An Approximation Algorithm for the Euclidean Bottleneck Steiner Tree Problem

A. Karim Abu-Affash

Abstract

Given two sets of points in the plane, \( P \) of \( n \) terminals and \( S \) of \( m \) Steiner points, a Steiner tree of \( P \) is a tree spanning all points of \( P \) and some (or none or all) points of \( S \). A Steiner tree with length of longest edge minimized is called a bottleneck Steiner tree. In this paper, we study the Euclidean bottleneck Steiner tree problem: given two sets, \( P \) and \( S \), and a positive integer \( k \leq m \), find a bottleneck Steiner tree of \( P \) with at most \( k \) Steiner points. The problem has application in the design of wireless communication networks.

We first show that the problem is NP-hard and cannot be approximated within factor \( \sqrt{2} \), unless \( P = NP \). Then, we present a polynomial-time approximation algorithm with performance ratio 2.

1 Introduction

Consider a wireless communication network with \( n \) stations, each station has a limited power so that it can only communicate with stations within a limited range, and suppose that, in order to make the network connected and due to budget limits, we are only allowed to put at most \( k \) new stations in given potential locations in this network. Clearly, we would like to select locations such that distance between stations as small as possible. This application motivates the following problem:

The Bottleneck Steiner Tree \((k\text{-BST})\) problem. Given two sets in the plane, \( P \) of terminal points and \( S \) of Steiner points, and a positive integer \( k \), one is asked to find Steiner tree \( T \) of \( P \) with at most \( k \) Steiner points, such that the bottleneck (i.e., length of the longest edge) of \( T \) is minimized.

In the classical Steiner tree \((ST)\) problem, the goal is to find a Steiner tree \( T \) such the total length of edges of \( T \) is minimized. This problem has been shown to be NP-complete \cite{7} and many approximation algorithms have been proposed \cite{2,3,8}. A general version of the \( k\text{-BST} \), where \( k = |S| \), has been studied by Sarrafzadeh and Wong \cite{12}. They showed that this version can be solved in polynomial time.
Another version, where $S$ is the whole plane $\mathbb{R}^2$, has been studied extensively in the last decade. In [13], this version was shown to be NP-hard to approximate within ratio $\sqrt{2}$. The best known upper bound on approximation ratio is $1.866$ [14]. Bae et al. [1] presented an $O(n \log n)$ time algorithm to the problem for $k = 1$ and an $O(n^2)$ time algorithm for $k = 2$. Li et al. [11] presented a $(\sqrt{2} + \epsilon)$-approximation algorithm with inapproximability within $\sqrt{2}$ for a special case of the problem where there should be no edge connecting any two Steiner points in the optimal solution. These versions have many important applications in VLSI design, network communication and computational biology [4,6,9,10].

We are not aware of any previous work studying our version. However, in this paper, we show that the $k$-BST problem is NP-hard and we present a polynomial-time algorithm with constant factor approximation ratio for the problem.

2 Hardness Result

Given a set $P$ of $n$ terminals in the plane, a set $S$ of $m$ Steiner points and an integer $k \leq m$, the goal in the $k$-BST problem is to find a Steiner tree with at most $k$ Steiner points from $S$ and bottleneck as small as possible. In this section we prove hardness of the problem.

**Theorem 2.1.** The $k$-BST problem cannot be approximated within $\sqrt{2}$ in polynomial time, unless $P = \mathsf{NP}$.

The proof directly follows by a slight modification of the proof of Theorem 1 in [13].

3 2-Approximation Algorithm

In this section, we develop a polynomial-time approximation algorithm for computing a Steiner tree with at most $k$ Steiner points ($k$-ST for short) such that its bottleneck is at most 2 times the bottleneck of an optimal (minimum-bottleneck) $k$-ST.

Let $G = (V, E)$ be the complete graph over $V = P \cup S$. We assume, without loss of generality, that $E = \{e_1, e_2, \ldots, e_l\}$ such that $|e_1| \leq |e_2| \leq \ldots \leq |e_l|$. It is not hard to see that the bottleneck of an optimal $k$-ST is a length of an edge from $E$. For an edge $e_i \in E$, let $G_i = (V, E_i)$ be the graph with $E_i = \{e_j \in E : |e_j| \leq |e_i|\}$. The idea behind our algorithm is to devise a procedure that, for a given edge $e_i \in E$, does one of the following:

(i) It constructs a $k$-ST of $P$ in $G$ with bottleneck at most 2 times $|e_i|$.

(ii) It returns the information that $G_i$ does not contain any $k$-ST of $P$.

For two points $p, q \in P$, let $\delta_i(p, q)$ be a shortest Steiner path between $p$ and $q$ in $G_i$, i.e., a path connecting $p$ and $q$ with minimum number of Steiner points in $G_i$. Let $G_P = (P, E_P)$ be the complete graph over $P$. For each edge $(p, q)$ in $E_P$, we assign a weight $w(p, q)$ equal to the number of Steiner points in $\delta_i(p, q)$. Let $T$ be a minimum spanning tree of $G_P$ under $w$. We define the normalized weight of $T$ as $C(T) = \sum_{e \in T} w(e)/2$. 

2
Lemma 3.1. If $G_i$ contains a $k$-ST of $P$, then $C(T) \leq k$.

**Proof:** Let $T^*$ be a $k$-ST of $P$ in $G_i$. A Steiner tree is full if all terminals are leaves. We decompose $T^*$ into a union of full trees. For each full tree $T^*_j$ of $T^*$, we will construct a spanning tree $T'_j$ of the terminals of $T^*_j$ in $G_P$, such that the union of these tree is a spanning tree $T'$ of $P$ in $G_P$ with $C(T') \leq k$. We arbitrary select a Steiner point as the root of $T^*_j$; see Figure 1(a). The construction of $T'_j$ is bottom-up by an iterative process. In each iteration, we select the deepest leaf $p$ in the rooted tree, which is a terminal, and we connect it to its nearest terminal $q$ by an edge of weight equal to the number of Steiner points between them. Let $s$ be the first common parent of $p$ and $q$. We then remove the Steiner points between $p$ and $s$ (in the last iteration, we may remove all of the remaining points).

![Figure 1: (a) The rooted tree, and (b) the construction of $T'_j$.](image)

In the example in Figure 1(b), we first select the terminal $a$, which is the deepest one, we connect it to the terminal $b$ by an edge of weight 3 and we remove the points $s_1$ and $s_2$. Next, we select the terminal $d$, we connect it to the terminal $c$ by an edge of weight 2 and we remove the point $s_3$. In the last iteration, we select the terminal $b$, we connect it to the terminal $c$ by an edge of weight 3 and we remove all of the remaining points.

Notice that, since, in each iteration, we select the deepest terminal, we add an edge $(p, q)$, of weight $w(p, q)$, and we remove at least $\lfloor w(p, q)/2 \rfloor$ Steiner points from $T^*_j$. This implies that $C(T'_j) = \sum_{e \in T'_j} \lfloor w(e)/2 \rfloor \leq k_j$, where $k_j$ is the number of Steiner points in $T^*_j$. Moreover, the union $T'$ of the trees $T'_j$ is a spanning tree of $G'$ and has $C(T') \leq k$. Thus, since $T$ is a minimum spanning tree of $G'$, we have $C(T) \leq C(T') \leq k$.

We now describe our approximation algorithm. We traverse the edges of $E$ in the sorted order and, for each edge $e_i \in E$, we construct a minimum spanning tree $T$ of $G_P = (P, E_P)$ and check whether $C(T) \leq k$. If so, we construct a $k$-ST of $P$, otherwise, we move to the next edge $e_{i+1}$.
Algorithm 1 EBST($G = (V, E), P, k$)

1: $C(T) \leftarrow \infty$
2: $G_P = (P, E_P) \leftarrow$ the complete graph over $P$
3: $i \leftarrow 0$
4: while $C(T) > k$ do
5: $i \leftarrow i + 1$
6: construct the graph $G_i$
7: for each edge $(p, q) \in E_P$ do
8: $w(p, q) \leftarrow$ the number of Steiner points in $\delta_i(p, q)$
9: construct a minimum spanning tree $T$ of $G_P$ under $w$
10: $C(T) \leftarrow \sum_{e \in T} \lfloor w(e)/2 \rfloor$
11: $\text{Construct-}k\text{-ST}(T, G_i)$

The construction of a $k$-ST is done as follows. For each edge $e = (p, q) \in T$, we select $\lfloor w(e)/2 \rfloor$ Steiner points on any shortest Steiner path between $p$ and $q$ in $G_i$, such that, the path from $p$ to $q$ that passes through these points has a bottleneck at most $2|e_i|$, and we connect these points to form a path; see Figure 2. Clearly, the obtained Steiner tree contains at most $k$ Steiner points and its bottleneck is at most $2|e_i|$.

Figure 2: The constructed $k$-ST consists of the black circles and the dotted lines.

Lemma 3.2. The algorithm above constructs a $k$-ST of $P$ with bottleneck at most 2 times the bottleneck of an optimal $k$-ST.

Proof: Let $e_i$ be the first edge satisfying the condition $C(T) \leq k$. Thus, by Lemma 3.1, the bottleneck of any $k$-ST in $G$ is at least $|e_i|$, and, therefore, the constructed $k$-ST has a bottleneck at most 2 times the bottleneck of an optimal $k$-ST.

Lemma 3.3. The algorithm above has a polynomial running time.
**Proof:** $G_i$ can be constructed in $O((n + m)^2)$ time. In order to construct the graph $G_P$, we can compute in $O((n+m)^3)$ time the shortest Steiner paths between each pair of points in $P$ [5]. Once $G_P$ is constructed, computing a minimum spanning tree of $G_P$ can be done in $O(n^2)$ time, and selecting the relevant Steiner points can be done in $O(k(n + m))$ time.

By combining Lemma 3.2 and Lemma 3.3, we get the following theorem.

**Theorem 3.4.** There exists a polynomial-time approximation algorithm with performance ratio 2 for the $k$-BST problem.

4 Conclusion

In this paper, we studied the problem of finding bottleneck Steiner trees in the Euclidean plane. We proved that the $k$-BST problem in the plane does not admit any approximation algorithm with performance ratio less than $\sqrt{2}$, unless $P = NP$, and that there exists a polynomial-time approximation algorithm with performance ratio 2. It would be interesting to find better approximation algorithm for the $k$-BST problem. Another interesting question is how efficient can one solve the $k$-BST problem for a constant $k > 0$?

References

[1] S.W. Bae, C. Lee, and S. Choi. On exact solutions to the euclidean bottleneck Steiner tree problem. *Information Processing Letters*, 110:672–678, 2010.

[2] P. Berman and V. Ramaiyer. Improved approximation for the Steiner tree problem. *Journal of Algorithms*, 17:381–408, 1994.

[3] A. Borchers and D.Z. Du. The $k$-Steiner ratio in graphs. *SIAM Journal on Computing*, 26:857–869, 1997.

[4] X. Cheng and D.Z. Du. *Steiner Tree in Industry*. Kluwer Academic Publishers, Dordrecht, Netherlands, 2001.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd edition*. MIT Press, 2001.

[6] D.Z. Du, J.M. Smith, and J.H. Rubinstein. *Advances in Steiner Tree*. Kluwer Academic Publishers, Dordrecht, Netherlands, 2000.

[7] M.R. Garey, R.L. Graham, and D.S. Johnson. The complexity of computing Steiner minimal trees. *SIAM Journal of Applied Mathematics*, 32(4):835–859, 1977.
[8] S. Hougardy and H.J. Prommel. A 1.598 approximation algorithm for the Steiner problem in graphs. In Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA ’00), pages 448–453, 1999.

[9] F.K. Hwang, D.S. Richards, and P. Winter. The Steiner Tree Problem. Annals of Discrete Mathematics, Amsterdam, 1992.

[10] A.B. Kahng and G. Robins. On Optimal Interconnection for VLSI. Kluwer Academic Publishers, Dordrecht, Netherlands, 1995.

[11] Z.-M. Li, D.-M. Zhu, and S.-H. Ma. Approximation algorithm for bottleneck Steiner tree problem in the Euclidean plane. Journal of Computer Science and Technology, 19(6):791–794, 2004.

[12] M. Sarrafzadeh and C.K. Wong. Bottleneck Steiner trees in the plane. IEEE Transactions on Computers, 41(3):370–374, 1992.

[13] L. Wang and D.-Z. Du. Approximations for a bottleneck Steiner tree problem. Algorithmica, 32:554–561, 2002.

[14] L. Wang and Z.-M. Li. Approximation algorithm for a bottleneck k-Steiner tree problem in the Euclidean plane. Information Processing Letters, 81:151–156, 2002.