Performance Analysis of Evolutionary Algorithms for the Minimum Label Spanning Tree Problem

Xinsheng Lai, Yuren Zhou, Jun He, *Member, IEEE,* and Jun Zhang, *Senior Member, IEEE*

***Abstract*—A few experimental investigations have shown that evolutionary algorithms (EAs) are efficient for the minimum label spanning tree (MLST) problem. However, we know little about that in theory. In this paper, we theoretically analyze the performances of the (1+1) EA, a simple version of EA, and a simple multiobjective evolutionary algorithm called GSEMO on the MLST problem. We reveal that for the MLST$_b$ problem, the (1+1) EA and GSEMO achieve a $(b+1)/2$-approximation ratio in expected polynomial runtime with respect to n, the number of nodes, and k, the number of labels. We also find that GSEMO achieves a $(2\ln n+1)$-approximation ratio for the MLST problem in expected polynomial runtime with respect to n and k. At the same time, we show that the (1+1) EA and GSEMO outperform local search algorithms on three instances of the MLST problem. We also construct an instance on which GSEMO outperforms the (1+1) EA.**

***Index Terms*—Approximation ratio, evolutionary algorithm, minimum label spanning tree, multiobjective, runtime complexity.**

I. Introduction

The MINIMUM label spanning tree (MLST) problem is an issue arising from practice, which seeks a spanning tree with the minimum number of labels in a connected undirected graph with labeled edges. This problem has many applications in real-life. For example, in communication networks, a communication node may communicate with other nodes via different types of channels, such as optic fiber, coaxial cable, microwave, and telephone line [1]. Given a set of communication nodes, we want to find a spanning tree (a connected communication network) that uses the minimum possible number of types of channels, which can reduce the construction cost and the complexity of the network [2]. Another example is that the MLST problem has been used in data compression to increase compression rates [3], [4]. The MLST problem, proposed in [2], has been proved to be NP-hard.

For this problem, two heuristic algorithms were proposed in [2]. One is the edge replacement algorithm (ERA) and the other is the maximum vertex covering algorithm (MVCA). Their experiments showed that ERA is not stable and MVCA is very efficient.

The genetic algorithm, belonging to the larger class of EAs, is a general purpose optimization algorithm [5]–[7] with a strong globally searching capacity [8]. So, Xiong *et al.* [9] proposed a one-parameter genetic algorithm for the MLST problem. The experimental results on extensive instances randomly generated showed that the genetic algorithm outperforms MVCA. Nummela and Julstrom [10] also proposed an efficient genetic algorithm for solving the MLST problem.

Besides, many methods have been recently proposed for solving this NP-hard problem. Consoli and Moreno-Pérezb [11] proposed a hybrid local search combining variable neighborhood search and simulated annealing. Chwatal and Raidl [12] presented exact methods including branch-and-cut and branch-and-cut-and-price. Cerulli *et al.* [13] utilized several metaheuristic methods for this problem, such as simulated annealing, reactive tabu search, the pilot method, and variable neighborhood search. Consoli *et al.* [14] proposed a greedy randomized adaptive search procedure and a variable neighborhood search for solving the MLST problem.

Since both ERA and MVCA are two original heuristic algorithms for the MLST problem, the worst performance analysis of these two algorithms, especially MVCA, has become a hot research topic in recent years. Krumke and Wirth [15] proved that MVCA has a logarithmic performance guarantee of $2\ln n + 1$, where n is the number of nodes in the input graph, and presented an instance to show that ERA might perform as badly as possible. Wan *et al.* [16] further proved that MVCA has a better performance guarantee of $\ln(n-1)+1$. Xiong *et al.* [17] proved another bound on the worst performance of MVCA for MLST$_b$ problems, i.e., $H_b = \sum_{i=1}^{b} \frac{1}{i}$, where the subscript b denotes that each label appears at most b times, and is also called the maximum frequency of the labels.

The performance of MVCA on the MLST problem has been deeply investigated. However, there is still no theoretical work that focuses on the performance analysis of EAs for the MLST

Manuscript received April 12, 2013; revised August 22, 2013; accepted October 31, 2013. Date of publication November 20, 2013; date of current version November 26, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant 61170081, Grant 61165003, Grant 61300044, and Grant 61332002, in part by the EPSRC under Grant EP/I009809/1, in part by the National High-Technology Research and Development Program (863 Program) of China No. 2013AA01A212, and in part by the NSFC for Distinguished Young Scholars under Grant 61125205. *(Corresponding author: Y. Zhou)*.

X. Lai and Y. Zhou are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: yrzhou@scut.edu.cn).

X. Lai is also with the School of Mathematics and Computer Science, Shangrao Normal University, Shangrao 334001, China.

J. He is with the Department of Computer Science, Aberystwyth University, Aberystwyth, Wales SY23 3DB, U.K.

J. Zhang is with the Department of Computer Science, Sun Yat-Sen University, Guangzhou 510275, China, the Key Laboratory of Digital Life, Ministry of Education, China, and also with the Key Laboratory of Software Technology, Education Department of Guangdong Province, Guangdong, China.

Digital Object Identifier 10.1109/TEVC.2013.2291790

problem, though a few experimental investigations have shown that EAs are efficient for this problem.

Recently, the theoretical analysis of EAs' performance on combinatorial optimization problems has received much attention. During the past few years, theoretical investigations about EAs focused on the runtime for finding globally optimal solutions of combinatorial optimization problems or their variants. These problems include plateaus of constant fitness [18], linear function problems [19]–[21], minimum cut problems [22], satisfiability problems [23], minimum spanning tree problems [24], Eulerian cycle problems [25], Euclidean traveling salesperson problems [26], the maximum matching problem [27], and so on.

Nevertheless, many combinatorial optimization problems, including the MLST problem, are NP-hard, and it is commonly believed that there is no polynomial time algorithm for them. Therefore, we usually only ask satisfying solutions to such NP-hard problems in practice. Thus, we are interested in whether an approximation solution with a given satisfying quality can be efficiently obtained. In fact, the approximation performance analysis of randomized heuristics, including EAs, on NP-hard problems has been receiving more and more attention.

Oliveto *et al*. [28] found that for minimum vertex cover problems the (1+1) EA may find arbitrarily bad approximation solutions on some instances, but can efficiently find the minimum cover of them by using a restart strategy. Friedrich *et al.* [29] proved that the (1+1) EA may find almost arbitrarily bad approximation solutions for minimum vertex cover problems and minimum set cover problems as well. Witt [30] proved that in the worst case the (1+1) EA and the randomized local search algorithm need an expected runtime $O(n^2)$ to produce a $\frac{4}{3}$-approximation solution to the partition problem.

On the approximation performance of multiobjective EAs, Friedrich *et al.* [29] revealed that the multiobjective EA efficiently finds an $\ln n$-approximation solution to the minimum set cover problem. Neumann and Reichel [31] found that multiobjective EAs can find a k-approximation solution for the minimum multicuts problem in expected polynomial runtime. Recently, Yu *et al*. [32] studied the approximation performance of SEIP, a simple evolutionary algorithm with isolated population, on set cover problems. They found that SEIP can efficiently obtain an H_n-approximation solution for unbounded set cover problems, and an $(H_k - \frac{k-1}{8k^9})$-approximation solution for k-set cover problems as well.

Though the MLST problem is NP-hard, and the minimum spanning tree problem is in the complexity class P, both belong to the problems of finding spanning trees with some measure to be optimized. The measure of the MLST problem is the number of labels used in the tree, whereas the measure of the minimum spanning tree problem is the total weights of the edges in the tree. The performance of EAs on the minimum spanning tree problem has been recently investigated. Neumann and Wegener [24] showed that the (1+1) EA solves the minimum spanning tree problem in expected polynomial runtime. They also showed that the minimum spanning tree problem can be solved more efficiently via multiobjective optimization [33]. Another spanning tree related problem is the multiobjective minimum spanning tree problem, which is NP-hard. Recently, the performance of EAs on this NP-hard problem has also been theoretically investigated [34], [35]. However, there is no theoretical work focusing on EAs' performance for the MLST problem.

In this paper, we concentrate on the performance analysis of the (1+1) EA and GSEMO (a simple multiobjective evolutionary algorithm with bit-wise mutation) for the MLST problem. We analyze the approximation performances of the (1+1) EA and GSEMO on the MLST problem. For the MLST$_b$ problem, we prove that the (1+1) EA and GSEMO are $(b+1)/2$-approximation algorithms. We also reveal that GSEMO can efficiently achieve a $(2\ln n+1)$-approximation ratio for the MLST problem. Though the MLST problem is NP-hard, we show that on three instances the (1+1) EA and GSEMO efficiently find their global optima, while local search algorithms may be trapped in local optima. Meanwhile, we construct an additional instance where GSEMO outperforms the (1+1) EA.

The rest of this paper is organized as follows. Section II describes the MLST problem and the algorithms considered in this paper. Section III analyzes the approximation performances of the (1+1) EA and GSEMO on the MLST problem, while Section IV investigates the performances of the (1+1) EA and GSEMO on four instances. Finally, Section V concludes this paper and gives possible directions for further research.

II. MLST Problem and Algorithms

First of all, we give the concepts of spanning subgraph, the MLST and MLST$_b$ problems.

*Definition 1 (*Spanning subgraph): Let $G = (V, E)$ and $H = (V', E')$ be two graphs, where V (V') is the set of nodes of G (H), and E (E') is the set of edges of G (H). If $V' = V$ and $E' \subseteq E$, then H is a spanning subgraph of G.

*Definition 2 (*MLST problem): Let $G = (V, E, L)$ be a connected undirected graph, where V, E, and $L = \{1, 2, \ldots, k\}$ are respectively the sets of n nodes, m edges, and k labels. Each edge associates with a label by a surjective function $l : E \rightarrow L$. The MLST problem is to seek a spanning tree with the minimum number of labels in the input graph G.

From the definition of the MLST problem, it is easy to see that each edge $e \in E$ has a unique label $l(e) \in L$, and each label in L has at least one edge associating with it.

*Definition 3 (*MLST$_b$ problem): For an MLST problem, if each label appears at most b times, then it is an MLST$_b$ problem.

Clearly, the MLST$_b$ problem is a special case of the MLST problem.

Our goal in this paper is to seek a connected spanning subgraph with the minimum number of labels rather than a spanning tree with the minimum number of labels, since any spanning tree contained in such a spanning subgraph is an MLST. This is an alternative formulation of the MLST problem that has been adopted in papers [9] and [10].

Since each edge has exactly one label, if a label is selected, then all edges with such a label are selected. Therefore, we

Algorithm 1 The (1+1) EA for the MLST problem

```
01: Begin
02:     Initialize a solution $X \in \{0, 1\}^k$ uniformly at random;
03:     While termination criterion is not fulfilled
04:         Obtain an offspring $Y$ by flipping each bit in $X$ with
              probability $\frac{1}{k}$;
05:         If $fit(Y) < fit(X)$ then $X := Y$;
06:     End while
07: End
```

aim to find a set of selected labels such that the edges with these labels construct a spanning subgraph of the input graph and the number of selected labels is minimized. Thus, a set of selected labels is a solution.

Further, we encode a solution as a bit string $X = (x_1, \ldots, x_k)$, which has been used in [9], where $k = |L|$ and bit $x_i \in \{0, 1\}$ corresponding to label i. If $x_i = 1 (i = 1, 2, \ldots, k)$, then label i is selected, otherwise it is not. Thus, a bit string X represents a label subset and $|X|$ represents the number of labels contained in X.

We consider the spanning subgraph $H(X)$ of G, where $H(X)$ is a spanning subgraph restricted to edges with labels that the corresponding bits in X are set to 1. We call a solution X such that $H(X)$ is a connected spanning subgraph a feasible solution. A feasible solution with the minimum number of labels is a globally optimal solution.

For solving the MLST problem, the (1+1) EA uses a fitness function. The fitness function is defined as

$$fit(X) = (c(H(X)) - 1) * k^2 + |X| \tag{1}$$

where $c(H(X))$ is the number of connected components in $H(X)$, k is the total number of labels in L, and $|X| = \sum_{i=1}^{k} x_i$, i.e., the number of labels contained in X.

The fitness function should be minimized. The first target is to make sure that $H(X)$ is a connected spanning subgraph, and the second target is to make sure that the number of labels in the connected spanning subgraph is minimized.

Recall that a feasible solution X satisfies that $H(X)$ is a connected spanning subgraph, i.e., $c(H(X)) = 1$. Therefore, the fitness value of a feasible solution equals to the number of labels contained in it.

We also define the fitness vector for GSEMO as a vector $(c(H(X)), |X|)$, where $c(H(X))$ and $|X|$ are simultaneously minimized by GSEMO.

The following algorithms are those considered in this paper.

The (1+1) EA as shown in Algorithm 1 starts with an arbitrary solution and repeatedly uses mutation operator to generate an offspring solution from the current one. If the offspring solution is strictly better than the current one, then the (1+1) EA uses it to replace the current solution.

Another algorithm for the MLST problem is the local search algorithm with the 2-switch neighborhood (LS2N), which is proposed by Brüggemanna *et al.* [36]. We now describe some concepts about it.

Algorithm 2 GSEMO for the MLST problem

```
01: Begin
02:     Initialize a solution $X \in \{0, 1\}^k$ uniformly at random;
03:     $P \leftarrow \{X\}$;
04:     While termination criterion is not fulfilled
05:         Choose a solution $X$ from $P$ uniformly at random;
06:         Obtain an offspring $Y$ by flipping each bit in $X$ with
              probability $\frac{1}{k}$;
07:         If $Y$ is not dominated by $\forall X \in P$ then
08:             $Q := \{X | X \in P$, and $Y$ dominates $X\}$;
09:             $P \leftarrow P \cup \{Y\} \setminus Q$;
10:         End if
11:     End while
12: End
```

We use $X_1 - X_2$ to denote the set of labels that are contained in X_1 but not in X_2, where X_1 and X_2 are two solutions. For example, if $X_1 = \{1, 2, 3, 4, 5\}$ and $X_2 = \{1, 2, 3, 6\}$, then we have $X_1 - X_2 = \{4, 5\}$ and $X_2 - X_1 = \{6\}$. Thus, we have $|X_1 - X_2| = 2$ and $|X_2 - X_1| = 1$.

Definition 4 ([36]*h*-switch neighborhood): Let $h \geq 1$ be an integer, and let X_1 and X_2 be two feasible solutions for some instance of the MLST problem. We say that X_2 is in h-switch neighborhood of X_1, denoted by $X_2 \in h\text{-}SWITCH(X_1)$, if and only if

$$|X_1 - X_2| \leq h \quad and \quad |X_2 - X_1| \leq h. \tag{2}$$

In other words, $X_2 \in h\text{-}SWITCH(X_1)$ means that X_2 can be derived from X_1 by first removing at most h labels from X_1 and then adding at most h labels to it.

LS2N: in Algorithm 1, if the initial solution X is an arbitrary feasible solution, and the offspring Y is a feasible solution selected from the 2-switch neighborhood of X, then it is LS2N [36].

GSEMO has been investigated on covering problems [29], minimum spanning tree problems [33]–[35], and also pseudo-Boolean functions [37], [38]. GSEMO for the MLST problem is described in Algorithm 2.

In Algorithm 2, P is a population used to preserve those solutions that cannot be dominated by any other from the population. The concept of dominance is defined as follows.

Suppose that the fitness vectors of solutions X and Y are $(c(H(X)), |X|)$ and $(c(H(Y)), |Y|)$, respectively. We say that X dominates Y, if one of the following two conditions is satisfied:
(1) $c(H(X)) < c(H(Y))$ and $|X| \leq |Y|$;
(2) $c(H(X)) \leq c(H(Y))$ and $|X| < |Y|$.

For the sake of completeness, we describe another two algorithms in the following, which are greedy algorithms.

The first one is the modified MVCA. It starts with a solution containing no labels and each time selects a label such that when this label is chosen the decrease in the number of connected components is the largest.

The second is called in this paper the modified MVCA with contraction, which has been investigated in [15].

In Algorithm 3, if we contract each connected component in H to a supernode after step 3, then we obtain the modified MVCA with contraction.

Algorithm 3 The modified MVCA [17]

```
Input: A given connected undirected graph $G = (V, E, L)$,
       $L = \{1, \ldots, k\}$.
01: Let $C$ be the set of used labels, $C := \emptyset$;
02: Repeat
03:     Let $H$ be the spanning subgraph of $G$ restricted to
        edges with labels from $C$;
04:     For all $i \in L \setminus C$ do
05:         Determine the number of connected components
            when inserting all edges labeled by $i$ in $H$;
06:     End for
07:     Choose label $i$ with the smallest resulting number of
        connected components: $C := C \cup \{i\}$;
08: Until $H$ is connected.
Output: $H$
```

III. Approximation Performances of the (1+1) EA and GSEMO on the MLST Problem

The following is the concept of approximation ratio (solution). Given a minimization problem R and an algorithm A, if for an instance I of R, the value of the best solution obtained in polynomial runtime by A is A(I), and $\max_{I \in R} \frac{\mathrm{A}(I)}{\mathrm{OPT}(I)} = r$, where OPT$(I)$ is the value of the optimal solution of I, then we say that A achieves an r-approximation ratio (solution) for R.

Although the MLST problem is NP-hard, we reveal that the (1+1) EA and GSEMO guarantee achievement of an approximation ratio for the MLST$_b$ problem in expected polynomial runtime with respect to n the number of nodes and k the number of labels, and that GSEMO guarantees the obtaining of an approximation ratio for the MLST problem in expected polynomial runtime with respect to n and k.

A. Approximation Guarantees of the (1+1) EA and GSEMO on the MLST$_b$ Problem

In this subsection, let $G = (V, E, L)$ be an arbitrary instance of the MLST$_b$ problem, where $|V| = n$, $|L| = k$, and $b \geq 2$, and let OPT(G) denote the number of labels used in the global optimum of G.

We show that the (1+1) EA and GSEMO guarantee achievement of a $(b+1)/2$-approximation ratio for G in expected polynomial runtime with respect to n and k in two steps. Since G is an arbitrary instance of the MLST$_b$ problem, we reveal that the (1+1) EA and GSEMO guarantee achievement of a $(b+1)/2$-approximation ratio for the MLST$_b$ problem in expected runtime polynomial in n and k. First, we prove that the (1+1) EA and GSEMO starting with any initial solution find a feasible solution for G in expected runtime polynomial in n and k, then prove that starting with any feasible solution the (1+1) EA and GSEMO find a $(b+1)/2$-approximation solution for G in expected polynomial runtime with respect to k by simulating the result proved in [36].

We now prove that starting with any initial solution, the (1+1) EA can efficiently find a feasible solution of G.

Lemma 1: The (1+1) EA starting with any initial solution finds a feasible solution for G in expected runtime $O(nk)$.

Proof: According to the fitness function (1), during the optimization process of the (1+1) EA, the number of connected components will never be increased.

Let X be the current solution of G. If X is not a feasible solution, then the number of connected components of the spanning subgraph $H(X)$ is greater than one. Note that G is connected. There must exist a label such that when it is added to X the number of connected components will be decreased by at least one. The probability of adding this label to X is $\frac{1}{k}(1-\frac{1}{k})^{k-1} \geq \frac{1}{ek}$, which implies that in expected runtime $O(k)$ the number of connected components will be decreased by at least one.

Since there are at most n connected components, a feasible solution of G will be found by the (1+1) EA starting with any initial solution in expected runtime $O(nk)$. ■

Brüggemann *et al.* [36] have proved that LS2N is a $(b+1)/2$-approximation algorithm for the MLST$_b$ problem, so we have the following lemma.

Lemma 2: LS2N can find a feasible solution for G with at most OPT$(G) \cdot (b+1)/2$ labels.

We partition all feasible solutions of G into two disjoint sets. One is $S_1 = \{X | X \in \{0,1\}^k$, X is a feasible solution of G, $|X| \leq \mathrm{OPT}(G) \cdot (b+1)/2\}$, and the other is $S_2 = \{X | X \in \{0,1\}^k$, X is a feasible solution of G, $|X| > \mathrm{OPT}(G) \cdot (b+1)/2\}$.

From Lemma 2, we derive a property regarding the 2-switch neighborhood.

Corollary 1: If X is a feasible solution of G, and $X \in S_2$, then there must exist a feasible solution $X' \in$ 2-*SWITCH*(X) whose fitness value is one or two less than that of X.

Next, we will show that starting with an arbitrary feasible solution, the (1+1) EA can efficiently find a $(b+1)/2$-approximation solution of G.

Lemma 3: For G, the (1+1) EA starting with an arbitrary feasible solution finds a $(b+1)/2$-approximation solution in expected runtime $O(k^4)$.

Proof: Let X be the current feasible solution of G. By Corollary 1, if $X \in S_2$, then there must exist a feasible solution $X' \in$ 2-*SWITCH*(X) whose fitness value is 1 or 2 less than that of X. So, replacing X with X' decreases the fitness value by at least 1. Since a feasible solution belonging to S_2 has at most k labels, then after at most $k - \mathrm{OPT}(G) \cdot (b+1)/2$ such replacing steps a feasible solution belonging to S_1 will be found.

Now, we calculate the expected runtime for the (1+1) EA to find X'. Since $X' \in$ 2-*SWITCH*(X) and $|X'| < |X|$, there exist three cases. The first is that X' is obtained by removing one exact label from X. The second is that X' is obtained by removing two exact labels from X. The third is that X' is obtained by removing two exact labels from X and adding one exact label to it.

Obviously, the worst case is the third one, since in this case three bits of X must be simultaneously flipped by the (1+1) EA. In this case, the probability that the (1+1) EA finds X' is $\frac{1}{k^3}(1-\frac{1}{k})^{k-3} \geq \frac{1}{ek^3}$. So, the expected runtime for the (1+1) EA to find a feasible solution $X' \in$ 2-*SWITCH*(X) is $O(k^3)$, which means that the expected runtime for the (1+1) EA to reduce the fitness value by at least one is $O(k^3)$.

Therefore, the expected runtime for the (1+1) EA starting with an arbitrary feasible solution to find a $(b+1)/2$-approximation solution for G is $O((k - \mathrm{OPT}(G) \cdot (b+1)/2)k^3) = O(k^4)$, as $\mathrm{OPT}(G) \cdot \frac{b+1}{2} \leq k$. ■

Combining Lemmas 1 and 3, and noting that G is an arbitrary instance of the MLST_b problem, we obtain the following theorem.

Theorem 1: The (1+1) EA starting with any initial solution finds a $(b+1)/2$-approximation solution for the MLST_b problem in expected runtime $O(nk + k^4)$.

As we will see below, GSEMO can also efficiently obtain such an approximation solution for the MLST_b problem.

Theorem 2: GSEMO starting with any initial solution finds a $(b+1)/2$-approximation solution for the MLST_b problem in expected runtime $O(nk^2 + k^5)$.

Proof: For GSEMO, the fitness vector of any solution X is $(c(H(X)), |X|)$. For a given value d of $|X|$, there is at most one solution in the population whose fitness vector is $(*, d)$, so the population size is $O(k)$ as $|X|$ takes value from $\{0, 1, \ldots, k\}$, where k is the number of labels contained in the label set.

Consider the arbitrary instance G, and let X be the solution among the population such that $c(H(X))$ is the minimum. If $c(H(X)) > 1$, then there exists a label l such that when it is added the number of connected components will be reduced by at least one, as G is connected. The probability of selecting X from the population is $\Omega(\frac{1}{k})$, as the population size is $O(k)$, and the probability of flipping only the bit corresponding to label l is $\frac{1}{k}(1 - \frac{1}{k})^{k-1} = \Omega(\frac{1}{k})$, so a solution X' such that $c(H(X'))$ is at least one less than $c(H(X))$ will be included in the population in expected runtime $O(k^2)$.

Since there are at most n connected components in the spanning subgraph induced by any solution, a feasible solution will be included in the population in expected runtime $O(nk^2)$.

At least one feasible solution is now included in the population. Let X be the feasible solution which has the minimum labels among all feasible solutions in the population. If the number of labels contained in X is greater than $\frac{b+1}{2} \cdot \mathrm{OPT}(G)$, then according to Corollary 1 there exists a feasible solution $X' \in$ 2-*SWITCH*(X) such that $|X'|$ is at least one less than $|X|$. According to the proof of Lemma 3, the expected runtime to generate such a solution X' from X is $O(k^3)$. Since the expected runtime to select solution X from the population is $O(k)$, such a solution X' will be included in the population in expected runtime $O(k^4)$.

Therefore, a $(b+1)/2$-approximation solution of G will be included in the population in expected runtime $O((k - \frac{b+1}{2} \cdot \mathrm{OPT}(G))k^4) = O(k^5)$ once a feasible solution is found.

Hence, GSEMO starting with any initial solution will find a $(b+1)/2$-approximation solution for G in expected runtime $O(nk^2 + k^5)$. Noting that G is an arbitrary instance of the MLST_b problem, we obtain the theorem. ■

B. Approximation Guarantee of GSEMO on the MLST Problem

In this subsection, let $G' = (V', E', L')$ be an arbitrary instance of the MLST problem, where $|V'| = n$ and $|L'| = k$, and let $\mathrm{OPT}(G')$ be the number of labels used in the minimum label spanning tree T^* of G'.

Now we analyze the approximation guarantee of GSEMO on the MLST problem by simulating the modified MVCA with contraction. Similar to Lemma 2 in [15], we have the following lemma.

Lemma 4: For G', if $n \geq 2$, then there exists a label such that the number of connected components of the spanning subgraph restricted to edges with this label is not more than $\lfloor n(1 - \frac{1}{2\mathrm{OPT}(G')})\rfloor$.

Proof: Note that G' is a connected undirected graph. Since the minimum label spanning tree T^* has exactly $n-1$ edges, there must exist a label, say j, such that the number of edges in T^* labeled by j is at least $\lceil \frac{n-1}{\mathrm{OPT}(G')} \rceil$. Hence, the number of connected components of the spanning subgraph, restricted to edges with label j, is not more than $n - \lceil \frac{n-1}{\mathrm{OPT}(G')} \rceil = \lfloor n(1 - \frac{1}{\mathrm{OPT}(G')}) + \frac{1}{\mathrm{OPT}(G')} \rfloor$. When $n \geq 2$, we have $\lfloor n(1 - \frac{1}{\mathrm{OPT}(G')}) + \frac{1}{\mathrm{OPT}(G')} \rfloor \leq \lfloor n(1 - \frac{1}{2\mathrm{OPT}(G')})\rfloor$. ■

Further, for a spanning subgraph $H(X)$ of G', we have the following corollary.

Corollary 2: Let s be the number of connected components of $H(X)$. If $s \geq 2$, then there is a label such that when it is added to X the number of connected components will be reduced to not more than $\lfloor s(1 - \frac{1}{2\mathrm{OPT}(G')})\rfloor$.

Proof: Contracting each connected component of $H(X)$ to a supernode, then G' is converted to G'' with s nodes. Suppose that the number of labels used in the minimum label spanning tree of G'' is $\mathrm{OPT}(G'')$. According to Lemma 4, there is a label in G'' such that the number of connected components of the spanning subgraph, restricted to edges with this label, is not more than $\lfloor s(1 - \frac{1}{2\mathrm{OPT}(G'')})\rfloor$. Noting that the number of labels used in the minimum label spanning tree of G' is $\mathrm{OPT}(G')$, it is clear that $\mathrm{OPT}(G'') \leq \mathrm{OPT}(G')$. Thus, $\lfloor s(1 - \frac{1}{2\mathrm{OPT}(G'')})\rfloor < \lfloor s(1 - \frac{1}{2\mathrm{OPT}(G')})\rfloor$. In other words, there is a label such that when it is added to X the number of connected components of $H(X)$ will be reduced to not more than $\lfloor s(1 - \frac{1}{2\mathrm{OPT}(G')})\rfloor$. ■

Based on Corollary 2, we prove that GSEMO guarantees to find a $(2\ln n + 1)$-approximate solution for the MLST problem in expected polynomial runtime with respect to n and k.

Theorem 3: GSEMO starting with any initial solution finds a $(2\ln n + 1)$-approximation solution for the MLST problem in expected runtime $O(k^2 \ln k + k^3 \ln n)$.

Proof: Consider the arbitrary instance G' of the MLST problem.

We first reveal that GSEMO starting with any initial solution will find the all-zeros bit string for G' in expected runtime $O(k^2 \ln k)$, then reveal that GSEMO finds a $(2\ln n + 1)$-approximation solution for G' in expected runtime $O(k^3 \ln n)$ after the all-zeros bit string being included in the population. Combining them, and noting that G' is an arbitrary instance of the MLST problem, we obtain the theorem.

We now investigate the expected runtime that GSEMO starting with any initial solution finds the all-zeros bit string with Pareto optimal fitness vector $(n, 0)$. Once it is found, it can never be removed from the population. If it is not included in the population, then GSEMO can choose a solution X from P which contains the minimum number of labels among all solutions in the population with probability $\Omega(\frac{1}{k})$,

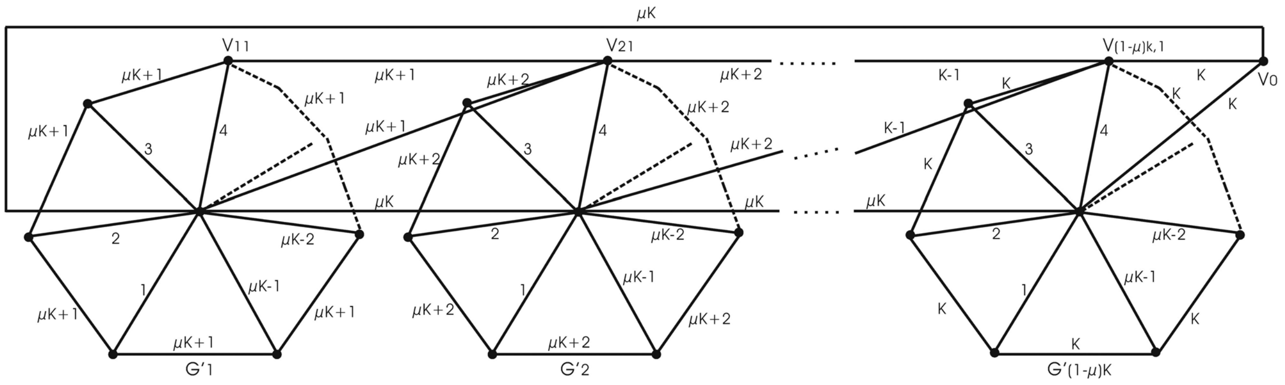


Fig. 1. Instance G_1.

as the population size is $O(k)$. The event of flipping one of $|X|$ bits whose value is 1 will decrease the number of labels, and the probability of this event is $\binom{|X|}{1}\frac{1}{k}(1-\frac{1}{k})^{k-1} \geq \frac{|X|}{ek}$. So, the expected runtime that GSEMO includes a solution which contains $|X|-1$ labels is $O(\frac{k^2}{|X|})$. Following this way, the all-zeros bit string will be included in the population in expected runtime $O(\sum_{i=|X|}^{1}\frac{k^2}{i}) = O(k^2 \ln |X|) = O(k^2 \ln k)$.

Now, the all-zeros bit string with fitness vector $(n, 0)$ is included in the population. It is easy to see that $n \geq 2$, otherwise G' is trivial. According to Corollary 2, there is a label such that when it is added to the all-zeros bit string the number of connected components will be reduced to not more than $n(1 - \frac{1}{2\mathrm{OPT}(G')})$. The probability of choosing this label is $\frac{1}{k}(1-\frac{1}{k})^{k-1} = \Omega(\frac{1}{k})$. Since the population size is $O(k)$, the probability that GSEMO selects the all-zeros bit string from P is $\Omega(\frac{1}{k})$. So a solution X^1 with fitness vector $(c_1, 1)$, where $c_1 \leq \lfloor n(1-\frac{1}{2\mathrm{OPT}(G')})\rfloor \leq n(1-\frac{1}{2\mathrm{OPT}(G')})$ can be included in the population in expected runtime $O(k^2)$.

If $c_1 \geq 2$, then there is still a label such that when it is added to X^1 the number of connected components will be reduced to not more than $n(1-\frac{1}{2\mathrm{OPT}(G')})^2$. So a solution X^2 with fitness vector $(c_2, 2)$, where $c_2 \leq n(1-\frac{1}{2\mathrm{OPT}(G')})^2$ can be included in the population in expected runtime $O(k^2)$ after X^1 being included in the population.

Similarly, suppose that solution X^{h-1} with fitness vector $(c_{h-1}, h-1)$, where $c_{h-1} \leq n(1-\frac{1}{2\mathrm{OPT}(G')})^{h-1}$, has now been included in the population. If $c_{h-1} \geq 2$, then a solution X^h with fitness vector (c_h, h), where $c_h \leq n(1-\frac{1}{2\mathrm{OPT}(G')})^h$, will be included in the population in expected runtime $O(k^2)$ after X^{h-1} being included in the population.

Note that when $h = 2 \cdot \mathrm{OPT}(G') \cdot \ln n$, we have $n(1-\frac{1}{2\mathrm{OPT}(G')})^h \leq 1$. So, a connected spanning subgraph with at most $\lceil 2 \cdot \mathrm{OPT}(G') \cdot \ln n \rceil \leq (2\ln n + 1) \cdot \mathrm{OPT}(G')$ labels will be finally included in the population in expected runtime $O((2\ln n+1)\cdot \mathrm{OPT}(G')\cdot k^2) = O(k^3 \ln n)$ after the all-zeros bit string being included in the population. ■

Table I summarizes the approximation performances of the (1+1) EA and GSEMO for the minimum label spanning tree problem. For the MLST$_b$ problem, the (1+1) EA and GSEMO can efficiently achieve a $(b+1)/2$-approximation ratio. However, the order of the upper bound on the expected runtime of GSEMO is higher than that of the (1+1) EA. This is because that GSEMO has to select a promising solution to mutate from a population of size $O(k)$. For the MLST problem, GSEMO efficiently achieves a $(2\ln n + 1)$-approximation ratio, but the approximation performance of the (1+1) EA is unknown.

TABLE I
APPROXIMATION PERFORMANCES OF THE (1+1) EA AND GSEMO. 'r' AND '—' REFER TO THE APPROXIMATION RATIO AND UNKNOWN, RESPECTIVELY

	The (1+1) EA		GSEMO	
	r	Runtime	r	Runtime
MLST$_b$	$\frac{b+1}{2}$	$O(nk+k^4)$	$\frac{b+1}{2}$	$O(nk^2+k^5)$
MLST	—	—	$2\ln n+1$	$O(k^2\ln k + k^3 \ln n)$

IV. PERFORMANCES OF THE (1+1) EA AND GSEMO ON FOUR INSTANCES

In this section, we first present an instance where GSEMO outperforms the (1+1) EA, then we show that the (1+1) EA and GSEMO outperform local search algorithms on three instances of the MLST problem.

A. Instance Where GSEMO Outperforms the (1+1) EA

In this subsection, we construct an instance $G_1 = (V_1, E_1, L_1)$ on which GSEMO is superior to the (1+1) EA, where $L_1 = \{1, \ldots, k\}$.

Given $\mu (0 < \mu < \frac{1}{2})$, we construct instance G_1 as follows. For simplicity, we assume that μk is an integer, thus $(1-\mu)k$ is also an integer. First, we construct $(1-\mu)k$ subgraphs $G'_1, \ldots, G'_{(1-\mu)k}$. $G'_i (1 \leq i \leq (1-\mu)k)$ contains a $(\mu k - 1)$-sided regular polygon whose edges are all labeled by the same label $\mu k + i$ and an inner node in the center of this regular polygon. Since the number of sides of a regular polygon is at least 3, $\mu k - 1 \geq 3$, i.e., $\mu k \geq 4$. From the inner node of G'_i, $(\mu k - 1)$ edges labeled by from 1 to $\mu k - 1$ connect to the $\mu k - 1$ outer nodes $v_{i1}, v_{i2}, \ldots, v_{i,\mu k-1}$. Then three edges are connected from G'_i $(1 \leq i \leq (1-\mu)k - 1)$ to G'_{i+1}: the first one labeled by $\mu k + i$ is from the inner node of G'_i to outer node $v_{i+1,1}$ of G'_{i+1}, the second one labeled by $\mu k + i$ is from outer node v_{i1} of G'_i to outer node $v_{i+1,1}$ of G'_{i+1}, and the third one labeled by μk is from the inner node of G'_i to the inner node of G'_{i+1}. Finally, v_0 is connected to the inner node and $v_{(1-\mu)k,1}$ of $G'_{(1-\mu)k}$ with two edges labeled by k, and v_0 is also connected to the inner node of G_1' with an edge labeled by μk. Fig. 1 shows the instance G_1.

When $0 < \mu < 1/2$, $X_1^* = (\overbrace{1, \dots, 1}^{\mu k}, \overbrace{0, \dots, 0}^{(1-\mu)k})$ is the global optimum of G_1. $X_1^l = (\overbrace{0, \dots, 0}^{\mu k}, \overbrace{1, \dots, 1}^{(1-\mu)k})$ is a local optimum for LS2N. This is because that X_1^* is the only feasible solution that contains fewer labels than X_1^l. However, $|X_1^l - X_1^*| = (1-\mu)k$ and $|X_1^* - X_1^l| = \mu k$, which implies that $X_1^* \notin$ 2-*SWITCH*(X_1^l) as $\mu k \geq 4$ and $(1-\mu)k > \mu k$.

For instance G_1, the expected runtime for the (1+1) EA to jump out of X_1^l is exponential.

Theorem 4: For instance G_1, starting with X_1^l, the expected runtime for the (1+1) EA to find the global optimum is $\Omega(k^{\mu k})$.

Proof: For instance G_1, when the current solution is X_1^l, the (1+1) EA only accepts the event that adds all μk labels from $\{1, \dots, \mu k\}$ and simultaneously removes more than μk labels from $\{\mu k + 1, \dots, k\}$. So, the probability of escaping from the local optimum is

$$\begin{aligned}\sum_{i=1}^{k-2\mu k} \binom{k-\mu k}{\mu k+i}\left(\tfrac{1}{k}\right)^{2\mu k+i}\left(1-\tfrac{1}{k}\right)^{k-2\mu k-i} &= \left(\tfrac{1}{k}\right)^{\mu k}\sum_{i=1}^{k-2\mu k} \binom{k-\mu k}{\mu k+i}\left(\tfrac{1}{k}\right)^{\mu k+i}\left(1-\tfrac{1}{k}\right)^{k-2\mu k-i}\\ &< \left(\tfrac{1}{k}\right)^{\mu k}.\end{aligned}$$

This is because

$$\begin{aligned}\sum_{i=1}^{k-2\mu k} \binom{k-\mu k}{\mu k+i}\left(\tfrac{1}{k}\right)^{\mu k+i}\left(1-\tfrac{1}{k}\right)^{k-2\mu k-i} &< \sum_{i=1}^{k-2\mu k} \binom{k-\mu k}{\mu k+i}\left(\tfrac{1}{k}\right)^{\mu k+i}\left(1-\tfrac{1}{k}\right)^{k-2\mu k-i}\\ &\quad+ \sum_{i=-\mu k}^{0} \binom{k-\mu k}{\mu k+i}\left(\tfrac{1}{k}\right)^{\mu k+i}\left(1-\tfrac{1}{k}\right)^{k-2\mu k-i}\\ &= \sum_{i=0}^{k-\mu k} \binom{k-\mu k}{i}\left(\tfrac{1}{k}\right)^{i}\left(1-\tfrac{1}{k}\right)^{k-\mu k-i} = 1.\end{aligned}$$

Thus, starting with X_1^l, the expected runtime for the (1+1) EA to find the global optimum of G_1 is $\Omega(k^{\mu k})$. ∎

While the (1+1) EA needs an expected exponential runtime to jump out of X_1^l, GSEMO can efficiently find the global optimum for instance G_1.

Theorem 5: For instance G_1, GSEMO finds the global optimum in expected runtime $O(k^2 \ln k)$.

Proof: We first determine the expected runtime that GSEMO starting with any initial solution finds the all-zeros solution. Then we determine the expected runtime that starting with the all-zeros solution GSEMO produces the whole Pareto front.

Adding a label from $L_1' = \{1, \dots, \mu k\}$ to the all-zeros bit string can reduce the number of connected components by $(1-\mu)k$, while adding a label from $L_1'' = \{\mu k + 1, \dots, k\}$ can reduce the number of connected components by μk. Note that when $0 < \mu < 1/2$, $(1-\mu)k$ is larger than μk. Hence, the Pareto front contains $\mu k + 1$ Pareto optimal solutions with fitness vectors $(n, 0)$, $(n - (1-\mu)k, 1)$, ..., $(n - (1-\mu)jk, j)$, ..., $(1, \mu k)$, respectively. It is clear that the population size is $O(k)$.

It has been proved in Theorem 3 that the expected runtime for GSEMO starting with any initial solution to include the all-zeros bit string in the population is $O(k^2 \ln k)$.

Now we calculate the expected runtime to produce the whole Pareto front after the all-zeros bit string being found. The worst case is from the all-zeros bit string to produce the whole Pareto front. Suppose that now in the population, there is a Pareto optimal solution X with fitness vector $(n - (1-\mu)jk, j)$, which has the maximum number of labels. Another Pareto optimal solution with fitness vector $(n - (1-\mu)(j+1)k, j+1)$ can be produced by adding a label from L_1' which is not in X. The probability of adding this label is $\binom{\mu k-j}{1}\frac{1}{k}(1-\frac{1}{k})^{k-1} \geq \frac{\mu k-j}{ek}$. This implies that the expected runtime is $O(\frac{k^2}{\mu k-j})$, as the expected runtime that GSEMO selects X from P is $O(k)$. So, considering the worst case of starting with the all-zeros bit string, the expected runtime for GSEMO to produce the whole Pareto front is $O(\sum_{j=0}^{\mu k-1}\frac{k^2}{\mu k-j}) = O(k^2 \ln k)$. ∎

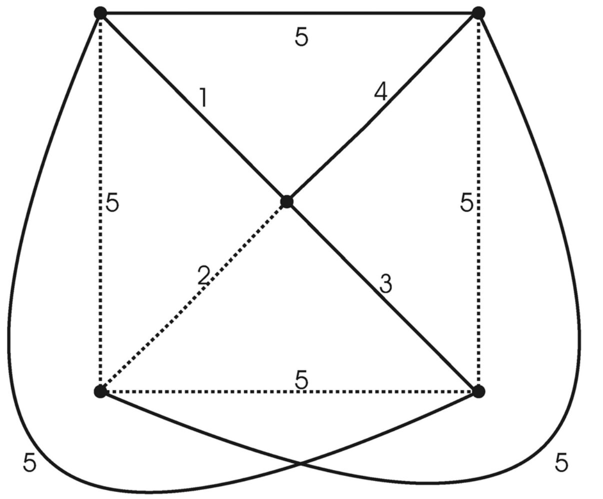


Fig. 2. Example of instance G_2 with $n = 5$.

B. Instance Where the (1+1) EA and GSEMO Outperform ERA

ERA is a local search algorithm. It takes an arbitrary spanning tree as input, then considers each non-tree edge and tests whether the number of used labels can be reduced by adding this non-tree edge and deleting a tree edge on the induced cycle.

In this subsection, we show that the (1+1) EA and GSEMO outperform ERA on an instance proposed by Krumke and Wirth [15], which is denoted by G_2 in this paper.

This instance can be constructed in two steps. First, a star shaped graph is constructed by selecting one node out of n nodes and connecting it to the other $n-1$ nodes with $n-1$ edges, which are labeled by $n-1$ distinct labels: 1, 2, ..., $n-1$. Second, a complete graph is constructed by adding edges with the same label k to the star shaped graph. Thus, we get a complete graph $G_2 = (V_2, E_2, L_2)$, where $|V_2| = n$, $|E_2| = n(n-1)/2$, and $L_2 = \{1, 2, \dots, k\}$ is the set of labels. It is clear that $|L_2| = k$ and $k = n$. Fig. 2 shows an example with $n = 5$, where the dashed edges construct a spanning tree with the minimum number of labels.

For instance G_2, the global optimum $X_2^* = (x_{21}^*, \dots, x_{2k}^*)$ contains two labels: one comes from $\{1, \dots, k-1\}$ and the other is label k, i.e., $|X_2^*| = 2$, $\sum_{i=1}^{k-1} x_{2i}^* = 1$, and $x_{2k}^* = 1$.

Krumke and Wirth [15] used instance G_2 to demonstrate that ERA might perform as badly as possible. In fact, $X_2^l = (\overbrace{1, \dots, 1}^{k-1}, 0)$ is a local optimum for ERA, since starting with

X_2^l the number of labels used in $H(X_2^l)$ cannot be reduced by adding any non-tree edge and deleting a tree edge on the induced cycle. The local optimum uses $k-1$ labels, while the global optimum uses only 2 labels. However, the (1+1) EA can efficiently solve instance G_2.

Theorem 6: For instance G_2, the (1+1) EA starting with any initial solution finds a global optimum in expected runtime $O(k \ln k)$.

Proof: For simplicity, let L_2' denote the label set $\{1, \ldots, k-1\}$.

Let $A = \{X | c(H(X)) = 1, x_k = 1, 2 \leq |X| \leq k-1\}$, i.e., a solution $X \in A$ contains label k and at least one but at most $k-2$ labels from L_2'. Obviously, any solution $X \in A$ is feasible.

To find a global optimum, a solution $X \in A$ should be found first. Once a solution $X \in A$ has been found, the global optimum can be found by removing all $|X|-2$ redundant labels from L_2'. Once such a label is removed from X, it cannot be added anymore. According to the Coupon Collector's theorem [39], all redundant labels contained in X will be removed in expected runtime $O(k \ln k)$.

In order to analyze the expected runtime to find a solution $X \in A$, we further partition all solutions that do not belong to A into five disjoint subsets B_1, B_2, B_3, B_4, B_5

$$B_1 = \{X | c(H(X)) = 1,\ |X| = k, \text{ and } x_k = 1\};$$
$$B_2 = \{X | c(H(X)) = 1,\ |X| = k-1, \text{ and } x_k = 0\};$$
$$B_3 = \{X | c(H(X)) > 1,\ 1 \leq |X| \leq k-2, \text{ and } x_k = 0\};$$
$$B_4 = \{X | c(H(X)) > 1,\ |X| = 1, \text{ and } x_k = 1\};$$
$$B_5 = \{X | c(H(X)) > 1,\ |X| = 0\}.$$

If $X \in B_1$, then X will be transformed into A by removing one label from L_2'. The probability of this event is $\binom{k-1}{1}\frac{1}{k}(1-\frac{1}{k})^{k-1} = \Omega(1)$, which implies that the expected runtime is $O(1)$.

If $X \in B_2$, then X will be transformed into A by adding label k and simultaneously removing one label from L_2'. The probability of this event is $\binom{k-1}{1}(\frac{1}{k})^2(1-\frac{1}{k})^{k-2} = \Omega(\frac{1}{k})$, which implies that the expected runtime is $O(k)$.

If $X \in B_3(B_4)$, then X will be transformed into A by adding label k (one label from L_2'). The probability of this event is $\frac{1}{k}(1-\frac{1}{k})^{k-1} = \Omega(\frac{1}{k})$ $\left(\binom{k-1}{1}\frac{1}{k}(1-\frac{1}{k})^{k-1} = \Omega(1)\right)$, which implies that the expected runtime is $O(k)$.

If $X \in B_5$, then X will be transformed into A by simultaneously adding label k and a label from L_2'. The probability is $\binom{k-1}{1}(\frac{1}{k})^2(1-\frac{1}{k})^{k-2} = \Omega(\frac{1}{k})$, which implies that the expected runtime is $O(k)$.

So, any solution will be transformed into A in expected runtime $O(k)$.

Combining the expected runtime to remove all redundant labels contained in a solution belonging to A, the expected runtime for the (1+1) EA starting with any initial solution to find a global optimum is $O(k \ln k)$. ■

As shown in the following theorem, GSEMO can also efficiently solve instance G_2.

Theorem 7: For instance G_2, GSEMO starting with any initial solution finds a global optimum in expected runtime $O(k^2 \ln k)$.

Proof: Let L_2' denote the label set $\{1, \ldots, k-1\}$. The optimization process consists of two independent phases: the first phase lasts until a solution with fitness vector $(1, *)$, i.e., a feasible solution, is included in the population, and the second phase ends when a global optimum is found.

We now analyze the expected runtime of the first phase. Let X be the solution with fitness vector $(c(H(X)), |X|)$, where $c(H(X))$ is the minimum among all solutions in the population. If $c(H(X)) > 1$, then there are three cases. The first one is that X contains no label, the second is that X contains label k but no label from L_2', and the third is that X contains at least one but at most $k-2$ labels from L_2' and no label k.

For all three cases, a solution with fitness vector $(1, *)$ will be included in expected runtime $O(k^2)$, since the probability of selecting X from P is $\Omega(\frac{1}{k})$, and the probability of transforming X into a solution with fitness vector $(1, *)$ is $\Omega(\frac{1}{k})$.

Once a solution with fitness vector $(1, *)$ is included in the population, we show that a global optimum will be found in expected runtime $O(k^2 \ln k)$. To this end, we partition the second phase into two subphases: the first subphase lasts until a solution belonging to $A = \{X | x_k = 1, 2 \leq |X| \leq k-1\}$ is found, i.e, such a solution contains label k and at least one but at most $k-2$ labels from L_2', the second subphase ends when a global optimum is found.

If a solution X with fitness vector $(1, *)$ and $X \notin A$, then there are two cases needed to be considered: the first is that X contains label k and all labels from L_2', and the other is that X contains all labels from L_2' excluding label k.

For the first case, removing any one of the labels from L_2' will transform X into A. The probability of this event is $\binom{k-1}{1}\frac{1}{k}(1-\frac{1}{k})^{k-1} = \Omega(1)$, which implies that the expected runtime is $O(1)$. For the second case, removing one label from L_2' and simultaneously adding label k will transform X into A. The probability of this event is $\binom{k-1}{1}(\frac{1}{k})^2(1-\frac{1}{k})^{k-2} = \Omega(\frac{1}{k})$, which implies that the expected runtime is $O(k)$. Noting that the probability of selecting X from the population is $\Omega(\frac{1}{k})$, a solution belonging to A will be included in the population in expected runtime $O(k^2)$ after a solution with fitness vector $(1, *)$ being included.

Now at least one solution belonging to A is included in the population. Let X be the solution which has the minimum number of labels among all solutions in the population belonging to A. If X is not the global optimum of G_2, then the global optimum will be found by removing all $|X|-2$ redundant labels from L_2'. Once such a label is removed from X, it cannot be added anymore. According to the Coupon Collector's theorem [39], all redundant labels will be removed in expected runtime $O(k \ln k)$, and the probability of selecting X from P is $\Omega(\frac{1}{k})$, so a global optimum will be found in expected runtime $O(k^2 \ln k)$.

Altogether, GSEMO starting with any initial solution finds a global optimum in expected runtime $O(k^2 \ln k)$. ■

C. Instance Where the (1+1) EA and GSEMO Outperform LS2N

Brüggemann *et al.* [36] proposed an instance, denoted by G_3 in this paper, to show that there exists a local optimum with respect to LS2N.

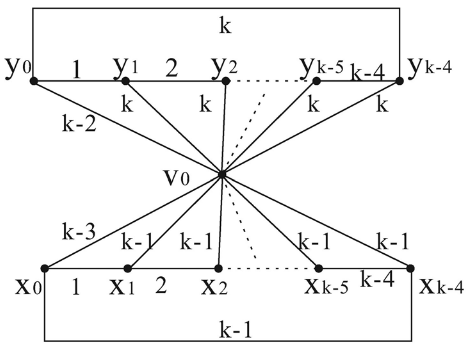


Fig. 3. Instance G_3.

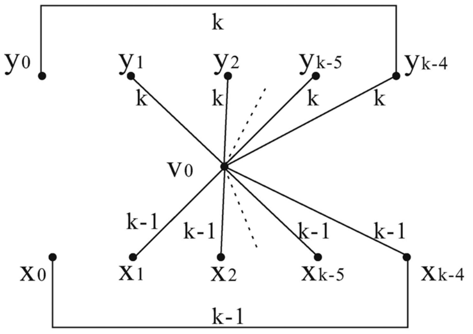


Fig. 4. MLST of instance G_3.

As shown in Fig. 3, this instance is a graph $G_3 = (V_3, E_3, L_3)$, where $V_3 = \{v_0, x_0, x_1, \ldots, x_{k-4}, y_0, y_1, \ldots, y_{k-4}\}$, $L_3 = \{1, 2, \ldots, k\}$, $|V_3| = 2k-5$, $|E_3| = 4k-12$, and $|L_3| = k$. Fig. 4 shows the minimum label spanning tree.

In this instance, the global optimum is

$$X_3^* = (\overbrace{0, \ldots, 0}^{k-2}, 1, \ 1).$$

LS2N might be trapped in the local optimum, which contains labels $1, 2, \ldots, k-2$. In fact, to jump out of this local optimum, at least three labels from $\{1, 2, \ldots, k-2\}$ should be removed and simultaneously two labels $k-1$ and k should be added, but the resulting solution is not in the 2-switch neighborhood of the local optimum. Hence, L2SN may not find the global optimum of G_3. However, the (1+1) EA is efficient for instance G_3.

Theorem 8: For instance G_3, the (1+1) EA starting with any initial solution finds the global optimum in expected runtime $O(k^2)$.

Proof: Let L_3' denote the label set $\{1, \ldots, k-2\}$, and let $C = \{X|c(H(X)) = 1, x_{k-1} = 1, x_k = 1, 2 \leq |X| \leq k-1\}$, i.e, a solution $X \in C$ contains labels $k-1$ and k and at most $k-3$ labels from L_3'.

Note that the global optimum contains only two labels $k-1$ and k. The optimization process consists of two independent phases: the first phase lasts until a solution $X \in C$ is constructed from an arbitrary solution, and the second phase ends when all $|X| - 2$ redundant labels from L_3' are removed.

For analyzing the expected runtime of finding a solution $X \in C$, we partition all solutions that do not belong to C into seven disjoint subsets $D_1, D_2, D_3, D_4, D_5, D_6, D_7$

$D_1 = \{X|c(H(X)) = 1, x_{k-1} = 0, x_k = 0, |X| = k-2\}$;
$D_2 = \{X|c(H(X)) = 1, x_{k-1} = 0, x_k = 1, |X| = k-2$ or $|X| = k-1\}$;
$D_3 = \{X|c(H(X)) = 1, x_{k-1} = 1, x_k = 0, |X| = k-2$ or $|X| = k-1\}$;
$D_4 = \{X|c(H(X)) = 1, x_{k-1} = 1, x_k = 1, |X| = k\}$;
$D_5 = \{X|c(H(X)) > 1, x_{k-1} = 0, x_k = 0\}$;
$D_6 = \{X|c(H(X)) > 1, x_{k-1} = 0, x_k = 1\}$;
$D_7 = \{X|c(H(X)) > 1, x_{k-1} = 1, x_k = 0\}$.

If $X \in D_1$, then X will be transformed into C by adding labels $k-1$, k, and simultaneously removing three labels from L_3'. The probability of this event is $\binom{k-2}{3}(\frac{1}{k})^5(1-\frac{1}{k})^{k-5} = \Omega(\frac{1}{k^2})$, which implies that the expected runtime is $O(k^2)$.

If $X \in D_2(D_3)$, then X will be transformed into C by adding label $k-1$ (k) and simultaneously removing two labels from L_3'. The probability of this event is $\binom{k-3}{2}(\frac{1}{k})^3(1-\frac{1}{k})^{k-3} = \Omega(\frac{1}{k})$, which implies that the expected runtime is $O(k)$.

If $X \in D_4$, then X will be transformed into C by removing a label from L_3'. The probability of this event is $\binom{k-2}{1}\frac{1}{k}(1-\frac{1}{k})^{k-1} = \Omega(1)$, which implies that the expected runtime is $O(1)$.

If $X \in D_5$, then X will be transformed into C by simultaneously adding labels $k-1$ and k. The probability of this event is $(\frac{1}{k})^2(1-\frac{1}{k})^{k-2} = \Omega(\frac{1}{k^2})$, which implies that the expected runtime is $O(k^2)$.

If $X \in D_6(D_7)$, then X will be transformed into C by adding label $k-1$ (k). The probability of this event is $\frac{1}{k}(1-\frac{1}{k})^{k-1} = \Omega(\frac{1}{k})$, which implies that the expected runtime is $O(k)$.

So, a solution belonging to C will be found in expected runtime $O(k^2)$.

In the second phase, removing each label contained in a solution belonging to C, which is from L_3', will reduce the fitness value, and once it is removed it cannot be added anymore. According to the Coupon Collector's theorem [39], the second phase ends in expected runtime $O(k \ln k)$.

Altogether, the expected runtime for the (1+1) EA starting with any initial solution to find the global optimum is $O(k^2)$. ■

GSEMO is also efficient for instance G_3.

Theorem 9: For instance G_3, the expected runtime for GSEMO starting with any initial solution to find the global optimum is $O(k^2 \ln k)$.

Proof: We first analyze the runtime that the all-zeros solution is found by GSEMO starting with an arbitrary solution, then analyze the runtime that the global optimum is found by GSEMO once the all-zeros solution is included in the population.

It has been proved in Theorem 3 that the expected runtime for GSEMO starting with any initial solution to find the all-zeros bit string is $O(k^2 \ln k)$.

Once the all-zeros bit string is included in the population, the Pareto optimal solution X^1 with fitness vector $(k-2, 1)$ will be found by adding label $k-1$ or k to the all-zeros bit string. The probability that GSEMO selects the all-zeros bit string from P is $\Omega(\frac{1}{k})$, and the probability of only flipping a bit corresponding to any one of such labels is $\frac{2}{k}(1-\frac{1}{k})^{k-1} = \Omega(\frac{1}{k})$. So, X^1 will be included in the population in expected runtime $O(k^2)$ after the all-zeros bit string being included. Then the global optimum X_3^* with fitness vector $(1, 2)$ will be found by adding the remaining label from $\{k-1, k\}$ to solution X^1, and the expected runtime to produce solution X_3^* from solution X^1 is also $O(k^2)$.

Therefore, the expected runtime for GSEMO starting with any initial solution to find the global optimum is $O(k^2 \ln k)$. ■

D. Instance Where the (1+1) EA and GSEMO Outperform the Modified MVCA

In this subsection, we show that the (1+1) EA and GSEMO outperform the modified MVCA on an instance proposed by Xiong *et al.* [17], which is denoted by G_4 in this paper.

Given the bound of the labels' frequency $b (b \geq 2)$, let $n = b \cdot b! + 1$. We construct $G_4 = (V_4, E_4, L_4)$ as follows, where $V_4 = \{1, 2, \ldots, n\}$, $|V_4| = n$, and $L_4 = I_b \cup I_{b-1} \cup \cdots \cup I_2 \cup I'$.

We construct $b!$ groups from V_4, each containing $b+1$ nodes, as follows:

$$V_1' = \{1, 2, \ldots, b+1\};$$
$$V_2' = \{b+1, b+2, \ldots, 2b+1\};$$
$$\ldots$$
$$V_j' = \{(j-1)b+1, (j-1)b+2, \ldots, jb+1\};$$
$$\ldots$$
$$V_{b!}' = \{(b!-1)b+1, (b!-1)b+2, \ldots, b!b+1\}.$$

In V_j' $(j = 1, 2, \ldots, b!)$, all pairs of consecutive nodes $((j-1)b+1, (j-1)b+2), \ldots, (jb, jb+1)$ are connected by b edges, and all these b edges are labeled by one label. Thus, $b!$ labels are needed, which constitute the label set I'.

In each V_j', node $(j-1)b+1$ is connected to nodes $(j-1)b+3, \ldots, jb+1$. The label subset I_h $(h = b, b-1, \ldots, 2)$ is obtained as follows. We choose edge $((j-1)b+1, (j-1)b+1+h)$ in each V_j', so there are $b!$ such edges. We label the first h edges with one label, and the next h edges with a second label, and so on. So, $\frac{b!}{h}$ labels are needed, and they construct I_h.

Hence, $|I_h| = \frac{b!}{h}$ $(h = b, b-1, \ldots, 2)$, $|I'| = b!$, and the total number of labels $k = \sum_{j=2}^{b} \frac{b!}{j} + b!$. The edges with $b!$ labels from I' construct the minimum label spanning tree, so in this instance the global optimum is

$$X_4^* = (\overbrace{0, \ldots, 0}^{\sum_{j=2}^{b} \frac{b!}{j}}, \overbrace{1, \ldots, 1}^{b!}).$$

Fig. 5 shows an example with $b = 4$, where the dashed edges construct the spanning tree with the minimum number of labels.

Xiong *et al.* used this instance to show that the modified MVCA may obtain a solution using all labels from $I_b \cup I_{b-1} \cup \cdots \cup I_2 \cup I'$, which is H_b-approximation solution, where $H_b = \sum_{i=1}^{b} \frac{1}{i}$. However, the (1+1) EA can efficiently find the global optimum of instance G_4.

Theorem 10: For instance G_4, the (1+1) EA starting with any initial solution finds the global optimum in expected runtime $O(nk)$, where $n = b \cdot b! + 1$, $k = \sum_{j=2}^{b} \frac{b!}{j} + b!$, and b is the maximum frequency of the labels.

Proof: The optimization process consists of two independent phases. The first phase ends when the (1+1) EA finds a feasible solution, and the second phase lasts until the (1+1) EA removes all redundant labels from $\{1, 2, \ldots, \sum_{j=2}^{b} \frac{b!}{j}\}$.

Let X be the current solution. Note that a feasible solution contains all labels from I'. If X is not a feasible solution, then there must exist a bit x_h from $\{x_i | \sum_{j=2}^{b} \frac{b!}{j} + 1 \leq i \leq \sum_{j=2}^{b} \frac{b!}{j} + b!\}$ whose value is 0. So, the (1+1) EA can decrease the number of connected components by at least one with probability $\frac{1}{k}(1-\frac{1}{k})^{k-1} \geq \frac{1}{ek}$. This is the probability of the event that bit x_h is flipped from 0 to 1 and the other bits keep unchanged. Hence, the expected runtime to decrease the number of connected components from n to 1 is $O(nk)$, i.e., a feasible solution will be found by the (1+1) EA in expected runtime $O(nk)$.

Once a feasible solution X is found, each bit from $\{x_i | \sum_{j=2}^{b} \frac{b!}{j} + 1 \leq i \leq \sum_{j=2}^{b} \frac{b!}{j} + b!\}$ takes value 1, and the flippings of them cannot be accepted by the (1+1) EA, as such flippings will create a disconnected spanning subgraph. For each bit $x_i (1 \leq i \leq \sum_{j=2}^{b} \frac{b!}{j})$, if $x_i = 1$, then it can be flipped from 1 to 0, since this will decrease the fitness value; otherwise, its flipping cannot be accepted by the (1+1) EA, as this will increase the fitness value. So, when all bits have been selected at least once to flip, the global optimum will be found. According to the Coupon Collector's theorem [39], the expected runtime for this to happen is $O(k \ln k)$.

Hence, the expected runtime for the (1+1) EA starting with any initial solution to find the global optimum is $O(nk + k \ln k) = O(nk)$. Note that $n = b \cdot b! + 1 > k = b!(1 + \frac{1}{2} + \cdots + \frac{1}{b})$, and $k > \ln k$. So, $n > \ln k$, and $nk > k \ln k$. ■

GSEMO can solve instance G_4 in expected runtime polynomial in k.

Theorem 11: For instance G_4, GSEMO starting with any initial solution finds the global optimum in expected runtime $O(k^3)$.

Proof: The optimization process consists of two phases. The first phase lasts until GSEMO starting with any initial solution finds a solution with fitness vector $(1, *)$, and the second phase ends when GSEMO finds the global optimum.

Note that a connected spanning subgraph contains all labels from I'. If X is a solution such that $c(H(X)) > 1$, then at least one label from I' is not contained in it, and the number of connected components can be decreased by adding such a label.

We now analyze the expected runtime that GSEMO starting with any initial solution finds a solution with fitness vector $(1, *)$. If such a solution has not been included in the population, then there is a solution X from P such that $c(H(X))$ is the minimal, and adding some label l from I' to X will reduce the number of connected components. The probability that GSEMO chooses X from P is $\Omega(\frac{1}{k})$, as the population size is $O(k)$, and the probability of flipping only the bit corresponding to label l is $\frac{1}{k}(1-\frac{1}{k})^{k-1} = \Omega(\frac{1}{k})$, so a solution with a smaller number of connected components will be found in expected runtime $O(k^2)$. After all labels from I' being added, a connected spanning subgraph will be constructed. Thus, a solution with fitness vector $(1, *)$ will included in expected runtime $O(b!k^2) = O(k^3)$.

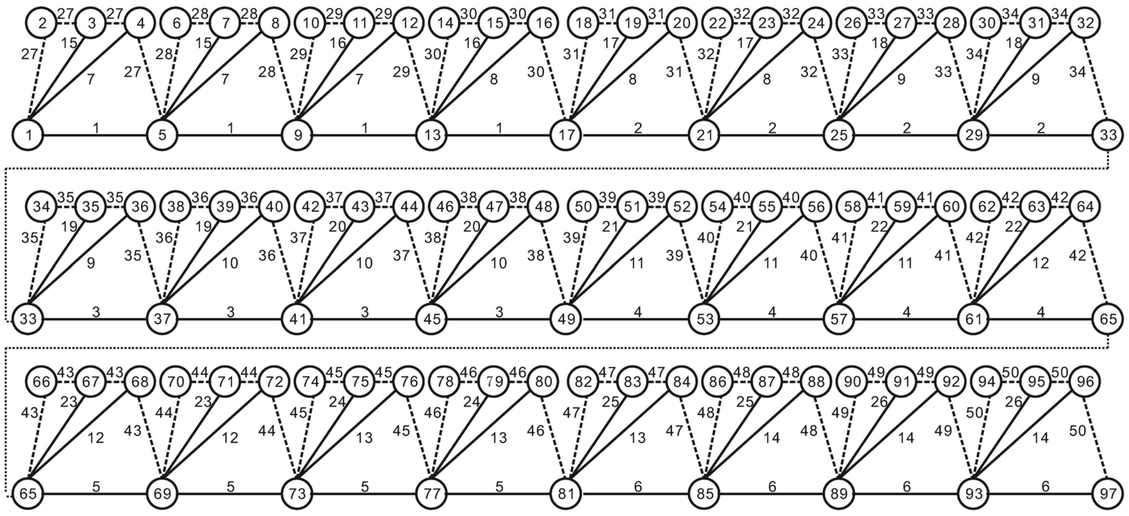


Fig. 5. Instance G_4 with b=4.

TABLE II

BOUNDS ON THE (EXPECTED) RUNTIME OF THE (1+1) EA AND GSEMO, ERA, LS2N, AND THE MODIFIED MVCA TO FIND THE GLOBAL OPTIMA ON FOUR INSTANCES. '—' AND 'INF' MEAN UNKNOWN AND INFINITE, RESPECTIVELY

	Instance G_1	Instance G_2	Instance G_3	Instance G_4
The (1+1) EA	$\Omega(k^{\mu k})$	$O(k \ln k)$	$O(k^2)$	$O(nk)$
GSEMO	$O(k^2 \ln k)$	$O(k^2 \ln k)$	$O(k^2 \ln k)$	$O(k^3)$
ERA	—	Inf	—	—
LS2N	—	—	Inf	—
The modified MVCA	—	—	—	Inf

Note: $\Omega(k^{\mu k})$ is the lower bound on the expected runtime that the (1+1) EA starting with the given initial solution X_1^l finds the global optimum of G_1.

Now at least one solution with fitness vector $(1, *)$ is included in the population. Let X be the solution which has the minimum number of labels among all solutions in the population with fitness vector $(1, *)$. If X is not the global optimum of G_4, then GSEMO can finish the second phase by removing all redundant labels from $I_b \cup I_{b-1} \cup \cdots \cup I_2$. Since the probability of selecting X form P is $\Omega(\frac{1}{k})$, and removing all redundant labels needs an expected runtime $O(k \ln k)$, the global optimum will be found in expected runtime $O(k^2 \ln k)$.

Combining the expected runtime in two phases, we finish the proof. ■

Table II shows that GSEMO outperforms the (1+1) EA on G_1. This is mainly because that GSEMO behaves greedily.

It also shows that ERA (LS2N, the modified MVCA) may be trapped in the local optimum of G_2 (G_3, G_4), thus it cannot efficiently solve G_2 (G_3, G_4). However, the (1+1) EA and GSEMO can efficiently solve them.

This theoretically shows that EAs outperform local search algorithms on some instances. The reason is that EAs are global search algorithms as they use global mutation, while a local search algorithm searches the neighborhood of a current solution, thus it may be trapped in a local optimum.

Nevertheless, local search algorithms are not always worse than EAs.

For example, the modified MVCA can efficiently solve instance G_2. Recall that the modified MVCA begins with the all-zeros bit string, then chooses at each step a label such that when this label is chosen the decrease in the number of connected components is the largest. Therefore, the modified MVCA first tests all labels and chooses label k, as when it is chosen the decrease in the number of connected components is the largest. Next, the modified MVCA tests the remaining $k-1$ labels: 1, 2, ..., $k-1$, and randomly chooses one of them as the decrease in the number of connected components is the same when each of them is chosen. Hence, a global optimum of G_2 is found by the modified MVCA in runtime $2k-1$, which is superior to that of the (1+1) EA and GSEMO.

V. CONCLUSION

In this paper, we investigate the performances of the (1+1) EA and GSEMO for the minimum label spanning tree problem. We reveal that the (1+1) EA and GSEMO can guarantee the achievement of some approximation ratios for the MLST problem. This shows that EAs can guarantee certain approximation ratios for difficult NP-hard problems. We also theoretically show that the (1+1) EA and GSEMO defeat local search algorithms in some instances, and that GSEMO outperforms the (1+1) EA in one instance.

On the MLST$_b$ problem, the (1+1) EA and GSEMO achieve the same approximation ratio of $(b+1)/2$. To find a $(b+1)/2$-approximation solution for the MLST$_b$ problem, the upper bound on the expected runtime of GSEMO is larger than that of the (1+1) EA. However, we still have no idea how to obtain the lower bounds on their runtime for such an approximation ratio.

As for the approximation ratio of the (1+1) EA on the MLST problem, we still know nothing about it. Apart from this, since the (1+1) EA and GSEMO are randomized algorithms, it is natural to ask whether they can achieve better approximate ratios than those guaranteed by some other algorithms or not.

Usually, EAs in practice, use a population of individuals, and in order to retain the diversity of individuals, an appropriate population size is needed, which is different from the (1+1) EA using only one individual. Thus, the performance analysis of population-based EAs is an important research topic (see [19], [40]–[42]).

How do population-based EAs such as (μ+λ) EAs perform on the MLST problem? Are they better than the (1+1) EA? Crossover is usually an important operator for population-based EAs. Several theoretical investigations have recently proven that crossover is essential or useful on some problems, e.g., [43] and [44]. Therefore, it is interesting to investigate the performance of the population-based EA using not only mutation but also crossover on the MLST problem.

REFERENCES

[1] A. S. Tanenbaum, *Computer Networks*. Englewood Cliffs, NJ, USA: Prentice Hall, 1989.

[2] R.-S. Chang and S.-J. Leu, "The minimum labeling spanning trees," *Inform. Process. Lett.*, vol. 63, no. 5, pp. 277–282, 1997.

[3] A. Chwatal, G. Raidl, and O. Dietzel, "Compressing fingerprint templates by solving an extended minimum label spanning tree problem," in *Proc. 7th MIC*, 2007, pp. 105.1–105.3.

[4] A. Chwatal, G. Raidl, and K. Oberlechner, "Solving a k-node minimum label spanning arborescence problem to compress fingerprint templates," *J. Math. Mod. Algor.*, vol. 8, no. 3, pp. 293–334, 2009.

[5] J. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.

[6] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York, NY, USA: Addison-Wesley, 1989.

[7] F. Herrera, M. Lozano and J. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artif. Intell. Rev.*, vol. 12, no. 4, pp. 265–319, 1998.

[8] K. Gallagher and M. Sambridge, "Genetic algorithms: A powerful tool for large-scale non-linear optimization problems," *Comput. Geosci.*, vol. 20, nos. 7–8, pp. 1229–1236, 1994.

[9] Y. Xiong, B. Golden, and E. Wasil, "A one-parameter genetic algorithm for the minimum labeling spanning tree problem," *IEEE Trans. Evol. Comput.*, vol. 9, no. 1, pp. 55–60, Feb. 2005.

[10] J. Nummela and B. Julstrom, "An effective genetic algorithm for the minimum-label spanning tree problem," in *Proc. GECCO*, 2006, pp. 553–558.

[11] S. Consoli and J. Moreno-Pérez, "Solving the minimum labelling spanning tree problem using hybrid local search," in *Proc. Euro. MEC*, vol. 39, 2012, pp. 75–82.

[12] A. Chwatal and G. Raidl, "Solving the minimum label spanning tree problem by mathematical programming techniques," Vienna Univ. Technol., Inst. Comput. Graph. Algorithms, Vienna, Autria, Tech. Rep. TR 186–1–10–03, Jun. 2010.

[13] R. Cerulli, A. Fink, M. Gentili, and S. Voß, "Metaheuristics comparison for the minimum labelling spanning tree problem," in *The Next Wave on Computing, Optimization, and Decision Technologies*, B. L. Golden, S. Raghavan, E. A. Wasil, eds. New York, NY, USA: Springer, 2005, pp. 93–106.

[14] S. Consoli, K. Darby-Dowman, N. Mladenović, and J. Moreno-Pérez, "Greedy randomized adaptive search and variable neighbourhood search for the minimum labelling spanning tree problem," *Eur. J. Oper. Res.*, vol. 196, no. 2, pp. 440–449, Jul. 2009.

[15] S. O. Krumke and H. Wirth, "On the minimum label spanning tree problem," *Inform. Process. Lett.*, vol. 66, no. 2, pp. 81–85, 1998.

[16] Y. Wan, G. Chen, and Y. Xu, "A note on the minimum label spanning tree," *Inform. Process. Lett.*, vol. 84, no. 2, pp. 99–101, 2002.

[17] Y. Xiong, B. Golden, and E. Wasil, "Worst-case behavior of the MVCA heuristic for the minimum labeling spanning tree problem," *Oper. Res. Lett.*, vol. 33, no. 1, pp. 77–80, 2005.

[18] S. Jansen and I. Wegener, "Evolutionary algorithms: How to cope with plateaus of constant fitness and when to reject strings of the same fitness," *IEEE Trans. Evol. Comput.*, vol. 5, no. 6, pp. 589–599, Dec. 2001.

[19] J. He and X. Yao, "Drift analysis and average time complexity of evolutionary algorithms," *Artif. Intell.*, vol. 127, no. 1, pp. 57–85, 2001.

[20] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+1) evolutionary algorithm," *Theor. Comput. Sci.*, vol. 276, nos. 1–2, pp. 51–81, 2002.

[21] J. He and X. Yao, "Towards an analytic framework for analysing the computation time of evolutionary algorithms," *Artif. Intell.*, vol. 145, nos. 1–2, pp. 59–97, 2003.

[22] F. Neumann, J. Reichel, and M. Skutella, "Computing minimum cuts by randomized search heuristics," *Algorithmica*, vol. 59, no. 3, pp. 323–342, 2011.

[23] Y. Zhou, J. He, and Q. Nie, "A comparative runtime analysis of heuristic algorithms for satisfiability problems," *Artif. Intell.*, vol. 173, no. 2, pp. 240–257, 2009.

[24] F. Neumann and I. Wegener, "Randomized local search, evolutionary algorithms, and the minimum spanning tree problem," *Theor. Comput. Sci.*, vol. 378, no. 1, pp. 32–40, 2007.

[25] F. Neumann, "Expected runtimes of evolutionary algorithms for the Eulerian cycle problem," *Comput. Oper. Res.*, vol. 35, no. 9, pp. 2750–759, 2008.

[26] A. Sutton and F. Neumann, "A parameterized runtime analysis of evolutionary algorithms for the Euclidean traveling salesperson problem," in *Proc. 26th Conf. AAAI*, 2012, pp. 1105–1111.

[27] O. Giel and I. Wegener, "Evolutionary algorithms and the maximum matching problem," in *Proc. 20th Annu. STACS*, 2003, vol. 2607, pp. 415–426.

[28] P. Oliveto, J. He, and X. Yao, "Analysis of the (1+1)-EA for finding approximate solutions to vertex cover problems," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1006–1029, Oct. 2009.

[29] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C. Witt, "Approximating covering problems by randomized search heuristics using multiobjective models," *Evol. Comput.*, vol. 18, no. 4, pp. 617–633, 2010.

[30] C. Witt, "Worst-case and average-case approximations by simple randomized search heuristics," in *Proc. 22nd Annu. STACS*, 2005, vol. 3404, pp. 44–56.

[31] F. Neumann and J. Reichel, "Approximating minimum multicuts by evolutionary multiobjective algorithms," in *Proc. 10th Int. Conf. PPSN*, 2008, pp. 72–81.

[32] Y. Yu, X. Yao, and Z. Zhou, "On the approximation ability of evolutionary optimization with application to minimum set cover," *Artif. Intell.*, vols. 180–181, pp. 20–33, Apr. 2012.

[33] F. Neumann and I. Wegener, "Minimum spanning trees made easier via multiobjective optimization," *Natural Comput.*, vol. 5, no. 3, pp. 305–319, 2006.

[34] F. Neumann, "Expected runtimes of a simple evolutionary algorithm for the multiobjective minimum spanning tree problem," *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1620–1629, 2007.

[35] C. Qian, Y. Yu, and Z.-H. Zhou, "An analysis on recombination in multiobjective evolutionary optimization," *Artif. Intell.*, vol. 204, pp. 99–119, Nov. 2013.

[36] T. Brüggemann, J. Monnot, and G. Woeginger, "Local search for the minimum label spanning tree problem with bounded color classes," *Oper. Res. Lett.*, vol. 31, no. 3, pp. 195–201, 2003.

[37] O. Giel, "Expected runtimes of a simple multiobjective evolutionary algorithm," in *Proc. IEEE Congr. Evol. Comput.*, 2003, pp. 1918–1925.

[38] M. Laumanns, L. Thiele, and E. Zitzler, "Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 170–182, Apr. 2004.

[39] M. Mitzenmacher and E. Upfal, *Propability and Computing*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

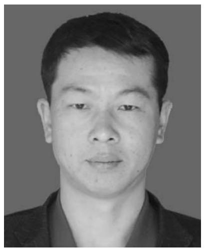
[40] T. Jansen, K. A. D. Jong, and I. Wegener, "On the choice of the offspring population size in evolutionary algorithms," *Evol. Comput.*, vol. 13, no. 4, pp. 413–440, 2005.

[41] T. Chen, J. He, G. Sun, G. Chen, and X. Yao, "A new approach to analyzing average time complexity of population-based evolutionary algorithms on unimodal problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 5, pp. 1092–1106, Oct. 2009.

[42] T. Chen, K. Tang, G. Chen, and X. Yao, "A large population size can be unhelpful in evolutionary algorithms," *Theor. Comput. Sci.*, vol. 436, no. 8, pp. 54–70, 2012.

[43] T. Jansen and I. Wegener, "Real royal road functions—Where crossover provably is essential," *Discrete Appl. Math.*, vol. 149, pp. 111–125, 2005.

[44] B. Doerr, E. Happ, and C. Klein, "Crossover can provably be useful in evolutionary computation," *Theor. Comput. Sci.*, vol. 425, pp. 17–33, 2012.



Xinsheng Lai received the M.Sc. degree in computer applied technology from Guizhou University, Guiyang, China, in 2004. He is currently working toward the Ph.D. degree from South China University of Technology, Guangzhou, China.

His current research interests include evolutionary computation, neural computation, and their real world applications.

Yuren Zhou received the B. Sc. degree in mathematics from Peking University, Beijing, China, in 1988, the M.Sc. degree in the mathematics from Wuhan University, Wuhan, China, in 1991, and the Ph.D. degree in computer science from the same University in 2003.

He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His current research interests include design and analysis of algorithms, evolutionary computation, and data mining.

Jun He (M'06) received the B.S. and M.Sc. degrees in mathematics, and the Ph.D. degree in computer science from Wuhan University, Wuhan, China, in 1989, 1992, and 1995, respectively.

He is currently a Senior Lecturer at Aberystwyth University, Wales, U.K. He has published over 80 papers in his research areas. His current research interests include evolutionary computation, global optimization, and network security.

Jun Zhang (M'02–SM'08) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Kowloon, Hong Kong, in 2002.

Since 2004, he has been with Sun Yat-Sen University, Guangzhou, China, where he is currently a Cheung Kong Professor with the School of Advanced Computing. He has authored 7 research books and book chapters, and over 100 technical papers in his research areas. His research interests include computational intelligence, cloud computing, high performance computing, data mining, wireless sensor networks, operations research, and power electronic circuits.

Dr. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China, in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, the IEEE TRANSACTIONS ON CYBERNETICS, and the *IEEE Computational Intelligence Magazine*.