Exact Algorithms for the Bottleneck Steiner Tree Problem*

(Extended Abstract)

Sang Won Bae[1], Sunghee Choi[2], Chunseok Lee[2], and Shin-ichi Tanigawa[3]

[1] Department of Computer Science and Engineering, POSTECH, Pohang, Korea
swbae@postech.ac.kr
[2] Division of Computer Science, KAIST, Daejeon, Korea
{sunghee, stonecold}@tclab.kaist.ac.kr
[3] Department of Architecture and Architectural Engineering, Kyoto University, Kyoto, Japan
is.tanigawa@archi.kyoto-u.ac.jp

Abstract. Given n terminals in the plane $\mathbb{R}^2$ and a positive integer k, the bottleneck Steiner tree problem is to find k Steiner points in $\mathbb{R}^2$ so that the longest edge length of the resulting Steiner tree is minimized. In this paper, we study this problem in any L_p metric. We present the first fixed-parameter tractable algorithm running in $O(f(k) \cdot n^2 \log n)$ time for the L_1 and the L_∞ metrics, and the first exact algorithm for any other L_p metric with $1 < p < \infty$ whose time complexity is $O(f(k) \cdot (n^k + n \log n))$, where $f(k)$ is a function dependent only on k. Note that prior to this paper there was no known exact algorithm even for the L_2 metric, and our algorithms take a polynomial time in n for fixed k.

1 Introduction

We consider the following problem, called the *bottleneck Steiner tree problem* (BST).

*Problem 1 (*BST*).* Given n points, called *terminals*, in the plane and a positive integer k, find a Steiner tree spanning all terminals and at most k Steiner points that minimizes the length of its longest edge.

Such a Steiner tree with the length of the longest edge minimized is called a *bottleneck Steiner tree* (also known as a *min-max Steiner tree*).

Unlike the classical Steiner tree problem where the sum of the edge lengths of the Steiner tree is minimized, this problem asks a Steiner tree where the maximum of the edge lengths is minimized and the Steiner points in the resulting tree can be chosen in the whole plane $\mathbb{R}^2$.

The bottleneck Steiner tree problem (shortly BST) has been studied with many applications, like VLSI layout [4], multi-facility location, and wireless communication network design [9]. Previous work on BST considered the Euclidean plane (L_2) or the

* Work by S.W.Bae was supported by the Brain Korea 21 Project. Work by C.Lee and S.Choi was supported by the Korea Science and Engineering Fouundation (KOSEF) grant funded by the Korea Government (MEST) (No.R01-2007-000-20865-0). Work by S.Tanigawa was supported by Grant-in-Aid for JSPS Research Fellowship for Young Scientists.

Y. Dong, D.-Z. Du, and O. Ibarra (Eds.): ISAAC 2009, LNCS 5878, pp. 24–33, 2009.

rectilinear plane (L_1). BST is known to be NP-hard to approximate within ratio $\sqrt{2}$ in the L_2 metric or ratio 2 in the L_1 metric [9]. For the L_1 metric, the matching upper bound was known by Wang and Du [9] and for the L_2 metric, the best known upper bounds on approximation ratio is 1.866 by Wang and Li [10]. For the special case of this problem where there should be no edge connecting any two Steiner points in the optimal solution, Li et al. [7] present a $(\sqrt{2}+\epsilon)$-factor approximation algorithm and inapproximability within ratio $\sqrt{2}$.

Also, there has been effort on devising an exact algorithm for finding the locations of k Steiner points. Many researchers considered the variation of BST where a topology $\mathcal{T}$ of Steiner trees is fixed, called the bottleneck Steiner tree with fixed topology (BST-FT); the resulting Steiner tree should have the same topology as a given tree $\mathcal{T}$ [8,5]. Once we have an exact algorithm for this problem BST-FT, we can find an exact solution to BST by enumerating all valid topology trees.

For the L_1 (or the L_∞) metric, Ganley and Salowe [5] presented an $O((n+k)^2)$ time exact algorithm for BST-FT, which leads to an exact algorithm for BST with running time $O((n+k)!(n+k)^2)$. The situation for the L_2 metric (or the Euclidean metric) is much worse. There was no known exact algorithm for BST in the L_2 metric but few partial results: Sarrafzadeh and Wong [8] presented an $O((n+k)\log(n+k))$ time algorithm for the decision version of BST-FT. More recently, Bae et al. [3] presented exact algorithms for $k \leq 2$; $O(n \log n)$ time for $k = 1$ and $O(n^2)$ time for $k = 2$.

The aim of this paper is to devise exact algorithms for BST in any L_p metric, which run fast for small fixed k. We present two exact algorithms for BST: (1) an $O(f(k) \cdot n^2 \log n)$-time algorithm for the L_1 or the L_∞ metric and (2) an $O(f(k) \cdot (n^k + n \log n))$-time algorithm for the L_p metric with $1 < p < \infty$, where $f(k)$ is a function dependent on k only. Note that both algorithms run in time polynomial in n for fixed k. The algorithm for the L_1 and the L_∞ metrics is a *fixed-parameter algorithm* with respect to parameter k. A fixed-parameter algorithm is one that has time complexity of the form $f(k) \cdot n^{O(1)}$. Thus, we show that BST in the L_1 metric is fixed-parameter tractable with respect to parameter k. On the other hand, we failed to achieve fixed-parameter tractability for the other case but it is worth noting that our algorithm for the L_p metric with $1 < p < \infty$ is the first exact algorithm for BST.

2 Properties of Bottleneck Steiner Trees

Let $P \subset \mathbb{R}^2$ be a set of n points; we call each point in P a *terminal*. A *bottleneck spanning tree* of P is a spanning tree of P such that the length of a longest edge is minimized. We call the length of a longest edge in a bottleneck spanning tree of P the *bottleneck of the set* P, denoted by $b(P)$. Note that $b(P)$ is dependent only on the set P, and not on a particular spanning tree of the points P. A *minimum spanning tree* $MST(P)$ of given points P is a spanning tree of P with minimum sum of the edge lengths. Obviously, $MST(P)$ is a bottleneck spanning tree of P. Throughout this section, the underlying distance function is assumed to be the L_p metric for any $1 \leq p \leq \infty$, and $d(a,b)$ denotes the L_p-distance between two points $a, b \in \mathbb{R}^2$.

Since computing a minimum or bottleneck spanning tree of a given set can be done easily, the bottleneck Steiner tree problem can be viewed as finding a set Q of k optimal

locations to minimize the bottleneck $b(P \cup Q)$ of $P \cup Q$ over all such sets of k points in the plane $\mathbb{R}^2$. We let $e_1, \ldots, e_{n-1}$ be the edges of $MST(P)$ in the order that their lengths are not increasing, and $|e_i|$ denote the length of e_i with respect to the L_p metric.

A solution to the bottleneck Steiner tree problem is a set Q of k Steiner points and its resulting Steiner tree, which is a bottleneck spanning tree T^* of $P \cup Q$. By definition, there can be many bottleneck Steiner trees for an instance of the problem. Here, we prove the following theorem for any L_p metric to restrict ourselves to Steiner trees with helpful properties.

Theorem 1. *There always exists a bottleneck Steiner tree T^* for the terminal set P with a set Q of k Steiner points that satisfies the following conditions **BST1–4**:*

BST1 *Each edge in T^* belongs to $MST(P)$ or is incident to a Steiner point in Q.*
BST2 *Each Steiner point $q \in Q$ is located at the center of the minimum enclosing L_p circle of its neighbors in T^*.*
BST3 *The degree of each Steiner point is bounded by a constant Δ. More specifically, $\Delta = 7$ for the L_1 and the L_∞ metrics; $\Delta = 5$ for any L_p metric with $1 < p < \infty$.*
BST4 *There is a positive integer c with $1 \leq c \leq (\Delta - 1)k$ such that T^* excludes $e_1, \ldots, e_c$ but includes $e_{c+1}, \ldots, e_{n-1}$.*

Bae et al. [3] have proved this theorem for the L_2 metric, and then exploited it to devise exact algorithms for the Euclidean bottleneck Steiner tree problem for $k \leq 2$.

Observe that **BST1** easily follows from the optimality of the minimum spanning tree; its proof can be found also in Bae et al. [3]. Furthermore, we can locally rearrange the locations of Steiner points to satisfy **BST2** without any combinatorial change of the Steiner tree. Thus, in this section, we mainly discuss **BST3** and **BST4**.

We start with bounding the number of nearest neighbors of each point.

Lemma 1. *Let q be a point in the plane and $p_1, \ldots, p_m$ be other m points around q in counter-clockwise order. If $d(p_i, q) \leq d(p_i, p_j)$ for any $1 \leq i, j \leq m$ with $i \neq j$, then we have either $m \leq 8$ for the L_1 and the L_∞ metrics or $m \leq 6$ for the L_p metric with $1 < p < \infty$. Moreover, the maximum value of m can be obtained only if we have equality $d(p_i, q) = d(p_i, p_{i-1}) = d(p_i, p_{i+1})$ for any $1 \leq i \leq m$.*

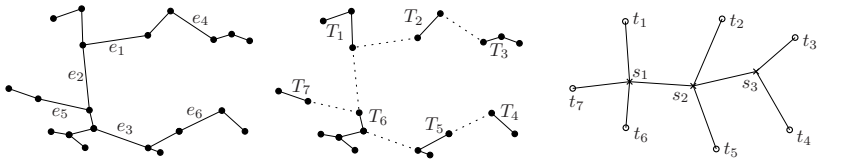
Lemma 2. *There exists a bottleneck Steiner tree T^* such that T^* fulfills **BST1–3** and there is a positive integer c where T^* excludes $e_1, \ldots, e_c$ but includes $e_{c+1}, \ldots, e_{n-1}$.*

Now, we give a lower bound on the bottleneck improvement depending on the number k of allowed Steiner points.

Lemma 3. *For any k points $q_1, \ldots, q_k \in \mathbb{R}^2$, we have $b(P \cup \{q_1, \ldots, q_k\}) \geq |e_{(\Delta-1)k+1}|$.*

Conversely, Lemma 3 gives an upper bound on the possible number c of removed edges from $MST(P)$ as appeared in **BST4**, finally proving Theorem 1.

Hence, a possible way to solve BST is as follows: For each c with $1 \leq c \leq (\Delta-1)k$, we remove $e_1, \ldots, e_c$ from $MST(P)$, to obtain $c + 1$ induced subtrees $T_1, \ldots, T_{c+1}$. Then, we find k Steiner points to reconnect the T_i with new edges incident to the Steiner points, minimizing the longest edge length. Finally, we choose the best solution among all c as an optimal solution for the original problem.



(a) $MST(P)$ (b) $MST(P) - \{e_1, \ldots, e_6\}$ (c) an abstract topology $\mathcal{T}_0$

Fig. 1. An illustration of the case when $k = 3$ and $c = 6$. (a) From $MST(P)$, (b) remove $c = 6$ longest edges $e_1, \ldots, e_6$ to have $c + 1 = 7$ subtrees $T_1, \ldots T_7$. (c) An abstract topology $\mathcal{T}_0$ is a tree on the t_i, representing these subtrees, and the s_i, representing Steiner points.

This motivates another variation of the problem, called the bottleneck Steiner tree problem with a *fixed topology on subtrees*.

*Problem 2 (*BST-FT-ST*).* Given a set P of n terminals in the plane, two positive integers k and c with $c \leq (\Delta - 1)k$, and a topology $\mathcal{T}_0$ on the $c + 1$ subtrees T_i induced by $MST(P)$ and k Steiner points, find an optimal placement of k Steiner points to obtain a bottleneck Steiner tree that has the same topology as $\mathcal{T}_0$.

The topology $\mathcal{T}_0$ in the above problem has $c + k + 1$ vertices $V := \{s_1, \ldots, s_k, t_1, \ldots, t_{c+1}\}$, where t_i represents subtree T_i and s_j represents a Steiner point. See Figure 1. We call each s_i a *Steiner vertex* in order to distinguish it from a Steiner point chosen as a point in $\mathbb{R}^2$, and let $S := \{s_1, \ldots, s_k\}$ be the set of Steiner vertices. A Steiner point can not have degree 1 but may have degree 2 in the bottleneck Steiner tree [8]. Together with **BST3**, we can assume that each s_j has degree between 2 and Δ in $\mathcal{T}_0$. Also, to distinguish $\mathcal{T}_0$ from a topology tree used in BST-FT [8,5], we call $\mathcal{T}_0$ an *abstract topology* while a topology tree on terminals is called a *concrete topology*.

Notice that an abstract topology $\mathcal{T}_0$ has at most $\Delta k + 1$ vertices. This means that the number of possible abstract topologies is independent of the number n of terminals. We end this section with an easy bound on the number of abstract topologies, which is based on the Prüfer code.

Lemma 4. *Given k Steiner points, the total number of abstract topologies over all $1 \leq c \leq (\Delta - 1)k$ is bounded by $O((\Delta k + 1)^{\Delta k - 1})$.*

3 Fixed-Parameter Algorithm for the L_1 and the L_∞ Metrics

In this section, we present a fixed-parameter algorithm for BST-FT-ST in the L_∞ (and equivalently the L_1) metric. Thus, throughout this section, $d(a, b)$ denotes the L_∞ distance between two points $a, b \in \mathbb{R}^2$. Recall that we are given an abstract topology $\mathcal{T}_0$ on $V = \{s_1, \ldots, s_k, t_1, \ldots, t_{c+1}\}$. Also, we assume that every internal vertex of $\mathcal{T}_0$ is a Steiner vertex. If we have $t_i \in V$ which is an internal vertex in $\mathcal{T}_0$, we can just split $\mathcal{T}_0$ at t_i into two abstract topologies and consider each separately since any optimal placement of Steiner points in one is independent from the terminals in the other.

We first consider the decision version of BST-FT-ST: for a given real value $\lambda > 0$, we would like to answer whether there exists a placement of k Steiner points such that

the maximum edge length in the resulting Steiner tree with the same topology as $\mathcal{T}_0$ is at most λ. Once we obtain an efficient decision algorithm, we do a binary search on *critical values* of λ to find the smallest λ^* that yields the positive answer "YES". Thus, λ^* is the optimal bottleneck value for a given abstract topology $\mathcal{T}_0$.

We should mention that the optimal objective value λ^* may be determined by $|e_{c+1}|$, the longest edge that is left from $MST(P)$. We, however, consider only the edges incident to any Steiner point and optimize their lengths; afterwards, we can simply compare the obtained value to $|e_{c+1}|$. Thus, in the remaining of this paper, we assume that λ^* is always determined by the length of an edge incident to a Steiner point.

3.1 Decision Algorithm

Our algorithm can be seen as an extension of the algorithm by Ganley and Salowe [5]; namely, from concrete topologies to abstract topologies. We choose an arbitrary Steiner vertex of $\mathcal{T}_0$ as a root, and consider $\mathcal{T}_0$ as a rooted tree. Let $C(v)$ be the set of children of a vertex v in $\mathcal{T}_0$. For each vertex v of $\mathcal{T}_0$, we shall associate a region $R_\lambda(v)$ as follows. If $v = t_i$, representing a subtree T_i of $MST(P)$, then $R_\lambda(v)$ is the set of terminals contained in T_i. Otherwise, if v is a Steiner vertex, $R_\lambda(v) := \bigcap_{u \in C(v)} R_\lambda(u) \oplus B_\lambda$, where B_λ denotes the L_∞-ball (i.e., a square) of radius λ centered at the origin and $\oplus$ denotes the Minkowski sum operation. We compute the regions $R_\lambda(v)$ in a bottom-up manner starting from the leaves, and answer "YES" if $R_\lambda(v) \neq \emptyset$ for all vertices v of $\mathcal{T}_0$; otherwise, report "NO". The correctness of the algorithm is straightforward.

To compute the regions $R_\lambda(v)$, we shall show that $R_\lambda(v)$ can be simply described in terms of squares centered at the terminals in P. We denote by $B(p, r)$ the L_∞-ball of radius r centered at point $p \in \mathbb{R}^2$. Also, for two vertices u and v in $\mathcal{T}_0$, let $\delta_{u,v}$ be the length of the path between u and v (i.e. the number of edges) in $\mathcal{T}_0$, and $L(v)$ be the set of leaves which are descendants of v in $\mathcal{T}_0$.

Lemma 5. *Let v be an internal vertex of a given abstract topology $\mathcal{T}_0$. Suppose $R_\lambda(u) \neq \emptyset$ for all descendants u of v in $\mathcal{T}_0$. Then, we have $R_\lambda(v) = \bigcap_{t_i \in L(v)} \bigcup_{p \in T_i} B(p, \lambda \cdot \delta_{t_i,v})$.*

Lemma 5 tells us the complexity of $R_\lambda(v)$ as well as how to compute it.

Lemma 6. *The decision version of BST-FT-ST can be solved in $O(kn^2)$ time.*

3.2 Optimization Algorithm

We shall show that the exact solution of BST-FT-ST can be obtained by $O(\log n)$ implementations of the decision algorithm. Following is a key lemma for our purpose.

Lemma 7. *Let λ^* be the smallest real number that the decision algorithm reports YES. Then, there exists a Steiner vertex s_i such that the area of $R_{\lambda^*}(s_i)$ is zero.*

For any Steiner vertex s_i and two terminals $p \in T_j$ and $p' \in T_{j'}$, consider the value λ such that $B(p, \lambda \cdot \delta_{s_i,t_j})$ touches $B(p', \lambda \cdot \delta_{s_i,t_{j'}})$. We call such λ a *critical value*. Lemma 7 implies that there is s_i such that we have two touching squares $B(p, \lambda^* \cdot \delta_{s_i,t_j})$ and $B(p', \lambda^* \cdot \delta_{s_i,t_{j'}})$ where $p \in T_j$ and $p' \in T_{j'}$; thus, λ^* is also a critical value.

Lemma 8. *BST-FT-ST in the L_∞ metric can be solved in $O(kn^2 \log n)$ time.*

In order to solve BST, we enumerate all possible abstract topologies and solve BST-FT-ST for each. The number of abstract topologies is at most $O((7k+1)^{7k-1})$ by Lemma 4 and Theorem 1. Finally, we obtain a fixed-parameter algorithm for BST in the L_1 or the L_∞ metric.

Theorem 2. *Given n terminals and a positive integer k, a bottleneck Steiner tree with k Steiner points in the L_1 or the L_∞ metric can be exactly computed in $O((7k+1)^{7k} \cdot n^2 \log n)$ time.*

4 Exact Algorithm for the Euclidean Metric and the L_p Metric

In this section we present an algorithm that finds an exact solution to BST-FT-ST in the L_2 (the Euclidean) metric, leading to the first exact algorithm for BST in the Euclidean metric. Throughout this section, $d(a, b)$ denotes the Euclidean distance between two points a and b. Also, we assume that every leaf of the given abstract topology $\mathcal{T}_0$ is a terminal vertex as in Section 3. Note that our algorithm works for any L_p metric by Theorem 1 while we mainly discuss bottleneck Steiner trees in the Euclidean metric.

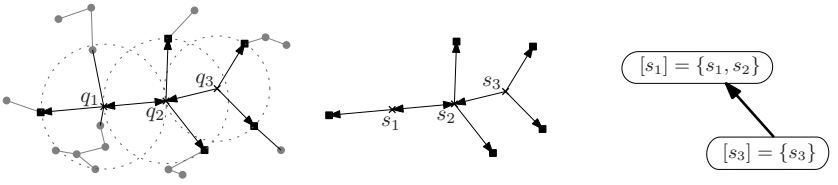
As Ganley and Salowe [5] pointed out earlier, it seems much harder to find an exact solution to BST in the Euclidean metric (or any L_p metric) than in the L_1 metric. One could try a similar approach as done in Section 3 but would immediately face with a difficulty; a nice description of the region $R_\lambda(v)$ like Lemma 5 is hardly obtained. Indeed, that is one of the reasons why no exact algorithm has been discovered yet for the Euclidean case. This also causes another difficulty in collecting critical values λ among which the optimal objective value λ^* can be found. To overcome all difficulty, we make full use of the properties of bottleneck Steiner trees revealed in Section 2, introducing some new concepts; determinators and primary clusters.

4.1 Determinators and Primary Clusters

Consider an optimal placement $Q = \{q_1, \ldots, q_k\}$ of k Steiner points for a given abstract topology $\mathcal{T}_0$. Recall that the corresponding Steiner tree T^* is supposed to satisfy **BST1–4**. Let r_i be the length of the longest edge incident to q_i in T^* and B_i be the disk centered at q_i with radius r_i. By **BST2**, each B_i has two or three points on its boundary, which are among the neighbors of q_i in T^*. Note that a disk is said to be *determined by three points* or *by two diametral points* if the three points lie on the boundary of the disk or if the two points define the diameter of the disk, respectively. We call the three or two points determining a disk the *determinators* of the disk or of its center. Note that if s_i is adjacent to t_j in $\mathcal{T}_0$ we can assume that, in the resulting Steiner tree, q_i is adjacent to the closest terminal in subtree T_j.

Let D_i be the set of determinators of q_i in T^*. D_i consists of two or three points in $P \cup Q$. In the case where there are more than three points on the boundary of B_i, we arbitrarily choose three of them. Then, the following is an immediate observation.

Lemma 9. *If $q_i \in D_j$ in T^*, then $r_i \geq r_j$.*



(a) BST T^* (b) concrete topology $\mathcal{T}$ (c) DAG G on $[S]$

Fig. 2. (a) A bottleneck Steiner tree T^* for abstract topology $\mathcal{T}_0$ in Figure 1; The arrows indicate the determinators D_i of each Steiner points and terminals that are determinators of some q_i are depicted as small solid squares. Dotted circles indicate B_i whose radius is r_i. (b) An induced concrete topology $\mathcal{T}$ taking only terminals that are determinators, $D_i \cap P$, with determinator information; An optimal placement of Steiner points is determined by these selected terminals only. Here, we have a primary cluster $[s_1] = \{s_1, s_2\}$. See $r_1 = r_2 \geq r_3$ in (a). (c) Directed acyclic graph G on $[S]$ based on the determinator relation on $\mathcal{T}$ induces a partial ordering on $[S]$.

Based on the sets of determinators, D_i, we can infer an ordering on the Steiner vertices. First, we build an equivalence relation $\equiv$ on $S = \{s_1, \ldots, s_k\}$; $s_i \equiv s_j$ if and only if $q_i \in D_j$ and $q_j \in D_i$. We denote by $[s_i]$ the equivalence class that includes s_i, and let $[S] := \{[s_i] \mid s_i \in S\}$. Now, we construct a directed graph G on $[S]$ in which there is a directed edge from $[s_i]$ to $[s_j]$ if and only if there are two indices i' and j' with $s_{i'} \in [s_i]$ and $s_{j'} \in [s_j]$ such that $q_{j'} \in D_{i'}$. Obviously, G is acyclic. We denote by $([S], \preceq)$ a partial ordered set induced by G. Lemma 9 implies that if $s_j \in [s_i]$, then $r_i = r_j$, and if $[s_i] \preceq [s_j]$ then $r_{i'} \leq r_{j'}$ for any i', j' with $s_{i'} \in [s_i]$ and $s_{j'} \in [s_j]$. Let $\mathcal{M}$ be the set of all the maximal elements in $[S]$ with respect to $\preceq$. More precisely, $\mathcal{M} = \{[s_i] \in [S] \mid$ there is no other $[s_j] \in [S]$ with $[s_i] \preceq [s_j]\}$. We call each subset of Steiner vertices in $\mathcal{M}$ a *primary cluster*. See Figure 2 for more illustrative explanation.

Since the optimal objective value λ^* for given abstract topology $\mathcal{T}_0$ is determined as $\max r_i$ (as assumed in the beginning of Section 3), it is obvious that λ^* is indeed determined by a primary cluster in $\mathcal{M}$.

Lemma 10. *Assume that all symbols and relations are induced from a bottleneck Steiner tree as above. Then, there always exists a primary cluster* $[s_i] \in \mathcal{M}$ *such that* $\lambda^* = r_i$.

4.2 Algorithm

Our algorithm enumerates all possible combinations of determinators $(D_1, \ldots, D_k)$ for a given abstract topology $\mathcal{T}_0$. By a fixed combination of determinators, the equivalence relation $(S, \equiv)$ and the partial ordering $([S], \preceq)$ are inferred as above. At last, we will be able to collect critical values by handling each primary cluster.

Thus, our algorithm for BST-FT-ST in the L_2 metric is summarized as follows:

(1) Enumerate all possible combinations of determinators D_i for abstract topology $\mathcal{T}_0$.
(2) For a fixed combination of the determinators D_i, build a concrete topology $\mathcal{T}$.
(3) For each primary cluster induced by the D_i, collect critical values.
(4) Do a binary search on the collected critical values to find the optimal value λ^*.

From now on, we describe each step of the algorithm in more details.

(1) Enumeration of all possible combinations of determinators For a given abstract topology $\mathcal{T}_0$, a rough calculation on the number of combinations of determinators gives us $20^k n^{3k}$: (i) we choose two or three neighbors from at most five neighbors of each s_i in $\mathcal{T}_0$ and, (ii) choose one terminal from at most n terminals in T_j if t_j was chosen for s_i at (i).

In order to prune a number of unnecessary combinations of determinators, we bring a known geometric structure, namely, the *farthest color Voronoi diagram*. The farthest color Voronoi diagram is a generalization of the standard farthest-neighbor Voronoi diagram to colored point sets [6,1]: Given a collection $\mathcal{C} = \{P_1, \ldots, P_l\}$ of l sets of colored points, define the *distance to a color* i for $1 \leq i \leq l$ as $d_i(x) := \min_{p \in P_i} d(x, p)$. Then, the farthest color Voronoi diagram $FCVD(\mathcal{C})$ is a decomposition of $\mathbb{R}^2$ into Voronoi regions VR_i for subset P_i, defined to be the set $\{x \in \mathbb{R}^2 \mid d_i(x) > d_j(x), 1 \leq j \leq l, j \neq i\}$. Each edge of $FCVD(\mathcal{C})$ is the set of points that have the same set of two farthest colors; each vertex is a point that has three or more farthest colors. Each Voronoi region VR_i is again refined into cells σ_p for each $p \in P_i$ such that $\sigma_p = \{x \in VR_i \mid d(x, p) = d_i(x) = \min_{q \in P_i} d(x, q)\}$. We regard $FCVD(\mathcal{C})$ as this refined diagram. Note that each cell, edge, or vertex of $FCVD(\mathcal{C})$ is determined by (thus, associated with) one, two, or three points in $\bigcup_i P_i$. For more details about the farthest color Voronoi diagram, we refer to Huttenlocher et al. [6] and Abellanas et al. [1]. We now introduce an interesting relation between the farthest color Voronoi diagram and the determinators.

Lemma 11. *Given the determinators D_i of q_i, let $m := |D_i \cap P|$, the number of terminals that are determinators of q_i. If $m > 0$, then q_i lies on the $(3 - m)$-face ϕ_i of $FCVD(\mathcal{C}_i)$ determined by $D_i \cap P$, where $\mathcal{C}_i = \{V(T_j) \mid t_j$ a neighbor of s_i in $\mathcal{T}_0\}$ and $V(T_j) \subseteq P$ denotes the vertex set of T_j.*

Thus, as a preprocessing, we compute $FCVD(\mathcal{C}_i)$ for each Steiner vertex s_i, where $\mathcal{C}_i$ is defined as in Lemma 11. Since we have at most 5 subsets of P in $\mathcal{C}_i$, the total combinatorial complexity of all $FCVD(\mathcal{C}_i)$ is $O(n)$ and $O(n \log n)$ time is sufficient to build them for any L_p metric with $1 < p < \infty$ [6].

To enumerate combinations of determinators, we first choose one face ϕ_i (of any dimension) from $FCVD(\mathcal{C}_i)$ to fix $D_i \cap P$. Then, for each edge between two Steiner vertices s_i, s_j in $\mathcal{T}_0$, we have four possibilities to fix $D_i \setminus P$; $q_i \in D_j$ or $q_i \notin D_j$, and $q_j \in D_i$ or $q_j \notin D_i$. Thus, we have $(O(n))^k$ combinations of $D_i \cap P$ and 4^k combinations of $D_i \setminus P$. Hence, we have at most $(O(n))^k \cdot 4^k = (O(n))^k$ possible combinations of the determinators in total.

(2) Building a corresponding concrete topology At this step, we are given the determinators D_i for each Steiner point. The D_i induce a concrete topology $\mathcal{T}$ from the abstract topology $\mathcal{T}_0$ by choosing the terminals appeared in any of the D_i. Note that $\mathcal{T}$ consists of at most $3k$ terminals and k Steiner points since D_i contains at most three terminals. We should mention that when building $\mathcal{T}$ from $\mathcal{T}_0$, we drop some edges, which are not related to the D_i. Since an optimal placement of Steiner points is determined only by the D_i, these dropped edges do not matter for getting critical values at Step (3).

(3) Collecting critical values In order to collect critical values λ, we search each primary cluster $[s_i] \in \mathcal{M}$. By Lemma 10, a primary cluster $[s_i]$ among $\mathcal{M}$ causes the longest edge in the resulting Steiner tree T^*; that is, $\lambda^* = r_i = r_j$ for any $s_j \in [s_i]$. Recall that the placement q_j for each $s_j \in [s_i]$ is determined by its determinators D_j. Thus, to precompute (candidates of) the value λ^*, we take a subtopology $\mathcal{T}_i$ of $\mathcal{T}$, which is an induced subtree by $[s_i]$ and their determinators.

In $\mathcal{T}_i$, every leaf is a terminal in P and every internal vertex is a Steiner vertex with degree 2 or 3. Observe that any degree-2 Steiner point is located at the midpoint of its two neighbors by **BST2**. Thus, degree-2 Steiner points are rather easy to find once we know an optimal placement of all degree-3 Steiner points. We modify $\mathcal{T}_i$ into a weighted tree $\mathcal{T}_i'$ by the following operations: (1) Initially, assign weight 1 to every edge of $\mathcal{T}_i$. (2) Whenever there is a degree-2 Steiner vertex, we remove it from $\mathcal{T}_i$ and its two incident edges are merged into one with summed weight.

Now, let m be the number of Steiner vertices in $\mathcal{T}_i'$, and $w(e)$ be the weight of an edge e of $\mathcal{T}_i'$. Then, $\mathcal{T}_i'$ consists of exactly $2m+1$ edges and m internal vertices (all Steiner). Without loss of generality, we assume that $s_1, \ldots, s_m$ are the Steiner vertices in $\mathcal{T}_i'$. For each edge e of $\mathcal{T}_i'$, we assign a function $h_e : (\mathbb{R}^2)^m \to \mathbb{R}$ defined to be

$$h_e((q_1, \ldots, q_m)) := \frac{1}{w(e)}\{\text{the length of } e \text{ when } s_j \text{ is placed at } q_j \in \mathbb{R}^2 \text{ for each } j\}.$$

Let $q_1^*, \ldots, q_m^*$ be an optimal placement of m Steiner points. Then, at $(q_1^*, \ldots, q_m^*)$, we have equality $h_e((q_1^*, \ldots, q_m^*)) = h_{e'}((q_1^*, \ldots, q_m^*))$ for any pair of two edges e, e' of $\mathcal{T}_i'$, by Lemma 9.

The function h_e indeed returns the distance between two points, so it defines an algebraic surface of degree 2 in $\mathbb{R}^{2m+1}$, which is the graph of h_e. Furthermore, at every point $(q_1, \ldots, q_m) \in (\mathbb{R}^2)^m$ that satisfies all the equalities, the $2m+1$ surfaces meet all together at a point $(q_1, \ldots, q_m, \lambda) \in \mathbb{R}^{2m+1}$; this point is a vertex (0-face) of the lower (or upper) envelope of the $2m+1$ surfaces. Fortunately, all 0-faces of the lower envelope of algebraic surfaces in high dimension can be computed by Agarwal et al. [2]; in our case, it costs $O((2m+1)^{2m+\epsilon})$ time for any positive ϵ. Once we find all the 0-faces of the lower envelope, we have at most $O((2m+1)^{2m+\epsilon})$ critical values by taking the height (the $(2m+1)$-st coordinate) of each 0-face of the lower envelope.

We do this procedure for every primary cluster, collecting $O((2k+1)^{2k+\epsilon})$ critical values in total in the same time bound.

(4) Binary search on critical values At this step, we do a binary search on the collected critical values using a decision algorithm running on the concrete topology $\mathcal{T}$. We make use of a modified version of the decision algorithm by Sarrafzadeh and Wong [8]. We choose an arbitrary internal vertex of $\mathcal{T}$ as a root and consider $\mathcal{T}$ as a rooted tree.

By Lemma 11, every Steiner point q_i must lie in the face ϕ_i chosen at Step (1). We thus redefine the region $R_\lambda(s_i)$ for each Steiner vertex as follows:

$$R_\lambda(s_i) := \phi_i \cap \bigcap_{u \in C(s_i)} (R_\lambda(u) \oplus B_\lambda),$$

where $C(s_i)$ is the set of children of s_i in $\mathcal{T}$ and B_λ is a unit disk centered at the origin.

If ϕ_i is a segment or a point, this additional intersection with ϕ_i does not increase the complexity. For easy analysis of the case where ϕ_i is a two-dimensional face, we triangulate each two-dimensional face of $FCVD(\mathcal{C}_i)$ at Step (1) and take ϕ_i as a triangle. Since the triangulation does not increase the complexity of $FCVD(\mathcal{C}_i)$, we still have $O(n)$ number of vertices, edges, and triangular faces in $FCVD(\mathcal{C}_i)$. Also, computing $R_\lambda(s_i)$ is done without additional cost; ϕ_i is either a point, a segment, or a triangle. Since $\mathcal{T}$ consists of $O(k)$ vertices, our modified decision algorithm runs in $O(k \log k)$ time, based on the analysis by Sarrafzadeh and Wong [8].

Hence, for each concrete topology $\mathcal{T}$, we can find an optimal objective value λ^* in $O(k \log k \cdot \log((2k+1)^{2k+\epsilon})) = O(k^2 \log^2 k)$ time. To see the total time complexity, initially we spend $O(n \log n)$ time to compute $FCVD(\mathcal{C}_i)$, Steps (3)–(4) take $O((2k+1)^{2k+\epsilon})$ time, and we repeat Steps (3)–(4) $(O(n))^k$ times; $O((2k+1)^{2k+\epsilon} \cdot (O(n))^k + n \log n)$ time to solve BST-FT-ST in total. Combining this with Lemma 4, we finally conclude:

Theorem 3. *Given n terminals and a positive integer k, a bottleneck Steiner tree with k Steiner points in the L_p metric with $1 < p < \infty$ can be exactly computed in $O(f(k) \cdot (n^k + n \log n))$ time, where $f(k) = O((5k+1)^{5k-1}(2k+1)^{2k+\epsilon}2^{O(k)}) = O(k^{7k} \cdot 2^{O(k)})$.*

Remark that our algorithm only finds an optimal placement Q of k Steiner points. To obtain a bottleneck Steiner tree, it suffices to compute the minimum spanning tree $MST(P \cup Q)$ for points $P \cup Q$.

References

1. Abellanas, M., Hurtado, F., Icking, C., Klein, R., Langetepe, E., Ma, L., Sacristan, V.: The farthest color Voronoi diagram and related problems. In: Proc. 17th European Workshop Comput. Geom., pp. 113–116 (2001)
2. Agarwal, P.K., Aronov, B., Sharir, M.: Computing envelopes in four dimensions with applications. SIAM J. Comput. 26(6), 1714–1732 (1997)
3. Bae, S.W., Lee, C., Choi, S.: On exact solutions to the Euclidean bottleneck Steiner tree problem. In: Das, S., Uehara, R. (eds.) WALCOM 2009. LNCS, vol. 5431, pp. 105–116. Springer, Heidelberg (2009)
4. Chiang, C., Sarrafzadeh, M., Wong, C.: A powerful global router: based on Steiner min-max trees. In: Proc. IEEE Int. Conf. CAD, pp. 2–5 (1989)
5. Ganley, J.L., Salowe, J.S.: Optimal and approximate bottleneck Steiner trees. Oper. Res. Lett. 19, 217–224 (1996)
6. Huttenlocher, D.P., Kedem, K., Sharir, M.: The upper envelope of Voronoi surfaces and its applications. Discrete Comput. Geom. 9, 267–291 (1993)
7. Li, Z.-M., Zhu, D.-M., Ma, S.-H.: Approximation algorithm for bottleneck Steiner tree problem in the Euclidean plane. J. Comput. Sci. Tech. 19(6), 791–794 (2004)
8. Sarrafzadeh, M., Wong, C.: Bottleneck Steiner trees in the plane. IEEE Trans. Comput. 41(3), 370–374 (1992)
9. Wang, L., Du, D.-Z.: Approximations for a bottleneck Steiner tree problem. Algorithmica 32, 554–561 (2002)
10. Wang, L., Li, Z.: An approximation algorithm for a bottleneck k-Steiner tree problem in the Euclidean plane. Inform. Process. Lett. 81, 151–156 (2002)