Computers & Operations Research ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

Multistart Iterated Tabu Search for Bandwidth Coloring Problem

Xiangjing Lai [a,b], Zhipeng Lü [b,*]

[a] School of Information Engineering, Jiangxi University of Science and Technology, 341000 Ganzhou, PR China
[b] School of Computer Science and Technology, Huazhong University of Science and Technology, 430074 Wuhan, PR China

ARTICLE INFO

Keywords:
Bandwidth Coloring
Bandwidth MultiColoring
Frequency Assignment
Iterated Tabu Search
Metaheuristics
Perturbation operator

ABSTRACT

This paper presents a Multistart Iterated Tabu Search (MITS) algorithm for solving Bandwidth Coloring Problem (BCP) and Bandwidth MultiColoring Problem (BMCP). The proposed MITS algorithm exhibits several distinguishing features, such as integrating an Iterated Tabu Search (ITS) algorithm with a multistart method and a problem specific perturbation operator. Tested on two sets of 66 public benchmark instances widely used in the literature, the MITS algorithm achieves highly competitive results compared with the best performing algorithms, improving the previous best known results for 22 instances while matching the previous best known results for 39 ones. Furthermore, two important features of the proposed algorithm are analyzed.

1. Introduction

Given an undirected graph $G=(V,E)$ with a set V of n vertices and an edge set E, the classical vertex coloring problem (VCP) consists of assigning a color c_i $(1 \le c_i \le k)$ to each vertex i $(1 \le i \le n)$ in graph G such that adjacent vertices receive different colors and the number of colors used, k, is minimized.

The Bandwidth Coloring Problem (BCP) is a generalization of VCP, where the distance constraint is imposed on each pair of adjacent vertices, i.e., for edge $e(i,j)$ the absolute value of the difference between the colors of vertices i and j must be larger than or equal to the associated edge weight $d(i,j)$, i.e., $|c_i-c_j| \ge d(i,j)$. Thus, it is obvious that the classical VCP is a special case of BCP with $d(i,j)=1$ for all the edges in the graph.

The Bandwidth MultiColoring Problem (BMCP) is a generalization of BCP, where each vertex i is associated with a vertex weight $w(i)$ and each edge $e(i,j)$ is associated with an edge weight $d(i,j)$. The objective of BMCP is to assign $w(i)$ distinct colors from 1 to k to vertex i, such that for each edge $e(i,j)$ the difference between any colors of vertices i and j must be not less than the associated edge weight $d(i,j)$. Here, $i=j$ implies that the difference between any two colors of vertex i must be at least $d(i,i)$. The objective of the BMCP problem is to minimize the number of colors used k. Therefore, one can observe that BCP is a special case of BMCP with $w(i)=1$ for all the vertices.

It should be noted that as suggested in [1,2], BMCP can be converted into BCP by splitting each vertex i into a clique with cardinality $w(i)$, with each edge of the clique having an edge weight $d(i,i)$, corresponding to the weight of the loop edge of vertex i in the original graph. The obtained new graph will have $\sum_{i=1}^{n} w(i)$ vertices. Thus, we only consider the BCP problem in this study.

Given the NP-hard nature of VCP, it can be concluded that BCP and BMCP are both NP-hard problems, since they generalize the VCP problem.

There are some important real-world applications that can be modeled by BCP and BMCP. For example, the minimum span frequency assignment problem (MS-FAP) is equivalent to BMCP. MS-FAP concerns allocation of frequencies, with the aim of minimizing the span, i.e., the difference between the maximum and minimum used frequencies. In BMCP, the vertices correspond to transmitters of MS-FAP, the weight $w(i)$ of vertex i corresponds to the number of frequencies that have to assign to the transmitter associated with vertex i, and the edge weight corresponds to the minimum distance between frequencies that can be used by adjacent transmitters. More information about frequency assignment problems (FAP) can be found at an FAP web site [3].

In order to solve BCP, BMCP and MS-FAP, a large number of successful methods were proposed in the literature. In the studies of MS-FAP, various metaheuristic algorithms were designed, such as greedy algorithms [4], local search [5], tabu search [6,7], simulated annealing [6], genetic algorithm [8], and constraint programming approaches [9]. In 2002, to promote research on BCP, BMCP, etc., a computational symposium on graph coloring and its generalizations was organized (see [10] for more details). In the symposium, Phan and Skiena designed a general heuristic, called Discropt for VCP and BCP [11]; Prestwich tackled BCP and BMCP by using a hybrid method, called FCNS, which integrates local search and constraint propagation [12]. In a subsequent work, an extended version of FCNS was developed by the same

* Corresponding author.
E-mail addresses: zhipeng.lv@hust.edu.cn, zhipeng.lui@gmail.com (Z. Lü).

0305-0548/$ - see front matter Crown Copyright © 2012 Published by Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.09.003

Please cite this article as: Lai X, Lü Z. Multistart Iterated Tabu Search for Bandwidth Coloring Problem. Computers and Operations Research (2012), http://dx.doi.org/10.1016/j.cor.2012.09.003

author [13]. In [1,14], Lim et al. developed two hybrid methods for solving VCP and its generalizations, including BCP and BMCP. Recently, Malaguti and Toth proposed an evolutionary approach for solving BCP and tested it on two sets of benchmark instances widely used in the literature, producing impressive results [2,15]. In a more recent work [16], Martí et al. proposed several heuristic approaches for BCP in order to study the use of memory structures in both constructive and improvement methods.

This paper presents a Multistart Iterated Tabu Search (MITS) algorithm, a hybrid metaheuristic algorithm integrating a tabu search procedure, iterated local search (ILS) [17,18] and a multistart method for solving the BCP and BMCP problems. The proposed algorithm is evaluated on two sets of benchmarks commonly used in the literature, and the previous best known results are improved for 22 out of 66 instances, showing the effectiveness of our proposed method.

The rest of this paper is organized as follows. Section 2 describes our MITS algorithm in detail. In Section 3, computational results are presented and compared with several reference algorithms from the literature. Two important features of the proposed algorithm are discussed and analyzed in Section 4. Finally, Section 5 is dedicated to the conclusion of the paper.

2. Multistart Iterated Tabu Search

BCP can be solved from the point of view of constraint satisfaction by solving a series of BCP problems with a given number of colors k. Starting from an initial number of colors k which is large enough, our algorithm solves the k-BCP problem. As soon as the k-BCP problem is solved, we decrease k to $k-1$ and solve again the k-BCP problem. This process is repeated until no legal k-coloring can be found. Therefore, we will only consider the k-BCP problem in this paper. It is clear that a smaller k for a given graph leads to a harder k-BCP problem and our algorithm thus solves a series of problems of increasing difficulty.

2.1. Search space and evaluation function

When designing a search algorithm for solving combinatorial optimization problems, it is indispensable to define a search space. For BCP, given the value of k, the search space used in our algorithm contains all possible k-colorings, including legal and illegal k-colorings. A legal k-coloring satisfies all the distance constraints on any two adjacent vertices, while an illegal k-coloring violates some distance constraints. Thus, the size of the whole search space is equal to k^n, where n is the number of vertices in the graph.

Given a k-coloring of a BCP problem, denoted by S, the evaluation function that we use to measure the degree of constraint violations can be defined as

$$f(S)=\sum_{e(i,j)\in E}\max\{0,d(i,j)-|c_i-c_j|\} \quad (1)$$

where $d(i,j)$ is the edge weight for edge $e(i,j)$, and c_i and c_j respectively represent colors of vertices i and j. Therefore, a solution S with $f(S)=0$ corresponds to a legal k-coloring.

2.2. Main scheme and initial solution

Iterated Tabu Search (ITS) is known to be a highly effective metaheuristic framework for solving a large number of combinatorial optimization problems [19–22], which combines ILS with a tabu search (TS) algorithm. Starting from an initial solution, a TS procedure is performed until a local optimum solution is reached. Then, the obtained local optimum solution is destructed by a perturbation operator and another round of TS procedure is performed from the destructed solution. Finally, an acceptance criterion is used to decide whether the new local optimum solution is accepted as the initial solution for next round of TS. This process is repeated until a stopping condition is met.

Our MITS algorithm adapts an ITS algorithm in a multistart fashion and its main scheme can be simply described in Algorithm 1. The initial solution of the ITS algorithm embedded in our MITS algorithm is randomly generated, i.e., each vertex in the graph is randomly assigned a color from 1 to k. The stopping condition of our MITS algorithm is that the timeout limit is reached or a legal k-coloring is found.

Algorithm 1. Multistart Iterated Tabu Search algorithm.

```
1:  Input: Graph G=(V,E), the number of colors (k) and the
    distance constraint matrix D_{n×n}
2:  Output: the best k-coloring found so far
3:  repeat
4:    S^0 ← Initial_solution, β ← 0    /* Section 2.2*/
5:    S' ← Tabu_Search(S^0)    /* Section 2.3*/
6:    repeat
7:      S* ← Perturbation_Operator(S') /* Section 2.4*/
8:      S*' ← Tabu_Search(S*) /* Section 2.3*/
9:      if f(S*') = 0 then
10:       return S*'
11:     end if
12:     if f(S*') < f(S') then
13:       β ← 0
14:     else
15:       β ← β+1
16:     end if
17:     S' ← Acceptance_Criterion(S*', S')    /* Section 2.5 */
18:   until β reaches a predetermined value β_max   /*
      Section 2.5 */
19: until The timeout limit is reached
```

2.3. Tabu Search

The TS algorithm was first proposed by Glover [23] and was widely used for solving a large number of combinatorial optimization problems [24,25]. It typically incorporates a tabu list as a "recency-based" memory structure to assure that solutions visited within a certain span of iterations, called tabu tenure, will not be revisited. TS then restricts consideration to moves not forbidden by the tabu list, and selects a move that produces the best move value to perform. In the case that two or more moves have the same best move value, a random best move is selected. In some cases, some of those neighborhood solutions forbidden by the tabu list are of excellent quality, where an aspiration criterion is applied to accept a tabu solution if it leads to a solution better than the best solution found so far. The ingredients of the present TS algorithm are described in detail in the following subsections.

2.3.1. Neighborhood structure

In a local search procedure, applying a move mv to a candidate solution S ($S=\{c_1,\ldots,c_n\}$) leads to a neighboring solution denoted by $S\oplus mv$. Let $M(S)$ be the set of all possible moves which can be applied to S, then the neighborhood of S is defined as: $N(S)=\{S\oplus mv|mv\in M(S)\}$.

For BCP, a neighborhood of a given k-coloring can be 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 $\langle u,c_i,c_j\rangle$), called

Please cite this article as: Lai X, Lü Z. Multistart Iterated Tabu Search for Bandwidth Coloring Problem. Computers and Operations Research (2012), http://dx.doi.org/10.1016/j.cor.2012.09.003

"critical one-move" neighborhood. Here, a vertex is considered to be conflicting if at least one of the distance constraints associated with this vertex is violated. Therefore, for a k-coloring S with cost $f(S)$, the size of this neighborhood is bounded by $O(f(S) \times k)$. For VCP problem, the same neighborhood structure was adopted by several authors in the literature [25,26].

2.3.2. Incremental evaluation of neighborhood moves

Given a neighborhood structure, it is particularly important to be able to rapidly determine the effect (called move value) of a move on the objective function $f(S)$. For this purpose, we employ a fast incremental evaluation technique to evaluate the Δ value of a move. Our TS procedure maintains an $n \times k$ matrix Q, from which the move values for all the possible moves can be rapidly calculated. The item $Q[i][q]$ $(1 \le i \le n, 1 \le q \le k)$ measures the constraint violation degree of vertex i if it receives color q

$$Q[i][q] = \sum_{e(i,j) \in E} \max\{0, d(i,j) - |q - c_j|\} \tag{2}$$

Note that this matrix is only initialized at the beginning of each TS procedure and will be updated in a fast manner during a single TS. With this memory structure, the value of a move $\langle u, c_i, c_j \rangle$ can be rapidly determined as follows:

$$\Delta f(S)_{\langle u,c_i,c_j \rangle} = Q[u][c_j] - Q[u][c_i] \tag{3}$$

We now describe how the matrix Q is quickly updated once a move is performed during the TS procedure. If a move $\langle u, c_i, c_j \rangle$ is performed, only the elements in those rows corresponding to the neighbors of vertex u need to be updated. Therefore, there are at most $\delta(u) \times k$ elements that need to be updated, where $\delta(u)$ is the degree of vertex u. More specifically, let v be a neighbor of vertex u, then the elements $Q[v][p]$ $(p = 1,2,\ldots,k)$ in the vth row of Q can be updated as follows:

$$Q[v][p]' = Q[v][p] - \max\{0, d(u,v) - |p - c_i|\} + \max\{0, d(u,v) - |p - c_j|\} \tag{4}$$

In fact, the number of elements that need to be updated in matrix Q can be further reduced and it is much less than $\delta(u) \times k$, since the value of the last two terms, $\max\{0, d(u,v) - |p - c_i|\}$ and $\max\{0, d(u,v) - |p - c_j|\}$ $(p = 1,2,\ldots,k)$, in Eq. (4) is equal to 0 in most cases. Through a simple analysis, it can be observed that they are necessary to be updated only if $p \in (c_i - d(u,v), c_i + d(u,v)) \bigcup (c_j - d(u,v), c_j + d(u,v))$. Therefore, it can be concluded that the number of elements which need to be updated in Q is no more than $\sum_{v \in N^*(u)} (4d(u,v) - 2)$, where $N^*(u)$ is the set of the neighbors of vertex u. In order to speed up the procedure, the updating of the matrix Q just considers these elements.

2.3.3. Tabu list management

Within TS, a tabu list is introduced to forbid the previously visited solutions to be revisited. In the present TS, when a move $\langle u, c_i, c_j \rangle$ is performed, vertex u is forbidden to accept color c_i for the next tt iterations, where tt denotes the tabu tenure and it is dynamically determined by

$$tt = f(S) + rand(C) + \left\lceil C \times \frac{Freq[u][c_i]}{\max_{1 \le s \le n, 1 \le q \le k}\{Freq[s][q]\}} \right\rceil \tag{5}$$

where C is a predetermined parameter and its value is empirically set to 10, $rand(C)$ represents a random number from 1 to C, and $Freq[s][q]$ represents the frequency of assigning the color q to vertex s. The first part of this function can be explained by the reason that a solution with high violations should have a long tabu tenure to escape from the local optimum trap, and the second part of this function aims to increase the randomness of the tabu tenure. The basic idea behind the last part is to penalize a move which repeats too often.

2.3.4. Aspiration criteria

Since attributes of a solution instead of the solutions themselves are recorded in the tabu list, it is possible that a candidate solution in the tabu list could lead to a solution better than the best solution found so far. Therefore, in our TS algorithm, a simple aspiration criterion is applied which allows the best move in the neighborhood to be performed in spite of being tabu if it leads to a solution better than the current best solution.

2.3.5. Stop condition of TS

Our TS algorithm stops when the current best solution cannot be improved within a given number α of iterations and we called this number the depth of TS.

2.4. Perturbation operator

When a TS phase terminates, we employ a perturbation operator to destruct the reached local optimum solution in order to jump out of this local optimum trap and explore a more promising region of the search space. For BCP problem, the perturbation operator employed in our algorithm can be described as follows. For the current solution $S = (c_1, c_2, \ldots, c_n)$, a new solution $S^{new} = (c_1^{new}, c_2^{new}, \ldots, c_n^{new})$ is generated as follows: $c_i^{new} = c_i + rand[-R,R]$ $(1 \le i \le n)$, such that $c_i^{new} \in \{1,2,\ldots,k\}$, where R is a predetermined integer and represents the scope of perturbation operator, and $rand[-R,R]$ represents a random integer between $-R$ and R.

The basic idea behind this perturbation operator can be stated as follows. In a local optimum solution, the distance constraints between two adjacent vertices are usually satisfied well. Therefore, a solution with a perturbation on a small number of vertices would very likely fall back into the previous local optimum solution in a subsequent TS. On the other hand, with a too strong perturbation on all the vertices, ITS in our algorithm will behavior like a multistart method. The present perturbation operator performs a relatively small perturbation on all the vertices such that the perturbed solution is not very far from the previous local optimum solution and the relative distance constraint of the perturbed solution is not drastically changed.

2.5. Acceptance and stop criterion of ITS used in the MITS algorithm

For ITS embedded in our MITS algorithm, when a new local optimum solution is obtained by TS, an acceptance criterion is used to decide whether to accept it as the incumbent solution. The acceptance criterion used in the ITS algorithm can be stated as follows: The incumbent solution S' is replaced by the new local optimum solution $S^{*'}$ if and only if $f(S^{*'}) \le f(S')$.

ITS in our MITS algorithm stops when the current objective function value cannot be improved within a given number β_{max} of consecutive perturbations, or a legal k-coloring is found, where β_{max} is called the improvement cutoff of perturbation.

3. Computational results and comparison

3.1. Benchmark instances

Two sets of benchmark instances are considered in our experiments. The first set of benchmarks is composed of 33 BCP instances generated by Michael Trick which is available in [10]. Three kinds of instances are contained in this set: The GEOMn instances with a sparse graph, and the GEOMa and GEOMb instances with a denser graph. The second set of benchmarks is composed of 33 larger instances transformed from the BMCP problem, corresponding to instances of the first set. Table 1 gives

Please cite this article as: Lai X, Lü Z. Multistart Iterated Tabu Search for Bandwidth Coloring Problem. Computers and Operations Research (2012), http://dx.doi.org/10.1016/j.cor.2012.09.003

Table 1
Benchmark instances and their characteristics.

Instance	BCP						BMCP					
	$\|V\|$	$\|E\|$	*Den*	d_{ave}	d_{min}	d_{max}	$\|V\|$	$\|E\|$	*Den*	d_{ave}	d_{min}	d_{max}
GEOM20	20	20	0.1053	2	0	4	118	1048	0.1518	17.76	4	27
GEOM20a	20	37	0.1947	3.7	0	7	100	1327	0.2681	26.54	0	44
GEOM20b	20	32	0.1684	3.2	1	6	40	132	0.1692	6.6	2	11
GEOM30	30	50	0.1149	3.33	1	6	143	1419	0.1398	19.85	5	35
GEOM30a	30	81	0.1862	5.4	2	10	171	3288	0.2262	38.45	12	63
GEOM30b	30	81	0.1862	5.4	2	10	69	447	0.1905	12.96	6	21
GEOM40	40	78	0.1	3.9	0	6	220	3074	0.1276	27.95	1	41
GEOM40a	40	146	0.1872	7.3	3	12	203	4473	0.2182	44.07	20	67
GEOM40b	40	157	0.2013	7.85	2	13	84	743	0.2131	17.69	3	27
GEOM50	50	127	0.1037	5.1	0	9	285	4935	0.1219	34.63	0	66
GEOM50a	50	238	0.1943	9.52	4	16	302	9649	0.2123	63.9	22	115
GEOM50b	50	249	0.2033	9.96	3	17	104	1140	0.2128	21.92	7	35
GEOM60	60	185	0.1045	6.17	1	10	315	6174	0.1248	39.2	7	59
GEOM60a	60	339	0.1915	11.3	6	18	362	13 105	0.2005	72.4	33	124
GEOM60b	60	366	0.2068	12.2	3	20	127	1785	0.2231	28.1	7	44
GEOM70	70	267	0.1106	7.63	1	13	384	8584	0.1167	44.71	11	69
GEOM70a	70	459	0.1901	13.11	7	20	379	14 821	0.2069	78.21	33	120
GEOM70b	70	488	0.202	13.94	4	24	148	2212	0.2033	29.89	10	53
GEOM80	80	349	0.1104	8.72	3	14	465	12 927	0.1198	55.6	21	89
GEOM80a	80	612	0.1936	15.3	7	23	389	15 545	0.2060	79.9	28	129
GEOM80b	80	663	0.2098	16.575	5	29	169	3028	0.2133	35.83	11	62
GEOM90	90	441	0.1101	9.8	5	15	530	16 180	0.1154	61.05	27	96
GEOM90a	90	789	0.1970	17.53	8	25	454	20 455	0.1989	90.11	37	136
GEOM90b	90	860	0.2147	19.11	6	34	184	3602	0.2139	39.15	12	64
GEOM100	100	547	0.1105	10.94	5	18	581	19 829	0.1177	68.25	28	104
GEOM100a	100	992	0.2	19.84	9	28	528	28 496	0.2048	107.93	51	180
GEOM100b	100	1050	0.2121	21	5	37	200	4429	0.2226	44.29	11	72
GEOM110	110	638	0.1064	11.6	6	19	643	24 799	0.1201	77.14	37	127
GEOM110a	110	1207	0.2013	21.94	12	32	602	38 783	0.2144	128.85	62	177
GEOM110b	110	1256	0.2095	22.84	5	39	220	5163	0.2143	46.94	12	85
GEOM120	120	773	0.1083	12.88	6	21	680	27 759	0.1202	81.64	33	126
GEOM120a	120	1434	0.2	23.9	13	35	664	46 429	0.2109	139.8	64	212
GEOM120b	120	1491	0.2088	24.85	5	43	235	5779	0.2102	49.18	11	81

Table 2
Descriptions and settings of important parameters.

Parameters	Section	Description	Values
α	2.3.5	Depth of TS	10^4
β_{max}	2.5	Improvement cutoff of perturbation	30
R	2.4	Scope of perturbation	2

the characteristics of these instances, including the number of vertices ($|V|$), the number of edges ($|E|$), the density of the graph (*Den*), the average degree of a vertex (d_{ave}), the minimum degree of a vertex (d_{min}) and the maximum degree of a vertex (d_{max}), where the number of edges in column 3 does not contain the number of the loop edges for the vertex. Moreover, the first and second sets are respectively denoted by BCP and BMCP in Table 1.

3.2. Experimental protocol

Our algorithm is programmed in C language and performed on a cluster with 2.8 GHz CPU and 4 Gb RAM. For all the considered problem instances, Table 2 gives the descriptions and settings of the parameters used in the algorithm. Notice that it is possible that better results would be found by using a set of instance-dependent parameters.

As stated before, we only deal with k-BCP problem. Based on this fact and limited by computational resource, the present study just considers three values of k for all the instances, i.e., k^*, k^*-1, and k^*-2, where k^* is the previous best known result. Given the stochastic nature of the present algorithm, for each instance and each given k, we carried out 30 independent runs with different random seeds. The timeout limits of our MITS algorithm for the BCP and BMCP instances are respectively 3600 s and 7200 s.

3.3. Computational results on BCP instances

Our first aim is to assess the MITS algorithm on the set of 33 BCP instances, and the statistics of computational results over 30 independent runs are summarized in Table 3, including the previous best known results (k^*), the minimum objective function value obtained for the given k (f_{min}), the average value (f_{ave}), the standard deviation (f_{dev}), the number of times for hitting a legal k-coloring (N_{hit}), and the average CPU time (in seconds) per hit of the legal k-coloring (T_{ave}) that is obtained via dividing the total time by N_{hit}, where "-" means that our algorithm fails to find a legal k-coloring.

From the table, it can be seen that our MITS algorithm is able to reach the previous best known results except for 4 instances, and for a given k, the value of f_{dev} and the difference between f_{ave} and f_{min} are both relatively small for most instances. These facts show the robustness of our ITS algorithm.

Moreover, our MITS algorithm is able to improve the previous best known results for 5 instances though the CPU time needed is relatively long, which shows the effectiveness of the present algorithm to some extent.

In order to further show the performance of our MITS algorithm, we compare our results with those of several best performing algorithms in the literature. Table 4 gives the detailed computational statistics of our MITS algorithm that are extracted from Table 3, as well as the results of 4 best performing reference algorithms in the literature, namely a hybrid algorithm [14] that combines a greedy method, squeaky wheel optimization method

Please cite this article as: Lai X, Lü Z. Multistart Iterated Tabu Search for Bandwidth Coloring Problem. Computers and Operations Research (2012), http://dx.doi.org/10.1016/j.cor.2012.09.003

Table 3
Statistics of computational results over 30 runs on BCP instances.

Instance	k^*	k^*					k^*-1					k^*-2				
		f_{min}	f_{ave}	f_{dev}	N_{hit}	T_{ave}	f_{min}	f_{ave}	f_{dev}	N_{hit}	T_{ave}	f_{min}	f_{ave}	f_{dev}	N_{hit}	T_{ave}
GEOM20	20	1	1	0	0	–	2	2	0	0	–	3	3	0	0	–
GEOM20a	20	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM20b	13	0	0	0	30	0	1	1	0	0	–	3	3	0	0	–
GEOM30	27	1	1	0	0	–	2	2	0	0	–	3	3	0	0	–
GEOM30a	27	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM30b	26	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM40	27	1	1	0	0	–	2	2	0	0	–	3	3	0	0	–
GEOM40a	37	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM40b	33	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM50	28	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM50a	50	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM50b	35	0	0	0	30	3	1	1	0	0	–	3	3	0	0	–
GEOM60	33	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM60a	50	0	0	0	30	1	1	1	0	0	–	3	3	0	0	–
GEOM60b	41	0	0	0	30	277	2	2.53	0.51	0	–	4	5.23	0.9	0	–
GEOM70	38	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM70a	61	0	0	0	30	45	1	1	0	0	–	2	2	0	0	–
GEOM70b	48	0	0	0	30	222	0	0.6	0.48	10	8685	2	3	0.37	0	–
GEOM80	41	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM80a	63	0	0	0	30	21	1	1	0	0	–	2	2	0	0	–
GEOM80b	60	0	0	0	30	322	1	1	0	0	–	2	2	0	0	–
GEOM90	46	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM90a	63	0	0	0	30	230	1	1	0	0	–	2	2.1	0.31	0	–
GEOM90b	70	0	0.56	0.5	13	6098	0	1.16	0.69	5	20 144	2	2.9	0.4	0	–
GEOM100	50	0	0	0	30	2	1	1	0	0	–	2	2	0	0	–
GEOM100a	68	0	0.03	0.18	29	1044	0	0.96	0.69	8	11 407	2	3.13	0.68	0	–
GEOM100b	73	0	0.97	0.56	5	19 312	0	1.53	0.86	4	24 561	2	2	0	0	–
GEOM110	50	0	0	0	30	2	1	1	0	0	–	2	2	0	0	–
GEOM110a	72	0	0.13	0.34	26	1529	1	2.06	0.82	0	–	3	4.06	0.8	0	–
GEOM110b	78	0	1.23	0.68	4	24 416	1	2.63	0.56	0	–	3	4.06	0.64	0	–
GEOM120	59	0	0	0	30	1	1	1	0	0	–	2	2	0	0	–
GEOM120a	84	0	0	0	30	82.5	0	0	0	30	6512	0	0.9	0.3	3	34 176
GEOM120b	84	2	4.16	0.79	0	–	4	5.73	0.74	0	–	6	7.9	0.76	0	–

[27] and a TS procedure, Discropt heuristic [11], FCNS [13] and an evolutionary algorithm [2]. The results of the 4 reference algorithms are directly extracted from [2]. Column 2 gives the previous best known results (k^*) reported in the literature. Columns 3–9 respectively give the results (k) and computational time (in seconds) for the 4 reference algorithms. Our results are reported in columns 10–13, including the best results obtained (k_{best}), the hit rates (R_{hit}), the average CPU time (in seconds) per successful hit (T_{ave}), and the difference between the best results obtained and the previous best known results ($k_{best}-k^*$), where the improved solutions are indicated in bold and "-" means that our algorithm fails to find the previous best known results.

Following the spirit of [2], to make the comparison as fair as possible, the computing time obtained by different machines should be scaled with respect to the performance obtained on a common benchmark program. In the present study, we employ a benchmark program (dfmax) which is commonly used in the literature and is available in [10], together with a benchmark instance (R500.5), to evaluate the computing speed of our machine, and the user time of our machine is 7.35 s on this benchmark program. According to the report in [2], for the same benchmark program, the user time of the machine used by the algorithm in [2] is 7.0 s. Therefore, our CPU time reported in Section 3 is scaled by multiplying 7.0/7.35 to make it comparable with that reported in [2]. Note that similar comparisons can also be found in [28,29].

From Table 4, one easily observes that our MITS algorithm respectively obtains better and worse results than the previous best known results for 5 and 4 instances. It should be mentioned that the Discropt heuristic algorithm is the only one that can obtain the best known results for the 3 small instances for which our algorithm fails to obtain the best known results. For the remaining 24 instances, our algorithm reproduces the previous best known results. These computational results show that our MITS algorithm is competitive in comparison with the 4 reference algorithms.

To the best of our knowledge, Malaguti and Toth's evolutionary algorithm [2] is the best performing algorithm in the literature. From the table, one observes that our algorithm gets better results than the algorithm in [2] for 5 instances and matches for the 27 rest ones, but fails to reproduce their result for the instance named GEOM120b. In addition, it should be noted that in terms of computational time our algorithm is comparable with the algorithm in [2] for many instances, but much more time-consuming for several difficult instances.

Compared with other 3 reference algorithms, our algorithm needs much longer time to reach the reported results on some difficult instances. Nevertheless, this is reasonable since our algorithm can get smaller k values, and a k-BCP problem with a smaller k is much more difficult than that with a larger k, as mentioned before.

The hit rate is another important criterion to evaluate the performance of an algorithm. From Table 4, one observes that the hit rate of our algorithm is 30/30 for most instances. However, for several difficult instances, our algorithm obtains low hit rates, which shows that further improvements over the present method are still possible.

3.4. Computational results on BMCP instances

Our second aim is to assess our MITS algorithm on the set of 33 larger instances transformed from the BMCP problem. The

Please cite this article as: Lai X, Lü Z. Multistart Iterated Tabu Search for Bandwidth Coloring Problem. Computers and Operations Research (2012), http://dx.doi.org/10.1016/j.cor.2012.09.003

Table 4
Comparison of the MITS algorithm with other reference algorithms on BCP instances.

Instance	k^*	[11]	[14]		[13]		[2]		This work			
		k	k	Time	k	Time	k	Time	k_{best}	R_{hit}	T_{ave}	$k_{best}-k^*$
GEOM20	20	20	21	0	21	0	21	0	–	–	–	–
GEOM20a	20	20	22	0	20	0	20	0	20	30/30	0	0
GEOM20b	13	13	14	0	13	0	13	0	13	30/30	0	0
GEOM30	27	27	29	0	28	0	28	0	–	–	–	–
GEOM30a	27	27	32	0	27	0	27	0	27	30/30	0	0
GEOM30b	26	26	26	0	26	0	26	0	26	30/30	0	0
GEOM40	27	27	28	1	28	0	28	0	–	–	–	–
GEOM40a	37	38	38	1	37	0	37	0	37	30/30	0	0
GEOM40b	33	36	34	1	33	0	33	0	33	30/30	0	0
GEOM50	28	29	28	1	28	0	28	0	28	30/30	0	0
GEOM50a	50	54	52	1	50	2	50	0	50	30/30	0	0
GEOM50b	35	40	38	2	35	0	35	0	35	30/30	3	0
GEOM60	33	34	34	0	33	0	33	0	33	30/30	0	0
GEOM60a	50	54	53	2	50	1	50	0	50	30/30	1	0
GEOM60b	41	47	46	1	43	0	41	29	41	30/30	277	0
GEOM70	38	40	38	0	38	0	38	0	38	30/30	0	0
GEOM70a	61	64	63	0	62	2	61	12	61	30/30	45	0
GEOM70b	48	54	54	0	48	1	48	52	**47**	10/30	8685	**−1**
GEOM80	41	44	42	2	41	0	41	0	41	30/30	0	0
GEOM80a	63	69	66	0	63	12	63	150	63	30/30	21	0
GEOM80b	60	70	65	8	61	0	60	145	60	30/30	322	0
GEOM90	46	48	46	0	46	3	46	0	46	30/30	0	0
GEOM90a	63	74	69	2	64	2	63	150	63	30/30	230	0
GEOM90b	70	83	77	6	72	2	70	1031	**69**	5/30	20 144	**−1**
GEOM100	50	55	51	9	50	0	50	2	50	30/30	2	0
GEOM100a	68	84	76	6	68	9	68	273	**67**	8/30	11 407	**−1**
GEOM100b	73	87	83	2	73	15	73	597	**72**	4/30	24 561	**−1**
GEOM110	50	59	53	1	50	4	50	3	50	30/30	2	0
GEOM110a	72	88	82	11	73	7	72	171	72	26/30	1529	0
GEOM110b	78	87	88	5	79	2	78	676	78	4/30	24 416	0
GEOM120	59	67	62	0	60	4	59	0	59	30/30	1	0
GEOM120a	84	101	92	1	84	4	84	614	**82**	3/30	34 176	**−2**
GEOM120b	84	103	98	1	86	9	84	857	–	–	–	–

statistics of computational results over 30 independent runs are reported in Table 5. The symbols are the same as in Table 3.

From the table, one observes that our algorithm is able to reproduce the previous best known results except for one instance named GEOM120, though the CPU time needed is long for some difficult instances. Moreover, our algorithm improves the previous best known results for 17 out of 33 instances, which clearly shows the effectiveness of our algorithm. In addition, for a given k, the value of f_{dev} and the difference between f_{ave} and f_{min} are both small for most instances, showing the robustness of the present algorithm again.

In order to make a meaningful comparison with the best performing algorithms in the literature, our results are summarized in Table 6 together with the results of the 4 reference algorithms, namely two hybrid algorithms [1,14], FCNS [13], and an evolutionary algorithm [2]. The results of the 4 reference algorithms are directly extracted from [2], our results are extracted from Table 5 and the symbols are the same as in Table 4.

From Table 6, it can be observed that our algorithm improves the previous best known results for more than one half of the instances, but takes much longer CPU time than the 4 reference algorithms on many instances. Nevertheless, it is still acceptable since our algorithm is able to improve the quality of solution on these instances. On the other hand, for some instances, our algorithm is able to improve very stably the previous best known results with a small computational effort, such as GEOM80b, GEOM100a and GEOM110, implying that the proposed algorithm is quite competitive compared with the best performing algorithms in the literature on these instances, and further improvement is still possible for these instances.

In addition, one observes that there exist several instances for which the hit rates of our algorithm are below 5/30, which shows that they are difficult cases for our algorithm. For example, for GEOM120b with $k=189$, the hit rate of our algorithm is only 1/30. Therefore, further improvements over the present algorithm are still needed.

4. Discussion and analysis

We now turn our attention to discussing and analyzing two important features of the proposed MITS algorithm, including the importance of the proposed incremental evaluation technique and the influence of the scope of perturbation, R.

4.1. Importance of the incremental evaluation technique

The main ingredient of our MITS algorithm is the TS procedure. For a local search method, it is particularly important to be able to rapidly determine the effect of a move on the objective function value. As described in Section 2.3.2, we propose an incremental evaluation technique for evaluating the neighborhood moves, where an $n \times k$ matrix Q is maintained to help to calculate the move values of all possible candidate moves. Once a move is performed, only the values of matrix Q affected by the move are accordingly updated.

In order to evaluate the effectiveness of this incremental evaluation technique, we carry out computational experiments to compare the performance of the TS algorithm with and without using this technique on two representative instances, i.e., GEOM120a of BCP and GEOM120a of BMCP. Note that

Please cite this article as: Lai X, Lü Z. Multistart Iterated Tabu Search for Bandwidth Coloring Problem. Computers and Operations Research (2012), http://dx.doi.org/10.1016/j.cor.2012.09.003

Table 5
Statistics of computational results over 30 runs on larger instances transformed from BMCP.

Instance	k^*	k^*					k^*-1					k^*-2				
		f_{min}	f_{ave}	f_{dev}	N_{hit}	T_{ave}	f_{min}	f_{ave}	f_{dev}	N_{hit}	T_{ave}	f_{min}	f_{ave}	f_{dev}	N_{hit}	T_{ave}
GEOM20	149	0	0	0	30	2	1	1	0	0	–	2	2	0	0	–
GEOM20a	169	0	0	0	30	15	1	1	0	0	–	2	2	0	0	–
GEOM20b	44	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM30	160	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM30a	209	0	0	0	30	10	1	1	0	0	–	2	2	0	0	–
GEOM30b	77	0	0	0	30	0	1	1	0	0	–	2	2	0	0	–
GEOM40	167	0	0	0	30	4	1	1	0	0	–	2	2	0	0	–
GEOM40a	213	0	0	0	30	328	1	1	0	0	–	2	2	0	0	–
GEOM40b	74	0	0	0	30	2	1	1	0	0	–	2	2	0	0	–
GEOM50	224	0	0	0	30	8	1	1	0	0	–	2	2	0	0	–
GEOM50a	316	0	0.1	0.3	27	2872	0	0.63	0.66	14	11 859	0	1.3	0.84	5	40 373
GEOM50b	83	0	0	0	30	1200	1	1	0	0	–	2	2	0	0	–
GEOM60	258	0	0	0	30	19	1	1	0	0	–	2	2	0	0	–
GEOM60a	357	0	1.3	1.17	10	17 955	0	2.73	1.83	3	38 570	2	4.8	1.88	0	–
GEOM60b	115	0	0	0	30	119	0	0	0	30	785	0	0.93	0.25	2	104 711
GEOM70	272	0	0	0	30	1212	0	0	0	30	2204	0	0	0	30	7602
GEOM70a	469	0	0	0	30	1816	0	0.47	0.51	16	8819	0	0.96	0.56	5	38 759
GEOM70b	117	0	0.83	0.59	8	22 359	0	2.16	0.75	1	213 545	3	4.13	0.57	0	7602
GEOM80	383	0	0.06	0.25	28	1893	0	0.7	0.65	12	14 608	0	1.67	0.61	1	212 213
GEOM80a	363	0	0.6	0.7	13	13 531	0	1.3	0.76	4	50 438	0	2.13	1.28	5	41 235
GEOM80b	141	0	0	0	30	41	0	0	0	30	61	0	0	0	30	255
GEOM90	332	0	0	0	30	258	0	0	0	30	568	0	0.06	0.25	28	4022
GEOM90a	377	0	0	0	30	321	0	0	0	30	920	0	0.5	0.51	15	10 427
GEOM90b	144	0	2.16	1.05	1	211 366	1	4	1.14	0	–	4	5.76	0.82	0	–
GEOM100	404	0	1.06	0.64	5	40 121	1	1.8	0.55	0	–	2	2.67	0.61	0	–
GEOM100a	444	0	0	0	30	270	0	0	0	30	280	0	0	0	30	381
GEOM100b	156	0	4.16	1.12	1	213 949	4	5.73	0.78	0	–	4	6.46	1.11	0	–
GEOM110	383	0	0	0	30	96	0	0	0	30	138	0	0	0	30	183
GEOM110a	490	0	0	0	30	565	0	0	0	30	597	0	0	0	30	926
GEOM110b	206	0	0	0	30	53	0	0	0	30	176	0	0	0	30	944
GEOM120	396	5	9.93	2.22	0	–	8	12	2.0	0	–	11	15.3	2.05	0	–
GEOM120a	556	0	0	0	30	599	0	0	0	30	734	0	0	0	30	1018
GEOM120b	191	0	0.47	0.51	16	9415	0	0.93	0.64	7	27 152	0	1.83	0.7	1	213 989

Table 6
Comparison of our MITS algorithm with other reference algorithms on BMCP instances.

Instance	k^*	[14]		[1]		[13]		[2]		This work			
		k	Time	k	Time	k	Time	k	Time	k_{best}	R_{hit}	T_{ave}	$k_{best}-k^*$
GEOM20	149	149	0	149	17	149	4	149	18	149	30/30	2	0
GEOM20a	169	169	4	169	16	170	2	169	9	169	30/30	15	0
GEOM20b	44	44	0	44	2	44	0	44	5	44	30/30	0	0
GEOM30	160	160	0	160	23	160	0	160	1	160	30/30	0	0
GEOM30a	209	211	3	209	40	214	11	210	954	209	30/30	10	0
GEOM30b	77	77	0	77	7	77	0	77	0	77	30/30	0	0
GEOM40	167	167	1	167	47	167	1	167	20	167	30/30	0	0
GEOM40a	213	214	99	213	54	217	299	214	393	213	30/30	328	0
GEOM40b	74	76	2	74	10	74	4	74	1	74	30/30	2	0
GEOM50	224	224	11	224	77	224	1	224	1197	224	30/30	8	0
GEOM50a	316	326	27	318	12	323	51	316	4675	**314**	5/30	40 373	**−2**
GEOM50b	83	87	15	87	15	86	1	83	197	83	30/30	1200	0
GEOM60	258	258	13	258	96	258	77	258	139	258	30/30	19	0
GEOM60a	357	368	1037	358	162	373	10	357	8706	**356**	3/30	38 570	**−1**
GEOM60b	115	119	83	116	23	116	12	115	460	**113**	2/30	104 711	**−2**
GEOM70	272	279	7	273	138	277	641	272	1413	**270**	30/30	7602	**−2**
GEOM70a	469	478	115	469	188	482	315	473	988	**467**	5/30	38 759	**−2**
GEOM70b	117	124	38	121	30	119	55	117	897	**116**	1/30	213 545	**−1**
GEOM80	383	394	1118	383	204	398	361	388	132	**381**	1/30	212 213	**−2**
GEOM80a	363	379	187	379	190	380	109	363	8583	**361**	5/30	41 235	**−2**
GEOM80b	141	145	894	141	39	141	37	141	1856	**139**	30/30	255	**−2**
GEOM90	332	335	1133	332	248	339	44	332	4160	**330**	28/30	4022	**−2**
GEOM90a	377	382	2879	377	245	382	13	382	5334	**375**	15/30	10 427	**−2**
GEOM90b	144	157	179	157	46	147	303	144	1750	144	1/30	211 366	0
GEOM100	404	413	175	404	311	424	7	410	3283	404	5/30	40 121	0
GEOM100a	444	462	48	459	334	461	26	444	12 526	**442**	30/30	381	**−2**
GEOM100b	156	172	1354	170	55	159	367	156	3699	156	1/30	213 949	0
GEOM110	383	389	160	383	368	392	43	383	2344	**381**	30/30	183	**−2**
GEOM110a	490	501	1292	494	441	500	29	490	2318	**488**	30/30	926	**−2**
GEOM110b	206	210	3	206	68	208	5	206	480	**204**	30/30	944	**−2**
GEOM120	396	409	505	402	408	417	9	396	2867	–	–	–	–
GEOM120a	556	564	1476	556	633	565	41	559	3873	**554**	30/30	1018	**−2**
GEOM120b	191	201	240	199	97	196	3	191	3292	**189**	1/30	213 989	**−2**

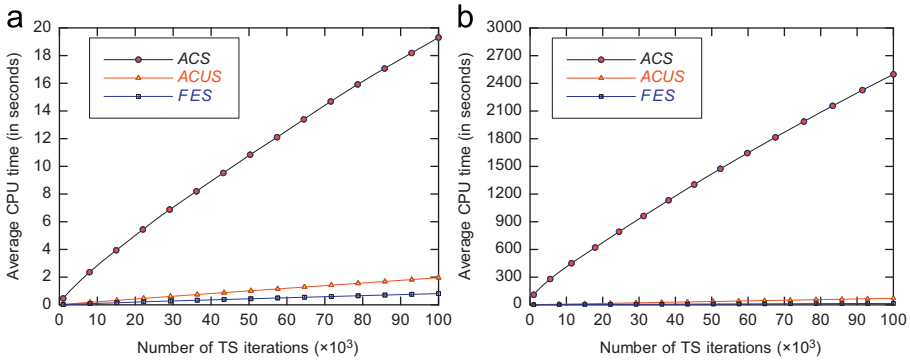

Fig. 1. Performance comparison of TS algorithm with and without the fast evaluation technique. (a) GEOM120a of BCP. (b) GEOM120a of BMCP.

similar results can be observed on other instances. The following three strategies are considered: Our fast incremental updating strategy used in this paper, called fast evaluation strategy (FES); The value of all the columns of the matrix Q is updated using Eq. (4), called all column updating strategy (ACUS); The fast incremental evaluation technique is not used and all the neighborhood moves are calculated from scratch, called all calculating strategy (ACS).

For both instances, TS algorithm is independently run for 10 times. The evolution of the average CPU time with the number of iterations is shown in Fig. 1. It can be clearly seen that the average CPU time of TS with the fast evaluation strategy (FES) is much faster than the two other strategies ACUS and ACS. For instance, for the instance GEOM120a of BCP, the FES strategy is respectively about 2 and 24 times faster than the ACUS and ACS strategies. Moreover, it should be stated that the larger the value of k is, the more efficient the FES strategy is, as shown on the instance GEOM120a of BMCP. For this instance, the FES strategy is about 5 and 170 times faster than the ACUS and ACS strategies, respectively. This experiment clearly demonstrates the importance of the fast evaluation strategy proposed in this paper.

4.2. Influence of scope of perturbation on the performance

It should be pointed out that the scope of perturbation operator, R, is one of the most important ingredients of the present MITS algorithm and it determines the effectiveness of the MITS algorithm. With a too large perturbation scope, the structure of the current solution will be completely destroyed and ITS used in our MITS algorithm will behavior as a multistart method. On the contrary, ITS will fall back into the local optimum solution just visited if the perturbation scope is too small. Therefore, the value of R should be carefully determined.

In order to show the influence of R on the performance of the ITS algorithm and to find the appropriate parameter setting for it, we perform some additional computational experiments on 6 representative instances from the second set of benchmark instances, for which our ITS algorithm is independently run for 200 times for each value of R from 1 to 4. The statistical results are summarized in Table 7, including the number of colors used (k), the scope of perturbation (R), the average objective function value over 200 runs (f_{ave}), standard deviation of objective function value (f_{dev}), and the average CUP time (in seconds) per run of ITS for the given R (T_{ave}).

From Table 7, one can find two significant facts. The first one is that a larger R usually corresponds to a larger T_{ave}, which means that a stronger perturbation operator usually leads to more computational effort. Thus, a small R is desirable from the point of view of saving CPU time. The second one is that in the range of $R \geq 2$ a larger R usually leads to a worse f_{ave}, which means also that a small R is desirable from the point of view of optimizing the objective function value. On the other hand, with a too small R, i.e., $R=1$, ITS usually obtains worse f_{ave} and f_{dev} compared with those achieved when using a larger R.

Table 7
Comparison of different strengths of perturbation operator.

Instance	k	R	f_{ave}	f_{dev}	T_{ave}
GEOM50a	315	1	7.82	3.52	106.45
		2	5.40	2.45	117.58
		3	5.86	2.01	145.45
		4	7.42	1.85	168.18
GEOM80a	362	1	7.83	3.39	125.64
		2	6.03	2.20	133.68
		3	7.93	1.70	157.21
		4	9.51	1.74	174.28
GEOM100	404	1	6.36	1.85	80.08
		2	4.67	1.49	97.50
		3	4.32	1.32	110.75
		4	5.22	1.35	114.50
GEOM100b	155	1	12.91	2.88	56.15
		2	11.43	2.50	62.92
		3	11.80	2.01	78.98
		4	13.67	1.70	137.04
GEOM120a	554	1	2.83	3.13	575.48
		2	2.65	3.20	413.72
		3	7.29	4.32	623.33
		4	12.10	4.60	700.91
GEOM120b	189	1	6.78	2.05	38.40
		2	6.05	1.62	45.29
		3	6.16	1.28	67.11
		4	6.67	1.16	80.23

Therefore, we can conclude from the above experiments that the scope of perturbation operator, R, is an important parameter for the performance of our MITS algorithm, and $R=2$ is generally appropriate for the instances considered in the present study.

5. Conclusion

In this paper, we have presented a MITS algorithm for solving the BCP and BMCP problems, which incorporates an ITS algorithm into the framework of a multistart method. We tested the proposed algorithm on 66 problem instances commonly used in the literature. Computational results show that the proposed algorithm is highly competitive in comparison with the best performing algorithms in the literature. It is noteworthy that new best known solutions are obtained for 22 instances.

Please cite this article as: Lai X, Lü Z. Multistart Iterated Tabu Search for Bandwidth Coloring Problem. Computers and Operations Research (2012), http://dx.doi.org/10.1016/j.cor.2012.09.003

Our further analysis indicates that the incremental evaluation technique proposed in this paper is essential to enhance the computational efficiency of our TS algorithm, and the scope of perturbation operator, *R*, is a key parameter to influence the performance of the proposed algorithm.

There are several directions to extend this work. One immediate possibility is to incorporate the present TS algorithm as well as an appropriate crossover operator into the framework of an evolutionary algorithm. The other possibility is to develop more efficient perturbation operators that consider more problem-specific knowledge to enhance the performance of the present MITS algorithm.

Acknowledgment

We would like to thank the anonymous referees for their helpful comments and questions that help to improve our paper. This work was partially supported by the National Natural Science Foundation of China (Grant nos. 61100144 and 61173180).

References

[1] Lim A, Zhu Y, Lou Q, Rodrigues B. Heuristic methods for graph coloring problems. In: Proceedings of the 2005 ACM symposium on applied computing. Santa Fe, New Mexico; 2005. p. 933–9.
[2] Malaguti E, Toth P. An evolutionary approach for bandwidth multicoloring problems. European Journal of Operational Research 2008;189(3):638–51.
[3] 〈http://fap.zib.de/index.php〉; May 2012.
[4] Zoellner JA, Beall CL. A breakthrough in spectrum conserving frequency assignment technology. IEEE Transaction on Electromagnetic Compatibility 1977;19(3):313–9.
[5] Wang W, Rushforth CK. An adaptive local search algorithm for the channel assignment problem (CAP). IEEE Transaction on Vehicular Technology 1996;45(3):459–66.
[6] Costa D. On the use of some known methods for T-coloring of graphs. Annals of Operations Research 1993;41(4):343–58.
[7] Hao JK, Dorne R, Galinier P. Tabu search for frequency assignment in mobile radio networks. Journal of Heuristics 1998;4(1):47–62.
[8] Velanzuela C, Hurley S, Smith D. A perturbation based genetic algorithm for the minimum span frequency assignment. In: Lecture notes in computer science, vol. 1498; 1999. p. 907–16.
[9] Walser JP. Feasible cellular frequency assignment using constraint programming abstractions. In: Proceeding of the workshop on constraint programming applications (CP96). Cambridge, MA, USA; 1996.
[10] Trick MA. Computational symposium: graph coloring and its generalization. 〈http://mat.gsia.cmu.edu/COLOR02/〉; May 2012.
[11] Phan V, Skiena S. Coloring graphs with a general heuristic search engine. In: Computational symposium on graph coloring and its generalization. Ithaca, NY; 2002. p. 92–9.
[12] Prestwich S. Constrained bandwidth multicoloration neighborhoods. In: Computational symposium on graph coloring and its generalization. Ithaca, NY, 2002; p. 126–33.
[13] Prestwich S. Generalized graph colouring by a hybrid of local search and constraint programming. Discrete Applied Mathematics 2008;156(2): 148–58.
[14] Lim A, Zhang X, Zhu Y. A hybrid method for the graph coloring and its related problems. In: Proceedings of MIC2003: the fifth metaheuristic international conference. Kyoto, Japan; 2003.
[15] Malaguti E, Toth P. A survey on vertex coloring problems. International Transactions in Operational Research 2010;17(1):1–34.
[16] Martí R, Gortazar F, Duarte A. Heuristics for bandwidth coloring problem. International Journal of Metaheuristics 2010;1(1):11–29.
[17] Lourenço HR, Martin O, Stützle T. Iterated local search. In: Glover FW, Kochenberger GA, editors. Handbook of metaheuristics. Boston: Kluwer Academic Publishers; 2003. p. 321–53.
[18] Blum C, Roli A. Metaheuristics in combinatorial optimization: overview and conceptual comparisonACM Computing Surveys 2003;35(3):268–308.
[19] Cordeau JF, Laporte G, Pasin F. Iterated tabu search for the car sequencing problem. European Journal of Operational Research 2008;191(3):945–56.
[20] Palubeckis G. Iterated tabu search for the unconstrained binary quadratic optimization problem. Informatica 2006;17(2):279–96.
[21] Palubeckis G. Iterated tabu search for the maximum diversity problem. Applied Mathematics and Computation 2007;189(1):371–83.
[22] Lü ZP, Huang WQ. Iterated tabu search for identifying community structure in complex networks. Physical Review E 2009;80(2):026130.
[23] Glover F, Laguna M. Tabu search. Boston: Kluwer Academic Publishers; 1997.
[24] Lü ZP, Hao JK. A memetic algorithm for graph coloring. European Journal of Operational Research 2010;203(1):241–50.
[25] Lü ZP, Hao JK. Adaptive Tabu Search for course timetabling. European Journal of Operational Research 2010;200(1):235–44.
[26] Wu QH, Hao JK. Coloring large graphs based on independent set extraction. Computers and Operations Research 2012;39(2):283–90.
[27] Joslin DE, Clemnents DP. Squeaky wheel optimization. Journal of Artificial Intelligence Research 1999;10:353–73.
[28] Campos V, Piñana E, Martí R. Adaptive memory programming for matrix bandwidth minimization. Annals of Operations Research 2011;183(1):7–23.
[29] Rodriguez-Tello E, Hao JK, Torres-Jimenez J. An effective two-stage simulated annealing algorithm for the minimum linear arrangement problem. Computers and Operations Research 2008;35(10):3331–46.

Please cite this article as: Lai X, Lü Z. Multistart Iterated Tabu Search for Bandwidth Coloring Problem. Computers and Operations Research (2012), http://dx.doi.org/10.1016/j.cor.2012.09.003