On the Hardness of Full Steiner Tree Problems*

Ahmad Biniaz† Anil Maheshwari† Michiel Smid†

October 23, 2014

Abstract

Given a weighted graph $G = (V, E)$ and a subset R of V, a Steiner tree in G is a tree which spans all vertices in R. The vertices in $V \setminus R$ are called Steiner vertices. A full Steiner tree is a Steiner tree in which each vertex of R is a leaf. The *full Steiner tree* problem is to find a full Steiner tree with minimum weight. The *bottleneck full Steiner tree* problem is to find a full Steiner tree which minimizes the length of the longest edge. The *k-bottleneck full Steiner tree* problem is to find a bottleneck full Steiner tree with at most k Steiner vertices. The *smallest full Steiner tree* problem is to find a full Steiner tree with the minimum number of Steiner vertices.

We show that the full Steiner tree problem in general graphs cannot be approximated within a factor of $O(\log^{2-\varepsilon}|R|)$ for any $\varepsilon > 0$. We also provide a polynomial-time approximation factor preserving reduction from the full Steiner tree problem to the group Steiner tree problem. Based on that, the first approximation algorithm for the full Steiner tree problem in general graphs is obtained. Moreover, we show that the same hardness result holds for the node-weighted version of the full Steiner tree problem. We prove that it is NP-hard to approximate the k-bottleneck full Steiner tree problem within a factor of $2 - \varepsilon$. The smallest full Steiner tree problem is shown to be NP-complete and does not admit any polynomial-time $O((1 - \varepsilon)\ln n)$-approximation algorithm. The presented reductions show the connection between the full Steiner tree, the group Steiner tree, and the connected set cover problems. In addition, we present an $O(|E|\log|V|)$ time algorithm for the bottleneck full Steiner tree problem which relaxes the assumption, that G is a complete graph, in Chen et al. [7] algorithm.

1 Introduction

Given a graph $G = (V, E)$, a subset R of V, and a weight function $w : E \to \mathbb{R}^+$, a *Steiner tree* of G is a tree which spans all vertices in R. The vertices in R are called *terminals*; these vertices must be in the tree. The vertices in $S = V \setminus R$ are called *Steiner vertices*. A *full Steiner tree* is defined as a Steiner tree in which each vertex of R is a leaf. The *full Steiner*

*Research supported by NSERC.

†School of Computer Science, Carleton University, Ottawa, Canada.

tree (FST) problem is to find a full Steiner tree T with minimum weight $w(T) = \sum_{e \in T} w(e)$. This problem is NP-hard, and cannot be approximated within a factor of $(1 - \varepsilon) \ln n$ in polynomial-time [9].

The *bottleneck full Steiner tree* (BFST) problem is to find a full Steiner tree T such that the weight of the longest edge in T is minimized. We refer to the weight of the longest edge in T as its *bottleneck*. This problem can be solved exactly in polynomial-time [7]. However, the *k-bottleneck full Steiner tree* (k-BFST) problem which is to find a bottleneck full Steiner tree which contains at most k Steiner vertices is NP-hard [1].

Another variant of the Steiner tree problem is the *node-weighted Steiner tree* (NST) problem. Given a weight function $w : V \to \mathbb{R}^+$ on the vertex set V, we are looking for a minimum node-weighted Steiner tree T, where $w(T) = \sum_{v \in T} w(v)$. Since any Steiner tree contains all vertices in R, we may set $w(r) = 0$, for all nodes $r \in R$. This problem cannot be approximated within a factor of $(1 - \varepsilon) \ln n$ in polynomial-time (personal communication with Berman; see the reference 2 in [20]). Symmetrically, the *node-weighted full Steiner tree* (NFST) problem is defined to compute a full Steiner tree with minimum node-weight.

For an unweighted graph G, the *smallest full Steiner tree* (SFST) problem is to compute a full Steiner tree with the minimum number of Steiner vertices. Clearly, the SFST problem is a special case of the NFST problem, where all the Steiner vertices have the same weight.

We show a connection between the (node-weighted) full Steiner tree problems and the (node-weighted) group Steiner tree problems. In addition, we show the connection between the smallest full Steiner tree problem and the connected set cover problem. We define these problems in Section 1.1. Based on that, we give new lower bounds on the inapproximablility of the full Steiner tree problems, as well as the first approximation algorithms.

1.1 Preliminaries

Set Cover: Given a finite set U of elements, a family $\mathcal{S}$ of subsets of U, the set cover (SC) problem is to find a smallest subset $\mathcal{R}$ of $\mathcal{S}$ such that every element of U is covered by at least one set in $\mathcal{R}$. We denote an instance of the SC problem by $(U, \mathcal{S})$.

Connected Set Cover: Given a finite set U of elements, a family $\mathcal{S}$ of subsets of U, a graph G on vertex set $\mathcal{S}$, the connected set cover (CSC) problem is to find a smallest subset $\mathcal{R}$ of $\mathcal{S}$ such that every element of U is covered by at least one set in $\mathcal{R}$, and the subgraph of G induced by $\mathcal{R}$ is connected. We denote an instance of the CSC problem by $(U, \mathcal{S}, G)$. Clearly, by taking G to be a complete graph, the SC problem is a special case of the CSC problem.

In an extension of the CSC problem, which is known as the weighted connected set cover (WCSC) problem, there are real weights assigned to the elements of $\mathcal{S}$, and we are looking for a connected set cover which minimizes the total weight.

Group Steiner Tree: Given an edge-weighted graph $G(V, E)$ with weight function $w : E \to \mathbb{R}^+$ and a set $\mathcal{G} = \{g_1, g_2, \cdots, g_k\}$, where each g_i is a subset of V. Each such subset g_i is called a "group" of terminals. The group Steiner tree (GST) problem is to find a minimum weight Steiner tree in G that contains at least one terminal from each group.

Another variant of the group Steiner tree problem is the *node-weighted group Steiner tree* (NGST) problem. Given a weight function $w : V \rightarrow \mathbb{R}^+$ on the vertex set V, we are looking for a minimum node-weighted Steiner tree that contains at least one terminal from each group. We denote an instance of a weighted group Steiner tree problem by $(G, \mathcal{G}, w)$.

Directed Steiner Tree: This is the directed version of the (undirected) Steiner tree problem. Given a directed graph $G = (V, E)$ with a weight function $w : E \rightarrow \mathbb{R}^+$, a root vertex $r \in V$, a subset R of V, the directed Steiner tree (DST) problem is to find a minimum weight out-branching tree in G, rooted at r, that spans all vertices in R.

1.2 Previous Work

Drake and Hougardy [9] showed that approximating the FST problem is at least as hard as approximating the set cover problem, and there is no polynomial time approximation algorithm for the FST problem with performance ratio better than $\ln n$ unless NP = $\tilde{\mathrm{D}}$.[1] When the input graph is metric, i.e., it is a complete graph and edge weights satisfy the triangle inequality, constant factor approximation algorithms are presented in [6, 7, 9, 13, 18, 21, 22]. Biniaz et al. [3] presented a 20-approximation algorithm for the FST problem in unit disk graphs.

Chen et al. [7] presented an $O(m \log n)$ time algorithm which solves the BFST problem in a complete graph G, where n is the number of vertices and $m = \Theta(n^2)$ is the number of the edges of G. The geometric version of this problem, where V is a set of points in the plane, G is a complete graph over V, and the edge weights are equal to the Euclidean distances between the points, was considered in [1]. Biniaz et al. [4] presented an algorithm which solves the Euclidean BFST problem optimally in $\Theta(n \log n)$ time. However, Abu-Affash [1] proved that the k-BFST problem is NP-hard, and the geometric version of this problem cannot be approximated better than $\sqrt{2}$, unless P=NP. For the metric version, he presented an approximation algorithm which computes a k-BFST with bottleneck at most four times the optimal bottleneck. He mentioned the improvement of the approximation ratio as an open problem. He also left an interesting version of the k-BFST as an open problem: Given a desired bottleneck, we are looking for a full Steiner tree which minimizes k to achieve that bottleneck. In this paper we show some results on the inapproximablility of these problems.

In 1991, Berman (personal communication; see the reference 2 in [20]) proved that unless P=NP, the NST problem cannot be approximated better than $\ln n$ by giving an approximation-preserving reduction from the set cover problem [12]. Asymptotically optimal approximation algorithms with performance ratio of $O(\ln n)$ are presented in [20, 16]. Zou et al. [26] gave a constant factor approximation algorithm for the NST problem in unit disk graphs. Naor et al. [23] considered an *online* version of the NST problem where G is a (node and edge) weighted graph which is known in advance, and the terminals appear on-

[1]Here we use $\tilde{\mathrm{D}}$ to denote the complexity class deterministic quasi-polynomial time, or DTIME($n^{\mathrm{poly} \log n}$), where DTIME(t) is the class of languages that have a deterministic algorithm that runs in time t.

line. They presented a polynomial-time randomized online algorithm for this problem with a competitive ratio of $O(\log n \log^2 k)$, where k is the number of terminals.

Halperin and Krauthgamer [17] proved that the GST problem cannot be approximated within a factor of $O(\log^{2-\varepsilon} |\mathcal{G}|)$, unless $\tilde{\mathrm{Z}} \supseteq \mathrm{NP}$.[2] Garg et al. [15] presented the first polylogarithmic approximation algorithm for the GST problem. They use a randomized algorithm that solves the GST problem on trees with the approximation ratio of $O(\log k \log N)$ where k is the number of groups and N is the size of the largest group. For a general graph G, they find a group Steiner tree of cost within $O(\log n \log\log n \log k \log N)$ of the cost of the best group Steiner tree by reducing the problem to the case where G is a tree and then applying the results of Bartal [2]. The improved approximation factor of $O(\log^2 n \log k)$ can be achieved by applying the results of [11]. This result can even be improved to get an algorithm with approximation factor $O(\log n \log^2 k)$ (personal communication with Gupta, Halperin, Kortsarz, Krauthgamer, Ravi, Srinivasan, and Wang; see the reference 14 in [17]).

The NGST problem cannot be approximated within a factor of $O(\log^{2-\varepsilon} n)$; this follows from the result in [10] where WCSC and NGST problems are shown to be equivalent and WCSC problem is $\Omega(\log^{2-\varepsilon} n)$-hard.

Demaine et al. [8] showed that there is a polynomial-time 6-approximation algorithm for the node-weighted Steiner tree problem in planar graphs. In general they proved that node-weighted Steiner trees have polynomial-time $O(1)$-approximation algorithms for any family of graphs that exclude a fixed minor. They also presented an $O(\log n \operatorname{poly}\log\log n)$ approximation algorithm for the special case where G is an embedded planar graph and each group is the set of nodes on a face.

Khandekar et al. [19] studied the *fault-tolerant* versions of the group Steiner tree problem. Given an edge or node-weighted graph G, a root vertex r, a set $\mathcal{G}$ of groups, the goal is to find a minimum weight subgraph of G that contains two edge or vertex-disjoint paths from each group in $\mathcal{G}$ to r. They showed that this problem cannot be approximated within a factor of $O(\log^{2-\varepsilon} k)$, where k is the number of groups. In addition, they presented a polynomial-time $O(\sqrt{n}\log n)$-approximation algorithm for this problem. The *online* version of the NGST problem is also considered in [23].

Halperin and Krauthgamer [17] proved that the DST problem cannot be approximated within a factor of $O(\log^{2-\varepsilon} n)$, unless $\tilde{\mathrm{Z}} \supseteq \mathrm{NP}$. Charikar et al. [5] presented an $O(i^2(i-1)k^{1/i})$ approximation algorithm for the DST problem running in $O(n^i k^{2i})$ time for any $i > 1$, where $k = |R|$. It achieves an $O(k^\varepsilon)$ approximation ratio in polynomial time for any $\varepsilon > 0$, and $O(\log^3 k)$ approximation ratio in quasi-polynomial time. Similar results are also obtained in [24].

The SC problem is NP-hard [14], and cannot be approximated within a factor of $(1 - \varepsilon)\ln n$, unless $\tilde{\mathrm{D}} \supseteq \mathrm{NP}$ [12]. It is obvious that the SC problem is a special case of the CSC problem, where the input graph G is complete. Thus, the CSC problem is NP-hard and cannot be approximated within a factor of $(1 - \varepsilon)\ln n$ in polynomial-time. Shuai and Hu

[2]Here we use $\tilde{\mathrm{Z}}$ to denote the complexity class Las-Vegas quasi-polynomial time, or ZTIME($n^{\operatorname{poly}\log n}$), where ZTIME($t$) denote the class of languages that have a probabilistic algorithm that runs in expected time t (with zero error probability).

Table 1: Summary of results (note that the set cover problem is a special case of the connected set cover problem where the input graph is complete).

Problem	**Inapproximability**	**Reference**
set cover	$(1-\varepsilon)\ln n$	[12]
connected set cover	$(1-\varepsilon)\ln n$	
weighted connected set cover	$\log^{2\text{-}\varepsilon} n$	[10]
group Steiner tree	$\log^{2\text{-}\varepsilon} \lvert\mathcal{G}\rvert$	[17]
directed Steiner tree	$\log^{2\text{-}\varepsilon} n$	[17]
full Steiner tree	$(1-\varepsilon)\ln n$	[9]
	$\log^{2\text{-}\varepsilon} \lvert R\rvert$	Theorem 1
node-weighted full Steiner tree	$\log^{2\text{-}\varepsilon} \lvert R\rvert$	Theorem 2
k-bottleneck full Steiner tree	$2-\varepsilon$	Theorem 5
smallest full Steiner tree	$(1-\varepsilon)\ln n$	Theorem 6

[25] showed that even when G is a spider graph (a graph with exactly one vertex of degree greater than two), the CSC problem is $(1-\varepsilon)\ln n$-inapproximable, unless $\tilde{\mathrm{D}} \supseteq \mathrm{NP}$. In that case, they presented a $(1+\ln n)$-approximation algorithm. In the case where all the vertices of G have degree at most two, they presented polynomial-time algorithms.

Elbassioni et al. [10] showed that the CSC problem is equivalent to the GST problem with all edge weights set to 1. Hence, by applying the results of Garg et al. [15] on the equivalent GST problem, a polylogarithmic approximation algorithm for CSC is obtained.

1.3 Our Results

Motivated by the open problems mentioned in [1] we present some results on the hardness of the FST, NFST, k-BFST, and SFST problems. The presented reductions show the relation between the full Steiner tree, the group Steiner tree, and the connected set cover problems.

In Section 2 we show that the FST problem cannot be approximated within a factor of $O(\log^{2\text{-}\varepsilon} |R|)$ unless $\tilde{\mathrm{Z}} \supseteq \mathrm{NP}$, using a reduction from the group Steiner tree problem. Then, we present a polynomial time approximation factor preserving reduction from the FST problem to the group Steiner tree and the directed Steiner tree problems. This leads to the first approximation algorithms for the FST problem in general graphs. In addition, we show that the similar results are achievable for the node-weighted FST problem. In Section 3, we first introduce the *largest* full Steiner tree problem in a given graph G, and present an algorithm which solves this problem in $O(n+m)$ time. Then, we show how to use this algorithm to solve the BFST problem in G in $O(m \log n)$ time. In addition we reduce the connected set cover problem to the metric version of the k-BFST problem and prove that it is NP-hard to approximate the k-BFST problem within a factor of $2-\varepsilon$, for any $\varepsilon > 0$. Using a reduction from the connected set cover problem, in Section 4 we show that the SFST problem is NP-complete and cannot be approximated within a factor of $(1-\varepsilon)\ln n$ unless $\tilde{\mathrm{D}} \supseteq \mathrm{NP}$. Table 1 summarizes the inapproximability results. We conclude this paper in Section 5.

2 Full Steiner Trees

Given a graph $G = (V, E)$, a subset R of V, and a weight function $w : E \to \mathbb{R}^+$, we are interested to find a full Steiner tree T with minimum weight. This problem cannot be approximated better than $\ln n$ unless NP = $\tilde{\mathrm{D}}$ [9]. In this section we give a tighter bound on the inapproximability of the FST problem, as well as the first approximation algorithm for this problem in general graphs. Since in any full Steiner tree all the terminals are leaves, we assume w.l.o.g. that there are no edges between the vertices of R. In [17], Halperin and Krauthgamer proved that the GST problem cannot be approximated within a factor of $O(\log^{2\text{-}\varepsilon} |\mathcal{G}|)$, unless $\tilde{\mathrm{Z}} \supseteq$ NP. Using a reduction from the GST problem, we show that the FST problem cannot be approximated within a factor of $O(\log^{2\text{-}\varepsilon} |R|)$. For a tree T, we define the *skeleton tree* of T, as a tree obtained by removing the leaves of T.

Theorem 1. *There is no polynomial-time $O(\log^{2\text{-}\varepsilon} |R|)$-approximation for the FST problem for any $\varepsilon > 0$ unless $\tilde{Z} \supseteq$ NP.*

Proof. Using contradiction, suppose that there is a polynomial-time $O(\log^{2\text{-}\varepsilon} |R|)$-approximation algorithm for the FST problem.

We reduce the GST problem to the FST problem in the following way. Let $(G_{gst}, \mathcal{G}, w)$ be an instance of the GST problem, where $G_{gst} = (V_{gst}, E_{gst})$. We construct an instance (G_{fst}, R, w) of the FST problem as follows.

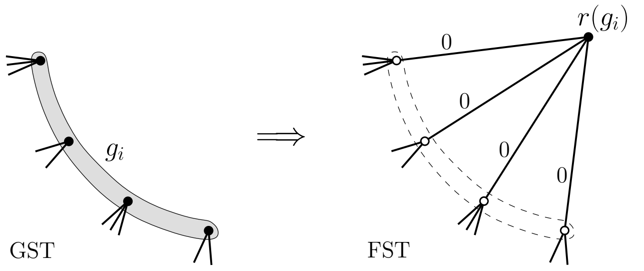

Figure 1: Reducing an instance of the GST problem to an instance of the FST problem. Solid circles show the terminals and light circles show the Steiner vertices.

For each group $g_i \in \mathcal{G}$, create a new vertex $r(g_i)$. Let R denote the set of these new vertices:

$$R = \{r(g_i) : g_i \in \mathcal{G}\}.$$

Let $E_i = \{(u, r(g_i)) : u \in g_i\}$ and $E = \bigcup_{g_i \in \mathcal{G}} E_i$. For each edge $e \in E$ let $w(e) = 0$. Let $G_{fst} = (V_{fst}, E_{fst})$, where $V_{fst} = V_{gst} \cup R$ and $E_{fst} = E_{gst} \cup E$. This gives an instance (G_{fst}, R, w) of the FST problem. See Figure 1. Since $|R| = |\mathcal{G}|$, for simplicity of notation we define $k = |R|$.

First, we show that the weight of an optimal solution for the FST problem is equal to the weight of an optimal solution for the GST problem. Let T^*_{fst} be an optimal solution for the FST problem. Let T^*_{gst} be the skeleton tree of T^*_{fst}, obtained by removing the leaves of T^*_{fst}. Note that in T^*_{fst} all leaves are terminals (and vice versa), and the terminals are incident on the edges of weight zero, thus, $w(T^*_{gst}) = w(T^*_{fst})$. We prove that T^*_{gst} is an optimal solution for the GST problem. Each group g_i in $\mathcal{G}$ has a corresponding vertex in R. In T^*_{fst}, each vertex $r(g_i) \in R$ is connected to T^*_{gst} via a vertex belonging to g_i. That is, at least one vertex from each $g_i \in \mathcal{G}$ is covered by the tree T^*_{gst}. Thus, T^*_{gst} is a group Steiner tree. We prove the optimality of T^*_{gst} by contradiction. Suppose that there is a solution T_{gst} for the GST problem, where $w(T_{gst}) < w(T^*_{gst})$. For each $g_i \in \mathcal{G}$, let $u_i \in g_i$ be a vertex that is covered by T_{gst}. By connecting u_i to $r(g_i)$ we obtain a full Steiner tree T_{fst} for the FST problem. Since $w(u_i, r(g_i)) = 0$ for all $g_i \in \mathcal{G}$, we have $w(T_{fst}) = w(T_{gst})$. Thus, $w(T_{fst}) = w(T_{gst}) < w(T^*_{gst}) = w(T^*_{fst})$, which contradicts the optimality of T^*_{fst}. Therefore, $w(T_{gst}) \geq w(T^*_{gst})$, which implies that T^*_{gst} is an optimal solution for the GST problem.

Now, let T^a_{fst} be the polynomial-time $O(\log^{2\text{-}\varepsilon} k)$-approximation solution for the FST problem. Then

$$w(T^a_{fst}) \leq O(\log^{2\text{-}\varepsilon} k) \cdot w(T^*_{fst}).$$

Let T^a_{gst} be the skeleton tree of T^a_{fst}. It is obvious that T^a_{gst} is a group Steiner tree, and $w(T^a_{gst}) = w(T^a_{fst})$. Therefore,

$$w(T^a_{gst}) \leq O(\log^{2\text{-}\varepsilon} k) \cdot w(T^*_{fst}) = O(\log^{2\text{-}\varepsilon} k) \cdot w(T^*_{gst}),$$

that is, T^a_{gst} is a polynomial-time $O(\log^{2\text{-}\varepsilon} k)$-approximation tree for the GST problem, which means that NP has quasi-polynomial Las-Vegas algorithms. □

Now, we prove that there is a polynomial-time approximation factor preserving reduction from the full Steiner tree problem to the group Steiner tree problem.

Lemma 1. *Any polynomial time α-approximation algorithm for the GST problem yields a polynomial time α-approximation algorithm for the FST problem.*

Proof. We will transform, in polynomial time, an instance (G_{fst}, R, w) of the FST problem, where $G_{fst} = (V_{fst}, E_{fst})$ and $S = V_{fst} \setminus R$, to an instance $(G_{gst}, \mathcal{G}, w)$ of the GST problem as follows. For each node $r \in R$, let $N(r)$ denote the adjacent vertices of r in G_{fst}. For each vertex $u \in N(r)$, create a new vertex $r(u)$. Let g_r denote the set of these new vertices:

$$g_r = \{r(u) : u \in N(r)\}.$$

We consider g_r to be one "group". Let $\mathcal{G} = \{g_r : r \in R\}$. Let $E = \{(u, r(u)) : r \in R, u \in N(r)\}$, and for each edge $(u, r(u)) \in E$ let $w(u, r(u)) = w(u, r)$ where $r(u) \in g_r$. Let $G_{gst} = (V_{gst}, E_{gst})$, where $V_{gst} = S \cup \{r(u) : r(u) \in g_r, g_r \in \mathcal{G}\}$ and $E_{gst} = G_{fst}[S] \cup E$, where $G_{fst}[S]$ denotes the subgraph of G_{fst} induced by vertex set S. This gives an instance $(G_{gst}, \mathcal{G}, w)$ of the GST problem. See Figure 2.

First, we show that the weight of an optimal solution for the GST problem is equal to the weight of an optimal solution for the FST problem. Let T^*_{gst} be an optimal solution for the GST problem. Note that the groups in $\mathcal{G}$ are pairwise disjoint, and the vertices in each group

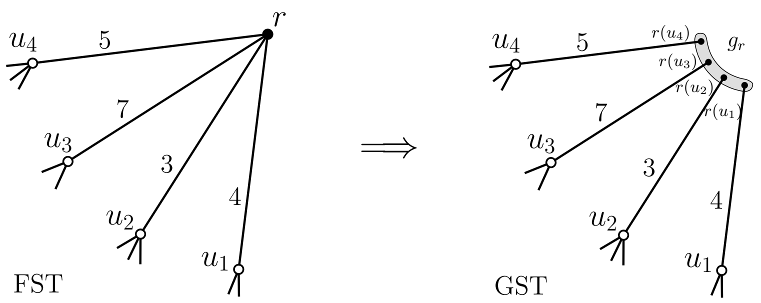

Figure 2: Reducing an instance of the FST problem to an instance of the GST problem. Solid circles show the terminals and light circles show the Steiner vertices.

have degree one. Thus, T^*_{gst} contains exactly one vertex $r(u)$ from each group g_r, where $r(u)$ is a leaf. Let T^*_{fst} be the tree obtained from T^*_{gst} by replacing each edge $(u, r(u)) \in T^*_{gst}$, where $r(u) \in g_r$, with its corresponding edge $(u, r) \in G_{fst}$. Clearly, T^*_{fst} is a full Steiner tree in G_{fst}, and $w(T^*_{fst}) = w(T^*_{gst})$ because $w(u, r) = w(u, r(u))$. Now, we prove that T^*_{fst} is an optimal full Steiner tree. Using contradiction, suppose that there is a solution T_{fst} for the FST problem where $w(T_{fst}) < w(T^*_{fst})$. Consider the skeleton tree T of T_{fst}. Recall that each group $g_r \in \mathcal{G}$ corresponds to a terminal $r \in R$. For a terminal $r \in R$, let u be the neighbor of r in T. Recall that for each $u \in N(r)$ there is a vertex $r(u) \in g_r$ where $w(u, r) = w(u, r(u))$. Let T_{gst} be the tree obtaining from T by connecting $r(u)$ to u for all $r \in R$. That is, T_{gst} is a group Steiner tree, and $w(T_{gst}) = w(T_{fst})$. Thus, $w(T_{gst}) = w(T_{fst}) < w(T^*_{fst}) = w(T^*_{gst})$, which contradicts the optimality of T^*_{gst}. Therefore, $w(T_{fst}) \geq w(T^*_{fst})$, which implies that T^*_{fst} is an optimal solution for the FST problem.

Now, let T^a_{gst} be an α-approximation for the GST problem. As described above we can obtain a full Steiner tree T^a_{fst} for the FST problem where $w(T^a_{fst}) = w(T^a_{gst})$. Therefore,

$$w(T^a_{fst}) = w(T^a_{gst}) \leq \alpha \cdot w(T^*_{gst}) = \alpha \cdot w(T^*_{fst}),$$

and hence T^a_{fst} is an α-approximation for the FST problem. □

By Lemma 1, the improved version (see the reference 14 in [17]) of Garg et al. [15] algorithm can be used to solve the FST problem (by reducing the FST problem to the equivalent GST problem first). Hence, an $O(\log n \log^2 |R|)$-approximation algorithm for the FST problem is obtained. Moreover, as a direct consequence of the following lemma, the polynomial-time $O(k^\varepsilon)$-approximation and the quasi-polynomial-time $O(\log^3 k)$-approximation algorithms of Charikar et al. [5] for the directed Steiner tree problem, carries over to the full Steiner tree problem, where $k = |R|$.

Lemma 2. *Any polynomial time α-approximation algorithm for the DST problem yields a polynomial time α-approximation algorithm for the FST problem.*

Proof. We will transform an instance of the FST problem on graph G to an instance of the DST problem with the same number of terminals. Replace each Steiner edge (an edge both of whose endpoints are Steiner vertices) in G with a pair of anti-parallel directed edges of the same weight. Then replace each terminal edge (s, r), where $s \in S$ and $r \in R$, with a directed edge of the same weight from s to r. Let G' be the resulting directed graph. It is easy to verify that an out-going directed Steiner tree originating from a Steiner node to any terminal in G' can be converted to a full Steiner tree of the same weight in G. Let T^*_{dst} be the minimum weight directed Steiner tree among all the directed Steiner trees in G' obtained by picking any of the Steiner vertices as the root vertex. Clearly, the weight of an optimal full Steiner tree T^*_{fst} in G is at least the weight of T^*_{dst}, i.e., $w(T^*_{fst}) \geq w(T^*_{dst})$.

Now, consider an α-approximation algorithm $\mathcal{A}_{dst}$ for the DST problem. Let T^a_{dst} be the minimum weight tree among all the trees obtained by running $\mathcal{A}_{dst}$ on all the Steiner vertices of G'. Let T^a_{fst} be the tree obtained from T^a_{dst} by replacing the directed edges with undirected edges of the same weight. It is obvious that T^a_{fst} is a full Steiner tree in G and $w(T^a_{fst}) = w(T^a_{dst})$. Therefore,

$$w(T^a_{fst}) = w(T^a_{dst}) \leq \alpha \cdot w(T^*_{dst}) \leq \alpha \cdot w(T^*_{fst}),$$

which implies that T^a_{fst} is an α-approximation for the FST problem. □

2.1 Node Weighted Full Steiner Trees

Given a graph $G = (V, E)$, a subset R of V, and a weight function $w : V \to \mathbb{R}^+$, we are interested in finding a full Steiner tree T with minimum node weight. Since any full Steiner tree contains all the vertices in R, we assign w.l.o.g. $w(r) = 0$ for each r in R. In this section we show that the same inapproximability results presented in Section 2 for the FST problem, hold for the NFST problem, using a reduction from the node-weighted GST problem. The NGST problem cannot be approximated within a factor of $O(\log^{2\text{-}\varepsilon} n)$; this follows from the result in [10] where WCSC and NGST problems are equivalent and WCSC problem is $\Omega(\log^{2\text{-}\varepsilon} n)$-hard.

Theorem 2. *There is no polynomial-time $O(\log^{2\text{-}\varepsilon} n)$-approximation for the NFST problem for any $\varepsilon > 0$ unless $\tilde{Z} \supseteq NP$.*

Proof. We reduce the NGST problem to the NFST problem. Given an instance of the NGST problem, we construct an instance (G_{fst}, R, w) of the NFST problem with the same node weight function, as described in the proof of Theorem 1. For each $r(g_i)$ in R corresponding to a group $g_i \in \mathcal{G}$ we assign $w(r_i) = 0$. Each solution T_{nfst} for the NFST problem gives a solution T_{ngst} for the NGST problem of the same weight, where T_{ngst} is the skeleton tree of T_{nfst}. Using the same analysis as in the proof of Theorem 1, the statement of the lemma holds. □

Using the same reduction as described in the proof of Lemma 1, and by assigning weight zero to each node $r(u) \in g_r$, we have the following lemma. Note that for each terminal r, we have $w(r) = 0$.

Lemma 3. *Any polynomial time α-approximation algorithm for the NGST problem yields a polynomial time α-approximation algorithm for the NFST problem.*

3 Bottleneck Full Steiner Trees

Given a graph $G = (V, E)$, a subset R of V, and a weight function $w : E \to \mathbb{R}^+$, the *bottleneck full Steiner tree* (BFST) problem is to find a full Steiner tree T_{bot} in G such that the weight of the maximum-weight edge in T_{bot} is minimized. This problem can be solved exactly in polynomial time [1, 4, 7]. Chen et al. [7] presented an $O(m \log n)$ time algorithm that solves the BFST problem in a complete graph. Their algorithm maintains a forest F of the full Steiner trees and a tree which contains the bottleneck edge. Initially, for each Steiner vertex u, they add to F a star T_u centered at u which contains all vertices of R as its leaves. Each star is a full Steiner tree. Let T^* be the tree in F with minimum bottleneck. Then, they consider the Steiner edges iteratively in the increasing order of their weights. If the addition of a Steiner edge (p, q) reduces the current bottleneck, they add (p, q) to F—which results in merging the trees containing p and q—and update T^*. In order to merge the trees, disjoint-set operations are used. Since in each iteration F is a forest of full Steiner trees, the input to their algorithm is a complete graph, thus, the running time of their algorithm is $O(n^2 \log n)$. We relax the assumption that G is a complete graph, and present an algorithm for the BFST problem running in $O(m \log n)$ time. The geometric version of the BFST problem is also of interest; in [1], the authors presented an $O(n \log^2 n)$ time algorithm for this problem which was improved to $O(n \log n)$ [4].

The k-BFST problem is to find a bottleneck full Steiner tree which contains at most k Steiner vertices. This problem is NP-hard. For metric graphs, a 4-approximation algorithm is presented by Abu-Affash [1].

The rest of this section is organized as follows. In Section 3.1 we define the largest full Steiner tree problem, and present a linear time algorithm for this problem. In Section 3.2 we show how to use the largest full Steiner tree algorithm to solve the BFST problem in $O(m \log n)$ time. In Section 3.3 we show that it is NP-hard to approximate the k-BFST problem in metric graphs within a factor of two.

3.1 Largest Full Steiner Tree Problem

Let $G = (V, E)$ be a graph with n vertices and m edges. Given a terminal vertex set $R \subset V$; in the case that G does not have a full Steiner tree which spans all the terminals in R, we define a *largest full Steiner tree* as a full Steiner tree in G with maximum cardinality. The *cardinality* of a tree T is the number of terminal vertices in T, i.e., $car(T) = |T \cap R|$. The *largest full Steiner tree* (LFST) problem is to find a largest full Steiner tree.

We present an algorithm which computes a largest full Steiner tree T_{max} in G in $O(n+m)$ time. Algorithm 1 receives a graph G and a set R as input and returns a largest full Steiner tree. It computes a forest F containing all the maximal full Steiner trees of G. A full Steiner tree T in G is *maximal* if by adding any vertex of G (terminal or Steiner) to T, the resulting tree is not a full Steiner tree. A largest full Steiner tree in G is a maximal tree T_{max} in F which has the maximum cardinality (recall that the cardinality of a tree is defined as the number of its terminal vertices). Now we describe how to compute F. We run a modified version of the depth first search algorithm (ModifiedDFS) on the Steiner vertices, and add the tree returned by the ModifiedDFS to F. Note that the DFS algorithm explores the graph from a given vertex by first exploring all the children of that vertex. To avoid looping through cycles, DFS marks each vertex upon first visiting it and does not explore the children of a previously visited vertex. In the ModifiedDFS algorithm, we mark all the terminal vertices as "visited" in advance, and start exploring G from a Steiner vertex, say u. That is, while exploring G, if we see a Steiner vertex s, we visit and explore s, but, if we see a terminal vertex r, we visit r but we do not explore it. Since the ModifiedDFS does not explore the terminal vertices, the resulting tree, say T_u, is a full Steiner tree in G. If G contains some unvisited Steiner vertices, we repeatedly run the ModifiedDFS on an unvisited Steiner vertex. See Algorithm 1.

Algorithm 1 LFST(G, R)

```
Input: an undirected graph $G(V, E)$, and a set $R \subset V$.
Output: a largest full Steiner tree $T_{max}$ in $G$.
1: $S \leftarrow V \backslash R$
2: $F \leftarrow \emptyset$
3: while $S$ is not empty do
4:     $u \leftarrow$ a vertex of $S$
5:     $T_u \leftarrow$ ModifiedDFS$(G, u)$
6:     $F \leftarrow F \cup \{T_u\}$
7:     $S \leftarrow S \setminus S(T_u)$
8: $T_{max} \leftarrow$ a tree in $F$ with the maximum cardinality
9: return $T_{max}$
```

For a full Steiner tree T we define $R(T)$ and $S(T)$ as the set of terminals and Steiner vertices of T, respectively. In addition, we define $T(R)$ as the set of all terminal edges in T (recall that a terminal edge is an edge incident on a terminal vertex), and $T(S)$ as the *skeleton tree* obtained from T by removing $T(R)$. Actually, $T(S)$ is the induced subgraph of T by $S(T)$, and contains all the Steiner edges of T. For two vertices p and q in G, we say that p is *accessible* from q if there is path in G between p and q which does not contain any terminal vertex as an internal node. It is obvious that in an undirected graph G, the accessibility of p and q is symmetric.

Observation 1. *In a full Steiner tree T, each terminal in $R(T)$ is accessible from each Steiner vertex in $S(T)$.*

Observation 2. *Consider a Steiner vertex $u \in G$ and the tree T_u obtained by running the* ModifiedDFS *on u. Then, $R(T_u) \cup S(T_u)$ is the set of all vertices accessible from u in G.*

Theorem 3. *The algorithm* LFST *returns a largest full Steiner tree of G in $O(m)$ time.*

Proof. The ModifiedDFS subroutine in algorithm LFST, does not explore any terminal vertex, and hence the resulting tree T_u does not contain any terminal vertex as an internal node. Thus, T_u is a full Steiner tree in G. Now we prove the maximality of T_u. While exploring a Steiner vertex u, the ModifiedDFS visits all the unvisited vertices (terminals and Steiners) adjacent to u, and explore all the unexplored Steiner vertices. By this argument and by Observation 2 the ModifiedDFS visits all the vertices accessible from u, and hence T_u is a maximal full Steiner tree in G. In addition, we repeatedly run ModifiedDFS on unvisited Steiner vertices. According to the symmetry of the accessibility in G, any choice of u in line 4, in any order, results in the same full Steiner trees. Therefore, after the while loop, F contains all the maximal full Steiner trees of G.

Now we show that the tree T_{max} returned by the algorithm LFST has the maximum cardinality. Let T be a full Steiner tree in G. Consider the skeleton tree $T(S)$ with the vertex set $S(T)$. Let u be the first vertex of $S(T)$ that is visited by ModifiedDFS. Clearly, $R(T)$ is accessible by u. In algorithm LFST, let T_u be the tree in F which contains u. By Observation 1, T_u contains all the terminals accessible from u. Thus, $R(T) \subseteq R(T_u)$, and hence, $car(T) \leq car(T_u)$. In line 9, T_{max} is a tree in F with maximum cardinality, then, $car(T_u) \leq car(T_{max})$. Therefore, $car(T) \leq car(T_{max})$, implies that T_{max} is a maximum full Steiner tree of G.

Now we analyze the time complexity of the algorithm. In the while loop, when we run the ModifiedDFS starting at a Steiner vertex u, it visits all the vertices that are accessible from u, but explores only the Steiner vertices. In the successor iterations of the while loop, the Steiner vertices accessible by u are never visited again. Thus, LFST explores each Steiner vertex exactly once. The total number of times that the ModifiedDFS visits a vertex (terminal or Steiner vertex) is at most the number of edges incident on that vertex. Therefore, the total running time of the algorithm is $O(n + m)$. □

3.2 Bottleneck Full Steiner Tree Problem

In this section we are looking for a bottleneck full Steiner tree T_{bot} in G, i.e., a largest full Steiner tree in which the weight of the longest edge is minimized. We combine the binary search algorithm with the LFST algorithm—presented in Section 3.1—to compute T_{bot} in $O(m \log n)$ time. We define the cardinality of a graph G, $car(G)$, as the cardinality of a maximum full Steiner tree in G. Using the LFST algorithm, we compute $car(G)$ in $O(n+m)$ time. The first step is to sort the edges by weight. This takes $O(m \log n)$ time. Let $e_1, e_2, \ldots, e_m$ be the edges of G, in nondecreasing order by their weight. Let $G_i = (V, E_i)$ be the graph with vertex set V and edge set E_i, where $E_i = \{e_1, e_2, \ldots, e_i\}$. Then, we use binary search to find the minimum value i^* of i such that G_{i^*} has the same cardinality as G, i.e., $car(G_{i^*}) = car(G)$. Therefore, the largest full Steiner tree in G_{i^*} is a bottleneck full Steiner tree in G. In each iteration of the binary search we run the algorithm LFST to compute a maximum full Steiner tree in G_i with cardinality $car(G_i)$ and test whether the "guess" i for i^* is too high or too low. The LFST algorithm takes $O(n + m)$ time per execution and the total number of times we execute LFST is $O(\log n)$, thus, the total running time of this algorithm is $O((n + m) \log n)$. W.l.o.g. one may only consider a connected component of G which contains all the terminals, that is $n = O(m)$, which results in the following theorem.

Theorem 4. *The BFST problem can be solved exactly in $O(m \log n)$ time.*

3.3 k-Bottleneck Full Steiner Tree Problem

In this section we show that it is NP-hard to approximate the k-BFST problem in metric graphs within a factor of two. This is achieved by providing a reduction from the connected set cover problem, which is known to be NP-complete.

Connected Set Cover: Given a finite set U of elements, a family $\mathcal{S}$ of subsets of U, a graph G on vertex set $\mathcal{S}$, does there exist a set $\mathcal{S}^* \subseteq \mathcal{S}$, such that $|\mathcal{S}^*| \leq k$ and the subgraph $G[\mathcal{S}^*]$ is a connected cover?

Theorem 5. *It is NP-hard to approximate the metric version of the k-BFST problem within a factor of $2 - \varepsilon$, for any $\varepsilon > 0$.*

Proof. We present a reduction from the CSC problem. Consider an instance $(U, \mathcal{S}, G_{csc}, k)$ of the CSC problem, where $G_{csc} = (V_{csc}, E_{csc})$ and $V_{csc} = \mathcal{S}$. We construct an instance (G_{fst}, R, w, k) of the metric k-BFST problem, such that G_{csc} has a connected set cover of size at most k if and only if G_{fst} has a full Steiner tree with at most k Steiner vertices and bottleneck at most $2 - \varepsilon$, for some $\varepsilon > 0$.

Let $R = \{u : u \in U\}$, $V_{fst} = \mathcal{S} \cup R$, and $E = E_{csc} \cup \{(S, u) : S \in \mathcal{S}, u \in S\}$. Let $G_{fst} = (V_{fst}, E_{fst})$ be the complete graph over V_{fst}. For each edge $e \in E$, let $w(e) = 1$ if $e \in E$, and $w(e) = 2$, otherwise. This gives an instance of the metric k-BFST problem. Note that the vertices in G_{csc} are analogous to Steiner vertices in G_{fst}. The induced subgraph of G_{fst} by $\mathcal{S}$ which contains the edges of weight 1 is isomorphic to $G_{csc}[\mathcal{S}]$.

Let $\mathcal{S}^* \subseteq \mathcal{S}$ be a connected vertex cover of size at most k in G_{csc}. Clearly, $G_{fst}[\mathcal{S}^*]$ is connected. Let T be a spanning tree in $G_{fst}[\mathcal{S}^*]$. By connecting each $u \in R$ to one of its neighbors in T we obtain a full Steiner tree T^*_{fst} in G_{fst} with at most k Steiner vertices and the length of each edge is exactly 1.

Conversely, suppose that there exists a full Steiner tree T^a_{fst} in G_{fst} with at most k Steiner vertices and bottleneck at most $2 - \varepsilon$. It is obvious that T^a_{fst} does not contain any edge in $E_{fst} \setminus E$. Let $\mathcal{S}^a$ denote the set of the Steiner vertices of T^a_{fst}, and let T denote the skeleton tree of T^a_{fst} which contains only the edges of weight 1. Clearly, $G_{csc}[\mathcal{S}^a]$ is connected. Each element $u \in U$ corresponding to a terminal vertex in R, is covered by at least one set in $\mathcal{S}^a$, and hence $\mathcal{S}^a$ is a connected set cover in G_{csc}. On the other hand, $|\mathcal{S}^a| = k$, thus, $\mathcal{S}^a$ is a solution for the CSC problem. $\square$

4 Smallest Full Steiner Trees

Abu-Affash [1] introduced a variant, or dual version, of the k-BFST as an open problem. Given a fixed desired bottleneck λ, we are interested in finding a full Steiner tree with the minimum number of Steiner points to achieve this bottleneck. In this section, we show that this problem is NP-complete and $\Omega((1 - \varepsilon) \ln n)$-hard, for all $\varepsilon > 0$. Without loss of generality, we can remove all edges of weight greater than λ from the input graph, and look for the full Steiner tree with the minimum number of Steiner vertices in the resulting graph. Hence, we define the *smallest full Steiner tree* (SFST) problem to compute a full Steiner tree which contains the smallest number of Steiner vertices. In the SFST problem we can assume that the input graph G is unweighted, or all edges of G have equal weight. W.l.o.g. assume that G is a *unit-weighted graph*, where all edge weights are set to 1.

Lemma 4. *An optimal solution for the FST problem on a unit-weighted graph G can be reconstructed from an optimal solution of the SFST problem on G, and vice versa.*

Proof. Consider a unit-weight instance (G, R) of the FST problem. Consider any full Steiner tree T of G. Partition the edges of T into two sets A and B, where A is the set of all edges incident to terminals in R, and B is the set of all edges in the skeleton tree of T. Clearly, $|A| = |R|$, and

$$w(T) = w(A) + w(B) = |R| + |B| - 1.$$

Since $w(A)$ is fixed for all full Steiner trees of G, an optimal full Steiner tree in G (say T^*) minimizes $w(B)$, which is the number of edges in B, and consequently it is the number of Steiner vertices of T^*. Therefore, to compute an optimal full Steiner tree in G, one can compute a smallest full Steiner tree in G, and vice versa. □

To prove the hardness of the problem, we present a reduction from the connected set cover problem. First, we introduce the decision version of the SFST problem: Given a graph $G = (V, E)$ and a subset R of V, does there exist a full Steiner tree in G which contains at most k Steiner vertices?

Lemma 5. *The smallest full Steiner tree problem is NP-complete.*

Proof. It is easy to see that smallest full Steiner tree problem belongs to NP. Now, we present a reduction from the connected set cover problem. Consider an instance $(U, \mathcal{S}, G_{csc}, k)$ of the CSC problem, where $G_{csc} = (V_{csc}, E_{csc})$ and $V_{csc} = \mathcal{S}$. We construct an instance (G_{sfst}, R, k) of the SFST problem, such that G_{csc} has a connected set cover of size at most k if and only if G_{sfst} has a full Steiner tree with at most k Steiner vertices.

Define $R = \{u : u \in U\}$, $G_{sfst} = (V_{sfst}, E_{sfst})$, where $V_{sfst} = \mathcal{S} \cup R$ and $E_{sfst} = E_{csc} \cup \{(S, u) : S \in \mathcal{S}, u \in S\}$. This gives an instance of the SFST problem. Note that the vertices in G_{csc} are analogous to Steiner vertices in G_{sfst}, and the induced subgraphs by $\mathcal{S}$ are isomorphic, i.e., $G_{sfst}[\mathcal{S}] \cong G_{csc}[\mathcal{S}]$. Let T be a full Steiner tree with k Steiner vertices in G_{sfst}. It is obvious that the skeleton tree of T (obtained by removing its leaves) is a connected set cover of size k for the CSC problem. On the other hand, if $\mathcal{S}^*$ is a connected set cover of size k in G_{csc}, then any spanning tree of $G_{csc}[\mathcal{S}^*]$ is the skeleton tree of a full Steiner tree with k Steiner vertices in G_{sfst}. Therefore, G_{csc} has a connected set cover of size at most k, if and only if, G contains a full Steiner tree with at most k Steiner vertices. □

By the reduction in the proof of Lemma 5, we have the following corollary.

Corollary 1. *There is a polynomial-time approximation factor preserving reduction from the smallest full Steiner tree problem to the connected set cover problem.*

Thus, the inapproximability results for the CSC problem hold for the SFST problem.

Theorem 6. *There is no polynomial-time* $(1-\varepsilon)\ln n$*-approximation algorithm for the SFST problem for any* $\varepsilon > 0$ *unless* $\tilde{D} \supseteq NP$*.*

As a direct result of Lemma 4 and Theorem 6 we have the following corollary.

Corollary 2. *There is no polynomial-time* $(1-\varepsilon)\ln n$*-approximation algorithm for the FST problem on a unit-weighted graph, for any* $\varepsilon > 0$*, unless* $\tilde{D} \supseteq NP$*.*

5 Conclusion

We considered full Steiner tree problems and presented their hardness results. We proved that the full Steiner tree problem is $\Omega(\log^{2\text{-}\varepsilon}|R|)$-hard, and the node-weighted full Steiner tree problem is $\Omega(\log^{2\text{-}\varepsilon} n)$-hard. We also presented approximation factor preserving reductions from the FST and NFST problems to the GST and NGST problems, respectively. We presented an $O(|E|\log|V|)$ time algorithm for the bottleneck full Steiner tree problem; which relaxes the assumption in Chen et al. [7] algorithm that the input graph is complete. As for the k-bottleneck full Steiner tree problem we showed that it cannot be approximated within a factor $2-\varepsilon$ on metric graphs. We introduced the unweighted (or smallest) full Steiner tree problem and proved that this problem is NP-complete, and $\Omega((1-\varepsilon)\ln n)$-hard. We also presented an approximation factor preserving reduction from this problem to the connected set cover problem. The presented reductions in this paper, show the connection between the full Steiner tree, the group Steiner tree, and the connected set cover problems.

References

[1] A. K. Abu-Affash. The Euclidean bottleneck full Steiner tree problem. To appear in *Algorithmica.*

[2] Y. Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 161–168, 1998.

[3] A. Biniaz, A. Maheshwari, and M. Smid. Approximating full Steiner tree in a unit disk graph. In *the Proceedings of the 26th Canadian Conference in Computational Geometry (CCCG 2014)*, 2014.

[4] A. Biniaz, A. Maheshwari, and M. Smid. An optimal algorithm for the Euclidean bottleneck full Steiner tree problem. *Computational Geometry: Theory and Applications*, 47(3):377–380, 2014.

[5] M. Charikar, C. Chekuri, T.-Y. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. *J. Algorithms*, 33(1):73–91, 1999.

[6] Y. H. Chen. An improved approximation algorithm for the terminal Steiner tree problem. In *ICCSA (3)*, pages 141–151, 2011.

[7] Y. H. Chen, C. L. Lu, and C. Y. Tang. On the full and bottleneck full Steiner tree problems. In *COCOON*, pages 122–129, 2003.

[8] E. D. Demaine, M. Hajiaghayi, and P. N. Klein. Node-weighted Steiner tree and group Steiner tree in planar graphs. In *ICALP (1)*, pages 328–340, 2009.

[9] D. E. Drake and S. Hougardy. On approximation algorithms for the terminal Steiner tree problem. *Inf. Process. Lett.*, 89(1):15–18, 2004.

[10] K. M. Elbassioni, S. Jelic, and D. Matijevic. The relation of connected set cover and group Steiner tree. *Theor. Comput. Sci.*, 438:96–101, 2012.

[11] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

[12] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[13] B. Fuchs. A note on the terminal Steiner tree problem. *Inf. Process. Lett.*, 87(4):219–220, 2003.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[15] N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000.

[16] S. Guha and S. Khuller. Improved methods for approximating node weighted Steiner trees and connected dominating sets. *Inf. Comput.*, 150(1):57–74, 1999.

[17] E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *STOC*, pages 585–594, 2003.

[18] S.-Y. Hsieh and W.-H. Pi. On the partial-terminal Steiner tree problem. In *ISPAN*, pages 173–177, 2008.

[19] R. Khandekar, G. Kortsarz, and Z. Nutov. Approximating fault-tolerant group-Steiner problems. *Theor. Comput. Sci.*, 416:55–64, 2012.

[20] P. N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. *J. Algorithms*, 19(1):104–115, 1995.

[21] G.-H. Lin and G. Xue. On the terminal Steiner tree problem. *Inf. Process. Lett.*, 84(2):103–107, 2002.

[22] F. V. Martinez, J. C. de Pina, and J. Soares. Algorithms for terminal Steiner trees. *Theor. Comput. Sci.*, 389(1-2):133–142, 2007.

[23] J. Naor, D. Panigrahi, and M. Singh. Online node-weighted Steiner tree and related problems. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 210–219, 2011.

[24] T. Rothvoß. Directed Steiner tree and the Lasserre hierarchy. *CoRR*, abs/1111.5473, 2011.

[25] T. Shuai and X.-D. Hu. Connected set cover problem and its applications. In *AAIM*, pages 243–254, 2006.

[26] F. Zou, X. Li, S. Gao, and W. Wu. Node-weighted Steiner tree approximation in unit disk graphs. *J. Comb. Optim.*, 18(4):342–349, 2009.