On Full Steiner Trees in Unit Disk Graphs

Ahmad Biniaz* Anil Maheshwari* Michiel Smid*

April 24, 2014

Abstract

Given an edge-weighted graph $G = (V, E)$ and a subset R of V, a Steiner tree of G is a tree which spans all the vertices in R. A full Steiner tree is a Steiner tree which has all the vertices of R as its leaves. The full Steiner tree problem is to find a full Steiner tree of G with minimum weight. In this paper we consider the full Steiner tree problem when G is a unit disk graph. We present a 20-approximation algorithm for the full Steiner tree problem in G. As for λ-precise unit disk graphs we present a $(10 + \frac{1}{\lambda})$-approximation algorithm, where λ is the length of the shortest edge in G.

1 Introduction

Given a graph $G = (V, E)$ of n vertices with a weight function $w : E \to \mathbb{R}^+$ on edges and a subset R of V. The vertices in R are called the *terminals* and the vertices in $V \backslash R$ are called *Steiner vertices* (usually denoted by S, i.e., $S = V \backslash R$); see Figure 1(a). A *Steiner tree* of G is a tree which contains all the vertices in R; see Figure 1(b). The weight of a tree T is defined as the sum of the weights of all the edges in T; i.e., $w(T) = \sum_{e \in T} w(e)$. The *Steiner tree problem* is to find a Steiner tree T of G with minimum weight [13]. This problem is known to be MAX SNP-hard [1, 2]. Robins and Zelikovsky [17] presented a 1.55-approximation algorithm for this problem. The approximation ratio was improved to $\ln(4) + \varepsilon < 1.39$ by Byrka et al. [3].

Motivated by the reconstruction of evolutionary trees in biology [15] and VLSI global routing and telecommunications [14], a *full Steiner tree* of G is defined as a Steiner tree which has all the vertices of R as its leaves; see Figure 1(c). The *full Steiner tree problem* is to find a full Steiner tree T of G with minimum weight. This problem is also known as the *terminal Steiner tree problem*. In a full Steiner tree problem one may assume that G does not have any edge between the vertices of R.

A *metric graph* is defined as a complete graph, whose edge weights satisfy the *triangle inequality*, i.e., for any three vertices x, y, and z, $w(x, y) \leq w(x, z) + w(y, z)$ [7, 20]. Lin and Xue [14] showed that the full Steiner tree problem for metric graphs is NP-complete

*School of Computer Science, Carleton University, Ottawa, Canada. Research supported by NSERC.

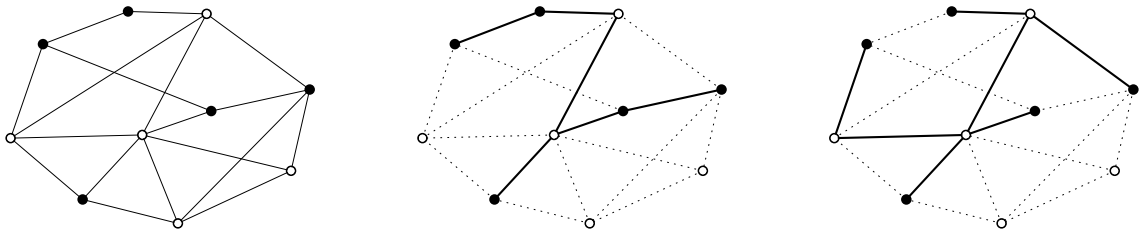


Figure 1: (a) The input graph G; terminals are indicated by filled circles, non-terminals by non-filled circles. (b) A Steiner tree and (c) a full Steiner tree of G.

and MAX SNP-hard, even when the lengths of the edges are restricted to be either 1 or 2. Many approximation algorithms have been proposed for the full Steiner tree problem in a metric graph [4, 5, 7, 8, 11, 14, 16]. Lin and Xue [14] presented an approximation algorithm with performance ratio $2 + \rho$, where ρ is the approximation ratio for the Steiner tree problem. The currently best-known approximation ratio for the Steiner tree problem is 1.39 [3]. The approximation ratio was improved to 2ρ in [5, 7, 8], and further to $2\rho - (\frac{\rho}{3\rho-2})$ in [16], and $2\rho - \frac{(\rho\alpha^2-\rho\alpha)}{(\alpha+\alpha^2)(\rho-1)+2(\alpha-1)^2}$ in [4] for any $\alpha \geq 2$. The straightforward 2ρ-approximation algorithms [5, 7, 8] start by computing a Steiner tree T of G which has no edge between any pair of terminals. Then, for each non-leaf terminal r in T, pick one of its adjacent Steiner vertices, say s, and connect all other Steiner neighbors of r to s [5, 7, 8].

Drake and Hougardy [7] showed that approximating the non-metric version of the full Steiner tree problem is at least as hard as approximating the set cover problem. They showed that there is no polynomial time approximation algorithm for the full Steiner tree problem with performance ratio better than $(1 - o(1)) \ln n$ unless $NP = DTIME(n^{O(\log \log n)})$.

1.1 Our Results

Let P denote a set of n points in the plane. The unit disk graph, $UDG(P)$, is defined to have P as its vertex set and there is an edge between two points p and q if their Euclidean distance is at most 1, i.e., $|pq| \leq 1$. Given a set of terminals $R \subset P$, we are interested in computing the minimum-weight full Steiner tree of $UDG(P)$; thus the Steiner vertices must be chosen from the set $P \setminus R$. We assume that the weight of an edge (p, q) is equal to the Euclidean distance between p and q; i.e., $w(p, q) = |pq|$. It is not known whether this problem can be solved in polynomial time.

The *aspect ratio* of an edge set E is the ratio of the length of a longest edge in E to the length of a shortest edge in E. The aspect ratio of a graph is defined as the aspect ratio of its edge set. For a constant $\lambda > 0$, a λ-*precise* (or λ-*precision*) UDG is a unit disk graph in which no two vertices are at distance smaller than λ; it is also known as λ-*civilized* UDG [6, 12]. Thus, the aspect ratio of any λ-precise UDG is at most $\frac{1}{\lambda}$. Most often wireless devices in a wireless network cannot be too close, so it is reasonable to model a wireless ad-hoc network as a λ-precise UDG [6]. It can be seen that grid graphs are λ-precise UDG.

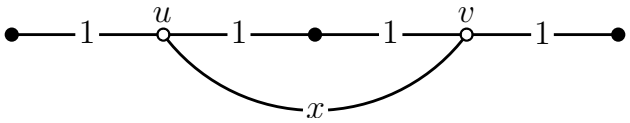


Figure 2: A non-metric instance for the full Steiner tree problem.

In this paper we present two polynomial-time approximation algorithms for the full Steiner tree problem in UDG and λ-precise UDG. Note that all previous results [4, 5, 7, 8, 11, 14, 16] are only applicable when the input graph G is metric. Thus, they cannot be applied to UDG, because it is not necessarily a complete graph. When the input graph G is a λ-precise UDG, we present a $(10 + \frac{1}{\lambda})$-approximation algorithm in Section 2. In Section 3 we extend our idea and present a 20-approximation algorithm for any unit disk graph. The combination of these two algorithms give us a full Steiner tree of approximation ratio $\min\{20, (10 + \frac{1}{\lambda})\}$ for UDG. Using the same technique as in Section 3, we can compute a full Steiner tree of approximation ratio 2Δ for general graphs, where Δ is the maximum vertex degree in the graph.

1.2 Preliminaries

Vazirani [20] showed that in polynomial time one can transform an instance of a Steiner tree problem into an equivalent instance of the metric Steiner tree problem; the transformation preserves the approximation factor. The transformation is as follows. For a given graph $G = (V, E)$, consider a complete graph G' with vertex set V. Define the weight of an edge (u, v) in G' as the weight of the shortest path between u and v in G. Next compute a minimum Steiner tree T' in G' and replace each edge (u, v) in T' by the corresponding shortest path in G. Finally compute a spanning tree T of the resulting subgraph. For any edge $(x, y) \in G$, its cost in G' is no more than its cost in G. Therefore, the cost of an optimal solution in G' is no more than the cost of an optimal solution in G [20].

Theorem (Vazirani [20]). *There is a polynomial-time approximation factor preserving reduction from the Steiner tree problem to the metric Steiner tree problem.*

Drake and Hougardy [7] showed that the similar transformation cannot be applied for the full Steiner tree problem. Figure 2 which is borrowed from [7] shows that if $x > 2$ then the shortest path between Steiner vertices u and v passes through a terminal vertex. Finally when replacing (u, v) in T' with the original shortest path in G, the resulting Steiner tree T is not a full Steiner tree. As a result we have the following observation:

Observation 1. *If the shortest path between any pair of vertices in G does not contain any terminal as an internal vertex, the above transformation can be applied for the full Steiner tree problem.*

Group Steiner Tree (GST): The group Steiner tree problem is defined as follows. Given an edge-weighted graph $G(V, E)$, a set $\mathcal{G}' = \{g_1, g_2, \ldots, g_k\}$, where each g_i is a subset of V. Each such subset g_i is called a group of terminals. The objective is to find a minimum weight tree T that contains at least one terminal from each group. The classical Steiner tree problem is a special case of GST when each group $g_i, 1 \leq i \leq k$, contains one vertex. Since the GST problem is a generalization of the classical Steiner tree problem, it is also NP-hard.

It is well known that the GST problem is at least as hard as the set cover problem. Therefore, it cannot be approximated to a factor $o(\ln k)$ unless $P = NP$, where $k = |\mathcal{G}'|$. In [10], the authors showed that the GST problem cannot be approximated better than $\Omega(\log^{2-\varepsilon} n)$, even when the host graph is a tree. Garg et al. [9] gave an approximation algorithm which, with high probability, finds a GST of cost within $O(\log^2 n \log\log n \log k)$ of the cost of an optimal GST. Slavík [18, 19] developed a 2σ approximation algorithm for GST problem, where σ is the size of the largest group in $\mathcal{G}'$. As part of their algorithm, they compute a complete graph G' on vertex set V with a shortest-path metric inherited from G. Without loss of generality, for the group Steiner tree problem one can assume that the given graph is metric, i.e., it is a complete graph and the edge lengths satisfy the triangle inequality.

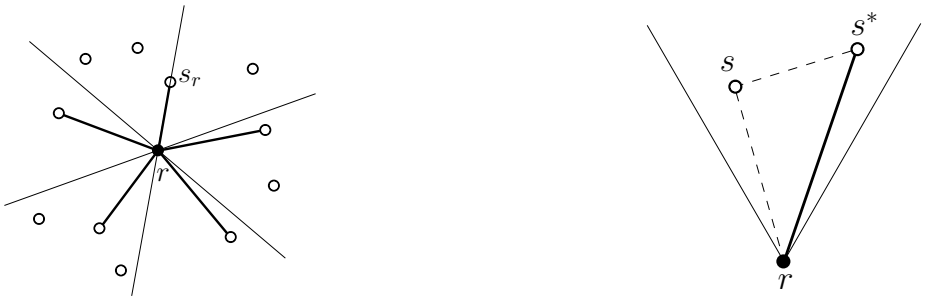
2 Approximation algorithm for λ-precise UDG

Let P be a set of n points in the plane and let $R \subset P$ be a set of terminals. In this section we present a $(10 + \frac{1}{\lambda})$-approximation algorithm for computing a full Steiner tree in a λ-precise unit disk graph $G(P) = UDG(P)$. Recall that in a λ-precise UDG, edges are of lengths at least λ and at most 1. We define a *terminal edge* as an edge connected to a terminal vertex. For a full Steiner tree T let $T(R)$ denote the set of all terminal edges in T and let $T(S)$ denote the *skeleton tree* obtained from T by removing $T(R)$. Clearly, $T(S)$ does not contain any terminal vertex. Let T_{opt} be an optimal full Steiner tree of $G(P)$. Let $T_{opt}(R)$ denote the set of its terminal edges and $T_{opt}(S)$ denote its skeleton tree.

Let $G'(P)$ be the graph obtaining from $G(P)$ in the following way. For each terminal $r \in R$, consider a collection of six cones each of angle $\pi/3$, all having their apex at r, that cover the plane. Let s_r be the nearest Steiner neighbor of r in $G(P)$. Place the cones in such a way that s_r is shared between two cones; see Figure 3(a). For each cone C for which $C \cap S \neq \emptyset$ consider the nearest Steiner neighbor s of r in $G(P)$ in C. Remove from $G(P)$ all the edges incident on r in C except the edge (r, s). Denote the resulting graph by $G'(P)$. For simplicity of notation in the rest of the paper we use G and G' instead of $G(P)$ and $G'(P)$, respectively.

Lemma 1. *The weight of an optimal full Steiner tree in $G'(P)$ is at most two times the weight of an optimal full Steiner tree in $G(P)$.*

Proof. Recall that T_{opt} is an optimal full Steiner tree of G. We describe a method that transforms T_{opt} to a tree T' which is a full Steiner tree of G' and has weight at most two times $w(T_{opt})$. For each terminal $r \in R$, let s^* be the neighbor of r in T_{opt}, and let $C_r(s^*)$ be



(a) (b)

Figure 3: (a) s_r is the nearest neighbor of r which is shared between two cones; bold edges are added to G'. (b) The edge $(r, s^*) \in T_{opt}$ is replaced by two edges (r, s) and (s, s^*).

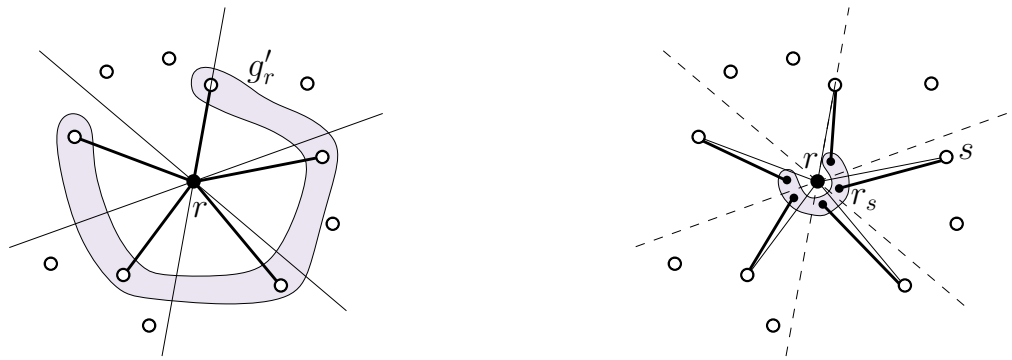
the cone with apex at r which contains s^*. If (r, s^*) is not an edge in G', then let s be the nearest Steiner neighbor of r in $C_r(s^*)$; see Figure 3(b). Clearly, s does not belong to T_{opt} because otherwise, we can replace the edge (r, s^*) by (r, s); the weight of the resulting tree is smaller than the weight of T_{opt} which is a contradiction. Thus, $s \notin T_{opt}$. We replace the edge (r, s^*) by the edges (r, s) and (s, s^*). Let T' denote the resulting tree. Clearly, (r, s) is an edge of G' and hence T' is a full Steiner tree of G'. Since $|rs| < |rs^*|$ and $\angle srs^* \leq \frac{\pi}{3}$, we have $\angle rss^* > \frac{\pi}{3}$. Thus, in triangle $\triangle rss^*$ the segment rs^* is the longest. Therefore, $|rs| + |ss^*| \leq 2|rs^*|$ and hence the weight of T' is at most two times the weight of T_{opt}. □

Consider the tree T' as described in the proof of Lemma 1. We have the following corollary:

Corollary 1. *The weight of the skeleton tree of T' is at most the weight of an optimal full Steiner tree of G; i.e., $w(T'(S)) \leq w(T_{opt})$.*

Proof. As described in Lemma 1, for each terminal edge (r, s^*) in T_{opt}, the tree T' contains either (r, s^*) or (r, s) and (s, s^*). In both cases the skeleton tree $T'(S)$ does not contain the edge (r, s). In addition, $|ss^*| \leq |rs^*|$. Therefore, $w(T'(S)) \leq w(T_{opt})$. □

For each terminal r, let g'_r denote the set of neighbors of r in G'; we consider g'_r to be one "group". See Figure 4(a). Define $\sigma = \max_{r \in R} |g'_r|$. Note that $1 \leq \sigma \leq 5$. Let $\mathcal{G}' = \{g'_r : r \in R\}$ and let $G[S]$ denote the subgraph of G, induced by S. The algorithm FSTUDG1 receives a λ-precise unit disk graph $G(P)$, a terminal set R, and returns a full Steiner tree T. It runs the function GroupSteinerTree$(G[S], \mathcal{G}')$ presented by Slavík [18, 19]. This function returns a group Steiner tree T'_g which is a 2σ-approximation for the group Steiner tree problem on the graph $G[S]$ with respect to $\mathcal{G}'$. Finally it computes a full Steiner tree T by appending each $r \in R$ to its nearest neighbor in T'_g. The GroupSteinerTree first computes a complete graph using the shortest path distances in G and then uses integer programming to formulate an optimal solution. Finally it approximates the optimal solution in polynomial time by using linear programming relaxation.



(a) (b)

Figure 4: (a) Group g'_r of the neighbors of a terminal r in G'. (b) Group g''_r of the copies of a terminal r in G''.

Algorithm 1 FSTUDG1$(G(P), R)$

Input: a λ-precise unit disk graph $G(P)$ and a subset $R \subset P$.
Output: a full Steiner tree T of $G(P)$.

1: $S \leftarrow P \backslash R$, $\mathcal{G}' \leftarrow \emptyset$
2: compute G' from $G(P)$
3: **for** each $r \in R$ **do**
4: $\quad g'_r \leftarrow$ set of neighbors of r in G'
5: $\quad \mathcal{G}' \leftarrow \mathcal{G}' \cup \{g'_r\}$
6: **end for**
7: $T'_g \leftarrow$ GroupSteinerTree$(G[S], \mathcal{G}')$
8: $T \leftarrow$ connect each $r \in R$ to its nearest neighbor in T'_g
9: **return** T

Lemma 2. *The weight of the tree T'_g computed by algorithm* FSTUDG1 *is at most* 10 *times the weight of T_{opt}.*

Proof. Recall that $T_{opt}(S)$ denotes the skeleton tree of the optimal full Steiner tree T_{opt} of $G(P)$, and $T_{opt}(R)$ denotes the set of its terminal edges. Let $T'(S)$ be the skeleton tree of the tree T' described in the proof of Lemma 1. By Corollary 1, $w(T'(S)) \leq w(T_{opt})$. On the other hand $T'(S)$ is a solution for the group Steiner tree problem in $G[S]$ with respect to $\mathcal{G}'$. Thus, the weight of T'_g in line 7 is at most 10 times (i.e., two times the maximum size of any group) the weight of $T'(S)$, that is

$$w(T'_g) \leq 10 \cdot w(T'(S)) \leq 10 \cdot w(T_{opt}).$$

□

Theorem 1. *Algorithm* FSTUDG1 *runs in polynomial time. The tree T returned by this algorithm is a $(10 + \frac{1}{\lambda})$-approximation for an optimal full Steiner tree T_{opt} of a λ-precise unit disk graph $G(P)$.*

Proof. As shown in [18, 19] the GroupSteinerTree routine in line 7 can be done in polynomial time. Therefore, FSTUDG1 runs in polynomial time.

By Lemma 2, the weight of T'_g is at most 10 times the weight of T_{opt}. For each terminal $r \in R$, let s be a vertex in T'_g that is closest to r. Note that $s \in g'_r$. Let $T(R)$ be the set of all (r, s) edges considered in line 8. Then T is the union of T'_g and $T(R)$. Clearly, T is a full Steiner tree of G and $w(T) = w(T'_g) + w(T(R))$. Note that in T, r is connected to s and in T_{opt} it is connected to s^*. Since G is a λ-precise UDG, the length of the edge (r, s) is at most $\frac{1}{\lambda}$ times the length of the edge (r, s^*), and hence $w(T(R)) \leq \frac{1}{\lambda} \cdot w(T_{opt}(R))$. Therefore,

$$\begin{aligned} w(T) &= w(T'_g) + w(T(R)) \\ &\leq 10 \cdot w(T_{opt}) + \frac{1}{\lambda} \cdot w(T_{opt}(R)) \\ &= 10 \cdot w(T_{opt}(S)) + 10 \cdot w(T_{opt}(R)) + \frac{1}{\lambda} \cdot w(T_{opt}(R)) \\ &= 10 \cdot w(T_{opt}(S)) + (10 + \frac{1}{\lambda}) \cdot w(T_{opt}(R)) \\ &\leq (10 + \frac{1}{\lambda}) \cdot w(T_{opt}). \end{aligned}$$

□

3 Approximation algorithm for UDG

The approximation ratio of the tree T that is returned by algorithm FSTUDG1, is directly related to the aspect ratio of the input graph G. In this section we present another approximation algorithm for the full Steiner tree problem in UDG which is independent of the aspect ratio. By a modification of algorithm FSTUDG1 we present a 20-approximation algorithm for unit disk graphs as follows. Consider again the graph G' of Section 2. Let G'' be the graph obtaining from G' in the following way. Recall g'_r that is a group of (at most five) Steiner neighbors of a vertex r in G'. For each $s \in g'_r$ create a copy of r and call it r_s; see Figure 4(b). Connect r_s to s with an edge of weight equal to $w(r, s)$, i.e., $w(r_s, s) = w(r, s)$, then remove r and the edges incident on r. Denote the resulting graph as G''. Note that in G'' all the terminals (new vertices) have degree one, and hence they cannot be internal vertex of any simple path. Thus, by Observation 1 we can use Vazirani's method to transform G'' to a metric graph, i.e., a complete graph which satisfies the triangle inequality. Note that in the GST approximation algorithm presented by Slavík [18, 19] the input graph should be metric. For each $r \in R$, let g''_r be the group of copies of r in G'' as shown in Figure 4(b). Let $\mathcal{G}''$ denote the set of groups g''_r, for all $r \in R$. Algorithm FSTUDG2 computes the group Steiner tree of G'' with respect to $\mathcal{G}''$. Finally it computes the full Steiner tree T by replacing one of the covered vertices of g''_r by r. The GroupSteinerTree is the 2σ-approximation algorithm presented by Slavík [18, 19].

Lemma 3. *An optimal group Steiner tree in G'' (with respect to $\mathcal{G}''$) can be transformed to an optimal full Steiner tree in G' with the same weight.*

Algorithm 2 FSTUDG2$(G(P), R)$

Input: a unit disk graph $G(P)$ and a subset $R \subset P$.
Output: a full Steiner tree T of $G(P)$.

1: $S \leftarrow V \backslash R$, $\mathcal{G}'' \leftarrow \emptyset$
2: compute G' from G
3: compute G'' from G'
4: **for** each $r \in R$ **do**
5: $\quad g''_r \leftarrow$ set of $|g'_r|$ many copies of r in G''
6: $\quad \mathcal{G}'' \leftarrow \mathcal{G}'' \cup \{g''_r\}$
7: **end for**
8: $T'' \leftarrow$ GroupSteinerTree$(G'', \mathcal{G}'')$
9: $T \leftarrow$ for each $r \in R$, replace one of the vertices in $T'' \cap g''_r$ with r, and discard all other vertices of g''_r from T''
10: **return** T

Proof. Let T''_{opt} denote an optimal group Steiner tree of G'' and let T'_{opt} denote an optimal full Steiner tree of G'. We show how one can transform T''_{opt} to a full Steiner tree T' of G' with $w(T') = w(T'_{opt})$. Note that in G'' all terminals have degree one, and hence T''_{opt} does not contain any internal terminal vertex. In addition, T''_{opt} is optimal, so it covers exactly one terminal from each group g''_r, and it does not contain any Steiner vertex as a leaf. For each g''_r, where $r \in R$, let r_s be the member of g''_r covered by T''_{opt}. Recall that for each edge (r_s, s) in G'' there is an edge (r, s) in G' with the same weight. Let T' be the tree obtained from T''_{opt} by replacing each edge (r_s, s) by (r, s). Since $w(r, s) = w(r_s, s)$, $w(T') = w(T''_{opt})$. Clearly, T' is a full Steiner tree of G' and hence $w(T') \geq w(T'_{opt})$. Now we prove that $w(T') = w(T'_{opt})$. Using contradiction, suppose that $w(T') > w(T'_{opt})$. Consider the skeleton tree $T'_{opt}(S)$. Each terminal $r \in R$ is connected to a Steiner vertex $s \in T'_{opt}(S)$. Note that $T'_{opt}(S) \subseteq G[S] \subseteq G''$. By connecting $r_s \in g''_r$ to s we obtain a solution T'' for the group Steiner tree problem which has the same weight as T'_{opt}. Thus, $w(T'') = w(T'_{opt}) < w(T') = w(T''_{opt})$, which contradicts the optimality of T''_{opt}. Thus, $w(T') = w(T'_{opt})$ and T' is an optimal full Steiner tree of G'. □

Theorem 2. *Algorithm* FSTUDG2 *runs in polynomial time. The tree T returned by this algorithm is a 20-approximation for an optimal full Steiner tree T_{opt} of a unit disk graph $G(P)$.*

Proof. As shown in [18, 19] the GroupSteinerTree routine in line 8 can be done in polynomial time. Therefore, FSTUDG2 runs in polynomial time.

Let T''_{opt} denote an optimal group Steiner tree of G'' with respect to $\mathcal{G}''$. The tree T'' obtained in line 8 is a 10-approximation for T''_{opt}. By using the same argument as in Lemma 3, the tree T obtained in line 9 is a full Steiner tree for G' and $w(T) = w(T'')$. According to the statement of Lemma 3, we argue that

$$w(T) = w(T'') \leq 10 \cdot w(T''_{opt}) = 10 \cdot w(T'_{opt}).$$

By Lemma 1, the weight of an optimal full Steiner tree in G' is at most 2 times the weight of an optimal full Steiner tree in G. Therefore,

$$w(T) \leq 10 \cdot w(T'_{opt}) \leq 20 \cdot w(T_{opt}).$$

□

Consider the input unit disk graph G and algorithms FSTUDG1 and FSTUDG2. To compute a full Steiner tree of G, if the aspect ratio of G is at most 10, i.e., $\lambda \geq \frac{1}{10}$, we use algorithm FSTUDG1, otherwise we use algorithm FSTUDG2. Thus, we can find a full Steiner tree of G with the approximation ratio of $\min\{20, (10 + \frac{1}{\lambda})\}$.

4 Conclusion

We considered the problem of computing a minimum-weight full Steiner tree in a unit disk graph. In Section 2 we presented a $(10 + \frac{1}{\lambda})$-approximation for λ-precise UDGs; where the length of the smallest edge is at least λ. For general unit disk graphs we presented a 20-approximation algorithm in Section 3. The combination of these two algorithms gives a full Steiner tree of approximation ratio $\min\{20, (10 + \frac{1}{\lambda})\}$ for UDG.

We leave as an open problem to improve the approximation ratio. We also leave as an open problem whether the exact solution can be computed in polynomial time.

The proposed algorithm for a UDG in Section 3 can be extended to any simple graph G as follow. Let G'' be the graph obtaining from G in the following way. Let $N(r)$ denote the Steiner neighbors of a terminal vertex r in G. For each $s \in N(r)$, where $r \in R$, create a copy of r and call it r_s. Connect r_s to s with an edge of weight equal to $w(r, s)$, then remove r and its adjacent edges. Denote the resulting graph by G''. Then we run algorithm FSTUDG2 from line 4. The tree T'' obtained in line 8 is a 2Δ-approximation for the group Steiner tree problem where $\Delta = \max\{|N(r)| : r \in R\}$; Δ is possibly the maximum vertex degree in G. Finally, by using the same argument as in Lemma 3, we conclude that the tree T obtained in line 9 is a 2Δ-approximation for an optimal full Steiner tree in G.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[2] M. W. Bern and P. E. Plassmann. The Steiner problem with edge lengths 1 and 2. *Inf. Process. Lett.*, 32(4):171–176, 1989.

[3] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6, 2013.

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

[5] 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.

[6] M. Damian. A simple Yao-Yao-based spanner of bounded degree. Online, http://arxiv.org/abs/0802.4325v2, 2008.

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

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

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

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

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

[12] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms*, 26(2):238–274, 1998.

[13] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem. Annals of Discrete Mathematics*. North-Holland, Elsevier, Amesterdam, 1992.

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

[15] C. L. Lu, C. Y. Tang, and R. C.-T. Lee. The full Steiner tree problem. *Theor. Comput. Sci.*, 306(1-3):55–67, 2003.

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

[17] G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *SODA*, pages 770–779, 2000.

[18] P. Slavík. The errand scheduling problem. Technical report, Department of Computer Science and Engineering, University of New York at Buffalo. http://www.cse.buffalo.edu/tech-reports/, 1997.

[19] P. Slavík. *Approximation algorithms for set cover and related problems*. PhD thesis, Department of Computer Science and Engineering, University of New York at Buffalo. http://www.cse.buffalo.edu/tech-reports/, 1998.

[20] V. V. Vazirani. *Approximation algorithms*. Springer, 2001.