analysis of the proposed LHS approach.

II. Learning-based Hybrid Search (LHS) for BCP and BMCP

The Learning-based Hybrid Search (LHS) approach is designed for BCP (more precisely, the k-BCP problem), since one can easily convert BMCP into BCP by defining a new graph as follows (see [7] for an example). For each vertex v_i of BMCP, we define a clique $\{v_{i_1}, v_{i_2}, ..., v_{p_i}\}$ of cardinality $p(i)$ with a distance $d(i,i)$ for each edge of the clique. For each edge $\{v_i, v_j\} \in E$, the distance $d(i,j)$ is duplicated for each pair of vertices $\{v_{i_x}, v_{j_y}\}$ of the two corresponding cliques. Then the new graph has $\sum_{v_i \in V} p(i)$ vertices and $\sum_{\{v_i,v_j\} \in E} p(i) \times p(j)$ edges.

To approximate BCP, we solve a series of k-BCP as performed in [7], [18], [20], i.e., to seek a legal bandwidth coloring with k colors (k can be initially fixed to be slightly a value larger than or equal to the best-known k in the literature). As soon as a legal k-coloring is found, we decrease the current value k to $k-1$ and solve the new k-BCP. We repeat this process until no legal k-coloring can be found and return the last k for which a legal k-coloring is reached.

We describe below the LHS algorithm which basically solves the decision k-BCP problem.

A. The general LHS procedure

Our Learning-based Hybrid Search (LHS) approach repeats the following two phases: a *coloring construction phase* (Section II-C) to extend in a step-by-step way a partial legal solution by coloring a new vertex at each step and a *repair phase* (Section II-D) using tabu search [12] to solve constraint violations when the partial legal solution cannot be further extended. The main procedure of the LHS approach is summarized in Algorithm 1.

The coloring construction phase operates with partial (legal) solutions and tries to expand a partial solution to a complete solution without violating the problem constraints. Starting from an empty solution, the construction procedure selects at each step, according to a learning-based guiding function (see Section II-B), an uncolored vertex and tries to assign to it an available color. For the selected vertex, if a color can be assigned to it without violating any distance constraints, the vertex receives the color and the construction phase continues. If no feasible coloring is possible for the selected vertex, a dead-end is encountered (in this case, the last selected vertex is called a *dead-end vertex*) and LHS switches to the tabu search repair phase to escape from the dead-end.

Suppose the partial legal solution is composed of $L-1$ colored vertices when the dead-end is encountered. Then the tabu search repair phase takes as its input the partial solution and extends it by assigning a randomly selected color from the given k colors to the dead-end vertex. Obviously, this extension leads inevitably to an illegal coloring (with L vertices) which violates some distance constraints. The purpose of the repair phase is then to try to find a legal coloring for the set of L vertices by re-coloring these vertices. At the end of the repair process, there are two possibilities. If the dead-end is resolved, i.e., a legal partial coloring is found for the set of L vertices, LHS switches back to the construction phase to continue its coloring construction. On the other hand, if the repair procedure fails to find a legal partial coloring for the set of L vertices, LHS drops the on-going process and prepares to restart a new round of construction-repair process. In order to learn from this failure, LHS updates the guiding function of some critical variables (see Section II-B) with the help of an adaptive reinforcement learning strategy. As such, the next round of the construction phase will benefit from some learned information to re-order the vertices such that the critical vertices which are difficult to color will be considered with a high priority.

LHS repeats the above process until a pre-fixed number of maximum tries is reached or a complete legal k-coloring is obtained.

Algorithm 1 Learning-based Hybrid Search for the Bandwidth Coloring Problem

Require: A graph $G = (V, E)$, an integer k
Ensure: A feasible k-coloring C_* found or null
$T \leftarrow 0$ /* T counts the failed 'construction-repair' rounds */
$C \leftarrow \emptyset$ /* C is the current feasible coloring under construction */
while $T \leq maxTries$ **do**
  repeat
    /* Construction phase */
    $(C, v_i) \leftarrow$ Construct_partial_solution(C) /* v_i is the dead-end vertex encountered, Sect. II-C */
    /* Tabu search based conflict repair phase */
    $C_* \leftarrow$ Tabu_search_repair(C, v_i) /* Apply tabu search to solve conflicts, Sect. II-D */
    if C_* is still a conflicting coloring **then**
      /* The current round of construction-local search fails */
      Update the learning-based guiding function $\mathcal{F}$ /* Sect. II-B */
      $C \leftarrow \emptyset$; $C_* \leftarrow \emptyset$
      break
    else
      $C \leftarrow C_*$
    end if
  until $|C_*| = |V|$
  if $|C_*| = |V|$ **then**
    return C_* /* C_* is a complete and legal k-coloring, return C_* */
  end if
  $T \leftarrow T + 1$
end while
return C_*

B. Learning-based guiding function

As we explain above and in Section II-C, the construction procedure employs a guiding function $\mathcal{F}$ to dynamically determine the order of vertices for color assignment. This guiding function constitutes thus one of the most critical components of the LHS algorithm and needs to be designed with care.

In our case, the guiding function $\mathcal{F}$ dynamically ranks each vertex v by taking into account both static and learning-based dynamic information and is called at each step of the construction process to select a vertex for color assignment (a vertex with the highest rank is selected, ties are broken at random). This function takes the following form:

$$\mathcal{F}(v) = \begin{cases} deg(v) + fr\_deg(v), & T = 0 \\ fb\_val(v) + fr\_deg(v), & T > 0, \forall v \in V \end{cases} \quad (3)$$

where T is the number of the failed 'construction-repair' rounds.

The $deg(v)$ part of $\mathcal{F}$ represents the degree of vertex v, i.e., $deg(v) = |A(v)|$ where $A(v) = \{u \in V : \{v, u\} \in E\}$. For a given graph, this part remains static and captures a basic and main characteristic of the graph. The use of this information within $\mathcal{F}$ is based on the consideration that a vertex with a large degree exhibits a stronger influence to its adjacent vertices than a vertex with a small degree. So a vertex with a high degree is selected with a higher priority for color assignment. This static part of $\mathcal{F}$ is only considered for the first round of the construction-repair process ($T = 0$) when there is no learning-based information available yet.

The freedom degree $fr\_deg(v)$ is the number of adjacent vertices of vertex v which received a color. Let $K(v) = \{u \in A(v) : u \text{ is colored}\}$, then $fr\_deg(v) = |K(v)|$. Clearly, $fr\_deg(v)$ takes values in $\{0, \ldots, deg(v)\}$. For each vertex v, $fr\_deg(v)$ is initially set to 0. Then each time an adjacent vertex of v receives a color, $fr\_deg(v)$ is increased by 1. As such, $\mathcal{F}$ evolves dynamically with fr_deg to favor the coloring of those vertices which become more constrained by the coloring of its adjacent vertices. The fr_deg component of $\mathcal{F}$ is based on the following consideration. When $fr\_deg(v)$ is small relative to $deg(v)$ (say close to 0), few of its adjacent vertices are colored. As a consequence, vertex v has a large freedom in the sense that it is easy to color. In this case, vertex v will be given a low $\mathcal{F}$ value, thus a low rank. Reversely, if $fr\_deg(v)$ is close to its maximum value $deg(v)$, almost all of its neighboring vertices have already received a color. In this case, coloring vertex v is more difficult since this is strongly constrained by the colors of its adjacent vertices. Consequently, we give a high $\mathcal{F}$ value (thus a high rank) to such a vertex whose coloring becomes critical.

The feedback value $fb\_val(v)$ is used to learn from each failed construction-repair process in order to influence the rank of vertices for the next round of coloring construction process. This part is initially set to 0 for each vertex at the beginning of the whole search process. Then two types of updates are dynamically operated after each failed construction-repair process, i.e., when a dead-end is encountered, which cannot be unblocked by the subsequent tabu search repair process (see Sections II-C and II-D).

- *Update of the dead-end vertex*: Suppose that v is the last vertex under consideration when the construction phase encounters a dead-end (i.e., no color can be assigned to v without violating some distance constraints). Since the dead-end involving vertex v cannot be resolved by the subsequent tabu search repair phase, we consider the underlying vertex v to be difficult or critical to color. In order to favor the selection of this vertex for color assignment for the next round of the coloring construction phase, we increase its feedback value $fb\_val(v)$ (thus its $\mathcal{F}$ rank) by a quantity $\Delta = \max_{j \in 1,\ldots,n}\{deg(u) : u \in V\}$.
- *Update of the conflicting vertices*: Suppose that c is the conflicting partial coloring after an improving or plateau move of the tabu search repair process. We consider that the vertices that are still involved in constraint violations are difficult or critical to color. In order to favor the selection of these vertices for the next round of the construction phase, we raise their rank. Precisely, let X be the set of colored vertices in c. We first identify the set CV of conflicting vertices in c such that $CV = \{v \in X : \exists u \in K(v), |c(v) - c(u)| < d(u, v)\}$ where $K(v)$ is the set of colored vertices adjacent to v. Then for each vertex v of CV, its feedback value $fb\_val(v)$ is increased by 1.

As such, if a vertex is repeatedly involved in constraint violations which are difficult to repair, its rank will progressively be augmented and the vertex will be selected for coloring with a high priority during the next round of the construction-repair process.

C. Construction phase with forward checking

The construction phase is the main component of the LHS approach responsible for generating legal k-colorings. The whole procedure of the construction phase is illustrated in Algorithm 2. During the construction phase, we maintain two sets of vertices $\mathcal{S} \subset V$ and $\mathcal{U} = V \setminus \mathcal{S}$. $\mathcal{S}$ is the current partial legal solution representing the set of vertices with their respective assigned colors while $\mathcal{U}$ contains the set of remaining vertices waiting for color assignment.

The construction phase initially starts with $\mathcal{S} = \emptyset$ an $\mathcal{U} = V$ and then iteratively extends $\mathcal{S}$ by including a new vertex v from $\mathcal{U}$ with a legal color $c(v)$ (i.e., $c(v)$ satisfies the distance constraints expressed in Eq. (2)). The construction phase relies on two key elements: the learning-based guiding function $\mathcal{F}$ (Section II-B) and a constraint satisfaction technique called forward-checking [15].

The learning-based guiding function $\mathcal{F}$ provides a dynamic order for the uncolored vertices of $\mathcal{U}$. Using $\mathcal{F}$, the construction procedure selects always the vertex with the largest $\mathcal{F}$ value (ties are broken at random). For each selected vertex, the forward-checking technique is applied to remove incompatible colors for the selected vertex v with respect to its adjacent vertices (i.e., the distance constraints). Forward-checking is an important component of the construction procedure. We explain its functioning in the rest of this section.

Let $v \in \mathcal{U}$ designate the selected vertex for color assignment, let $D(v)$ be the set of the currently available colors of v

Algorithm 2 Pseudo-code of the construction phase with forward checking

Require: $\mathcal{S}$ a partial feasible coloring
Ensure: Either $\mathcal{S}$ a complete legal coloring or $(\mathcal{S}, v)$ an extended partial coloring $\mathcal{S}$ with a dead-end vertex v
1: $\mathcal{U} \leftarrow V \backslash \mathcal{S}$ /* $\mathcal{U}$ is the set of uncolored vertices of graph G */
2: **while** $\mathcal{U} \neq \emptyset$ **do**
3: Select a vertex $v \in \mathcal{U}$ with the largest $\mathcal{F}$ value (break ties randomly)
4: $D(v) \leftarrow \{1, 2, \ldots, k\}$ /* Initial color set of vertex v */
5: **for** each $\kappa \in D(v)$ **do**
6: /* Let $A(v)$ be the set of vertices adjacent to vertex v */
7: **for** each $\mu \in A(v)$ **do**
8: **if** μ is a colored adjacent vertex of v **then**
9: **if** $|\kappa - c(\mu)| < d(v, \mu)$ **then**
10: /* Distance constraint violation, delete κ from $D(v)$ */
11: $D(v) \leftarrow D(v) - \{\kappa\}$
12: **end if**
13: **else**
14: /* μ is an uncolored adjacent vertex of v */
15: **if** $\kappa + d(v, \mu) > k$ **and** $\kappa - d(v, \mu) < 1$ **then**
16: /* Color κ for v is incompatible with an uncolored vertex μ, delete κ from $D(v)$ */
17: $D(v) \leftarrow D(v) - \{\kappa\}$
18: **end if**
19: **end if**
20: **end for**
21: **end for**
22: **if** $D(v) \neq \emptyset$ **then**
23: Choose the smallest color $\kappa \in D(v)$ and assign κ to v
24: Extend $\mathcal{S}$ with v and its color κ
25: For each uncolored vertex $\mu \in A(v)$, update its guiding function value $\mathcal{F}(\mu)$
26: **else**
27: /* $D(v)$ becomes empty, a dead-end is detected, return $\mathcal{S}$ and the dead-end vertex v for the repair phase */
28: **return** (S, v)
29: **end if**
30: **end while**
31: **return** $\mathcal{S}$

initially set to $\{1, 2, \ldots, k\}$, and let $A(v) = \{\mu \in V : \{\mu, v\} \in E\}$ be the set of vertices adjacent to v. Two forward-checking operations are triggered to reduce $D(v)$ with the computational complexity $\Theta(k \times |A(v)|)$.

- *First forward checking operation*: This operation aims to eliminate any color from $D(v)$ which is incompatible with the *colored* vertices in $A(v)$. More precisely, for a color $\kappa \in D(v)$, if there exists an adjacent colored vertex $\mu \in A(v)$ such that $|\kappa - c(\mu)| < d(v, \mu)$, color κ can not be assigned to vertex v (due to distance constraint violation) and can be removed from $D(v)$ (See Algorithm 2, lines 8-12).
- *Second forward checking operation*: This operation aims to eliminate any color from $D(v)$ which is incompatible with the *uncolored* vertices in $A(v)$ if the color is assigned to v. More precisely, for a color $\kappa \in D(v)$, if there exists an adjacent uncolored vertex $\mu \in A(v)$ such that $\kappa + d(v, \mu) > k$ and $\kappa - d(v, \mu) < 1$, color κ cannot be assigned to vertex v (since the uncolored vertex μ has no available colors if κ is assigned to v) and can be removed from $D(v)$ (see Algorithm 2, lines 14-18).

After these forward checking operations, if the color domain $D(v)$ is not empty, the current partial solution is extended by vertex v with the smallest color of $D(v)$. Before moving to the next iteration of the construction phase, the algorithm updates the guiding function value $\mathcal{F}(\mu)$ (i.e., $fr\_deg(\mu)$) for each uncolored vertex $\mu \in A(v)$ (Algorithm 2, lines 22-25).

On the other hand, if the color domain $D(v)$ becomes empty (i.e., no color can be assigned to v without violating some distance constraints), a dead-end is detected. At this point, the construction procedure is stopped and the search process switches to the tabu search repair procedure (Algorithm 2, lines 26-28).

Figure 1 illustrates the construction phase with forward checking. In the example, we use six colors (k = 6) to color a graph G with five vertices $a, b, \ldots, g$ and six edge weights. At the first construction step, the guiding function is $\mathcal{F}\{a, b, d, e, g\}$ = {2, 3, 2, 2, 3}, b and g are thus the vertices with the largest $\mathcal{F}$ value. Suppose a random selection between b and g gives vertex b and $D(b) = \{1, 2, 3, 4, 5, 6\}$. According to the first forward checking operation, no color can be removed from $D(b)$. Then, according to the second forward checking operation, {2, 3, 4, 5} can be removed from $D(b)$. Hence, $D(b) = \{1, 6\}$ and the smallest color 1 is chosen to color the vertex b. Then the function $\mathcal{F}\{a, b, d, e, g\}$ is updated to {3, -, 3, 2, 4}. At this point, one construction step is successfully accomplished. The next steps of the construction phase will handle the remaining vertices g, d, e and a in this order and assign them colors 4, 6, 1 and 5 respectively.

In this example, no dead-end is encountered during the construction phase. Generally, the application of forward-checking can eliminate all the colors of the vertex currently under consideration. In this case, the search switches to the repair phase for conflict resolution that we explain in the next section.

D. Tabu search repair phase

When the construction phase encounters a dead-end where the selected vertex v has no available colors (all its colors are removed by the forward-checking operations, see Section II-C), LHS assigns a random color from $\{1, ..., k\}$ to vertex v and updates the guiding function value $\mathcal{F}(\mu)$ (i.e., $fr\_deg(\mu)$) for each uncolored vertex $\mu \in A(v)$. By doing this, some distance constraints are inevitably violated causing the current partial solution to be illegal. The purpose of the repair phase is then to try to transform this conflicting solution into a legal partial bandwidth coloring in order to switch back to the construction phase. To achieve this, we develop a tabu search [12] repair procedure (TSRP) for the k-BCP which combines a basic tabu search procedure (TS) and a simple perturbation mechanism. The TS procedure is an adaptation of the Tabucol algorithm first introduced in [16] for the conventional graph coloring problem and later improved in [7], [9].

1) Tabu search procedure: The general scheme of the tabu search repair procedure is shown in Algorithm 3. As shown in the algorithm, TSRP alternates between the tabu search procedure (lines 4-18, Algorithm 3) and the perturbation mechanism.

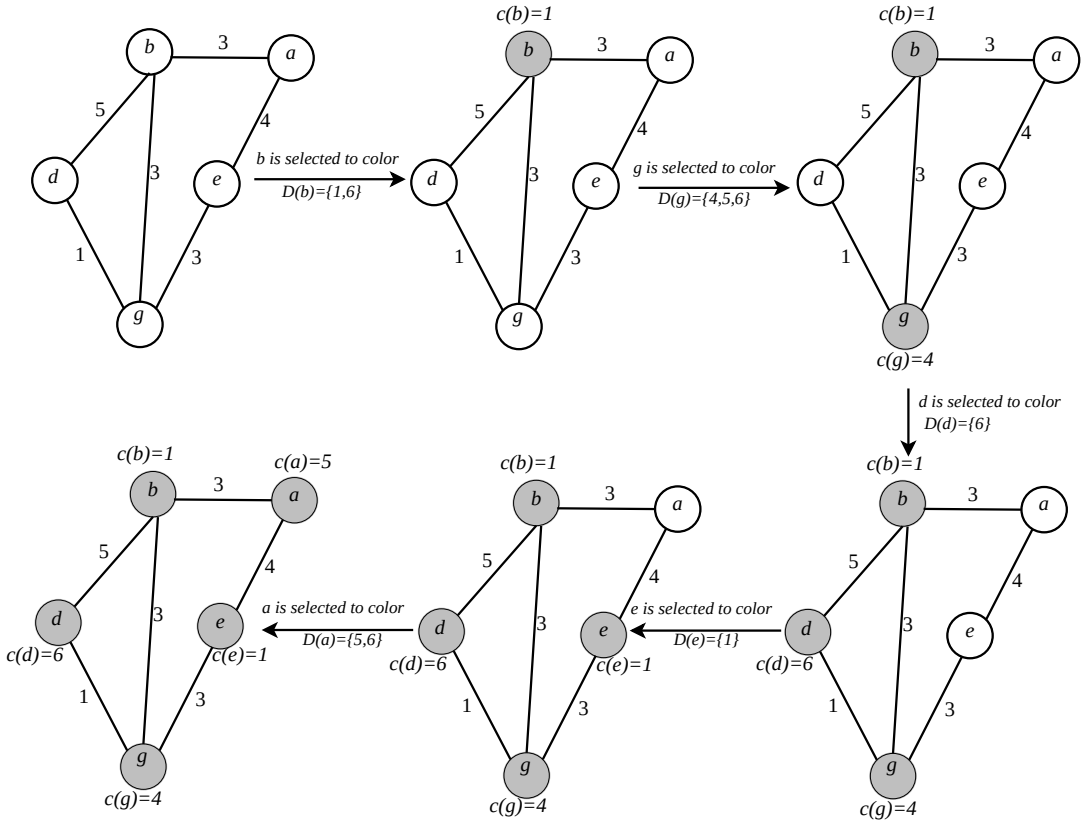

Fig. 1. An illustrative example of the construction phase with forward checking.

Suppose that the partial illegal solution $\mathcal{S}$ is composed of L vertices $\{v^1, ..., v^L\} \subset V$. Then the TS procedure explores a subset of the space $\Omega = \{1, ..., k\}^L$ to seek a legal bandwidth coloring by using an evaluation function f, a neighborhood N and a tabu list (see Section II-D2). Notice that Ω contains both legal and illegal bandwidth k-colorings. The purpose of TS is to find a legal solution by making successive improvements.

From the partial illegal coloring $\mathcal{S}$, TS improves its solutions by iteratively moving from the current solution to one neighboring solution guided by the evaluation function. The best solution (in terms of the evaluation function f) is recorded in $\mathcal{S}_*$. At each iteration, TS moves from the current solution $\mathcal{S}$ to a best authorized neighboring solution, records the transition in the tabu list to prevent the search from revisiting solution $\mathcal{S}$ and possibly updates the best solution $\mathcal{S}_*$ (lines 7-15). If the best solution cannot be improved for a given number $maxIters$ of consecutive iterations, the search is considered to be trapped in a local optimum. To escape from the local optimum, TSRP triggers a perturbation mechanism (see Section II-D3) to modify the current solution which becomes the starting solution of the next round of the tabu search procedure (lines 19-20). The repair phase using TSRP stops either when a legal coloring is found by the tabu search procedure (lines 10-11) or after reaching a prefixed number $maxTSruns$ of the tabu search runs (or perturbations) (lines 3, 22-23).

2) Evaluation function, constrained neighborhood and tabu list: We next describe the key elements of the tabu search procedure, i.e., the evaluation function to measure the quality of a candidate solution (bandwidth) coloring, the neighborhood to identify the neighboring solutions that can be attained at each iteration, and the tabu list to avoid short-term cycling.

Evaluation function f: Recall that a distance $d(i, j)$ is defined for each edge $\{v_i, v_j\} \in E$ and the distance constraint states that the absolute value of the difference between the colors assigned to adjacent vertices v_i and v_j must be at least the distance $d(i, j)$. Given a partial (illegal) solution $\mathcal{S}$, we use the evaluation function defined in Eq. (4) [18] to quantify the quality of $\mathcal{S}$.

$$f(\mathcal{S}) = \sum_{\{i,j\} \in E} max\{0, d(i, j) - |c(v_i) - c(v_j)|\} \quad (4)$$

This function basically measures the degree of constraint violations induced by a solution. Given two solutions $\mathcal{S}_1$ and $\mathcal{S}_2$, if $f(\mathcal{S}_1) < f(\mathcal{S}_2)$ (i.e., $\mathcal{S}_1$ has a smaller degree of constraint violations than $\mathcal{S}_2$), $\mathcal{S}_1$ is better than $\mathcal{S}_2$. If $f(\mathcal{S})$ equals to 0, $\mathcal{S}$ is a feasible solution.

Constrained neighborhood N: TS uses a constrained neighborhood N which can be described by the move operator $OneMove(v, i, j)$. Let $\mathcal{S}$ be a solution composed of L vertices $X = \{v^1, ..., v^L\}$. Let CV be the set of conflicting vertices such that $CV = \{v \in X$: the color of v is conflicting with the color of at least one vertex$\}$. The $OneMove$ operator changes the current color i of a conflicting vertex $v \in CV$ to another color j. Let $OneMove(v, i, j)$ designate such a move and $\mathcal{S} \oplus OneMove(v, i, j)$ be the resulting neighboring solution from $\mathcal{S}$. Then the neighborhood N of $\mathcal{S}$ is composed of all possible solutions that can be obtained by applying the $OneMove$ operator to $\mathcal{S}$, i.e.,

Algorithm 3 Tabu search repair procedure

Require: Graph G, color number k, partial illegal solution to be repaired $\mathcal{S}$, maximum number of launching TS $maxTSruns$
Ensure: a legal bandwidth k-coloring if found or the best solution found

```
S* ← S                    /* S* records the best solution found so far */
α ← 0                     /* Counts the number of launching TS */
while α < maxTSruns do
    /* Tabu search, see Section II-D2 */
    β ← 0                 /* Counts consecutive iterations failing to improve S* */
    repeat
        Select one best eligible move OneMove(v, i, j)
        S ← S ⊕ OneMove(v, i, j)
        Update the tabu list
        if f(S) = 0, i.e., S is legal coloring then
            return S
        end if
        if f(S) < f(S*), i.e., S is better than S* then
            S* ← S; β ← 0
        else
            β ← β + 1
        end if
    until β = maxIters
    /* The perturbation mechanism, see Section II-D3 */
    S ← Perturbation(S*) /* The perturbed solution becomes the starting point of the next TS run */
    α ← α + 1
end while
return S*
```

$$N(\mathcal{S}) = \{\mathcal{S} \oplus OneMove(v,i,j) \ : v \in CV\}$$

With this neighborhood, TS explores a much restricted space of $k^{|CV|}$ (instead of k^L) where CV is usually a very small subset of vertices of the current coloring $\mathcal{S}$. In order to render the neighborhood exploration as fast as possible, we adopt the incremental technique based on special data structures as explained in [7] to streamline the calculations.

Given this neighborhood, each iteration of TS selects the best $OneMove(v,i,j)$ operator among all the eligible candidate moves to be applied to the current solution. Ties are broken at random.

Tabu list management: The tabu list is introduced to record forbidden moves that have been performed in the recent past. Each time TS makes a move $OneMove(v,i,j)$, the pair (v,j) is added to the tabu list, meaning that it is forbidden to remove color j from vertex v for the next TT (tabu tenure) iterations. In our case, the tabu tenure tt is adaptively determined by $tt = f(\mathcal{S}) + random(10)$ [7], [9] where $f(\mathcal{S})$ is the value of the evaluation function defined in Eq. (4) and $random(10)$ is a random integer from 1 to 10. Moreover, a forbidden move is always accepted if it leads to a neighboring solution better than the best solution found so far (*aspiration criterion*).

3) The perturbation mechanism: The tabu search procedure described above can be trapped in a local optimum, leading to search stagnation. When this happens, we employ a simple perturbation strategy to change (or perturb) the current solution $\mathcal{S}$. Recall that $\mathcal{S}_*$ records the best solution found so far. The perturbation procedure always replaces $\mathcal{S}$ by $\mathcal{S}_*$ and thus uses $\mathcal{S}_*$ as the new starting point of the next round of TS.

The perturbation strategy is based on the consideration that $\mathcal{S}_*$ is a high quality configuration (i.e., having few conflicts) which could be close to a proper (legal) coloring. Using $\mathcal{S}_*$ as its starting point, the tabu search procedure will explore a new search trajectory and hopefully encounters a proper coloring.

E. Discussions

In this section, we discuss the relation of our LHS algorithm with the Forward Checking Coloration Neighborhood Search (FCNS) approach [21] which is probably the closest study for BCP. Indeed, both approaches employ forward checking to build a partial color assignment. However there are three notable differences between LHS and FCNS. First, unlike LHS, the heuristic used for vertex selection of FCNS does not integrate any learning technique. Second, when a dead-end is encountered (i.e., no color can be assigned to the vertex under consideration), LHS employs a tabu search procedure to repair the conflicting coloring while FCNS just uncolors one or more vertices in a heuristic way to resolve the conflict. Third, the process of dropping an on-going construction phase provides LHS with an opportunity of learning from previous failures while FCNS does not use such similar technique.

From a more general perspective, LHS is also related to the general GRASP metaheuristic [8] in the sense that both LHS and GRASP are composed of a construction phase and a local optimization phase. Yet, unlike GRASP which applies a local search procedure to improve a complete solution (complete color assignment in our case), the tabu search routine of LHS repairs partial (and conflicting) solutions.

As shown in the next section, the proposed LHS algorithm, equipped with its particular features, is a very competitive method for the bandwidth coloring problem.

III. Experimental Results

A. Benchmark instances and experimental protocol

Our LHS approach was tested on two sets of 66 well-known benchmark instances (33 graphs for BCP and 33 for BMCP [5]). These instances belong to three types: GEOMn, GEOMna and GEOMnb (where n denotes the number of vertices of the graph). The first type refers to sparse graphs, while the two other types correspond to dense graphs.

The LHS algorithm is coded in C++ and compiled using g++ with the '-O2' option on an Intel Xeon E5440 processor (2.83GHz and 4GB RAM). The run time required for solving the DIMACS machine benchmarks (available at: ftp://dimacs.rutgers.edu/pub/dsj/clique/) on our machine is 0.44, 2.63 and 9.85 seconds for graphs r300.5, r400.5 and r500.5 respectively. The computational results reported in this section were obtained with the parameter values shown in Table I. Given the stochastic nature of our LHS algorithm, each problem instance is independently solved 20 times. As explained in Section II, LHS solves the k-BCP problem by decreasing the k values. For the experiments reported in this paper, we set the initial value of k to be the best-known (i.e., the smallest) value k_* from the literature for all the graphs.

TABLE I
SETTINGS OF PARAMETERS

Parameter	Sect.	Description	Value
$maxTries$	II-A	Maximum number of tries	10^3
$maxIters$	II-D	Maximum number of non-improving moves for TS	10^4
$maxTSruns$	II-D	Maximum iterations of launching TS	50

B. Computational results of LHS on the BCP instances

This section is dedicated to an evaluation of the LHS's performance for the bandwidth coloring problem using the 33 BCP benchmark graphs. Columns 1–4 in Table II present the characteristics of the tested graphs and column k_* gives the current best-known results reported in the published literature and the unpublished paper [19]. The current best-known k_* reported in the published literature is also given in parentheses. Columns 6–8 present detailed computational results of our LHS algorithm: The best result in terms of the number of colors (k), the success rate (SR, number of runs over 20 to attain the best result k) and the average running time to reach k (t, in seconds).

From Table II, one observes that except for three small instances (indicated in italics), LHS can match the best-known results of the other 30 instances. Remarkably, LHS is able to improve the current best-known result for two hard instances (GEOM100b and GEOM110b indicated in bold). Furthermore, LHS achieves robust results with a success rate SR = 20/20 except for five cases (GEOM100a, GEOM100b, GEOM110a, GEOM110b and GEOM120b). Besides, Table II also lists the k value of these five graphs when SR = 20/20. The average running time of LHS ranges from 0 second to 1 hour. Each computing time corresponds to the average time for LHS to reach a legal coloring with the k value indicated in the table.

In order to further evaluate our LHS method, we compare our results with those obtained by four best performing algorithms in the literature: forward checking colouration neighbourhood search (FCNS) [21], evolutionary algorithm (EA) [20], multistart iterated tabu search (MITS) [18] and path relinking (PR) [19]. For this purpose, we restrict our attention to solution quality, i.e., the smallest k used to obtain a legal k-coloring. Computing times are included only for indicative purposes since there is no sense to compare the computing times of two methods if they achieve colorings with different k values. As one can observe in Table III, there are many such cases. Indeed, it is generally more difficult to find a legal k-coloring than a legal $(k+1)$-coloring. This is particularly true when k is close to the best-known value k_* (see for instance the cases GEOM100a, GEOM100b, GEOM110a and GEOM110b in Table III). Finally, the experimental platforms used by the reference algorithms are as follows: A 733MHz Pentium III PC for FCNS, a PIV 2.4MHz computer with 512 MB RAM for EA and a 2.8GHz computer with 4GB RAM for MITS and PR.

Table III presents the comparative results of LHS and the four reference methods (FCNS, EA, MITS and PR). The "-" marks for the reference MITS algorithm in Table III mean that MITS fails to reach the best-known result for the tested graph and the best obtained k is not reported for these graphs. From Table III, one observes that the reference algorithms can achieve the best reported k_* for 17, 24, 28 and 29 instances respectively, while LHS achieves the best-known results for 30 instances. Table III also discloses that LHS obtains no worse results than FCNS, EA, MITS and PR. More importantly, LHS can improve the best-known results in the literature for two instances (entries in bold). Finally, to find a legal coloring with the same k value, LHS requires comparable computing times with respect to FCNS and EA, and shorter times than MITS and PR for many cases. These outcomes provide evidence of the efficacy of our LHS approach for BCP.

C. Computational results of LHS on the BMCP instances

We turn now our attention to an evaluation of the LHS algorithm on the bandwidth multicoloring problem using the set of 33 "GEOM" benchmark instances for BMCP. For this purpose, we first present our detailed computational results, and then show a comparison between LHS and the five state-of-the-art algorithms OF-SW [4], FCNS [21], EA [20], MITS [18] and PR[19].

Table IV shows the detailed characteristics of each graph, the best-known result k_* (the best-known k_* in the published literature is also given in parentheses) and the result of our LHS approach. In addition to the best colors obtained (k) with the average running time to reach k (t in seconds), we also provide the success rate (SR) of LHS for attaining the best result k. Furthermore, we list the k value for each graph when LHS could achieve the robust results with a SR = 20/20. Table V reports the comparative results between LHS and the five reference algorithms. The experimental platform used by OF-SW algorithm is a 2GHz AMD Athlon MP 2400+ processor with 256 KB cache and 1 GB RAM, and the platforms of the other four reference algorithms are the same as given in Section III-B.

From the results in Table IV, we observe that our LHS approach can match the best-known k_* for all 33 instances. More importantly, LHS finds an improved best result for 12 out of 33 instances (entries in bold). This is remarkable given that 14 of these 33 best-known results were reported very recently in the unpublished work [19]. In particular, LHS can consistently achieve the best-known k_* in the literature with a perfect success rate for all graphs except GEOM80a, GEOM90a and GEOM110b. Besides, the indicated computing time corresponds to the average time for LHS to reach a legal coloring with the k value indicated in the table. From Table IV, one also observes that the running time increases when k decreases since this makes the problem more difficult to solve.

Table V lists the comparative results of LHS and the five reference algorithms (OF-SW, FCNS, EA, MITS and PR). The "-" marks for the OF-SW algorithm mean that no result is available for the concerned graphs. From Table V, one

TABLE II
LHS: DETAILED COMPUTATIONAL RESULTS ON BCP INSTANCES

Characteristics of the graphs				k_*	LHS		
Name	$\|V\|$	$\|E\|$	Den		k	SR	$t(s)$
GEOM20	20	20	0.1053	20	*21*	20/20	0.0
GEOM20a	20	37	0.1947	20	20	20/20	0.0
GEOM20b	20	32	0.1684	13	13	20/20	0.0
GEOM30	30	50	0.1149	27	*28*	20/20	0.0
GEOM30a	30	81	0.1862	27	27	20/20	0.0
GEOM30b	30	81	0.1862	26	26	20/20	0.0
GEOM40	40	78	0.1000	27	*28*	20/20	0.0
GEOM40a	40	146	0.1872	37	37	20/20	0.0
GEOM40b	40	157	0.2013	33	33	20/20	0.0
GEOM50	50	127	0.1037	28	28	20/20	0.0
GEOM50a	50	238	0.1943	50	50	20/20	0.1
GEOM50b	50	249	0.2033	35	35	20/20	1.2
GEOM60	60	185	0.1045	33	33	20/20	0.0
GEOM60a	60	339	0.1915	50	50	20/20	0.1
GEOM60b	60	366	0.2068	41	41	20/20	214.7
GEOM70	70	267	0.1106	38	38	20/20	0.0
GEOM70a	70	459	0.1901	61	61	20/20	23.7
GEOM70b	70	488	0.2020	47	47	20/20	665.4
GEOM80	80	349	0.1104	41	41	20/20	0.1
GEOM80a	80	612	0.1936	63	63	20/20	6.6
GEOM80b	80	663	0.2098	60	60	20/20	19.9
GEOM90	90	441	0.1101	46	46	20/20	0.0
GEOM90a	90	789	0.1970	63	63	20/20	23.8
GEOM90b	90	860	0.2107	69	69	20/20	779.2
GEOM100	100	547	0.1105	50	50	20/20	1.0
GEOM100a	100	992	0.2000	67	67	8/20	1557.4
					68	20/20	189.8
GEOM100b	100	1050	0.2121	72	**71**	12/20	2038.6
					72	20/20	759.6
GEOM110	110	638	0.1064	50	50	20/20	1.3
GEOM110a	110	1207	0.2013	71(72)	71	19/20	2218.7
					72	20/20	324.2
GEOM110b	110	1256	0.2095	78	**77**	10/20	2598.7
					78	20/20	94.3
GEOM120	120	773	0.1083	59	59	20/20	0.5
GEOM120a	120	1434	0.2000	82	82	20/20	171.1
GEOM120b	120	1491	0.2088	84	84	2/20	3568.1
					85	20/20	1829.7

observes that OF-SW, FCNS, EA and MITS reach the best-known results for 6, 8, 13 and 18 cases respectively. While both the unpublished PR algorithm and our LHS algorithm attain the best-known results for all 33 instances (in italics), LHS requires a much shorter computing time for most instances. More importantly, our LHS algorithm can find an improved best solution for 12 instances (in bold).

Once again, we do not emphasize on computing time since the compared approaches provide results with different k values. The computing times of FCNS [21] are much shorter, but its results are much worse in terms of solution quality. To find solutions of the same quality (with the same k), LHS does not consume more time than OF-SW [4] and EA [20]. Compared to the most recent and the two best performing algorithms MITS [18] and PR [19], LHS requires less computing times to find equal or better solutions.

In summary, LHS competes very favorably with the five high performing reference algorithms in the literature for BMCP.

IV. DISCUSSION

In this section, we perform an additional experiment to assess the impact of the learning-based guiding function $\mathcal{F}$ defined in Eq. (3) (Section II-B) which is a key element of the LHS approach.

Indeed, as explained in Section II-A, the construction phase uses the guiding function $\mathcal{F}$ to decide the next vertex for color assignment. This experiment aims to show its influence to the performance of LHS. For this purpose, we compare LHS (with $\mathcal{F}$ of Eq. (3)) and a LHS variant called LHS$_{random}$. LHS$_{random}$ discards the guiding function $\mathcal{F}$ and selects randomly vertices for color assignment.

For this experiment, we focus on 24 most difficult and challenging benchmark instances for BMCP. With the same experimental protocol, we run 20 times each of these two LHS procedures (LHS and LHS$_{random}$) to solve the 24 BMCP benchmark instances. The computational outcomes are reported in Table VI.

From Table VI, one observes that LHS achieves always a better or equal result compared to LHS$_{random}$. In particular, the result of LHS is better for 15 out of the 24 instances, i.e., LHS requires a smaller number of colors to find a legal coloring, with a color reduction ranging from 1 up to 8. Given that finding a legal k-coloring with k close to the best-known value k^* is already a difficult task, the improvement of LHS over LHS$_{random}$ is significant. Moreover, LHS requires less average time to reach its best solutions compared to LHS$_{random}$. In summary, discarding the guiding function $\mathcal{F}$ makes LHS less effective and weakens its performance.

Finally, one notices that even the weakened LHS$_{random}$ procedure is competitive compared with the best existing methods since LHS$_{random}$ is able to match the best-known results in most cases and even finds two improved best results (indicated in bold). Thus this experiment indirectly shows the

TABLE III
COMPARISONS WITH FOUR STATE-OF-THE-ART ALGORITHMS ON BCP INSTANCES

Graph		FCNS [21]		EA [20]		MITS [18]		PR [19]		LHS	
Name	k_*	k	$t(s)$	k	$t(s)$	k	$t(s)$	k	$t(s)$	k	$t(s)$
GEOM20	20	21	0.0	21	0.0	-	-	21	0.0	21	0.0
GEOM20a	20	*20*	0.0	*20*	0.0	*20*	0.0	*20*	0.0	*20*	0.0
GEOM20b	13	*13*	0.0	*13*	0.0	*13*	0.0	*13*	0.0	*13*	0.0
GEOM30	27	28	0.0	28	0.0	-	-	28	0.0	28	0.0
GEOM30a	27	*27*	0.0	*27*	0.0	*27*	0.0	*27*	0.0	*27*	0.0
GEOM30b	26	*26*	0.0	*26*	0.0	*26*	0.0	*26*	0.0	*26*	0.0
GEOM40	27	28	0.0	28	0.0	-	-	28	0.0	28	0.0
GEOM40a	37	*37*	0.0	*37*	0.0	*37*	0.0	*37*	0.0	*37*	0.0
GEOM40b	33	*33*	0.0	*33*	0.0	*33*	0.0	*33*	0.0	*33*	0.0
GEOM50	28	*28*	0.0	*28*	0.0	*28*	0.0	*28*	0.0	*28*	0.0
GEOM50a	50	*50*	2.0	*50*	0.0	*50*	0.0	*50*	0.0	*50*	0.1
GEOM50b	35	*35*	0.0	*35*	0.0	*35*	3.0	*35*	1.0	*35*	1.2
GEOM60	33	*33*	0.0	*33*	0.0	*33*	0.0	*33*	0.0	*33*	0.0
GEOM60a	50	*50*	1.0	*50*	0.0	*50*	1.0	*50*	0.0	*50*	0.1
GEOM60b	41	43	0.0	*41*	29.0	*41*	277.0	*41*	105.0	*41*	214.7
GEOM70	38	*38*	0.0	*38*	0.0	*38*	0.0	*38*	0.0	*38*	0.0
GEOM70a	61	62	2.0	*61*	12.0	*61*	45.0	*61*	47.0	*61*	23.7
GEOM70b	47	48	1.0	48	52.0	*47*	8685.0	*47*	6678.0	*47*	665.4
GEOM80	41	*41*	0.0	*41*	0.0	*41*	0.0	*41*	0.0	*41*	0.1
GEOM80a	63	*63*	12.0	*63*	150.0	*63*	21.0	*63*	12.0	*63*	6.6
GEOM80b	60	61	0.0	*60*	145.0	*60*	322.0	*60*	191.0	*60*	19.9
GEOM90	46	*46*	3.0	*46*	0.0	*46*	0.0	*46*	0.0	*46*	0.0
GEOM90a	63	64	2.0	*63*	150.0	*63*	230.0	*63*	191.0	*63*	23.8
GEOM90b	69	72	2.0	70	1031.0	*69*	20144.0	*69*	23560.0	*69*	779.2
GEOM100	50	*50*	0.0	*50*	2.0	*50*	2.0	*50*	2.0	*50*	1.0
GEOM100a	67	68	9.0	68	273.0	*67*	11407.0	*67*	5556.0	*67*	1557.4
										68	189.8
GEOM100b	72	73	15.0	73	597.0	*72*	24561.0	*72*	41832.0	**71**	2038.6
										72	759.6
GEOM110	50	*50*	4.0	*50*	3.0	*50*	2.0	*50*	5.0	*50*	1.3
GEOM110a	71(72)	73	7.0	72	171.0	72	1529.0	*71*	5140.0	*71*	2218.7
										72	324.2
GEOM110b	78	79	2.0	*78*	676.0	*78*	24416.0	*78*	18136.0	**77**	2598.7
										78	94.3
GEOM120	59	60	4.0	*59*	0.0	*59*	1.0	*59*	2.0	*59*	0.5
GEOM120a	82	84	4.0	84	614.0	*82*	34176.0	*82*	62876.0	*82*	171.1
GEOM120b	84	86	9.0	*84*	857.0	-	-	85	66301.0	*84*	3568.1
										85	1829.7

interest of the general cooperative approach of LHS which combines a construction procedure and a repair procedure.

V. CONCLUSION

In this paper, we presented the Learning-based Hybrid Search approach for the Bandwidth Coloring Problem (BCP) and the Bandwidth MultiColoring Problem (BMCP). LHS alternates between an informed construction phase and a repair procedure until attaining a feasible solution. The construction phase is guided by a learning-based function to choose the next vertex for color assignment and applies a forward checking technique to eliminate incompatible colors for unassigned vertices. The tabu search based repair procedure is used to resolve dead-end situations when the construction phase cannot further extend the current partial solution.

Experimental evaluations on two sets of 66 benchmark instances showed that the proposed LHS approach is highly competitive in comparison with the current most effective algorithms for BCP and BMCP. LHS can reach the best-known results for most benchmarks of both BCP and BMCP. In particular, LHS improves the best-known results for two BCP instances and 12 BMCP instances.

One observes that the transformation from bandwidth multi-coloring instances to bandwidth coloring instances introduces extra symmetry in the resulting BCP graph. Studies on the impact of these extra symmetries on search algorithms are reported in [20], [21]. An interesting perspective to improve the proposed LHS algorithm would be to integrate dedicated symmetry handling techniques. For instance, special terms may be built into the guiding function to distinguish a transformed vertex from an original vertex. To further consider coloring symmetries during the repairing phase, it would be advantageous to replace the tabu search routine by a memetic algorithm using a grouping crossover operator like the greedy partition crossover (GPX) introduced in [9].

Finally, the general LHS algorithm follows a new framework which is different from the existing approaches. In the future, we hope to investigate its usefulness for solving other constrained combinatorial problems.

ACKNOWLEDGMENT

We are grateful to the anonymous referees for valuable suggestions and comments which helped us to improve the paper. The work is partially supported by the Region of Pays de la Loire (France) via the RaDaPop (2009-2013) and LigeRO (2009-2013) projects, the FMJH Foundation via the PGMO program (2014-2015). Support for Yan Jin from the China Scholarship Council is also acknowledged.

REFERENCES

[1] K.I. Aardal, C.P.M. van Hoesel, A.M.C.A. Koster, C. Mannino and A. Sassano. Models and solution techniques for the frequency assignment problem. *Annals of Operations Research*, 153(1): 79–129, 2007.

TABLE IV
DETAILED COMPUTATIONAL RESULTS OF LHS ON THE SET OF 33 BMCP INSTANCES

Characteristics of the graphs				k_*	LHS		
Name	$\|V\|$	$\|E\|$	Den		k	SR	$t(s)$
GEOM20	118	1048	0.1518	149	149	20/20	1.8
GEOM20a	100	1327	0.2681	169	169	20/20	0.5
GEOM20b	40	132	0.1692	44	44	20/20	0.0
GEOM30	143	1419	0.1398	160	160	20/20	0.1
GEOM30a	171	3288	0.2262	209	209	20/20	16.2
GEOM30b	69	447	0.1905	77	77	20/20	0.0
GEOM40	220	3074	0.1276	167	167	20/20	0.2
GEOM40a	203	4473	0.2182	213	213	20/20	9.0
GEOM40b	84	743	0.2131	74	74	20/20	1.5
GEOM50	285	4935	0.1219	224	224	20/20	0.3
GEOM50a	302	9649	0.2123	312(314)	**311**	20/20	1452.6
					312	20/20	307.5
GEOM50b	104	1140	0.2128	83	83	20/20	72.1
GEOM60	315	6174	0.1248	258	258	20/20	1.3
GEOM60a	362	13105	0.2005	354(356)	**353**	2/20	9007.1
					354	20/20	4996.0
GEOM60b	127	1785	0.2231	113	113	20/20	910.7
GEOM70	384	8584	0.1167	266(270)	266	20/20	2534.0
GEOM70a	379	14821	0.2069	466(467)	**465**	6/20	36604.9
					466	20/20	12622.1
GEOM70b	148	2212	0.2033	116	**115**	3/20	3640.7
					116	20/20	1844.7
GEOM80	465	12927	0.1198	380(381)	**379**	20/20	357.8
					380	20/20	164.0
GEOM80a	389	15545	0.2060	358(361)	**357**	2/20	43403.0
					358	7/20	25852.0
					360	20/20	13302.9
GEOM80b	169	3028	0.2133	138(139)	138	20/20	46.5
GEOM90	530	16180	0.1154	328(330)	328	20/20	162.2
GEOM90a	454	20455	0.1989	372(375)	372	3/20	16782.1
					373	20/20	3164.5
GEOM90b	184	3602	0.2139	144	**142**	7/20	7680.8
					143	20/20	4858.5
					144	20/20	1331.8
GEOM100	581	19829	0.1177	404	404	20/20	64.9
GEOM100a	528	28496	0.2048	436(442)	**429**	1/20	78363.1
					434	20/20	10767.1
					436	20/20	2310.7
GEOM100b	200	4429	0.2226	156	**153**	1/20	10840.1
					155	20/20	3147.6
					156	20/20	726.3
GEOM110	643	24799	0.1201	375(381)	375	20/20	1598.8
GEOM110a	602	38783	0.2144	482(488)	**478**	1/20	49457.1
					480	20/20	2921.5
					482	20/20	375.3
GEOM110b	220	5163	0.2143	201(204)	201	3/20	5388.4
					203	20/20	303.1
GEOM120	680	27759	0.1202	396	396	20/20	626.1
GEOM120a	664	46429	0.2109	539(554)	**536**	2/20	69518.6
					539	20/20	20286.1
GEOM120b	235	5779	0.2102	189	**187**	8/20	8025.8
					188	20/20	1642.0
					189	20/20	315.2

[2] S.M. Allen, D.H. Smith and S. Hurley. Generation of lower bounds for minimum span frequency assignment. *Discrete Applied Mathematics*, 119(1–2): 59–78, 2002.

[3] F. Bernardo, R. Agustí, J. Pérez-Romero, O. Sallent. An application of reinforcement learning for efficient spectrum usage in next-generation mobile cellular networks systems. *IEEE Transactions on Man, and Cybernetics, Part C: Applications and Reviews*, 40(4): 477–484, 2010.

[4] M. Chiarandini and T. Stützle. Stochastic local search algorithms for graph set T-colouring and frequency assignment. *Constraints*, 12(3): 371–403, 2007.

[5] COLOR02/03/04: Graph Coloring and its Generalizations http://mat.gsia.cmu.edu/COLOR02/.

[6] D. Costa. On the use of some known methods for T-coloring of graphs. *Annals of Operations Research*, 41(4): 343–358, 1993.

[7] R. Dorne and J.K. Hao. Tabu search for graph coloring, T-colorings and set T-colorings. *Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Chapter 6, pp77–92, S. Voss, S. Martello, I.H. Osman, C. Roucairol (Eds.), Kluwer, 1998.

[8] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6: 109–133, 1995.

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

[10] A. Gamst. Some lower bounds for a class of frequency assignment problems. *IEEE Transactions of Vehicular Technology*, 35(1): 8–14, 1986.

[11] M.R. Garey and D.S. Johnson. Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman and Company, San Francisco, 1979.

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

[13] W.K. Hale. Frequency assignment: theory and applications. *Proceedings of the IEEE*, 68(12): 1497–1514, 1980.

[14] J.K. Hao, R. Dorne and P. Galinier. Tabu search for frequency assignment in mobile radio networks. *Journal of Heuristics*, 4(1): 47–62, 1998.

[15] R.M. Haralick and G.L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14(3): 263–313, 1980.

[16] A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 39: 345–351, 1987.

[17] D.S. Johnson, A. Mehrotra, and M.A. Trick (Eds.). Special issue on computational methods for graph coloring and its generalizations. *Discrete Applied Mathematics*, 156(2): 145–156, 2008.

[18] X. Lai, and Z. Lü. Multistart iterated tabu search for bandwidth coloring problem. *Computers and Operations Research*, 40(5): 1401–1409, 2013.

TABLE V
COMPARISONS OF LHS WITH FIVE STATE-OF-THE-ART ALGORITHMS ON THE SET OF 33 BMCP INSTANCES

Graph		OF-SW [4]		FCNS [21]		EA [20]		MITS [18]		PR [19]		LHS	
Name	k_*	k	$t(s)$	k	$t(s)$	k	$t(s)$	k	$t(s)$	k	$t(s)$	k	$t(s)$
GEOM20	149	-	-	*149*	4.0	*149*	18.0	*149*	2.0	*149*	1.0	*149*	1.8
GEOM20a	169	-	-	170	2.0	*169*	9.0	*169*	15.0	*169*	7.0	*169*	0.5
GEOM20b	44	*44*	30.0	*44*	0.0	*44*	5.0	*44*	0.0	*44*	0.0	*44*	0.0
GEOM30	160	-	-	*160*	0.0	*160*	1.0	*160*	0.0	*160*	0.0	*160*	0.1
GEOM30a	209	*209*	380.0	214	11.0	210	954.0	*209*	10.0	*209*	26.0	*209*	16.2
GEOM30b	77	*77*	80.0	*77*	0.0	*77*	0.0	*77*	0.0	*77*	0.0	*77*	0.0
GEOM40	167	-	-	*167*	1.0	*167*	20.0	*167*	0.0	*167*	1.0	*167*	0.2
GEOM40a	213	214	500.0	217	299.0	214	393.0	*213*	328.0	*213*	133.0	*213*	9.0
GEOM40b	74	*74*	140.0	*74*	4.0	*74*	1.0	*74*	2.0	*74*	4.0	*74*	1.5
GEOM50	224	-	-	*224*	1.0	*224*	1197.0	*224*	8.0	*224*	2.0	*224*	0.3
GEOM50a	312(314)	315	1080.0	323	51.0	316	4675.0	314	40373.0	*312*	270860.0	**311**	1452.6
												312	307.5
GEOM50b	83	84	200.0	86	1.0	*83*	197.0	*83*	1200.0	*83*	723.0	*83*	72.1
GEOM60	258	*258*	710.0	*258*	77.0	*258*	139.0	*258*	19.0	*258*	23.0	*258*	1.3
GEOM60a	354(356)	356	1420.0	373	10.0	357	8706.0	356	38570.0	*354*	34580.0	**353**	9007.1
												354	4996.0
GEOM60b	113	117	300.0	116	12.0	115	460.0	*113*	104711.0	*113*	63579.0	*113*	910.7
GEOM70	266(270)	267	1060.0	277	641.0	272	1413.0	270	7602.0	*266*	130844.0	*266*	2534.0
GEOM70a	466(467)	478	1470.0	482	315.0	473	988.0	467	38759.0	*466*	6952.0	**465**	36604.9
												466	12622.1
GEOM70b	116	120	380.0	119	55.0	117	897.0	*116*	213545.0	*116*	26110.0	**115**	3640.7
												116	1844.7
GEOM80	380(381)	382	1490.0	398	361.0	388	132.0	381	212213.0	*380*	34493.0	**379**	357.8
												380	164.0
GEOM80a	358(361)	360	1510.0	380	109.0	363	8583.0	361	41235.0	*358*	41772.0	**357**	43403.0
												360	13302.9
GEOM80b	138(139)	139	490.0	141	37.0	141	1856.0	139	255.0	*138*	705.0	*138*	46.5
GEOM90	328(330)	334	1810.0	339	44.0	332	4160.0	330	4022.0	*328*	134941.0	*328*	162.2
GEOM90a	372(375)	377	1910.0	382	13.0	382	5334.0	375	10427.0	*372*	282456.0	*372*	16782.1
												373	3164.5
GEOM90b	144	147	590.0	147	303.0	*144*	1750.0	*144*	211366.0	*144*	14648.0	**142**	7680.8
												144	1331.8
GEOM100	404	*404*	2170.0	424	7.0	410	3283.0	*404*	40121.0	*404*	16355.0	*404*	64.9
GEOM100a	436(442)	437	2500.0	461	26.0	444	12526.0	442	381.0	*436*	9108.0	**429**	78363.1
												436	2310.7
GEOM100b	156	159	690.0	159	367.0	*156*	3699.0	*156*	213949.0	*156*	86308.0	**153**	10840.1
												156	726.3
GEOM110	375(381)	378	2510.0	392	43.0	383	2344.0	381	183.0	*375*	25401.0	*375*	1598.8
GEOM110a	482(488)	490	3120.0	500	29.0	490	2318.0	488	926.0	*482*	9819.0	**478**	49457.1
												482	375.3
GEOM110b	201(204)	208	790.0	208	5.0	206	480.0	204	944.0	*201*	47653.0	*201*	5388.4
												203	303.1
GEOM120	396	397	2730.0	417	9.0	*396*	2867.0	-	-	*396*	15341.0	*396*	626.1
GEOM120a	539(554)	549	3690.0	565	41.0	559	3873.0	554	1018.0	*539*	45147.0	**536**	69518.6
												539	20286.1
GEOM120b	189	191	910.0	196	3.0	191	3292.0	*189*	213989.0	*189*	14371.0	**187**	8025.8
												189	315.2

TABLE VI
ASSESSMENT OF THE LEARNING-BASED GUIDING FUNCTION

Graph		LHS_{random}			LHS		
Name	k_*	k	SR	$t(s)$	k	SR	$t(s)$
GEOM50	224	*224*	20/20	0.3	*224*	20/20	0.3
GEOM50a	312(314)	313	2/20	28288.6	**311**	20/20	1452.6
GEOM50b	83	*83*	16/20	3492.4	*83*	20/20	72.1
GEOM60	258	*258*	20/20	1.6	*258*	20/20	1.3
GEOM60a	354(356)	*354*	7/20	12349.2	**353**	2/20	9007.1
GEOM60b	113	*113*	1/20	3549.1	*113*	20/20	910.7
GEOM70	266(270)	*266*	7/20	14235.4	*266*	20/20	2534.0
GEOM70a	466(467)	*466*	9/20	30681.1	**465**	6/20	36604.9
GEOM70b	116	117	3/20	7880.7	**115**	3/20	3640.7
GEOM80	380(381)	**379**	6/20	18108.5	**379**	20/20	357.8
GEOM80a	358(361)	360	1/20	64551.1	**357**	2/20	43403.0
GEOM80b	138(139)	*138*	20/20	529.6	*138*	20/20	46.5
GEOM90	328(330)	*328*	3/20	15547.1	*328*	20/20	162.2
GEOM90a	372(375)	373	3/20	26154.4	*372*	3/20	16782.1
GEOM90b	144	145	3/20	11794.4	**142**	7/20	7680.8
GEOM100	404	*404*	20/20	302.8	*404*	20/20	64.9
GEOM100a	436(442)	437	3/20	45299.8	**429**	1/20	78363.1
GEOM100b	156	159	4/20	7565.8	**153**	1/20	10840.1
GEOM110	375(381)	*375*	3/20	32435.4	*375*	20/20	1598.8
GEOM110a	482(488)	**481**	6/20	24425.6	**478**	1/20	49457.1
GEOM110b	201(204)	202	1/20	15397.1	*201*	3/20	5388.4
GEOM120	396	*396*	20/20	14012.6	*396*	20/20	626.1
GEOM120a	539(554)	542	1/20	76969.1	**536**	2/20	69518.6
GEOM120b	189	190	2/20	12950.6	**187**	8/20	8025.8

[19] X. Lai, Z. Lü, J.K. Hao, F. Glover and L. Xu. Path relinking for bandwidth coloring problem. Unpublished working paper, November 2013. Posted on CoRR 3 Sep 2014. http://arxiv.org/abs/1409.0973

[20] E. Malaguti and P. Toth. An evolutionary approach for bandwidth multicoloring problems. *European Journal of Operational Research*, 189(3): 638–651, 2008.

[21] S. Prestwich. Generalized graph colouring by a hybrid of local search and constraint programming. *Discrete Applied Mathematics*, 156(2): 148–158, 2008.

[22] F.S. Roberts. T-colorings of graphs: recent results and open problems. *Discrete Mathematics*, 93(2-3): 229–245, 1991.

[23] S. Salcedo-Sanz, R. Santiago-Mozos, and C. Bousoño-Calzón. A hybrid Hopfield network-simulated annealing approach for frequency assignment in satellite communications systems. *IEEE Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(2): 1108–1116, 2004.

[24] J.P. Walser. Feasible cellular frequency assignment using constraint programming abstractions. In: *Proceeding of the Workshop on Constraint Programming Applications (CP96)*. Cambridge, MA, USA, 1996. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.5731

[25] L. Wang, W. Liu, and H.X. Shi. Noisy chaotic neural networks with variable thresholds for the frequency assignment problem in satellite communications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2): 209–217, 2008.

[26] J.A. Zoellner and C.L. Beall. A breakthrough in spectrum conserving frequency assignment technology. *IEEE Transaction on Electromagnetic Compatibility*, 19(3): 313–319, 1977.