Performance Analysis of Evolutionary Algorithms for Steiner Tree Problems

Xinsheng Lai
School of Mathematics and Computer Science, Shangrao Normal University, Shangrao, 334001, China; School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510006, China

Yuren Zhou∗
yrzhou@scut.edu.cn
School of Data and Computer Science, Sun Yat-sen University, Guangzhou, 510006, China

Xiaoyun Xia
scutxxxy@gmail.com
College of Mathematics, Physics and Information Engineering, Jiaxing University, Jiaxing, 314001, China; School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510006, China

Qingfu Zhang
qingfu.zhang@cityu.edu.hk
Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong SAR

doi:10.1162/EVCO_a_00200

Abstract
The Steiner tree problem (STP) aims to determine some Steiner nodes such that the minimum spanning tree over these Steiner nodes and a given set of special nodes has the minimum weight, which is NP-hard. STP includes several important cases. The Steiner tree problem in graphs (GSTP) is one of them. Many heuristics have been proposed for STP, and some of them have proved to be performance guarantee approximation algorithms for this problem. Since evolutionary algorithms (EAs) are general and popular randomized heuristics, it is significant to investigate the performance of EAs for STP. Several empirical investigations have shown that EAs are efficient for STP. However, up to now, there is no theoretical work on the performance of EAs for STP. In this article, we reveal that the (1+1) EA achieves $\frac{3}{2}$-approximation ratio for STP in a special class of quasi-bipartite graphs in expected runtime $O((r+s-1) \cdot w_{\max})$, where $r$, $s$, and $w_{\max}$ are, respectively, the number of Steiner nodes, the number of special nodes, and the largest weight among all edges in the input graph. We also show that the (1+1) EA is better than two other heuristics on two GSTP instances, and the (1+1) EA may be inefficient on a constructed GSTP instance.

Keywords
Steiner tree, evolutionary algorithms, computational complexity, approximation ratio, multi-objective.

1 Introduction
The STP problem, named after Jakob Steiner, looks for a tree spanning a given set of nodes with the minimum weight by introducing some auxiliary nodes. All nodes in the
given set are called special nodes. This problem is a fundamental NP-hard combinatorial optimization problem (Garey et al., 1977).

The STP problem has wide applications in many fields, such as circuit layout, networks design, and identification of subnetwork for a given set of seed genes or proteins (Sadeghi and Fröhlich, 2013).

The problem depends on the way the weight between two nodes is determined; thus several special cases are derived. In this article, we consider one important case of STP, that is, the Steiner tree problem in graphs (GSTP), which is also NP-hard (Hwang et al., 1992).

The GSTP problem: given an undirected graph \( G = (V, E) \) and a weighting function \( w : E \rightarrow R^+ \), where \( V \) and \( E \) are, respectively, the sets of nodes and edges, and given a set \( S \subseteq V \) of special nodes, the GSTP problem is to find a tree that spans all special nodes in \( S \) and possibly some nodes from \( V \setminus S \) with the minimum sum of edge weights. We call such a tree the minimum weight Steiner tree.

For a GSTP problem, if \( w \) is a metric, then we call such a GSTP problem a metric STP problem. The concept of metric will be defined in the next section.

If the number of special nodes is 2, that is, \( |S| = 2 \), then the GSTP problem is reduced to the “shortest path problem”; if \( S = V \), then it is reduced to the “minimum spanning tree problem.” Both can be efficiently solved (see, e.g., Dijkstra, 1959; Cheriton and Tarjan, 1976).

In this article, let \( R := V \setminus S \), i.e., \( R \) is the set of all Steiner nodes, \( |V| = n \), and \( |E| = m \). If there is no edge connecting two nodes in \( R \), then \((G, S)\) is called quasi-bipartite (Rajagopalan and Vazirani, 1999).

For NP-hard combinatorial optimization problems, including STP, it is believed that there exists no polynomial-time algorithm so far. Thus, quite a few approximation algorithms have been developed for the STP problem over the past thirty years.

For GSTP, a 2-approximation algorithm has been presented for the first time (Takahashi and Matsuyama, 1980; Kou et al., 1981), which is a heuristic based on the minimum spanning tree. Prömel and Steger (1997) proposed an approximation algorithm for GSTP, obtaining an improved approximation ratio of \( \frac{5}{3} + \epsilon \) for any constant \( \epsilon > 0 \). Vazirani (2000) surveyed approximation algorithms for GSTP developed before 2000. Recently, Robins and Zelikovsky (2005) presented a heuristic algorithm that achieves 1.55-approximation ratio for GSTP in general graphs and 1.28-approximation ratio in quasi-bipartite graphs.

The EA is a randomized heuristic and a general purpose problem solver, so it is natural to investigate the performance of this algorithm for the STP problem. Several experimental investigations have shown that GAs, which belong to the larger class of EAs, are efficient for STP (Hesser et al., 1989; Rabkin, 2002; Kapsalis et al., 1993; Haghighat et al., 2002). However, we know nothing in theory about the efficiencies of EAs for STP.

In fact, extensive attention has been paid recently to the theoretical analysis of evolutionary algorithms’ performance on combinatorial optimization problems. These problems range from simple pseudo-Boolean functions (Jansen and Wegener, 2001; He and Yao, 2001; Droste et al., 2002; He and Yao, 2003), to classic combinatorial optimization problems such as minimum spanning tree problems (Neumann and Wegener, 2007), Eulerian cycle problems (Neumann, 2008), satisfiability problems (Zhou et al., 2009), minimum cut problems (Neumann et al., 2011), and Euclidean traveling salesperson problems (Sutton and Neumann, 2012).
Recently, the performance of EAs has been studied on the single-objective minimum spanning tree problem (Neumann and Wegener, 2007), a problem belonging to the complexity class \(\mathcal{P}\), and also on the multi-objective minimum spanning tree problem (Neumann, 2007; Qian et al., 2013), an NP-hard problem. The STP is another spanning tree-related optimization problem.

This article is devoted to theoretically investigating how EAs perform on the STP problem, which includes the approximation ability of EAs on this problem. Since we are in practice satisfied with good approximation solutions, the approximation performance analysis of EAs for NP-hard combinatorial optimization problems has recently become a hot topic (Giel and Wegener, 2003; Oliveto et al., 2009; Friedrich et al., 2010; Witt, 2005; Yu et al., 2012; Jansen et al., 2013). We reveal that the (1+1) EA is a \(2/3\)-approximation algorithm for the metric STP problem in quasi-bipartite graphs when weights on edges are positive integer numbers and polynomially bounded. Investigations on two GSTP instances show that the (1+1) EA outperforms the so-called average weight heuristic and the heuristic based on the minimum spanning tree. On one constructed GSTP instance the (1+1) EA may need exponential expected runtime to find its optimal solution.

The next section describes some definitions, notations, and algorithms discussed in this article, and Section 3 discusses the approximation performance of the (1+1) EA for the metric STP problem in a class of quasi-bipartite graphs. Section 4 investigates the performance of the (1+1) EA on GSTP instances. The last section concludes the article.

2 Preliminaries

At first, we describe the concepts of subgraph and induced subgraph, which can be found in any textbook on graph theory (see Bondy and Murty, 2008).

**Definition 1 (Subgraph):** Let \(G = (V, E)\) and \(G' = (V', E')\) be two graphs, where \(V(V')\) is the node set of \(G(G')\), and \(E(E')\) is the edge set of \(G(G')\). If \(V' \subseteq V\) and \(E' \subseteq E\), then \(G'\) is a subgraph of \(G\).

**Definition 2 (Induced subgraph):** Let \(G = (V, E)\) and \(G' = (V', E')\) be two graphs, where \(V(V')\) is the node set of \(G(G')\), and \(E(E')\) is the edge set of \(G(G')\). If \(V' \subseteq V\) and \(E' = \{(u, v) | u, v \in V', (u, v) \in E\}\), then \(G'\) is a subgraph induced by \(V'\), which is denoted by \(G[V']\).

The weight of a tree \(T\) is the sum of the weights of all edges in this tree; that is, \(w(T) = \sum_{e \in T} w_e\). For induced subgraph \(G[V']\), we denote the minimum spanning tree of \(G[V']\) as \(mst(G[V'])\), and the weight of \(mst(G[V'])\) as \(w(mst(G[V']))\).

As mentioned earlier, we aim to find a subset of Steiner nodes for GSTP such that the minimum spanning tree over them and the special nodes has the minimum weight. Therefore, a subset of the set \(R\) of Steiner nodes is a solution. Assume that the number of Steiner nodes is \(r\); that is, \(|R| = r\). We sort all Steiner nodes in a fixed order. Thus, in the (1+1) EA a bit string \(X = (x_1, x_2, \ldots, x_r) \in \{0, 1\}^r\) represents a solution, where \(x_i = 1\) if Steiner node \(i\) is selected, and \(x_i = 0\) otherwise. Thus, a bit string \(X\) represents a subset of Steiner nodes, and vice versa. So, the terms bit string, solution, and a subset of Steiner nodes will be interchangeably used in this article.

Let \(X\) be a solution for GSTP, \(c(G[S \cup X])\) be the number of connected components of \(G[S \cup X]\), and \(w_i\) be the weight of the minimum spanning tree in the \(i\)-th connected component \((i = 1, \ldots, c(G[S \cup X]))\). Following one of the fitness functions for the standard minimum spanning tree problem by Neumann and Wegener (2007), we define the
fitness function of the (1+1) EA for GSTP as follows:

\[
    fit(X) = (c(G[S \cup X]) - 1) \times W^2 + \sum_{i=1}^{c(G[S \cup X])} w_i,
\]

where \( W = m \cdot w_{\text{max}} \), \( w_{\text{max}} \) is the largest weight among all edges in \( E \); that is, \( w_{\text{max}} = \max\{w_e | e \in E\} \), and as previously mentioned, \( m \) is the total number of edges in the input graph \( G \), that is, \( m = |E| \). If \( c(G[S \cup X]) = 1 \), then we call \( X \) a feasible solution. This fitness function can be efficiently computed as follows. By using a binary search function, \( w_{\text{max}} \) can be computed in advance. For a given solution \( X \), we first compute the induced subgraph \( G[S \cup X] \), then compute the number \( c(G[S \cup X]) \) of connected components in \( G[S \cup X] \), and finally compute the minimum spanning tree in each component using Prim’s algorithm (Prim, 1957) and add up their weights.

Clearly, for a feasible solution \( X \) which satisfies \( c(G[S \cup X]) = 1 \), the fitness value of \( X \) is \( w(\text{mst}(G[S \cup X])) \).

Fitness function (1) has to be minimized by the (1+1) EA. The first target leads to find a feasible solution \( X \), and the second leads to decrease the weight of the spanning tree over \( X \) and \( S \).

The (1+1) EA for GSTP can be described as follows. Typically, the (1+1) EA accepts a new solution as long as it is not worse than the current one. In this article, the (1+1) EA accepts a new solution if and only if it is better than the current one. Jansen and Wegener (2001) showed that the criterion for accepting a new solution has an important effect on the behavior of the (1+1) EA for plateaus of constant fitness. In this sense, if a fitness function contains no plateaus, then the analysis of these two simple (1+1) EAs on this fitness function is the same.

**Algorithm 1: The (1+1) EA for GSTP**

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

In this article, the runtime of the (1+1) EA refers to the number of fitness evaluations until some termination criterion is fulfilled. If the (1+1) EA can efficiently solve a problem, we are interested in the expected optimization runtime; otherwise, we care about what approximation performance guarantee it can efficiently achieve.

Consider that a randomized heuristic algorithm \( A \) is used to find a minimum solution for a combinatorial optimization problem \( B \). If \( q = \max_{I \in B} \left\{ \frac{f(A(I))}{f(OPT(I))} \right\} \), where \( f(A(I)) \) is the value of the solution obtained by \( A \) for an instance \( I \) of \( B \) in an expected polynomial runtime and \( f(OPT(I)) \) denotes the value of the global optimum of \( I \), then we say that algorithm \( A \) achieves a \( q \)-approximation solution (ratio) for problem \( B \).

For the Steiner tree problem in quasi-bipartite graphs, which is NP-hard (Chlebik and Chlebikova, 2002), too, Rizzi (2003) proved that the following heuristic
algorithm achieves a $\frac{3}{2}$-approximation ratio, which is called the iterated 1-Steiner heuristic (ISH).

**Algorithm 2: The iterated 1-Steiner heuristic (ISH) (Rizzi, 2003)**

01: $J \leftarrow \emptyset$; $T \leftarrow$ any minimum spanning tree of $G[S \cup J]$;
02: While $\exists x \in R \setminus J$ such that $w(mst(G[S \cup J \cup \{x\})) < w(T)$ do
03: $J \leftarrow J \cup \{x\}$; $T \leftarrow$ any minimum spanning tree of $G[S \cup J]$;
04: Remove from $J$ all nodes with degree one in $T$; update $T$ accordingly (drop the corresponding leafs);
05: Remove from $J$ all nodes with degree two in $T$; update $T$ accordingly
   (shortcut the pairs of consecutive edges $(u, y)$ and $(y, v)$ with the single edge $(u, v)$);
06: End while
07: Return $T$.

For completeness, we describe two other heuristic algorithms for GSTP in the following.

The first heuristic algorithm for GSTP is the minimum spanning tree-based heuristic algorithm (MSTA) proposed by Takahashi and Matsuyama (1980), and it is also independently proposed by Kou et al. (1981).

Given an input graph $G = (V, E)$, and a set $S(\subseteq V)$ of special nodes, MSTA first constructs a complete graph $G'$ on $S$, where the weight on each edge connecting two special nodes equals the weight of the shortest path between them in the input graph. Then, MSTA finds the minimum spanning tree on $G'$. By replacing edges in this minimum spanning tree with their corresponding shortest path in the input graph, MSTA constructs a subgraph of $G$. Finally, MSTA finds a minimum spanning tree of this subgraph, and constructs a Steiner tree from the minimum spanning tree. The following describes MSTA.

**Algorithm 3: The minimum spanning tree-based heuristic algorithm (MSTA) (Kou et al., 1981)**

01: Construct a complete graph $G'$ on $S$ from the input graph $G = (V, E)$;
02: Find a minimum spanning tree $T_s$ from $G'$;
03: Construct subgraph $G_s$ of $G$ by replacing each edge in $T_s$ with its corresponding shortest path in $G$;
04: Find a minimum spanning tree $T_{s'}$ of $G_s$.
05: Construct a Steiner tree $T$ from $T_{s'}$ by deleting edges in $T_{s'}$, if necessary, such that all leaves in $T$ are special nodes;
06: Output: $T$.

The second heuristic algorithm for GSTP is proposed by Rayward-Smith, which is called the average distance heuristic (Rayward-Smith, 1983; Bern and Plassmann, 1989; Waxman and Imase, 1988). Since the concept of “distance” in this heuristic algorithm is equivalent to the concept of “weight” in this article, we call it the average weight heuristic (AWH), where $w(u, v)$ means the weight on the edge connecting nodes $u$ and $v$. The following description of AWH is taken from Bern and Plassmann (1989).

**Algorithm 4: The average weight heuristic (AWH) (Bern and Plassmann, 1989)**

01: Find a node $v$ (Steiner or special) and a set $S'$ of special nodes (possibly containing
(1 + 1) EA achieves a \(\frac{3}{2}\)-approximation ratio for the metric STP problem in a special class of quasi-bipartite graphs as long as \(w_{\text{max}}\) is polynomially bounded.

The first three conditions are usually satisfied in an edge-weighted undirected graph. The last condition is called the triangle inequality. If the weight on edges of a given undirected graph \(G = (V, E)\) is a metric, then the edge weights satisfy triangle inequality; that is, the weight of an edge that forms a triangle with two other edges is less than or equal to the sum of the weights of the other two.

### 3 The Approximation Performance Guarantee of the (1+1) EA for the Metric STP Problem in Quasi-Bipartite Graphs

In this section, we show that the (1+1) EA achieves a \(\frac{3}{2}\)-approximation ratio for the metric STP problem in a special class of quasi-bipartite graphs as long as \(w_{\text{max}}\) is polynomially bounded.

The following lemma has been proven by Rizzi, which shows that ISH described in Algorithm 2 produces a \(\frac{3}{2}\)-approximation solution for the metric STP problem, when \((G, S)\) is quasi-bipartite.

**Lemma 1 (Rizzi, 2003):** Let \(X\) be a solution such that each Steiner node in \(\text{mst}(G[S \cup X])\) connects at least three special nodes and \(w(\text{mst}(G[S \cup X \cup \{x\}])) \geq w(\text{mst}(G[S \cup X]))\) for every Steiner node \(x \in R \setminus X\), then \(X\) is a \(\frac{3}{2}\)-approximation solution to the metric STP problem, when \((G, S)\) is quasi-bipartite.

Further, let weights on edges be integer numbers. We now prove that starting with any initial solution the (1+1) EA achieves a \(\frac{3}{2}\)-approximation solution for the metric STP problem in expected runtime \(O(r(r + s - 1) \cdot w_{\text{max}})\), which is polynomial as long as \(w_{\text{max}}\) is polynomially bounded.

**Theorem 1:** Starting with any initial solution, the (1+1) EA achieves a \(\frac{3}{2}\)-approximation ratio for the metric STP problem in expected runtime \(O(r(r + s - 1) \cdot w_{\text{max}})\), when \((G, S)\) is quasi-bipartite and weights on edges are integer numbers.

**Proof:** For a solution \(X\), consider the Steiner nodes in \(\text{mst}(G[S \cup X])\). In \(\text{mst}(G[S \cup X])\) there is no edge connecting two Steiner nodes, as \((G, S)\) is quasi-bipartite. Let \(C_1(X)\) denote that each Steiner node in \(\text{mst}(G[S \cup X])\) connects at least three special nodes, and let \(C_2(X)\) denote that \(w(\text{mst}(G[S \cup X \cup \{x\}])) \geq w(\text{mst}(G[S \cup X]))\) for every Steiner node \(x \in R \setminus X\).
We partition the solution space $A = \{0, 1\}^r$ into two disjoint subspaces. One is $A_1 = \{X | C_1(X) \land C_2(X)\}$; the other is its complement set $A_2 = \{X | \neg C_1(X) \lor \neg C_2(X)\}$.

The main idea behind the proof is that if a solution $X$ is in subspace $A_1$, then according to Lemma 1, $X$ is a $\frac{3}{2}$-approximation solution to the metric STP problem in $G$; if $X$ is in subspace $A_2$, then the fitness value of $X$ can be decreased by at least one in expected time $O(r)$, and thus be efficiently transformed to a solution in subspace $A_1$ as long as $w_{\text{max}}$ is polynomially bounded.

Note that $c(G[S \cup X]) = 1$ holds for any solution $X$, as $w$ is a metric. Thus, each $X \in \{0, 1\}^r$ is a feasible solution, implying that the fitness value of $X$ is $w(\text{mst}(G[S \cup X]))$.

Let $X$ be the current solution. If $X$ is not in $A_1$, then it is in $A_2$, i.e., $\neg C_1(X) \lor \neg C_2(X)$ holds.

If $\neg C_1(X)$ holds, then there is at least one Steiner node $x$ in $\text{mst}(G[S \cup X])$ connecting one or two special nodes. If it connects only one special node, then it is a leaf node in $\text{mst}(G[S \cup X])$. In this case, removing $x$ and the edge incident to it will produce a Steiner tree covering all special nodes whose weight is less than $w(\text{mst}(G[S \cup X]))$.

Hence, removing $x$ from $X$ results in solution $X' = X \setminus \{x\}$ such that $w(\text{mst}(G[S \cup X'])) < w(\text{mst}(G[S \cup X]))$. If it connects two special nodes, say $u$ and $v$, then removing $x$ and two edges $(x, u)$ and $(x, v)$ and simultaneously connecting the two special nodes $u$ and $v$ with edge $(u, v)$ will produce a Steiner tree whose weight is less than that of the current one. This improvement follows from the triangle inequality. Hence, removing $x$ from $X$ can also result in solution $X' = X \setminus \{x\}$ such that $w(\text{mst}(G[S \cup X'])) < w(\text{mst}(G[S \cup X]))$.

Altogether, if $\neg C_1(X)$ holds, then the fitness value can be decreased by removing a Steiner node from $X$.

If $\neg C_2(X)$ holds, then there is a Steiner node $z \in R \setminus X$ such that $w(\text{mst}(G[S \cup X \cup \{z\}])) < w(\text{mst}(G[S \cup X]))$.

Altogether, if $X$ is in $A_2$, that is, $\neg C_1(X) \lor \neg C_2(X)$ holds, then the event that removes one specific Steiner node from $X$, or adds some specific Steiner node $x \in R \setminus X$ to $X$, results in a new solution whose fitness value is less than $w(\text{mst}(G[S \cup X]))$. The probability of this event is $\frac{1}{2}(1 - \frac{1}{r})^{-1} = \Omega(\frac{1}{r})$, which implies that the expected time is $O(r)$. Since the weight of each edge is a positive integer, the fitness value of $X$ can be decreased by at least one in expected time $O(r)$.

For an arbitrary solution, the minimum spanning tree covering all special nodes contains at most $r + s - 1$ edges, which contains all special nodes and all Steiner nodes. So the weight of this minimum spanning tree is at most $(r + s - 1) \cdot w_{\text{max}}$. Hence, a $\frac{3}{2}$-approximation solution can be found in expected runtime $O((r + s - 1) \cdot w_{\text{max}})$. □

4 Performance Analysis of the (1+1) EA on GSTP Instances

In this section, we show that the (1+1) EA is better than two other heuristics on two instances. At the end of this section, another instance is constructed to show that the (1+1) EA cannot always be efficient for GSTP.

4.1 An Instance Where the (1+1) EA Outperforms the MSTA

Takahashi and Matsuyama (1980) constructed an instance of GSTP, which we call $G_1$ in this article, to show that the approximation ratio produced by the MSTA described in Algorithm 3 is tight.

$G_1 = (V_1, E_1)$ is a complete graph, where $V_1$ is the set of nodes $\{v_1, v_2, \ldots, v_n\}$, $E_1 = \{(v_i, v_j) | i, j = 1, 2, \ldots, n\}$, and $S_1 = \{v_1, v_2, \ldots, v_s\}$, $2 \leq s \leq n$. The weight on the
edge $w(v_i, v_j)$ is defined as follows:

$$w(v_i, v_j) = \begin{cases} 
1 & i = 1, 2, \ldots, s; j = s + 1, \\
2 & i = 1, \ldots, s - 1; j = i + 1, \\
10 & \text{otherwise}.
\end{cases}$$

As shown in Figure 1, the solid edges construct the minimum Steiner tree of $G_1$, which covers all special nodes $\{v_1, v_2, \ldots, v_s\}$ and Steiner node $v_{s+1}$. Clearly, the weight of the minimum weight Steiner tree is $s$. In this article, Steiner nodes and special nodes are represented by hollow and solid circles, respectively.

For $G_1$, MSTA achieves a $2(1 - \frac{1}{2})$-approximation ratio. A complete graph $G'$ is constructed by MSTA on the set $S_1$ of special nodes. Since the weight on each edge of the complete graph $G'$ is 2, the total weight of the minimum spanning tree on $G'$ is $2(s - 1)$.

In this subsection, we show that the $(1+1)$ EA can efficiently find the minimum weight Steiner tree of $G_1$.

For $G_1$, $r = n - s$, thus a solution can be represented as $(x_1, x_2, \ldots, x_r) (\in \{0, 1\}^r)$, where $x_1$ corresponds to Steiner node $v_{s+1}$; $x_2$ corresponds to Steiner node $v_{s+2}$; $\ldots$; $x_r$ corresponds to Steiner node $v_n$. It is clear that the global optimum is $(1, 0, \ldots, 0)$. If $s = 2$, then $(0, \ldots, 0)$ is also a global optimum.

**Theorem 2:** The $(1+1)$ EA starting with any initial solution finds the global optimum of $G_1$ in expected runtime $O(r \ln r)$.

**Proof:** Note that $G_1$ is complete. Any solution $X$ is feasible, as the induced subgraph $G[S_1 \cup X]$ is connected. Let $X$ denote the current solution, and let $L_1 = \{v_{s+2}, \ldots, v_n\}$.

The main idea behind the proof is that if $X$ contains a Steiner node from $L_1$, then it can be removed from $X$.

If $X$ contains Steiner nodes from $L_1$, then each of them must be a leaf node incident to a special node by an edge in $\text{mst}(G[S_1 \cup X])$ as $G_1$ is complete. The weight of this edge is 10. Removing any one of such Steiner nodes from $X$ will result in a solution $X'$ whose fitness value is 10 less than that of $X$. Similar to the OneMax analysis from Droste et al. (2002), the $(1+1)$ EA accepts the mutations where the number of Steiner nodes from $L_1$ contained in $X$ is decreased, and the number of such Steiner nodes contained in $X$ can be decreased by at least one in such mutations. So, at most $r$ such mutations will make $X$ contain no Steiner nodes from $L_1$. If there are $i$ Steiner nodes coming from $L_1$ contained in $X$, then the probability of occurring such mutations is
at least \((\frac{1}{2})^k (1 - \frac{1}{2})^{r-1}\). So, the upper bound of the expected time until \(X\) contains no Steiner nodes from \(L_1\) is \(\sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i (1 - \frac{1}{2})^{r-1} = O(r \ln r)\).

Now \(X\) contains no Steiner node from \(L_1\). If \(X\) contains Steiner node \(v_{i+1}\), then it is the global optimum. If \(X\) contains neither Steiner node \(v_{i+1}\) nor Steiner node from \(L_1\), then there are two cases to be considered. The first is that \(s = 2\). In this case, \(X\) is also a global optimum. The second is that \(2 < s \leq n\). In this case, the global optimum will be found by adding Steiner node \(v_{i+1}\) to \(X\). The probability of this event is \(\Omega(\frac{1}{r})\), which implies the expected time is \(O(r)\).

Altogether, the global optimum of \(G_1\) can be found by the (1+1) EA starting with any initial solution in expected runtime \(O(r \ln r)\). \(\square\)

For instance \(G_1\), MSTA can achieve only a \(2(1 - \frac{1}{2})\)-approximation solution; however, the (1+1) EA can efficiently find its global optimum. Therefore, the (1+1) EA outperforms MSTA on instance \(G_1\).

### 4.2. An Instance Where the (1+1) EA is Superior to AWH

This subsection compares the (1+1) EA with AWH described in Algorithm 4 on an instance which we call \(G_2\) in this article. This instance is proposed by Waxman and Imase (1988) to show that for any \(\varepsilon > 0\), the weight of the Steiner tree found by AWH for \(G_2\) is larger than \((2 - \varepsilon)\) times the weight of the minimum weight Steiner tree of \(G_2\).

For the sake of clarity, we first give some concepts related to instance \(G_2\).

In a tree, if the minimum number of edges that must be visited from the root node to a node is \(i\), then we say that the node is in layer \(i\). Clearly, the root node is in layer 0. A perfect binary tree is a tree where all leaf nodes are in the bottom-most layer and each non-leaf node has two children. If a node \(v_f\) has two children \(v_l\) and \(v_r\), then node \(v_f\) is called the father node of \(v_l\) and \(v_r\), and \(v_l\) (respectively \(v_r\)) is called the brother node of \(v_f\) (respectively \(v_r\)). A perfect binary tree of height \(k\) refers to a perfect binary tree in which the minimum number of edges from the root node to a leaf node is \(k\). Denote by \(T_k\) a perfect binary tree of height \(k\). Thus, there are \(k + 1\) layers in \(T_k\): layer 0, layer 1, \(\ldots\), and layer \(k\), and the number of nodes in layer \(i\) (\(0 \leq i \leq k\)) is \(2^i\). Altogether, there are \(2^{k+1} - 1\) nodes including all leaf nodes and all non-leaf nodes in \(T_k\).

We now describe instance \(G_2\). Given a positive integer number \(k\), \(G_2 = (V_2, E_2)\) is a perfect binary tree \(T_k\) with a path connecting all leaf nodes. In \(G_2\), the set \(S_2\) of special nodes contains all leaf nodes, and the set \(R_2 := V_2 \setminus S_2\) of Steiner nodes contains all non-leaf nodes. Therefore, in \(G_2\) there are \(2^k\) special nodes and \(2^k - 1\) Steiner nodes, i.e., \(r = 2^k - 1\). All nodes in \(V_2\) are numbered starting from the root node layer by layer, and from left to right: \(v_0, v_1, v_2, \ldots, v_{2^{k+1} - 2}\). Figure 2 shows an example with \(k = 3\).

In this article, a subtree of \(G_2\) refers to a perfect binary subtree of \(T_k\), which contains all the nodes and edges branching downwards from a given node (the root node of the subtree) till the leaf nodes, and we denote it by the root node of the subtree. For example, when \(k = 3\), subtree \(v_1\) is the perfect binary tree over Steiner nodes \(v_1, v_3, v_4\), and special nodes \(v_7, v_8, v_9, v_{10}\), where \(v_1\) is its root node. Let \(v_i\) be a non-leaf node in layer \(k - h\) (\(1 \leq h \leq k\)), then the weight of the two edges leaving \(v_i\) downward is \(2^h - \frac{1}{2} + \delta\) (\(0 < \delta < 1\)), and the weight of the bottom edge connecting the two subtrees of \(v_i\) is \(2^{h+1} - 2\).

Let \(w_i(h)\) be the weight of a subtree of height \(1 \leq h \leq k\), which equals the sum of weights on each edge in the subtree, and let \(w_p(h)\) be the weight of the path connecting all leaf nodes of a subtree of height \(1 \leq h \leq k\), which is the sum of weights of all edges on this path. Clearly, \(w_i(h) = h2^h + (2^{h+1} - 2)\delta\), and \(w_p(h) = 2(h2^h - 2^h + 1)\).
The minimum weight Steiner tree of $G_2$ is the perfect binary tree $T_k$. As shown in Figure 2, the solid edges construct the minimum weight Steiner tree of $G_2$ with weight of $k2^k + (2^{k+1} - 2)\delta$, which covers all special nodes and all Steiner nodes. The dashed edges construct the tree that may be produced by AWH, whose weight is $2(2k^2 - 2k + 1)$.

While AWH may be trapped in the local optimum which contains no Steiner nodes, we will show that the (1+1) EA can efficiently find the minimum weight Steiner tree of $G_2$.

The following analysis will utilize the drift theorem which is described in Lemma 2.

**Lemma 2 (Drift theorem) (He and Yao, 2001):** Let $(X_t; t = 0, 1, 2, \ldots)$ be a Markov chain associated with an EA on a given optimization problem, and define a potential function $d : \{X_t; t = 0, 1, 2, \ldots\} \rightarrow \mathbb{R}^+ \cup \{0\}$. Let $T$ be the random variable that denotes the first point in time $t \in \{0, 1, 2, \ldots\}$ for which $d(X_t) = 0$. For any $t \geq 0$ and any $X_t$ with $d(X_t) > 0$, if the following inequality holds,

$$E[d(X_t) - d(X_{t+1}) | X_t] \geq c_{\text{low}},$$

where $c_{\text{low}} > 0$, then we have

$$E[T | X_0] \leq d(X_0)/c_{\text{low}}.$$

First consider the case where $0 < \delta < \frac{1}{3}$. In this case, $w_i(h) < w_p(h)$ for all $h \in \{2, \ldots, k\}$, i.e., the height of a subtree such that the weight of this subtree is less than the weight on the path connecting all its leaf nodes must be at least 2.

**Theorem 3:** For $G_2$ with $0 < \delta < \frac{1}{3}$, the (1+1) EA starting with any initial solution finds the global optimum in expected runtime $O(r^4 \ln r)$.

**Proof:** We first analyze the expected time that a feasible solution is found by the (1+1) EA starting with any initial solution. Then we utilize the drift theorem to derive the expected time until the global optimum is found once a feasible solution has been constructed.

Let $X$ be the current solution.

If the number of connected components in the subgraph induced by $S_2 \cup X$ is greater than 1, that is, $c(G[S_2 \cup X]) > 1$, then there must exist a connected component consisting of only Steiner node(s). Otherwise, all connected components contain special nodes, then these connected components can be connected by some proper edges on
the path; that is, the number of connected components is 1, which contradicts the assumption that the number of connected components is greater than 1.

Such a connected component is either a tree consisting of Steiner nodes or only one isolated Steiner node. If it is the former, then removing a Steiner node which is a leaf node in this tree results in a cheaper tree. If it is the latter, removing the unique Steiner node deletes this component. Altogether, removing some Steiner nodes will decrease the fitness value, which can be accepted by the (1+1) EA. The probability of this event is $\frac{1}{2}(1 - \frac{1}{r})^{r-1} = \Omega(\frac{1}{r})$, which implies that the expected time is $O(r)$. A connected component consisting of only Steiner node(s) will be deleted in expected time $O(r^2)$, as the number of Steiner nodes contained in such a connected component is $O(r)$. When all connected components only consisting of Steiner nodes are deleted, a feasible solution will be finally constructed. Since the number of such connected components is $O(r)$, a feasible solution will be found in expected time $O(r^3)$.

Next, we utilize the drift theorem to derive the expected time for the (1+1) EA to find the global optimum starting from any feasible solution.

Let $d(X_t) = w(X_t) - w_{opt}$, where $X_t$ is the feasible solution after $t$ iterations, $w(X_t)$ is the weight of the minimum spanning tree of the subgraph induced by $S_2 \cup X_t$; that is, $w(X_t) = w(\text{mst}(G[X_t \cup S_2]))$ and $w_{opt}$ is the weight of the minimum spanning tree of the subgraph induced by $S_2$ and the global optimum. If $d(X_T) = 0$, then the global optimum has been found and $T$ is the time to find the global optimum.

We first estimate the upper bound of $d(X_0)$. We have $d(X_0) = w(X_0) - w_{opt} \leq (w_p(k) + w_i(k)) - w_{opt} = w_p(k)$, since the weight of the minimum spanning tree of the subgraph induced by $S_2 \cup X_0$ is not larger than the sum of weights on all edges of the input graph; that is, $w(X_0) \leq w_p(k) + w_i(k)$, and the weight of the minimum spanning tree of the subgraph induced by $S_2$ and the global optimum is $w_i(k)$, that is, $w_{opt} = w_i(k)$. Thus, $d(X_0) \leq 2k(2^k - 2^k + 1) < 2k2^k$ as $k$ is a positive integer number, that is, $k \geq 1$.

Then, we estimate the lower bound of $E[d(X_t) - d(X_{t+1})|X_t]$. Note that $d(X_t) - d(X_{t+1}) = w(X_t) - w(X_{t+1})$, and $w(X_t) - w(X_{t+1})$ is always nonnegative as long as $X_t$ is not the global optimum, since $w(X_{t+1}) \leq w(X_t)$ holds according to the acceptance condition of the (1+1) EA.

All feasible solutions of $G_2$ can be partitioned into two sets: $C$ and its complement $\overline{C}$. $C$ is the set of feasible solutions that do not contain at least one Steiner node in layer $k - 1$, and $\overline{C}$ is the set of feasible solutions that contain all Steiner nodes in layer $k - 1$.

If $X_i \in C$, then at least one Steiner node in layer $k - 1$ is not contained in $X_i$. Assume that $v$ is such a Steiner node, and denote its father node and brother node by $v_f$ and $v_b$, respectively. Now we consider the subtree $v_f$ whose height is 2.

Since $v$ is not contained in $X_i$, not all the edges from subtree $v_f$ are contained in $\text{mst}(G[S \cup X_i])$. Therefore, the weight of $X_i$ can be reduced by constructing subtree $v_f$. There are four cases that need to be considered with respect to whether $v_f$ and $v_b$ are contained in $X_i$. Among these four cases, the worst case is that neither $v_f$ nor $v_b$ is contained in $X_i$, since in this case for constructing subtree $v_f$ three nodes $v, v_f,$ and $v_b$ should be simultaneously added. This means that among these four cases, the lower bound of the probability of constructing subtree $v_f$ is $(\frac{1}{2})^3(1 - \frac{1}{r})^{-3}$ on the other hand, among these four cases, the lower bound of the reduction of $w(X_i)$ is $2 - 6\delta$, which is the reduction of $w(X_i)$ in the cases where $v_b$ is not contained in $X_i$.

If $X_i \in \overline{C}$, then all Steiner nodes in layer $k - 1$ are contained in $X_i$. Since $X_i$ is not the global optimum, there is at least one Steiner node not contained in $X_i$. Assume that among all such Steiner nodes $v'$ is in the largest layer, say layer $i \in [0, k - 2]$; that is, Steiner nodes in layers from $i + 1$ to $k - 1$ are all contained in $X_i$. Then, $w(X_i)$ can be
decreased by $2^{k-i} - 2 - 2\delta$ through constructing a subtree of height $k - i$ by adding Steiner node $v'$ to connect two subtrees of height $k - i - 1$, i.e., $w(X_i) - w(X_{i+1}) = 2^{k-i} - 2 - 2\delta$ with probability $\frac{1}{r}(1 - \frac{1}{r})^{r-1}$.

Clearly, $2^{k-i} - 2 - 2\delta \geq 2 - 6\delta$ as $i \in [0, k - 2]$, and $(\frac{1}{r})^3(1 - \frac{1}{r})^{r-3} \leq \frac{1}{r}(1 - \frac{1}{r})^{r-1}$.

Then, we have

$$E[d(X_i) - d(X_{i+1})|X_i] = E[w(X_i) - w(X_{i+1})|X_i]$$

$$\geq (2 - 6\delta)(\frac{1}{r})^3(1 - \frac{1}{r})^{r-3}$$

$$\geq (2 - 6\delta)\frac{1}{er^3}.$$  

By Lemma 2, and note that $2^k = r - 1$, we have

$$E[T|X_0] \leq 2k2^k er^3/(2 - 6\delta) \in O(r^4 \ln r).$$

Altogether, the global optimum of $G_2$ will be found in expected runtime $O(r^3 + r^4 \ln r)$.

Next, consider the case where $\frac{1}{3} \leq \delta < \frac{5}{7}$. In this case, $w_i(h) < w_p(h)$ for all $h \in \{3, \ldots, k\}$, that is, the height of a subtree such that the weight of this subtree is less than the weight on the path connecting all its leaf nodes should be at least 3.

Note that feasible solution $X_t$ belongs to either set $C$ or its complement $\overline{C}$, where $C$ is the set of feasible solutions that do not contain at least one Steiner node in layer $k - 1$ or layer $k - 2$, and $\overline{C}$ is the set of feasible solutions that contain all Steiner nodes of layer $k - 1$ and layer $k - 2$.

If $X_i \in C$, a subtree of height 3 can be constructed by simultaneously adding at most 7 Steiner nodes, and $w(X_i) - w(X_{i+1})$ can be reduced by at least 10. While if $X_i \in \overline{C}$, adding a Steiner node that is in the largest layer among all Steiner nodes not contained in $X_i$, say layer $i \in [0, k - 3]$, reduces $w(X_i)$ by $2^{k-i} - 2 - 2\delta$.

Similar to the proof of Theorem 3, we have the following theorem.

**Theorem 4:** For $G_2$ with $\frac{1}{3} \leq \delta < \frac{5}{7}$, the (1+1) EA starting with any initial solution finds the global optimum in expected runtime $O(r^3 \ln r)$.

Finally consider the case where $\frac{5}{7} \leq \delta < 1$. In this case, $w_i(h) < w_p(h)$ for all $h \in \{4, \ldots, k\}$, that is, the height of a subtree such that the weight of this subtree is less than the weight on the path connecting all its leaf nodes should be at least 4.

Note that feasible solution $X_t$ belongs to either set $C$ or its complement $\overline{C}$, where $C$ is the set of feasible solutions that do not contain all Steiner nodes in layers from $k - 3$ to $k - 1$, and $\overline{C}$ is the set of feasible solutions that contain all Steiner nodes in layers from $k - 3$ to $k - 1$. If $X_i \in C$, a subtree of height 4 can be constructed by simultaneously adding at most 15 Steiner nodes, and $w(X_i) - w(X_{i+1})$ can be reduced by at least 34. While if $X_i \in \overline{C}$, adding a Steiner node that is in the largest layer among all Steiner nodes not contained in $X_i$, say layer $i \in [0, k - 4]$, reduces $w(X_i)$ by $2^{k-i} - 2 - 2\delta$.

Similar to the proof of Theorem 3, we have the following theorem.

**Theorem 5:** For $G_2$ with $\frac{5}{7} \leq \delta < 1$, the (1+1) EA starting with any initial solution finds the global optimum in expected runtime $O(r^{16} \ln r)$.

Theorems 3, 4, and 5 show that the (1+1) EA can efficiently find the global optimum for $G_2$ with $0 < \delta < 1$; however, AWH may be trapped in the local optimum of $G_2$. Therefore, the (1+1) EA is superior to AWH on $G_2$. 

718 Evolutionary Computation Volume 25, Number 4
In the proofs of Theorems 3, 4, and 5, we utilize the drift theorem and define the potential function as the difference between the fitness values of a feasible solution and the global optimum. It seems that these three theorems can also be proved by the fitness level method. However, defining the concrete fitness levels may be tedious in this instance. Thus, the utilization of the drift theorem is justified.

4.3. An Instance Where the (1+1) EA May Need Expected Exponential Optimization Runtime

In this subsection, we construct an instance called $G_3$ for which the expected optimization runtime of the (1+1) EA may be exponential.

Given three constant numbers $d(d > 0)$, $w(w > 0)$, and $t(t \geq 2)$ such that $w - (2t - 1)d > 0$, thus $w - d > 0$, $G_3 = (V_3, E_3)$, where $V_3 = S_3 \cup R_3$, $S_3 = \{v_1, \ldots, v_{i+2}\}$, and $R_3 = \{u_1, \ldots, u_t, u_{t+1}, \ldots, u_{2t+2}\}$. Obviously, the total number of Steiner nodes in $G_3$ is $2t + 2$; that is, $r = 2t + 2$. As shown in Figure 3, we construct $G_3$ in four steps. First, each pair of adjacent special nodes $v_i$ and $v_{i+1}$ ($i = 1, 2, \ldots, t + 1$) is connected by an edge $(v_i, v_{i+1})$, whose weight is $2w$. Second, each pair of adjacent Steiner nodes $u_i$ and $u_{i+1}$ ($i = 1, 2, \ldots, t - 1$) is connected by an edge $(u_i, u_{i+1})$, whose weight is $w - (2i + 1)d$, and each pair of adjacent Steiner nodes $u_i$ and $u_{i+1}$ ($i = t + 1, t + 2, \ldots, 2t + 1$) is connected by an edge $(u_i, u_{i+1})$, whose weight is $w + 2(i - t)d$. Third, $u_1$ and $u_i$ are respectively incident to $v_1$ by edge $(u_1, v_1)$ of weight $w - d$ and incident to $v_{i+2}$ by edge $(u_i, v_{i+2})$ of weight $w - td$, and Steiner node $u_i$ ($i = 1, 2, \ldots, t$) is incident to special node $v_{i+1}$ by edge $(u_i, v_{i+1})$ of weight $w + 2i d$. Finally, Steiner node $u_i$ ($i = t + 1, t + 2, \ldots, 2t + 2$) is incident to special node $v_{i-1}$ by edge $(u_i, v_{i-1})$ of weight $w$. The solid edges construct the minimum weight Steiner tree of $G_3$, which covers all special nodes and all Steiner nodes $X^*_3 = \{u_1, u_2, \ldots, u_t\}$, and the weight of the minimum weight Steiner tree is $(2t + 1)w$. Therefore, the optimal solution of $G_3$ is $X^*_3$.

The (1+1) EA need an exponential expected runtime to find the optimal solution for $G_3$ when it starts with the all-zeros solution.

**Theorem 6**: For $G_3$, the (1+1) EA starting with the all-zeros solution finds the optimal solution in expected runtime $\Omega(r^{\frac{r}{2}})$.

**Proof**: Let $R'_3 \subseteq R_3$. Note that the fitness value of the all-zeros solution is $(2t + 2)w$. If $R'_3 \neq X^*_3$, then adding all Steiner nodes in $R'_3$ to the all-zeros solution cannot be accepted by the (1+1) EA, as the weight of the minimum spanning tree of $G[S_3 \cup R'_3]$ is not less than the fitness value of the all-zeros solution. However, when all Steiner nodes in $X^*_3$
are added, the weight of the minimum spanning tree of \( G[S_3 \cup X_3^*] \) is \((2t + 1)w\), which is less than the fitness value of the all-zeros solution. So the (1+1) EA accepts only the event of simultaneously adding all Steiner nodes in \( X_3^* \) to the all-zeros solution. The probability of this event is \((1)\frac{1}{2}(1 - \frac{1}{2})^{2r + 2 - t} = O((\frac{1}{r})^r) = O((\frac{1}{r})^{2-1})\), which implies the expected runtime is \( \Omega(r^{2-1}) \).

Starting with any initial solution, the expected optimization runtime of the (1+1) EA may also be exponential.

**Theorem 7:** For \( G_3 \), the (1+1) EA starting with any initial solution finds the optimal solution in expected runtime \( O(r^{2-1}) \).

**Proof:** We first prove that starting with any initial solution, the Steiner nodes from \( R_3 \setminus X_3^* \) can be removed by the (1+1) EA in expected time \( O(r \ln r) \). Then we prove that the optimal solution could be found by the (1+1) EA in expected time \( O(r^{2-1}) \).

Let \( X \) be the current solution, then \( X \) is a feasible solution, as each Steiner node can be connected to a special node by an edge in the input graph. This is to say, \( c(G[S_3 \cup X]) = 1 \).

If \( X \) contains Steiner nodes from \( R_3 \setminus X_3^* \), then each of them must be a leaf node connecting with a special node in \( mst(G[S_3 \cup X]) \) by an edge of weight \( w \). The fitness value can be decreased by \( w \) through removing any one of such Steiner nodes from \( X \). So, the (1+1) EA accepts the mutations which decrease the number of Steiner nodes in \( X \) coming from \( R_3 \setminus X_3^* \). Since the number of such Steiner nodes in \( X \) will be decreased by at least one in such mutations, at most \( r \) such mutations will make \( X \) contain no Steiner nodes from \( R_3 \setminus X_3^* \). If there are \( i \) Steiner nodes coming from \( R_3 \setminus X_3^* \) contained in \( X \), then the probability of such mutations occurring is at least \((\frac{1}{r})^r(1 - \frac{1}{r})^{-1} \). Therefore, the expected time until \( X \) contains no Steiner nodes from \( R_3 \setminus X_3^* \) is \( \sum_{i=1}^{\infty}(\frac{1}{r})^r(1 - \frac{1}{r})^{-1} = O(r \ln r) \).

After removing from \( X \) all Steiner nodes which come from \( R_3 \setminus X_3^* \), if \( X \) now contains all Steiner nodes from \( X_3^* \), then it is already the optimal solution. Otherwise, the optimal solution can be found by the (1+1) EA from \( X \) in one step with probability at least \((\frac{1}{r})^r(1 - \frac{1}{r})^{-1} = \Omega((\frac{1}{r})^r) \). Therefore, the expected time is \( O(r^r) = O(r^{2-1}) \).

Theorems 6 and 7 show that the (1+1) EA may need expected exponential optimization runtime to find the optimal solution of \( G_3 \).

5 Conclusions

We investigate the performance of evolutionary algorithms for Steiner tree problems in this article. We reveal that the (1+1) EA achieves an approximation ratio of \( \frac{1}{2} \) for the metric STP problem in a special class of quasi-bipartite graphs, which are also NP-hard. This exemplifies that evolutionary algorithms are good approximation algorithms for NP-hard combinatorial optimization problem, though they are randomized heuristic algorithms. While the Steiner tree problems are NP-hard, we find that the (1+1) EA efficiently finds the global optima of two GSTP instances where other heuristics may be trapped in the local optima. However, on one constructed instance we show that the (1+1) EA may need an exponential expected optimization runtime, which implies that the (1+1) EA may not be always efficient for GSTP.

A question is whether the (1+1) EA can achieve an approximation ratio for the metric STP problem in the special class of quasi-bipartite graphs better than that we have obtained by simulating the iterated 1-Steiner heuristic, after all it is a randomized
heuristic. For the STP problem in general graphs, we still know nothing about the approximation performance of the (1+1) EA.

Another question is that we theoretically know little about the performance of the (1+1) EA on other cases of STP, such as rectilinear Steiner tree problems until now.

Population-based EAs are used in practice, which use not only mutation operator but also crossover. The performance of population-based EAs on combinatorial optimization problems is recently a hot topic (Chen et al., 2009; Jansen et al., 2005; Chen et al., 2012; Doerr et al., 2012). Hence, the analysis on the performance of population-based EAs for the STP problem will be another interesting work.

Acknowledgments

The authors thank the anonymous reviewers for their constructive feedback and valuable comments. This article is supported by the National Natural Science Foundation of China (grant nos. 61472143 and 61562071), the Scientific Research Special Plan of Guangzhou Science and Technology Programme (grant no. 201607010045), and the Natural Science Foundation of Jiangxi Province (grant nos. 20151BAB217008 and 20151BAB207020).

References


Performance Analysis of Evolutionary Algorithms for Steiner Tree Problems


