


Article

# An Efficient Memetic Algorithm for the Minimum Load Coloring Problem

Zhiqiang Zhang <sup>1,2,\*</sup> , Zhongwen Li <sup>1,2,\*</sup>, Xiaobing Qiao <sup>3</sup> and Weijun Wang <sup>2</sup>

<sup>1</sup> Key Laboratory of Pattern Recognition and Intelligent Information Processing, Institutions of Higher Education of Sichuan Province, Chengdu University, Chengdu 610106, China

<sup>2</sup> School of Information Science and Engineering, Chengdu University, Chengdu 610106, China; wangweijun@cdu.edu.cn

<sup>3</sup> College of Teachers, Chengdu University, Chengdu 610106, China; qiaoxiaobing@cdu.edu.cn

\* Correspondence: zqzhang@cdu.edu.cn (Z.Z.); lizw@cdu.edu.cn (Z.L.)

Received: 29 March 2019; Accepted: 21 May 2019; Published: 25 May 2019



**Abstract:** Given a graph  $G$  with  $n$  vertices and  $l$  edges, the load distribution of a coloring  $q: V \rightarrow \{\text{red, blue}\}$  is defined as  $d_q = (r_q, b_q)$ , in which  $r_q$  is the number of edges with at least one end-vertex colored red and  $b_q$  is the number of edges with at least one end-vertex colored blue. The minimum load coloring problem (MLCP) is to find a coloring  $q$  such that the maximum load,  $l_q = 1/l \times \max\{r_q, b_q\}$ , is minimized. This problem has been proved to be NP-complete. This paper proposes a memetic algorithm for MLCP based on an improved K-OPT local search and an evolutionary operation. Furthermore, a data splitting operation is executed to expand the data amount of global search, and a disturbance operation is employed to improve the search ability of the algorithm. Experiments are carried out on the benchmark DIMACS to compare the searching results from memetic algorithm and the proposed algorithms. The experimental results show that a greater number of best results for the graphs can be found by the memetic algorithm, which can improve the best known results of MLCP.

**Keywords:** minimum load coloring; memetic algorithm; evolutionary; local search

## 1. Introduction

The minimum load coloring problem (MLCP) of the graph, discussed in this paper, was introduced by Nitin Ahuja et al. [1]. This problem is described as follows: a graph  $G = (V, E)$  is given, in which  $V$  is a set of  $n$  vertices, and  $E$  is a set of  $l$  edges. The load of a  $k$ -coloring  $\varphi: V \rightarrow \{1, 2, 3, \dots, k\}$  is defined as

$$1/l \times \max_{i \in \{1,2,3,\dots,k\}} |\{e \in E | \varphi^{-1}(i) \cap e \neq \emptyset\}|,$$

the maximum fraction of edges with at least one end-point in color  $i$ , where the maximum is taken over all  $i \in \{1,2,3, \dots, k\}$ . The aim of the minimum load coloring problem is to minimize the load over all  $k$ -colorings.

This paper is dedicated to the NP-complete minimum load coloring problem [1]. We focus on coloring the vertices with the colors of red and blue. A graph  $G = (V, E)$  is given, in which  $V$  is a set of  $n$  vertices, and  $E$  is a set of  $l$  edges. The load distribution of a coloring  $q: V \rightarrow \{\text{red, blue}\}$  is defined as  $d_q = (r_q, b_q)$ , in which  $r_q$  is the number of edges with at least one end-vertex colored red, and  $b_q$  is the number of edges with at least one end-vertex colored blue. The objective of MLCP is to find a coloring  $q$  such that the maximum load,  $l_q = 1/l \times \max\{r_q, b_q\}$ , is minimized. MLCP can be applied to solve the wavelength division multiplexing (WDM) problem of network communication, and build the WDM network and complex power network [1–3].

This paper proposes an effective memetic algorithm for the minimum load coloring problem, which relies on four key components. Firstly, an improved K-OPT local search procedure, combining a tabu search strategy and a vertices addition strategy, is especially designed for MLCP to explore the search space and escape from the local optima. Secondly, a data splitting operation is used to expand the amount of data in the search space, which enables the memetic algorithm to explore in a larger search space. Thirdly, to find better global results, through randomly changing the current search patterns a disturbance operation is employed to improve the probability of escaping from the local optima. Finally, a population evolution mechanism is devised to determine how the better solution is inserted into the population.

We evaluate the performance of memetic algorithm on 59 well-known graphs from benchmark DIMACS coloring competitions. The computational results show that the search ability of memetic algorithm is better than those of simulated annealing algorithm, greedy algorithm, artificial bee colony algorithm [4] and variable neighborhood search algorithm [5]. In particular, it improves the best known results of 16 graphs in known literature algorithms.

The paper is organized as follows. Section 2 describes the related work of heuristic algorithms. Section 3 describes the general framework and the components of memetic algorithm, including the population initialization, the data splitting operation, the improved K-OPT local search procedure of individuals, the evolutionary operation and the disturbance operation. Section 4 describes the design process of simulated annealing algorithm. Section 5 describes the design process of greedy algorithm. Section 6 describes the experimental results. Section 7 describes the conclusion of the paper.

## 2. Related Work

In [6,7], the parameterized and approximation algorithms are proposed to solve the load coloring problem, and theoretically prove their capability in finding the best solution. On the other hand, considering the theoretical intractability of MLCP, several heuristic algorithms are proposed to find the best solutions. Heuristic algorithms use rules based on previous experience to solve a combinatorial optimization problem at the cost of acceptable time and space, and, at the same time, comparatively better results can be obtained. The heuristic algorithms used here include an artificial bee colony algorithm [4], a tabu search algorithm [5] and a variable neighborhood search algorithm [5] to solve MLCP.

Furthermore, to find the best solutions of the other combinatorial optimization problems, several heuristic algorithms are employed, such as a variable neighborhood search algorithm [8,9], a tabu search algorithm [10–13], a simulated annealing algorithm [14–17], and a greedy algorithm [18].

Local search algorithm, as an important heuristic algorithm, has been improved and evolved into many updated forms, such as a variable depth search algorithm [19], a reactive local search algorithm [20], an iterated local search algorithm [21], and a phased local search algorithm [22].

Memetic algorithm [23,24] is an optimization algorithm which combines population-based global search and individual-based local heuristic search, whose application is found in solving combinatorial optimization problems. Memetic algorithm is also proposed to solve the minimum sum coloring problem of graphs [24].

## 3. A Memetic Algorithm for MLCP

In this paper, we propose an efficient memetic algorithm to solve MLCP of graphs. In our algorithm, there are several important design parts.

- (1) Construct the population for the global search.
- (2) Search heuristically the individuals to find better solutions.
- (3) Evolve the population to find better solutions.

Memetic algorithm is summarized in Memetic\_D\_O\_MLCP (Algorithm 1). After population initialization, the algorithm randomly generates a population  $X$  consisting of  $p$  individuals (Algorithm 1,

Line 2, Section 3.2). Then, the memetic algorithm repeats a series of generations (limited to a stop condition) to explore the search space defined by the set of all proper 2-colorings (Section 3.1). For each generation, by data splitting operation, the population  $X$  is expanded to population  $Z$  with twice as much as the data amount (Algorithm 1, Line 5, Section 3.3). An improved K- OPT local search is carried out for each individual  $Z_j$  ( $0 \leq j < |Z|$ ) of the population  $Z$  to find the best solution of MLCP (Algorithm 1, Line 8, Section 3.4). If the improved solution has a better value, it is then used to update the best solution found so far (Algorithm 1, Lines 9-10). Finally, an evolutionary operation is conducted in population  $Z$  to get a replaced one instead of population  $X$  (Algorithm 1, Line 14, Section 3.5). To further improve the search ability of the algorithm and find better solutions, we add a disturbance operation into the memetic algorithm (Algorithm 1, Line 15, Section 3.6).

---

**Algorithm 1** Memetic\_D\_O\_MLCP ( $G, m, p, b, k, X, Z$ ).

---

Require:

$G: G = (V, E), |V| = n, |E| = l$

$m$ : initial number of red vertices in graph  $G$

$p$ : number of individuals in the population

$b, k$ : control parameters of disturbance operation

$X$ : set that stores the population

$Z$ : set that stores the extended population of  $X, |Z| = 2 \times |X|$

Output:  $s_1$ , the best solution found by the algorithm

$f(s_1)$ , the value of the objective function

begin

1  $d1 \leftarrow 0, d2 \leftarrow 0$ ; /\* control variables of disturbance operation, Section 3.6 \*/

2 Init\_population( $X, m, p$ ); /\* generates population  $X$  consisting of  $p$  individuals, Section 3.2 \*/

3  $W_{best} \leftarrow 0$ ;

4 repeat

5  $Z \leftarrow$  Data\_splitting( $X$ ); /\* population  $X$  is extended to population  $Z$  with twice as much the data amount, Section 3.3 \*/

6  $j \leftarrow 0$ ;

7 while  $j < 2 \times p$  do

8  $W \leftarrow$  New\_K-OPT\_MLCP ( $G, Z_j, T, L$ );

/\* a heuristic search is carried out for individual  $Z_j$ , ( $T$  is tabu table and  $L$  is tabu tenure value,

Section 3.4) \*/

9 if  $f(W) > W_{best}$  then

10  $W_{best} \leftarrow f(W), s_1 \leftarrow W$ ;

11 end if

12  $j \leftarrow j + 1$ ;

13 end while

14  $X \leftarrow$  Evolution\_population ( $Z, X$ ); /\* Section 3.5 \*/

15 ( $s_1, W_{best}, d1, d2$ )  $\leftarrow$  Disturbance\_operation( $X, b, k, d1, d2, W_{best}$ ); /\* Section 3.6 \*/

16 until stop condition is met;

17 return  $s_1, W_{best}$ ;

end

---

### 3.1. Search Space and Objective Function

In [1], the following description is considered to be MLCP's equivalent problem. A graph  $G = (V, E)$  is given, in which  $V$  is a set of  $n$  vertices, and  $E$  is a set of  $l$  edges.  $(V_{red}, V_{blue})$  is a two-color load coloring bipartition scheme of  $V$ , in which  $V_{red}$  is the set of vertices which are red, and  $V_{blue}$  is the set of vertices which are blue, here  $V = V_{red} \cup V_{blue}$ . The aim is to find the maximum value of  $\min\{|Er(V_{red})|, |Er(V_{blue})|\}$  from all bipartition schemes of  $(V_{red}, V_{blue})$  such that  $l_q$  can minimize. The maximum value is the minimum two-color load problem solution of graph  $G$ . Here,  $Er(V_{red})$  is the set of edges with both end-points in  $V_{red}$ , and  $Er(V_{blue})$  is the set of edges with both end-points in  $V_{blue}$ .

The algorithm conducts a searching within the bipartition scheme  $(V_{red}, V_{blue})$ , here  $|V_{red}| \subset V$ ,  $V_{blue} = V \setminus V_{red}$ , when  $|Er(V_{red})| \approx |Er(V_{blue})|$ ,  $(V_{red}, V_{blue})$  is the solution of the MLCP found by the algorithm. The search space  $S$  of the algorithm is defined as follows:

$$S = \{(V_{red}, V_{blue}) \mid V_{red} \subset V, V_{blue} = V \setminus V_{red}\}. \tag{1}$$

The objective function is as follows:

$$\begin{cases} f((V_{red}, V_{blue})) = \min\{|Er(V_{red})|, |Er(V_{blue})|\} \\ Er(V_{red}) = \{(v, w) \mid \forall (v, w) \in E, v \in V_{red}, w \in V_{red}\} \\ Er(V_{blue}) = \{(v, w) \mid \forall (v, w) \in E, v \in V_{blue}, w \in V_{blue}\} \end{cases} . \tag{2}$$

We define the best solution of the MLCP as follows:

$$f_b((V_{red}, V_{blue})) = \max_{1 \leq j \leq t} \{f((V_{red}, V_{blue})_j)\}. \tag{3}$$

Here,  $t$  is the number of all solutions that can be found by the algorithm in graph  $G$ , and  $(V_{red}, V_{blue})_j$  is the  $j$ th solution of the MLCP found by the algorithm.

Suppose a graph  $G = (V, E)$  is given in Figure 1. Let  $|V| = 6$ ,  $|E| = 8$ , and then the best solution for MLCP of graph  $G$  is shown in Figure 2, and its best value is 2.

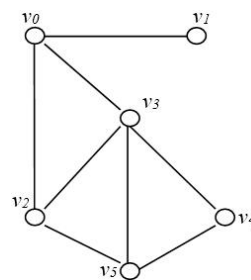


Figure 1. An instance of undirected graph  $G$ .

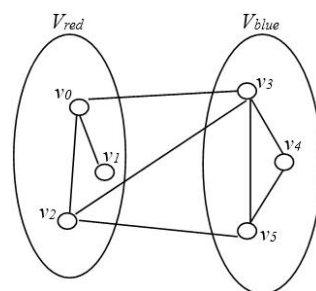


Figure 2. A best solution for MLCP of graph  $G$ .

### 3.2. Initial Population

The algorithm randomly generates population  $X$  consisting of  $p$  individuals. For the given graph  $G = (V, E)$ , in which  $V$  is a set containing  $n$  vertices, and  $E$  is a set containing  $l$  edges,  $m$  vertices are chosen at random from  $V$  to construct set  $V_{red}$  ( $m$  is the initial number of the red vertices); and the remaining vertices are used to construct set  $V_{blue}$ , that is,  $|V_{red}| = m$ ,  $V_{blue} = V \setminus V_{red}$ . Sets  $V_{red}$  and  $V_{blue}$  are seen as a bipartition scheme  $(V_{red}, V_{blue})$ , which is also treated as an individual in population  $X$ . In this way,  $p$  individuals are generated at random initially, and population  $X$  is thus constructed,  $|X| = p$ .

### 3.3. Data Splitting Operation

To avoid the defect of the local optima, we expand the data amount of population  $X$ , hence we get an expanded scope of data search. We use two data splitting strategies to split a bipartition scheme into two. Thus, by using the first data splitting strategy each individual  $X_i$  ( $0 \leq i < p$ ) in population  $X$  generates an individual  $Z_{2 \times i}$ , and by using the second data splitting strategy each individual  $X_i$  ( $0 \leq i < p$ ) generates an individual  $Z_{2 \times i + 1}$ . By doing this,  $p$  individuals in population  $X$  are divided into  $2 \times p$  individuals, and the enlarged population  $Z$  is constructed ( $|X| = p, |Z| = 2 \times p$ ). Figure 3 shows the population expansion, where the red arrow indicates the effects of the first data splitting strategy and the blue arrow the effects of the second data splitting strategy.

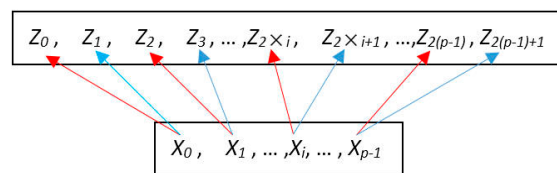


Figure 3. Expanding population  $X$  to population  $Z$ .

The first data splitting strategy of bipartition scheme  $(V_{red}, V_{blue})$  is an important part of memetic algorithm, which consists of five steps.

First step: Degree set  $Degree_{red}$  of all vertices in sub-graph  $G_1(V_{red})$  is calculated.

Second step: Find the minimum degree vertex,  $v$ , from  $Degree_{red}$ . If there is more than one vertex with the same minimum degree, randomly select a vertex among them.

Third step: Degree set  $Degree_{blue}$  of all vertices in sub-graph  $G_2(V_{blue})$  is calculated.

Fourth step: Find the minimum degree vertex,  $w$ , from  $Degree_{blue}$ . If there is more than one vertex with the same minimum degree, randomly select a vertex among them.

Fifth step: A new bipartition scheme  $(V'_{red}, V'_{blue})$  is generated by exchanging the vertices  $v$  and  $w$  in sets  $V_{red}$  and  $V_{blue}$ .

Suppose the number of red vertices is 4 in the given graph  $G = (V, E), V = \{v_0, v_1, \dots, v_9\}$ . We obtain a bipartition scheme  $(V_{red}, V_{blue})$ , as shown in Figure 4a, in which set  $V_{red} = \{v_0, v_3, v_8, v_9\}$ ,  $V_{blue} = \{v_1, v_2, v_4, v_5, v_6, v_7\}$ , where the degree of vertex  $v_9$  in set  $V_{red}$  is the smallest, and that of vertex  $v_4$  in set  $V_{blue}$  is the smallest. After exchanging the two vertices, a new bipartition scheme  $(V'_{red}, V'_{blue})$  is generated. The new bipartition scheme  $(V'_{red}, V'_{blue})$  after splitting is:  $V'_{red} = \{v_0, v_3, v_4, v_8\}$ ,  $V'_{blue} = \{v_1, v_2, v_5, v_6, v_7, v_9\}$ , as shown in Figure 4b.

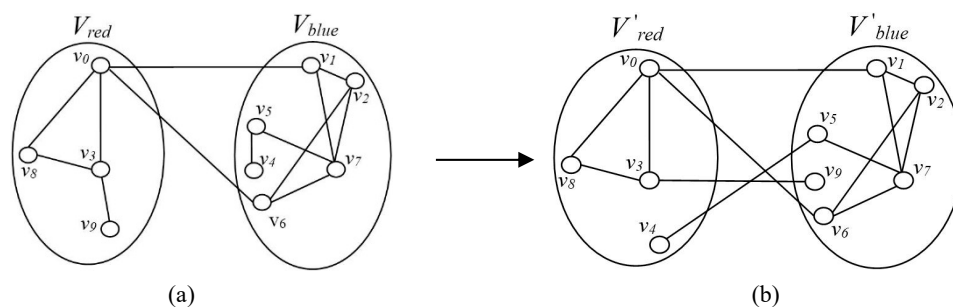


Figure 4. Bipartition scheme splitting: (a) bipartition scheme before the splitting; and (b) bipartition scheme after the splitting.

The second data splitting strategy is described as follows: for a given bipartition scheme  $(V_{red}, V_{blue})$ , in which  $|V_{red}| = m, V_{blue} = V \setminus V_{red}$ , randomly a vertex  $v$  in set  $V_{red}$  is chosen, and a vertex  $w$  in set  $V_{blue}$  is randomly chosen. Then, vertices  $v$  and  $w$  in set  $V_{red}$  and set  $V_{blue}$  are exchanged to generate a new bipartition scheme  $(V''_{red}, V''_{blue})$ .

### 3.4. Search for the Individuals

Memetic algorithm needs to carry out a heuristic search for each individual in the population by an effective and improved K-OPT local search algorithm designed.

We first obtain an individual  $Z_j$  which is a  $(V_{red}, V_{blue})_j, ((V_{red})_j \subset V, (V_{blue})_j = V \setminus (V_{red})_j, 0 \leq j < |Z|)$ . The local search algorithm is implemented to add as many selected vertices, acquired through our vertex adding strategy, as possible in set  $(V_{blue})_j$  to set  $(V_{red})_j$ , until the stop condition set by the algorithm is met. Thus, a new bipartition scheme  $(V_{red}, V_{blue})'_j$  is constructed. Generally speaking, in  $(V_{red}, V_{blue})'_j, |Er((V'_{red})_j)|$  is approximately equal to  $|Er((V'_{blue})_j)|$ . Then, the objective function value  $f((V_{red}, V_{blue})'_j)$  is calculated using Equation (2). If  $f((V_{red}, V_{blue})'_j) > f(s_1)$ , the memetic algorithm accepts the constructed bipartition as the new best solution. The improved K-OPT local search algorithm is implemented by the New\_K-OPT\_MLCP (Algorithm 2).

Our vertex adding strategy is described as follows:

We first need to define the following three vectors as the foundation on which the vertex adding strategy is constructed.

- $CC_{red}$ : The current set of red vertices in graph  $G$ .
- $PAV_{red}$ : The vertex set of possible additions, i.e., each vertex is connected to at least one vertex of  $CC_{red}$ .

$$PAV_{red} = \{v | v \in V_{blue}, \exists w \in CC_{red}, (v, w) \in E, V_{blue} = V \setminus CC_{red}\}. \tag{4}$$

- $GPAV_{red}$ : The degree set of vertices  $v_i \in PAV_{red}$  in sub-graph  $G'(PAV_{red})$ , where  $PAV_{red} \subseteq V_{blue}$ .

$$\begin{cases} GPAV_{red}[i] = degree_{G'(PAV_{red})}(v_i) = |\{a | \forall a \in PAV_{red}, (v_i, a) \in E\}| \\ v_i \in PAV_{red} \\ 0 \leq i \leq |PAV_{red}| - 1 \end{cases} . \tag{5}$$

To avoid the local optima defect, the vertex adding strategy is employed in two phases: vertex addition phase (Algorithm 2, Lines 8–12) and vertex deletion phase (Algorithm 2, Lines 14–18).

In the vertex addition phase of  $CC_{red}$ , we obtain  $PAV_{red}$  from the current  $CC_{red}$ , then select a vertex  $w$  from  $PAV_{red}$  and move it from  $V_{blue}$  to  $CC_{red}$ , and finally update  $PAV_{red}$ . The vertex addition phase is repeatedly executed until  $PAV_{red} = \emptyset$  or  $|Er(CC_{red})| > |Er(V_{blue})|$ .

In the vertex deletion phase of  $CC_{red}$ , we select a vertex  $u$  from  $CC_{red}$ , then delete the vertex  $u$  from  $CC_{red}$ , and add it to  $V_{blue}$ . Go back to the vertex addition phase again to continue the execution until the set ending conditions are met.

The approach to select vertex  $w$  is first to obtain a  $GPAV_{red}$  in sub-graph  $G'(PAV_{red})$ , then to calculate the vertex selection probability value  $\rho(w_i)$  of each vertex  $w_i$  ( $0 \leq i < |PAV_{red}|$ ) in  $PAV_{red}$ , and finally to select vertex  $w_i$  to maximize  $\rho(w_i)$ . If there are more than one vertex with the maximum value of  $\rho(w_i)$ , randomly select one.

$$\begin{cases} \maxd = \max_{w_i \in PAV_{red}, 0 \leq i < |PAV_{red}|} (degree_{G'(PAV_{red})}(w_i)) \\ \rho(w_i) = \frac{\maxd + 1 - GPAV_{red}[i]}{\maxd + 1} \\ w_i \in PAV_{red} \\ 0 \leq i < |PAV_{red}| \end{cases} . \tag{6}$$

A vertex  $w$  is selected according to the following criterion:

$$f_1(w) = \max_{0 \leq i < |PAV_{red}|} (\rho(w_i)). \tag{7}$$

We found that the larger the probability value  $\rho(w_i)$  of vertex  $w_i$  is, the smaller the degree value of vertex  $w_i$  becomes.

The approach of vertex  $u$  selection is as follows: we assume that  $(CC_{red}, V_{blue})^{(j)}$  is the bipartition scheme with no possible additions. We successively take the value of  $i$  from the range of  $0 - (|CC_{red}| - 1)$ , and then in turn execute  $CC_{red}^{(j)} \setminus \{u_i\}$  as follows: delete vertex  $u_i$  from  $CC_{red}^{(j)}$  successively to generate new bipartition schemes  $(CC'_{red}, V'_{blue})_i$ , i.e.,  $(CC'_{red})_i \leftarrow CC_{red}^{(j)} \setminus \{u_i\}$ ,  $(V'_{blue})_i \leftarrow V_{blue}^{(j)} \cup \{u_i\}$ ,  $u_i \in CC_{red}$ ,  $0 \leq i < |CC_{red}|$ , and finally obtain  $Ed((CC'_{red}, V'_{blue})_i)$ .

$$\begin{cases} Ed((CC'_{red}, V'_{blue})_i) = |\{(x, y) | \forall (x, y) \in E, x \in (CC'_{red})_i, y \in (V'_{blue})_i\}| \\ (V'_{blue})_i = V \setminus (CC'_{red})_i \\ 0 \leq i < |CC_{red}^{(j)}| \end{cases} \quad (8)$$

The maximum value  $maxdd$  is found from all the values of  $Ed$ , and the vertex selection probability value  $\rho_2(u_i)$  of vertex  $u_i$  can be calculated:

$$\begin{cases} maxdd = \max_{0 \leq i < |CC_{red}^{(j)}|} (Ed((CC'_{red}, V'_{blue})_i)) \\ \rho_2(u_i) = \frac{maxdd + 1 - Ed((CC'_{red}, V'_{blue})_i)}{maxdd + 1} \\ u_i \in CC_{red}^{(j)} \\ 0 \leq i < |CC_{red}^{(j)}| \end{cases} \quad (9)$$

A vertex  $u$  is selected according to the following criterion:

$$f_2(u) = \max_{u_i \in CC_{red}^{(j)}, 0 \leq i < |CC_{red}^{(j)}|} (\rho_2(u_i)). \quad (10)$$

We found that the larger the probability value  $\rho_2(u_i)$  of vertex  $u_i$  is, the smaller the corresponding  $Ed((CC'_{red}, V'_{blue})_i)$  becomes. If there are more than one vertices with the maximum value of  $\rho_2(u_i)$ , randomly select one.

At each generation, the variable  $gA$  stores the value of vertices number successfully added to the  $CC_{red}$  for now, and the variable  $gmaxA$  stores the value of vertices number successfully added to the  $CC_{red}$  during the previous generations. If  $gA > gmaxA$ , the incumbent  $CC_{red}$  has more red vertices than the previous ones found by the local search algorithm. Then,  $gmaxA$  is updated with the value of  $gA$  and the incumbent  $CC_{red}$  is stored to the set  $Abest$  (Algorithm 2, Line 12). In the vertex addition phase, the value of  $gA + 1$  replaces that of  $gA$  ( $gA \leftarrow gA + 1$ ) after a vertex is added. In the vertex deletion phase, the value of  $gA - 1$  replaces that of  $gA$  ( $gA \leftarrow gA - 1$ ) after a vertex is deleted.

At the completion of the inner loop statements, when  $gmaxA > 0$ ,  $CC_{red}$ , which has the greatest number of vertices, is stored in set  $Abest$ , then the incumbent  $CC_{red}$  is updated with  $Abest$ . When  $gmaxA = 0$ ,  $CC_{red}$ , which has the greatest number of vertices, is stored in set  $Aprev$ ; if the execution of the inner loop does not find any new set  $CC_{red}$  that has more vertices,  $Aprev$  is adopted as  $CC_{red}$  generated by the previous execution of the inner loop and will replace the incumbent  $CC_{red}$  (Algorithm 2, Lines 22–28).

The algorithm's search efficiency may be reduced because of the roundabout searching characteristics. To solve this problem, a restricting tabu table is added to the local search algorithm.

The tabu table can be presented by two-dimensional array or one-dimensional array. We adopt the one-dimensional array  $T$ , set the tabu tenure value as  $L$ , and store the iteration numbers of running the local search algorithm into the tabu table. When the algorithm runs reach iteration value  $c$ , and if  $(c - T[w]) < L$  or if  $(c - T[u]) < L$ , it means vertex  $w$  or  $u$  has been processed and the vertex should be re-selected. Otherwise, the current value  $c$  is stored in the tabu table, i.e.,  $T[w] \leftarrow c$  or  $T[u] \leftarrow c$ .

---

**Algorithm 2** New\_K-OPT\_MLCP ( $G, Z_j, T, L$ ).

---

Require:

$Z_j$ : the  $j$ th individual  $(V_{red}, V_{blue})_j, (V_{red})_j \subset V, (V_{blue})_j = V \setminus (V_{red})_j$

$T$ : tabu table

$L$ : tabu tenure value

Output:  $s_2$ , the solution found by local search algorithm

begin

1  $(CC_{red}, V_{blue}) \leftarrow (V_{red}, V_{blue})_j$ ;

2 according to  $CC_{red}$  and  $V_{blue}$ ,  $PAV_{red}$  and  $GPAV_{red}$  are obtained;

3 repeat

4  $Aprev \leftarrow CC_{red}, DA \leftarrow Aprev, PL \leftarrow \{v_0, v_1, \dots \dots v_{n-1}\}, gA \leftarrow 0, gmaxA \leftarrow 0, c \leftarrow 0$ ;

5 repeat

6  $c \leftarrow c + 1$ ;

7 if  $|PAV_{red} \cap PL| > 0$  and  $|Er(CC_{red})| < |Er(V_{blue})|$  then

8 select vertex  $w$  according to  $f_1(w)$ , if there are multiple vertices, select a vertex  $w$  randomly;

9 if  $c - T[w] < L$  then select a non-tabu vertex  $w$  according to  $f_1(w)$ ;

10  $T[w] \leftarrow c$ ;

11  $CC_{red} \leftarrow CC_{red} \cup \{w\}, V_{blue} \leftarrow V_{blue} \setminus \{w\}, gA \leftarrow gA + 1, PL \leftarrow PL \setminus \{w\}$ ;

12 if  $gA > gmaxA$  then  $gmaxA \leftarrow gA, Abest \leftarrow CC_{red}$ ;

13 else

14 select vertex  $u$  according to  $f_2(u)$ , if there are multiple vertices, select a vertex  $u$  randomly;

15 if  $c - T[u] < L$  then select a non-tabu vertex  $u$  according to  $f_2(u)$ ;

16  $T[u] \leftarrow c$ ;

17  $CC_{red} \leftarrow CC_{red} \setminus \{u\}, V_{blue} \leftarrow V_{blue} \cup \{u\}, gA \leftarrow gA - 1, PL \leftarrow PL \setminus \{u\}$ ;

18 if  $u \in Aprev$  then  $DA \leftarrow DA \setminus \{u\}$ ;

19 end if

20 based on  $CC_{red}$  and  $V_{blue}$ ,  $PAV_{red}$  and  $GPAV_{red}$  are updated;

21 until  $|DA| = 0$  or the cut-off time condition for CPU running is met;

22 if  $gmaxA > 0$  then

23  $CC_{red} \leftarrow Abest$ ;

24  $V_{blue} \leftarrow V \setminus CC_{red}$ ;

25 else

26  $CC_{red} \leftarrow Aprev$ ;

27  $V_{blue} \leftarrow V \setminus CC_{red}$ ;

28 end if

29 until  $gmaxA \leq 0$  or the cut-off time condition for CPU running is met;

30  $(V_{red}, V_{blue})_j \leftarrow (CC_{red}, V_{blue})$ ;

31  $s_2 \leftarrow (V_{red}, V_{blue})_j$ ;

32 return  $s_2$ ;

end

---

### 3.5. Evolutionary Operation of Population

An evolutionary operation in the population  $X$  is needed to quickly find the best solution of MLCP. We sort the individuals  $Z_j$  ( $0 \leq j < 2 \times p$ ) in population  $Z$  in ascending order according to the calculated value of objective function  $f_*$  in Equation (11). Then, we replace the individuals  $X_0 - X_{p-1}$  of population  $X$  with the individuals  $Z_0 - Z_{p-1}$  to complete the evolutionary operation.

$$f_*((V_{red}, V_{blue})_j) = |\{(a, b) | \forall (a, b) \in E, a \in (V_{red})_j, b \in (V_{blue})_j\}|, 0 \leq j < 2 \times p. \quad (11)$$

Evolution operation of the population is represented by Evolution\_population (Algorithm 3).



**Algorithm 3** Evolution\_population ( $Z, X$ ).

---

Require:  
 $Z$ : population,  $|Z| = 2 \times p$   
Output: population  $X$   
begin  
1  $j \leftarrow 0$ ;  
2 while  $j < 2 \times p$  do  
3  $R[j] \leftarrow f^*((V_{red}, V_{blue})_j)$ ;  
4  $j \leftarrow j + 1$ ;  
5 end while  
6 Individuals  $Z_0 - Z_{2 \times p - 1}$  of population  $Z$  are sorted in ascending order according to the value of set  $R$ ;  
7  $(X_0 - X_{p-1}) \leftarrow (Z_0 - Z_{p-1})$ ;  
8 return  $X$ ;  
end

---

**3.6. Disturbance Operation**

To further improve the search ability of the algorithm and find better values, we add a disturbance operation into the memetic algorithm. This disturbance operation is executed  $k$  times.

**Algorithm 4** Disturbance\_operation( $X, b, k, d1, d2, W_{best}$ ).

---

Require:  
 $X$ : set that stores the population  
 $b, k$ : control parameters of disturbance operation  
 $d1, d2$ : control variables of disturbance operation  
 $W_{best}$ : variable that stores the value of the objective function  $f(s_t)$ , in which  $s_t$  is the current best solution of MLCP  
Output:  $s_1$ , a better solution found by the local search algorithm  
 $f(s_1)$ , the value of the objective function  
 $d1, d2$ , values of the control variables  
begin  
1  $d1 \leftarrow d1 + 1$ ;  
2 if  $d1 = b$  then  
3  $d2 \leftarrow d2 + 1$ ;  
4 if  $d2 \leq k$  then  
5 if  $d2 = 1$  then  
6 randomly choose  $X_j$  from  $X$  and start disturbance to generate a new  $X'_j$ ;  
7  $W \leftarrow \text{New\_K-OPT\_MLCP}(G, X'_j, T, L)$ ;  
8 if  $f(W) > W_{best}$  then  $s_1 \leftarrow W, W_{best} \leftarrow f(W)$ ;  
9  $t \leftarrow X'_j$ ;  
10 else  
11 start disturbance  $t$  to generate a new  $t'$ ;  
12  $W \leftarrow \text{New\_K-OPT\_MLCP}(G, t', T, L)$ ;  
13 if  $f(W) > W_{best}$  then  $s_1 \leftarrow W, W_{best} \leftarrow f(W)$ ;  
14  $t \leftarrow t'$ ;  
15 end if  
16  $d1 \leftarrow d1 - 1$   
17 else  
18  $d1 \leftarrow 0, d2 \leftarrow 0$ ;  
19 end if  
20 end if  
21 return  $s_1, W_{best}, d1, d2$ ;  
end

---

When the number of iterations is  $b$ , disturbance operation begins and randomly selects an individual  $(V_{red}, V_{blue})_j$  ( $0 \leq j < |X|$ ) from population  $X$ , and chooses at random a vertex from set  $V_{red}$  and a vertex from set  $V_{blue}$ , then the two vertices are exchanged to generate a new  $(V_{red}, V_{blue})'$ . Then, employ the New\_K-OPT\_MLCP algorithm to search in  $(V_{red}, V_{blue})'$ ; if a better solution of MLCP is found, the memetic algorithm will accept it.

The disturbance operation is represented by Disturbance\_operation (Algorithm 4).

In Memetic\_D\_O\_MLCP algorithm, setting the value of  $b$  and  $k$  will determine the disturbance operation's starting condition and the number of times of its execution. In Disturbance\_operation algorithm, Lines 1, 3, 16, and 18 store the modified values of variables  $d1$  and  $d2$ , which are the threshold values needed to start off a new disturbance operation.

#### 4. Simulated Annealing Algorithm

Simulated annealing algorithm, a classical heuristic algorithm to solve combinatorial optimization problems, starts off from a higher initial temperature. With the decreasing of temperature parameters, the algorithm can randomly find the global best solution of problems instead of the local optima by combining the perturbations triggered by the probabilities.

For a given graph  $G$ , simulated annealing algorithm finds a coloring bipartition scheme  $(V_{red}, V_{blue})$  of  $V$  which maximizes  $\min\{|Er(V_{red})|, |Er(V_{blue})|\}$ . With parameters  $T_0$  (initial temperature value),  $\alpha$  (cooling coefficient) and  $T_{end}$  (the end temperature value), first, the algorithm divides the vertex set  $V$  into two sets, i.e.,  $V_{red}$  and  $V_{blue}$  ( $V_{red} = \emptyset, V_{blue} = V$ ) and the initial value of the best solution of MLCP  $C_{best}$  is set to 0. Next, a vertex is randomly selected in  $V_{blue}$  and moved from  $V_{blue}$  to  $V_{red}$ ; here,  $|V_{red}| = 1, V_{blue} = V \setminus V_{red}$ . Then, the algorithm repeats a series of generations to explore the search space defined by the set of all 2-colorings. At each generation, a vertex is randomly selected in  $V_{blue}$  and moved from  $V_{blue}$  to  $V_{red}$ . The additions will take place in the following three forms:

When  $2 > |V_{red}|$  and  $2 \leq |V_{blue}|$ , a vertex is randomly selected in  $V_{blue}$  and moved from  $V_{blue}$  to  $V_{red}$  to generate a new coloring bipartition scheme  $(V'_{red}, V'_{blue})$  and the new status is accepted.

When  $2 > |V_{blue}|$  and  $2 \leq |V_{red}|$ , a vertex is randomly selected in  $V_{red}$  and moved from  $V_{red}$  to  $V_{blue}$  to generate a new coloring bipartition scheme  $(V'_{red}, V'_{blue})$  and accepted as the new status.

When  $2 \leq |V_{red}|$  and  $2 \leq |V_{blue}|$ , a vertex is randomly selected from  $V_{red}$  and one randomly from  $V_{blue}$ , then the two vertices are exchanged to generate a new coloring bipartition scheme  $(V'_{red}, V'_{blue})$ , only when  $R_1((V'_{red}, V'_{blue})) \geq R_1((V_{red}, V_{blue}))$ , the scheme is accepted as a new status. Otherwise, the probability will decide whether to accept it as a new status or not.

Once the new status is accepted, if  $C_{best} < R_1((V'_{red}, V'_{blue}))$ , then the bipartition scheme  $(V'_{red}, V'_{blue})$  is accepted as the best solution of MLCP.

At the end of each generation, the temperature  $T$  cools down until  $T \leq T_{end}$  according to  $T = T \times \alpha$ , where  $\alpha \in (0,1)$ . The algorithm runs iteratively as per the above steps until the stop condition is met.

The best solution found by the algorithm is  $R_b((V_{red}, V_{blue}))$ , i.e.,

$$\begin{cases} R_1((V_{red}, V_{blue})_j) = \min\{|Er((V_{red})_j)|, |Er((V_{blue})_j)|\} \\ R_b((V_{red}, V_{blue})) = \max_{0 \leq j < t} \{R_1((V_{red}, V_{blue})_j)\} \end{cases} \quad (12)$$

Here,  $t$  is the number of all solutions that can be found by the simulated annealing algorithm in graph  $G$ , and  $(V_{red}, V_{blue})_j$  is the  $j$ th solution of MLCP.

The simulated annealing algorithm is represented by SA (Algorithm 5).

---

**Algorithm 5** SA( $G, V_{red}, V_{blue}, T_0, \alpha, T_{end}$ ).

---

Require:  $G = (V, E), |V| = n, |E| = l$

$V_{red}$ : a set of red vertices in graph  $G$

$V_{blue}$ : a set of blue vertices in graph  $G$

$T_0$ : initial temperature value

$\alpha$ : cooling coefficient

$T_{end}$ : end temperature value

Output:  $s_3$ , the best solution found by SA algorithm

$R_1(s_3)$ , value of the objective function

begin

1  $C_{best} \leftarrow 0$ ;

2 repeat

3  $T \leftarrow T_0$ ;

4 initialize  $V_{red}$  and  $V_{blue}$ , randomly select a vertex in  $V_{blue}$  and moved from  $V_{blue}$  to  $V_{red}$ ;

5 while  $T > T_{end}$  do

6 if  $2 > |V_{red}|$  and  $2 \leq |V_{blue}|$  then

7 a vertex is randomly selected in  $V_{blue}$  and moved from  $V_{blue}$  to  $V_{red}$  to generate a new bipartition scheme  $(V'_{red}, V'_{blue})$ ;

8  $(V_{red}, V_{blue}) \leftarrow (V'_{red}, V'_{blue})$ ;

9 if  $C_{best} < R_1((V'_{red}, V'_{blue}))$  then  $C_{best} \leftarrow R_1((V'_{red}, V'_{blue}))$ ,  $s_3 \leftarrow (V'_{red}, V'_{blue})$ ;

10 else if  $2 > |V_{blue}|$  and  $2 \leq |V_{red}|$  then

11 a vertex is randomly selected in  $V_{red}$  and moved from  $V_{red}$  to  $V_{blue}$  to generate a new bipartition scheme  $(V'_{red}, V'_{blue})$ ;

12  $(V_{red}, V_{blue}) \leftarrow (V'_{red}, V'_{blue})$ ;

13 if  $C_{best} < R_1((V'_{red}, V'_{blue}))$  then  $C_{best} \leftarrow R_1((V'_{red}, V'_{blue}))$ ,  $s_3 \leftarrow (V'_{red}, V'_{blue})$ ;

14 else if  $2 \leq |V_{red}|$  and  $2 \leq |V_{blue}|$  then

15 according to  $(V_{red}, V_{blue})$ , a vertex is randomly selected from  $V_{red}$  and a vertex randomly selected from  $V_{blue}$ ;

16 the two vertices are exchanged to generate a new bipartition scheme  $(V'_{red}, V'_{blue})$ ;

17 if  $R_1((V_{red}, V_{blue})) \leq R_1((V'_{red}, V'_{blue}))$  then

18  $(V_{red}, V_{blue}) \leftarrow (V'_{red}, V'_{blue})$ ;

19 if  $C_{best} < R_1((V'_{red}, V'_{blue}))$  then  $C_{best} \leftarrow R_1((V'_{red}, V'_{blue}))$ ,  $s_3 \leftarrow (V'_{red}, V'_{blue})$ ;

20 else if random number in  $(0, 1) < e^{\frac{R_1((V'_{red}, V'_{blue})) - R_1((V_{red}, V_{blue}))}{T}}$  then

21  $(V_{red}, V_{blue}) \leftarrow (V'_{red}, V'_{blue})$ ;

22 end if

23 end if

24  $T \leftarrow T \times \alpha$ ;

25 end while

26 until stop condition is met;

27 return  $s_3, C_{best}$ ; /\* $C_{best}$  is the value of the objective function  $R_1(s_3)$  \*/

end

---

## 5. Greedy Algorithm

Greedy algorithm aims at making the optimal choice at each stage with the hope of finding a global best solution. For a given graph  $G$ , greedy algorithm finds a coloring bipartition scheme  $(V_{red}, V_{blue})$  of  $V$  which maximizes  $\min\{|Er(V_{red})|, |Er(V_{blue})|\}$ .

When a graph  $G = (V, E)$  is given, the algorithm divides vertex set  $V$  into two sets, i.e.,  $V_{red}$  and  $V_{blue}$  ( $V_{red} = \emptyset, V_{blue} = V$ ), and the initial value of the best solution of MLCP  $C_{best}$  is set to 0. Next, a vertex is randomly selected in  $V_{blue}$  and moved from  $V_{blue}$  to  $V_{red}$ , here  $|V_{red}| = 1, V_{blue} = V \setminus V_{red}$ . Then, the algorithm repeats a series of generations to explore the search space defined by the set of all 2-colorings. At each generation, based on sub-graph  $G'(V_{blue})$ , choose a vertex  $w$  of the minimum degree ( $w \in V_{blue}$ ); if there are more than one vertex  $w$  with the same minimum degree, randomly select a

vertex among them. Then, add the vertex from  $V_{blue}$  to  $V_{red}$ , that is:  $V_{red} \leftarrow V_{red} \cup \{w\}$ ,  $V_{blue} \leftarrow V_{blue} \setminus \{w\}$ , thus a new bipartition scheme  $(V'_{red}, V'_{blue})$  is generated, and, when  $R_2((V'_{red}, V'_{blue})) > C_{best}$ , the scheme is accepted as the best solution. The generation will be repeated until  $|Er(V_{red})| > |Er(V_{blue})|$ .

The algorithm runs iteratively as per the above steps until the stop condition is met.

The best solution found by the algorithm is  $R_g((V_{red}, V_{blue}))$ , i.e.,

$$\begin{cases} R_2((V_{red}, V_{blue})_j) = \min\{|Er((V_{red})_j)|, |Er((V_{blue})_j)|\} \\ R_g((V_{red}, V_{blue})) = \max_{0 \leq j < t} \{R_2((V_{red}, V_{blue})_j)\} \end{cases} \quad (13)$$

Here,  $t$  is the number of all solutions that can be found by the greedy algorithm in graph  $G$ , and  $(V_{red}, V_{blue})_j$  is the  $j$ th solution of MLCP.

The greedy algorithm is represented by Greedy (Algorithm 6).

---

**Algorithm 6** Greedy ( $G, V_{red}, V_{blue}$ ).

---

Require:  $G: G = (V, E), |V| = n, |E| = l$

$V_{red}$ : a set of red vertices in graph  $G$

$V_{blue}$ : a set of blue vertices in graph  $G$

Output:  $s_4$ , the best solution found by greedy algorithm

$R_2(s_4)$ , value of the objective function

begin

1  $C_{best} \leftarrow 0$ ;

2 repeat

3  $V_{red} \leftarrow \emptyset, V_{blue} \leftarrow V$ ;

4 randomly select a vertex in  $V_{blue}$  and moved from  $V_{blue}$  to  $V_{red}$ ;

5 repeat

6  $(V'_{red}, V'_{blue}) \leftarrow (V_{red}, V_{blue})$ ;

7 if  $R_2((V'_{red}, V'_{blue})) > C_{best}$  then  $s_4 \leftarrow (V'_{red}, V'_{blue}), C_{best} \leftarrow R_2((V'_{red}, V'_{blue}))$ ;

8 select a vertex  $w$  with the minimum degree from sub-graph  $G'(V_{blue})$ , if there are multiple vertices, select a vertex  $w$  randomly;

9  $V_{red} \leftarrow V_{red} \cup \{w\}, V_{blue} \leftarrow V_{blue} \setminus \{w\}$ ;

10 until  $|Er(V_{red})| > |Er(V_{blue})|$ ;

11 until stop condition is met;

12 return  $s_4, C_{best}$ ; /\*  $C_{best}$  is the value of the objective function  $R_2(s_4)$  \*/

end

---

## 6. Experimental Results

All algorithms were programmed in C++, and run on a PC with Intel Pentium(R) G630 processor 2.70 GHz and 4 GB memory under Windows 7 (64 bits), and the test graphs adopted were the benchmark DIMACS proposed in [5]. We compared the search results by using memetic algorithm, simulated annealing algorithm, and greedy algorithm. Then, the results of memetic algorithm were compared with those obtained from using artificial bee colony algorithm [4], tabu search algorithm [5] and variable neighborhood search algorithm [5].

The first group of experiments was performed to adjust the key parameters and analyze their influence on Memetic\_D\_O\_MLCP. As is known to all, the most important parameters in Memetic\_D\_O\_MLCP implementations are the values of  $p$  and  $L$ , which determine the number of the individuals of the population and the tabu tenure value during the search process. To find the most suitable values of  $p$  and  $L$  for Memetic\_D\_O\_MLCP approach to MLCP, we performed experiments with different values of  $p$  and  $L$ . Memetic\_D\_O\_MLCP was run 10 times for each benchmark instance, and each test lasted 30 min.

The results of experiments are summarized in Table 1, organized as follows: in the first column, *Inst* the benchmark instance name is given, containing the vertices set  $V$ ; and, in the second column,

$m$  is the initial number of red vertices in the benchmark graph. For each  $p \in \{4, 12, 20\}$  and  $L \in \{10, 60, 90\}$ , column *Best* contains the best values of MLCP solution found by the algorithm, while column *Avg* represents the average values of MLCP solution found by the algorithm. For each instance, the best values of *Best* and *Avg* are shown in italics. The analysis of the obtained results shows that values of  $p$  and  $L$  influence the solution quality. For example, the number of best values of *Best* is 5 for combination  $p = 12$  and  $L = 90$ ; *Best* 3 for  $p = 4, L = 90$  and  $p = 20, L = 60$ ; *Best* 2 for  $p = 4$  and  $L = 10, p = 4$  and  $L = 60, p = 12$  and  $L = 60$ ; *Best* 1 for  $p = 20$  and  $L = 90$ ; *Best* 0 for  $p = 12$  and  $L = 10, p = 20$  and  $L = 10$ . Meanwhile, the number of best values of *Avg* is 2 for combinations  $p = 12$  and  $L = 90, p = 4$  and  $L = 90$ ; *Avg* 1 for  $p = 4$  and  $L = 10, p = 4$  and  $L = 60, p = 12$  and  $L = 60, p = 20$  and  $L = 10, p = 20$  and  $L = 60, p = 20$  and  $L = 90$ .

In Table 1, one observes that, for combination  $p = 12$  and  $L = 90$ , the number of instances where the Memetic\_D\_O\_MLCP achieved the best value for *Best* and *Avg* is 5 and 2, respectively. For all other combinations, these numbers are the biggest. Therefore, we used the combination in all other experiments.

The second groups of tests compared the search results of Memetic\_D\_O\_MLCP, SA algorithm and Greedy algorithm, each having been run 20 times for each benchmark instance with the cut-off time of 30 min. In simulated annealing algorithm, the initial temperature  $T_0$  is set at 1000, the cooling coefficient  $\alpha$  at 0.9999 and the end temperature  $T_{end}$  at 0.0001. The results of experiments are summarized in Table 2, organized as follows: in the second column,  $|V|$  is the number of vertices; and, in the third column,  $|E|$  is the number of edges. For each instance the best values of *Best* are shown in italics. Among 59 instances, the search results of Memetic\_D\_O\_MLCP, SA algorithm and Greedy algorithm were the same in the instances *myciel3.col*, *myciel4.col*, *queen5\_5.col* and *queen6\_6.col*. Memetic\_D\_O\_MLCP and Greedy algorithm could find equivalent best value of four instances (i.e., *queen7\_7.col*, *queen8\_8.col*, *queen8\_12.col*, and *queen9\_9.col*). In the remaining 51 instances, Memetic\_D\_O\_MLCP could find the best results of 38 instances (accounting for 75%), and Greedy algorithm could find the best results of 13 instances (accounting for 25%). The experiments showed that Memetic\_D\_O\_MLCP could find more instances of best values.

The third group of tests aimed at comparing the search results after each algorithm was run on four benchmark instances, namely *myciel6.col*, *homer.col*, *mulsol.i.5.col* and *inithx.i.1.col*, for the first one 100 s. The results that algorithms found were collected at an interval of 10 s. The running time was regarded as the X coordinate on the axis and the value of MLCP solution as the Y coordinate.

Figure 5 illustrates that Memetic\_D\_O\_MLCP can find the best result at each time node.

**Table 1.** Experiments with parameters  $p$  and  $L$ .

<i>Inst</i>	<i>m</i>	<i>p</i> = 4						<i>p</i> = 12						<i>p</i> = 20					
		<i>L</i> = 10		<i>L</i> = 60		<i>L</i> = 90		<i>L</i> = 10		<i>L</i> = 60		<i>L</i> = 90		<i>L</i> = 10		<i>L</i> = 60		<i>L</i> = 90	
		<i>Best</i>	<i>Avg</i>	<i>Best</i>	<i>Avg</i>	<i>Best</i>	<i>Avg</i>	<i>Best</i>	<i>Avg</i>	<i>Best</i>	<i>Avg</i>	<i>Best</i>	<i>Avg</i>	<i>Best</i>	<i>Avg</i>	<i>Best</i>	<i>Avg</i>	<i>Best</i>	<i>Avg</i>
fpsol2.i.1.col	V /5	3035	2844	3033	2857	3582	2928	3015	2840	3002	2845	3029	2837	3071	2929	2942	2727	2998	2843
fpsol2.i.2.col	V /5	2120	1965	2375	1953	2183	1860	2272	1972	2176	1944	2450	1969	2324	2016	2169	1932	2310	2038
fpsol2.i.3.col	V /5	2141	1930	2331	1931	2266	2048	2333	1960	2330	1930	2115	1900	1981	1817	2397	1986	2281	1951
DSJC125.1.col	V /5	255	252	254	252	254	251	254	252	255	252	255	252	254	252	255	253	254	252
DSJC125.5.col	V /5	1081	1072	1082	1067	1088	1078	1084	1076	1087	1080	1089	1080	1084	1078	1087	1072	1084	1074
queen15_15.col	V /5	1716	1699	1721	1678	1721	1694	1693	1650	1705	1692	1716	1681	1659	1632	1704	1638	1674	1641
queen16_16.col	V /5	2090	2050	2087	2049	2087	2055	2040	1976	2062	1994	2098	2026	2036	1990	2032	1995	2040	1996
multsol.i.4.col	V /5	1704	1694	1704	1698	1704	1694	1701	1695	1704	1696	1704	1698	1700	1697	1704	1697	1704	1694

**Table 2.** Test results of the Memetic\_D\_O\_MLCP, SA, and Greedy on benchmark instances.

Inst	V	E	Memetic_D_O_MLCP			SA		Greedy	
			<i>m</i>	<i>Best</i>	<i>Avg</i>	<i>Best</i>	<i>Avg</i>	<i>Best</i>	<i>Avg</i>
anna.col	138	986	V /5	200	198	160	154	131	131
david.col	87	812	V /5	158	157	133	130	140	140
DSJC125.1.col	125	736	V /5	255	252	222	217	240	238
DSJC125.5.col	125	3891	V /5	1091	1083	1025	1021	1075	1067
DSJC125.9.col	125	6961	V /5	1798	1789	1761	1756	1776	1772
fpsol2.i.1.col	496	11654	V /5	3091	2896	3016	2982	2510	2502
fpsol2.i.2.col	451	8691	V /5	2290	2046	2267	2242	1707	1703
fpsol2.i.3.col	425	8688	V /5	2387	1996	2291	2247	1664	1664
games120.col	120	1276	V /5	288	281	215	209	284	284
homer.col	561	3258	10	662	655	450	441	492	489
huck.col	74	602	V /5	130	129	111	109	113	113
inithx.i.1.col	864	18707	10	6644	6153	4861	4773	6167	6050
inithx.i.2.col	645	13979	10	5622	5104	3641	3597	3571	3481
inithx.i.3.col	621	13969	10	5589	4756	3643	3593	3131	3111
jean.col	80	508	V /5	111	110	98	95	106	106
latin_square_10.col	900	307350	10	85161	85072	77006	75770	85185	85185
le450_5a.col	450	5714	10	1824	1801	1516	1495	1834	1827
le450_5b.col	450	5734	10	1820	1801	1512	1498	1843	1831
le450_5c.col	450	9803	10	2985	2951	2541	2530	3014	2995
le450_5d.col	450	9757	10	2943	2913	2542	2519	2972	2958
le450_15b.col	450	8169	10	2395	2355	2138	2120	2409	2398
le450_15c.col	450	16680	10	4530	4476	4294	4267	4560	4539
le450_15d.col	450	16750	10	4586	4542	4320	4289	4626	4609
le450_25a.col	450	8260	10	2467	2434	2183	2148	2466	2454
le450_25b.col	450	8263	10	2664	2606	2172	2149	2682	2658
le450_25c.col	450	17343	10	4711	4652	4457	4438	4728	4714
le450_25d.col	450	17425	10	4872	4807	4470	4449	4883	4875
miles250.col	128	774	V /5	185	184	145	137	183	183
miles500.col	128	2340	V /5	522	522	393	381	518	518
miles750.col	128	4226	V /5	870	870	673	638	849	849
miles1000.col	128	6432	V /5	1183	1180	954	921	1156	1156
miles1500.col	128	10396	V /5	1645	1616	1461	1421	1485	1484
multsol.i.1.col	197	3925	V /5	1697	1690	1193	1152	1624	1624
multsol.i.2.col	188	3885	V /5	1685	1682	1153	1117	1202	1189
multsol.i.3.col	184	3916	V /5	1695	1692	1174	1131	1211	1174
multsol.i.4.col	185	2946	V /5	1704	1701	1172	1134	1218	1195
multsol.i.5.col	186	3973	V /5	1714	1713	1189	1144	1216	1210
myciel3.col	11	20	V /5	5	5	5	5	5	5
myciel4.col	23	71	V /5	21	21	21	21	21	21
myciel6.col	95	755	V /5	233	231	215	212	194	193
myciel7.col	191	2360	V /5	723	717	643	634	574	574
queen5_5.col	25	320	V /5	46	46	46	46	46	46
queen6_6.col	36	580	V /5	91	91	91	88	91	91
queen7_7.col	49	952	V /5	148	147	145	141	148	148
queen8_8.col	64	1456	V /5	236	232	219	214	236	228
queen8_12.col	96	2736	V /5	458	453	400	391	458	458
queen9_9.col	81	2112	V /5	340	336	308	304	340	334
queen10_10.col	100	2940	V /5	485	479	419	415	468	466
queen11_11.col	121	3960	V /5	644	643	563	556	633	633
queen12_12.col	144	5192	V /5	866	853	725	717	833	833
queen13_13.col	169	6656	V /5	1097	1093	918	909	1067	1067
queen14_14.col	196	8372	V /5	1407	1385	1148	1131	1346	1345
queen15_15.col	225	10360	V /5	1721	1706	1402	1376	1676	1675
queen16_16.col	256	12640	V /5	2107	2075	1668	1652	2051	2048
school1.col	385	19095	10	6633	6553	4951	4886	6644	6644
school1_nsh.col	352	14612	10	5545	5450	3838	3780	5548	5548
zeroin.i.1.col	211	4100	10	1210	1198	1113	1095	924	923
zeroin.i.2.col	211	3541	10	1135	1126	975	959	803	800
zeroin.i.3.col	206	3540	10	1134	1126	981	964	800	799

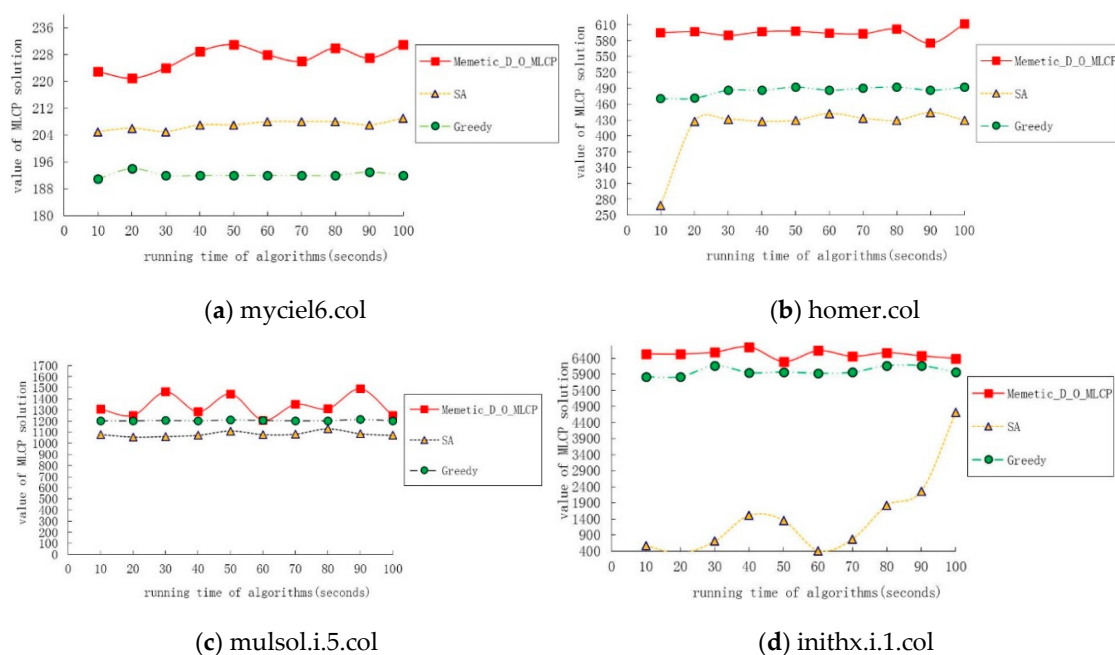


Figure 5. Running curves of the Memetic\_D\_O\_MLCP, SA and Greedy on benchmark instances.

The fourth group of tests compared the time each algorithm took to find the best results, each being run 20 times for 32 instances with the cut-off time of 30 min.

The results are summarized in Table 3. Compared with SA algorithm and Greedy algorithm, it took less time for Memetic\_D\_O\_MLCP to find the best results for the 11 instances (shown in italics). Accounting for 34% in the total, these 11 instances were: *fpsol2.i.2.col*, *huck.col*, *mulsol.i.3.col*, *mulsol.i.4.col*, *mulsol.i.5.col*, *myciel6.col*, *queen10\_10.col*, *queen11\_11.col*, *queen15\_15.col*, *inithx.i.3.col*, and *zeroin.i.2.col*. For six instances, namely  *david.col*, *DSJC125.9.col*, *games120.col*, *miles250.col*, *miles750.col*, and *jean.col*, which accounted for 19% in the total, the time spent by Memetic\_D\_O\_MLCP and Greedy algorithm showed little difference. Additionally, the former found better results than the latter. For the remaining 15 instances, although the time taken by Memetic\_D\_O\_MLCP was longer than that by Greedy algorithm, as it consumed more time for executing the operations of data splitting, searching, evolution and disturbance, the results found by the former were better than those by the latter. Of all 32 instances, comparing with Memetic\_D\_O\_MLCP, SA algorithm spent more time to find the best results; besides, the *Best* SA algorithm results were inferior.

The comparison between Memetic\_D\_O\_MLCP and artificial bee colony (ABC) algorithm [4] is summarized in Table 4. For each instance, the best values of *Best* are shown in italics. Of all 21 instances proposed in [4], except that the search results of instances *myciel3.col* and *myciel4.col* were equivalent to that of artificial bee colony algorithm, Memetic\_D\_O\_MLCP found better results (accounting for 90%) and improved the best-known result of instance *myciel5.col*.



**Table 3.** Running time of the Memetic\_D\_O\_MLCP, SA and Greedy on benchmark instances.

Inst	Memetic_D_O_MLCP				SA			Greedy		
	<i>m</i>	<i>Best</i>	<i>Avg</i>	<i>Time(min)</i>	<i>Best</i>	<i>Avg</i>	<i>Time(min)</i>	<i>Best</i>	<i>Avg</i>	<i>Time(min)</i>
anna.col	V /5	200	199	5.30	159	150	17.29	131	131	0.09
david.col	V /5	158	157	0.47	140	130	21.97	140	140	0.02
DSJC125.1.col	V /5	254	252	27.92	220	214	29.76	239	237	19.24
DSJC125.5.col	V /5	1086	1078	22.82	1026	1012	28.20	1075	1067	14.07
DSJC125.9.col	V /5	1797	1785	18.86	1757	1752	14.04	1782	1773	17.81
fpsol2.i.1.col	V /5	3073	2871	23.52	2984	2966	13.42	2510	2504	4.78
fpsol2.i.2.col	V /5	2274	1882	17.20	2250	2226	25.58	1707	1706	22.30
games120.col	V /5	288	280	1.05	216	205	29.74	284	284	0.05
huck.col	V /5	130	129	0.03	113	109	6.86	113	113	0.09
miles250.col	V /5	185	184	5.92	140	134	29.67	183	183	4.05
miles500.col	V /5	522	522	1.11	389	375	25.71	518	518	< 0.01
miles750.col	V /5	870	870	1.72	644	618	17.17	849	849	0.07
miles1000.col	V /5	1186	1178	18.53	938	892	24.83	1156	1156	0.22
miles1500.col	V /5	1619	1613	27.56	1453	1411	15.90	1485	1485	2.36
mulsol.i.1.col	V /5	1695	1689	20.94	1164	1089	28.06	1624	1624	0.29
mulsol.i.2.col	V /5	1685	1680	26.63	1157	1065	22.97	1202	1188	8.10
mulsol.i.3.col	V /5	1693	1687	22.76	1147	1112	23.56	1209	1183	25.14
mulsol.i.4.col	V /5	1704	1694	25.22	1139	1091	29.25	1218	1186	28.08
mulsol.i.5.col	V /5	1714	1705	23.77	1165	1093	20.23	1214	1207	28.68
jean.col	V /5	111	110	0.13	97	93	18.93	106	106	0.02
myciel6.col	V /5	232	231	20.01	213	211	19.19	194	192	26.07
myciel7.col	V /5	719	710	18.73	631	624	27.23	574	574	3.68
queen10_10.col	V /5	485	478	1.42	418	410	10.08	468	466	21.61
queen11_11.col	V /5	644	643	2.56	554	539	21.94	640	633	12.95
queen12_12.col	V /5	866	853	5.32	721	703	22.64	833	833	0.07
queen13_13.col	V /5	1097	1093	25.21	907	884	21.66	1067	1067	2.98
queen15_15.col	V /5	1697	1675	21.65	1377	1357	29.27	1676	1675	26.07
inithx.i.1.col	10	6622	5982	23.36	4774	4696	11.55	6169	6044	1.45
inithx.i.3.col	10	5362	4123	22.89	3616	3569	9.58	3151	3117	22.93
zeroin.i.1.col	10	1207	1189	12.76	1111	1083	19.77	924	923	5.78
zeroin.i.2.col	10	1131	1124	18.46	967	939	21.43	802	799	26.64
zeroin.i.3.col	10	1131	1125	28.96	959	937	28.32	800	798	10.53

**Table 4.** Comparison results on Memetic\_D\_O\_MLCP and ABC.

Inst	Memetic_D_O_MLCP		ABC
	<i>m</i>	<i>Best</i>	<i>Best</i>
DSJC125.1.col	V /5	255	209
DSJC125.5.col	V /5	1091	1005
DSJC125.9.col	V /5	1798	1746
fpsol2.i.1.col	V /5	3091	2956
fpsol2.i.2.col	V /5	2290	2231
fpsol2.i.3.col	V /5	2387	2207
inithx.i.1.col	10	6644	1295
inithx.i.2.col	10	5622	3574
inithx.i.3.col	10	5589	3548
myciel3.col	V /5	5	5
myciel4.col	V /5	21	21
myciel5.col	V /5	73	68
myciel6.col	V /5	233	207
myciel7.col	V /5	723	621
le450_5a.col	10	1824	1475
le450_5b.col	10	1820	1490
le450_5c.col	10	2985	2505
le450_5d.col	10	2943	2493
le450_15b.col	10	2395	2110
le450_15c.col	10	4530	4217
le450_15d.col	10	4586	4227

Furthermore, we compared the search results from Memetic\_D\_O\_MLCP, tabu search (Tabu) algorithm [5] and variable neighborhood search (VNS) algorithm [5]; the results are shown in Table 5

(the algorithms in the literature were run 20 times, each lasting 30 min for each benchmark instance). Memetic\_D\_O\_MLCP could find the best results of 26 instances (shown in italics), in which the best results of 11 instances equaled those found by Tabu algorithm. Hence, Memetic\_D\_O\_MLCP could improve the best-known results of the remaining 15 instances. Besides, of the 53 instances in Table 5, the best results of 22 instances found by Memetic\_D\_O\_MLCP were better than those by Tabu algorithm, and the best results of 42 instances found by Memetic\_D\_O\_MLCP were better than that of VNS algorithm.

**Table 5.** Comparison results on Memetic\_D\_O\_MLCP, Tabu and VNS.

<i>Inst</i>	Memetic_D_O_MLCP			Tabu		VNS	
	<i>m</i>	<i>Best</i>	<i>Avg</i>	<i>Best</i>	<i>Avg</i>	<i>Best</i>	<i>Avg</i>
anna.col	V /5	200	198	195	182	218	189
david.col	V /5	158	157	153	142	164	152
DSJC125.1.col	V /5	255	252	248	238	227	215
DSJC125.5.col	V /5	1091	1083	1078	1073	1047	1033
DSJC125.9.col	V /5	1798	1789	1794	1787	1793	1785
games120.col	V /5	288	281	282	269	192	181
homer.col	10	662	655	651	625	603	541
huck.col	V /5	130	129	130	126	123	110
inithx.i.1.col	10	6644	6153	7412	6272	6215	5838
inithx.i.2.col	10	5622	5104	5956	5831	4771	4478
inithx.i.3.col	10	5589	4756	5943	5818	4804	4464
jean.col	V /5	111	110	110	104	110	95
latin_square_10.col	10	85161	85072	76925	76925	77031	76956
le450_5a.col	10	1824	1801	1977	1923	1545	1520
le450_5b.col	10	1820	1801	1969	1923	1550	1522
le450_5c.col	10	2985	2951	3154	3124	2578	2553
le450_5d.col	10	2943	2913	3140	3108	2583	2546
le450_15b.col	10	2395	2355	2795	2719	2338	2268
le450_25b.col	10	2664	2606	2903	2863	2382	2337
le450_25d.col	10	4872	4807	5420	5376	4844	4747
miles250.col	V /5	185	184	183	172	134	126
miles500.col	V /5	522	522	502	483	402	367
miles750.col	V /5	870	870	836	833	708	648
miles1000.col	V /5	1183	1180	1114	1108	1035	963
miles1500.col	V /5	1645	1616	1517	1513	1565	1490
multsol.i.1.col	V /5	1697	1690	1649	1649	1313	1240
multsol.i.2.col	V /5	1685	1682	1685	1668	1319	1211
multsol.i.3.col	V /5	1695	1692	1695	1669	1260	1217
multsol.i.4.col	V /5	1704	1701	1704	1693	1276	1214
multsol.i.5.col	V /5	1714	1713	1697	1686	1296	1233
myciel3.col	V /5	5	5	5	5	7	7
myciel4.col	V /5	21	21	21	20	25	24
myciel6.col	V /5	233	231	231	223	247	237
myciel7.col	V /5	723	717	714	701	737	711
queen5_5.col	V /5	46	46	46	45	50	48
queen6_6.col	V /5	91	91	91	90	86	82
queen7_7.col	V /5	148	147	148	145	142	136
queen8_8.col	V /5	236	232	236	233	208	201
queen8_12.col	V /5	458	453	458	457	380	369
queen9_9.col	V /5	340	336	336	332	306	293
queen10_10.col	V /5	485	479	485	483	403	394
queen11_11.col	V /5	644	643	650	637	546	536
queen12_12.col	V /5	866	853	866	858	703	689
queen13_13.col	V /5	1097	1093	1106	1066	910	887
queen14_14.col	V /5	1407	1385	1407	1403	1127	1101
queen15_15.col	V /5	1721	1707	1722	1703	1388	1366
queen16_16.col	V /5	2107	2075	2136	2125	1682	1650
school1.col	10	6633	6553	6975	6752	5628	5398
school1_nsh.col	10	5545	5450	5721	5622	4169	4066
zeroin.i.1.col	10	1210	1198	1185	1166	1454	1358
zeroin.i.2.col	10	1135	1126	1105	1079	1294	1201
zeroin.i.3.col	10	1134	1126	1107	1082	1221	1158

## 7. Conclusions

In this paper, we propose a memetic algorithm (Memetic\_D\_O\_MLCP) to deal with the minimum load coloring problem. The algorithm employs an improved K-OPT local search procedure with a

combination of data splitting operation, disturbance operation and a population evolutionary operation to assure the quality of the search results and intensify the searching ability.

We assessed the performance of our algorithm on 59 well-known graphs from the benchmark DIMACS competitions. The algorithm could find the best results of 46 graphs. Compared with simulated annealing algorithm and greedy algorithm, which cover the best results for the tested instances, our algorithm was more competent.

In addition, we investigated the artificial bee colony algorithm, variable neighborhood search algorithm and tabu search algorithm proposed in the literature. We carried out comparative experiments between our algorithm and artificial bee colony algorithm using 21 benchmark graphs, and the experiments showed that the algorithm's best results of 19 benchmark graphs were better than those of artificial bee colony algorithm, and the best-known result of one benchmark graph was improved by our algorithm. More experiments were conducted to compare our algorithm with tabu search algorithm and variable neighborhood search algorithm, and proved that the best-known results of 15 benchmark graphs were improved by our algorithm.

Finally, we showed that the proposed Memetic\_D\_O\_MLCP approach significantly improved the classical heuristic search approach for the minimum load coloring problem.

**Author Contributions:** Writing and methodology, Z.Z.; Software, Z.Z.; Review and editing, Z.L.; Validation, X.Q.; and Supervision, W.W.

**Funding:** This research was supported by the Scientific Research Fund of Key Laboratory of Pattern Recognition and Intelligent Information Processing of Chengdu University (No. MSSB-2018-08), Chengdu Science and Technology Program (No. 2018-YF05-00731-SN), Sichuan Science and Technology Program (No. 2018GZ0247), and the Application Fundamental Foundation of Sichuan Provincial Science and Technology Department (No. 2018JY0320).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Ahuja, N.; Baltz, A.; Doerr, B.; Přívětivý, A.; Srivastav, A. On the minimum load coloring problem. *J. Discret. Algorithms* **2007**, *5*, 533–545. [[CrossRef](#)]
2. Baldine, I.; Rouskas, G.N. Reconfiguration and dynamic load balancing in broadcast WDM Networks. *Photonic Netw. Commun. J.* **1999**, *1*, 49–64. [[CrossRef](#)]
3. Rouskas, G.N.; Thaker, D. Multi-destination communication in broadcast WDM networks: A Survey. *Opt. Netw.* **2002**, *3*, 34–44.
4. Fei, T.; Bo, W.; Jin, W.; Liu, D. Artificial Bee Colony Algorithm for the Minimum Load Coloring Problem. *J. Comput. Theor. Nanosci.* **2013**, *10*, 1968–1971. [[CrossRef](#)]
5. Ye, A.; Zhang, Z.; Zhou, X.; Miao, F. Tabu Assisted Local Search for the Minimum Load Coloring Problem. *J. Comput. Theor. Nanosci.* **2014**, *11*, 2476–2480. [[CrossRef](#)]
6. Gutin, G.; Jones, M. Parameterized algorithms for load coloring problem. *Inf. Process. Lett.* **2014**, *114*, 446–449. [[CrossRef](#)]
7. Barbero, F.; Gutin, G.; Jones, M.; Sheng, B. Parameterized and Approximation Algorithms for the Load Coloring Problem. *Algorithmica* **2017**, *79*, 211–229. [[CrossRef](#)]
8. Hansen, P.; Mladenović, N.; Urošević, D. Variable neighborhood search for the maximum clique. *Discret. Appl. Math.* **2004**, *145*, 117–125. [[CrossRef](#)]
9. Dražić, Z.; Čangalović, M.; Kovačević-Vujčić, V. A metaheuristic approach to the dominating tree problem. *Optim. Lett.* **2017**, *11*, 1155–1167. [[CrossRef](#)]
10. Fadlaoui, K.; Galinier, P. A tabu search algorithm for the covering design problem. *J. Heuristics* **2011**, *17*, 659–674. [[CrossRef](#)]
11. Li, X.; Yue, C.; Aneja, Y.P.; Chen, S.; Cui, Y. An Iterated Tabu Search Metaheuristic for the Regenerator Location Problem. *Appl. Soft Comput.* **2018**, *70*, 182–194. [[CrossRef](#)]
12. Ho, S.C. An iterated tabu search heuristic for the Single Source Capacitated Facility Location Problem. *Appl. Soft Comput.* **2015**, *27*, 169–178. [[CrossRef](#)]

13. Palubeckis, G.; Ostreika, A.; Rubliauskas, D. Maximally diverse grouping: An iterated tabu search approach. *J. Oper. Res. Soc.* **2015**, *66*, 579–592. [[CrossRef](#)]
14. Tang, Z.; Feng, Q.; Zhong, P. Nonuniform Neighborhood Sampling based Simulated Annealing for the Directed Feedback Vertex Set Problem. *IEEE Access* **2017**, *5*, 12353–12363. [[CrossRef](#)]
15. Palubeckis, G. A variable neighborhood search and simulated annealing hybrid for the profile minimization problem. *Comput. Oper. Res.* **2017**, *87*, 83–97. [[CrossRef](#)]
16. Zhao, D.; Shu, Z. A Simulated Annealing Algorithm with Effective Local Search for Solving the Sum Coloring Problem. *J. Comput. Theor. Nanosci.* **2016**, *13*, 945–949. [[CrossRef](#)]
17. Li, X.; Li, S.-J.; Li, H. Simulated annealing with large-neighborhood search for two-echelon location routing problem. *Chin. J. Eng.* **2017**, *39*, 953–961.
18. Parekh, A.K. Analysis of a greedy heuristic for finding small dominating sets in graphs. *Inf. Process. Lett.* **1991**, *39*, 237–240. [[CrossRef](#)]
19. Katayama, K.; Hamamoto, A.; Narihisa, H. An effective local search for the maximum clique problem. *Inf. Process. Lett.* **2005**, *95*, 503–511. [[CrossRef](#)]
20. Battiti, R.; Protasi, M. Reactive local search for maximum clique. *Algorithmica* **2001**, *29*, 610–637. [[CrossRef](#)]
21. Xu, J.; Wu, C.C.; Yin, Y.; Lin, W.C. An iterated local search for the multi-objective permutation flowshop scheduling problem with sequence-dependent setup times. *Appl. Soft Comput.* **2017**, *52*, 39–47. [[CrossRef](#)]
22. Pullan, W. Phased local search for the maximum clique problem. *J. Comb. Optim.* **2006**, *12*, 303–323. [[CrossRef](#)]
23. Moscato, P.; Cotta, C. A gentle introduction to memetic algorithms. In *Handbook of Metaheuristics*; International series in operations research and management science; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2003; Volume 57, pp. 105–144, Chapter 5.
24. Jin, Y.; Hao, J.K.; Hamiez, J.P. A memetic algorithm for the Minimum Sum Coloring Problem. *Comput. Oper. Res.* **2014**, *43*, 318–327. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).