Algorithmica
DOI 10.1007/s00453-011-9553-y

Exact Algorithms for the Bottleneck Steiner Tree Problem

Sang Won Bae · Sunghee Choi · Chunseok Lee · Shin-ichi Tanigawa

Received: 28 February 2010 / Accepted: 7 July 2011

Abstract Given n points, called terminals, in the plane $\mathbb{R}^2$ and a positive integer k, the bottleneck Steiner tree problem is to find k Steiner points from $\mathbb{R}^2$ and a spanning tree on the $n+k$ points that minimizes its longest edge length. Edge length is measured by an underlying distance function on $\mathbb{R}^2$, usually, the Euclidean or the L_1 metric. This problem is known to be NP-hard. In this paper, we study this problem in the L_p metric for any $1 \leq p \leq \infty$, and aim to find an exact algorithm which is efficient for small fixed k. We present the first fixed-parameter tractable algorithm running in $f(k) \cdot n \log^2 n$ time for the L_1 and the L_∞ metrics, and the first exact algorithm for the L_p metric for any fixed rational p with $1 < p < \infty$ whose time complexity is $f(k) \cdot (n^k + n \log n)$, where $f(k)$ is a function dependent only on k.

A preliminary version of this paper was presented at the 26th International Symposium on Algorithms and Computation (ISAAC 2009). Work by S.W. Bae was supported by the Contents Convergence Software Research Center funded by the GRRC Program of Gyeonggi Province, South Korea. Work by C. Lee and S. Choi was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2010-0024092). Work by S. Tanigawa was supported by Grant-in-Aid for JSPS Research Fellowship for Young Scientists.

S.W. Bae (✉)
Department of Computer Science, Kyonggi University, Suwon, South Korea
e-mail: swbae@kgu.ac.kr

S. Choi · C. Lee
Division of Computer Science, KAIST, Daejeon, South Korea

S. Choi
e-mail: sunghee@tclab.kaist.ac.kr

C. Lee
e-mail: stonecold@tclab.kaist.ac.kr

S.-i. Tanigawa
Research Institute for Mathematical Sciences, Kyoto University, Kyoto, Japan
e-mail: tanigawa@kurims.kyoto-u.ac.jp

Published online: 30 July 2011

Note that prior to this paper there was no known exact algorithm even for the L_2 metric.

Keywords Steiner point · Bottleneck Steiner tree · Exact algorithm · Fixed-parameter tractability · Computational geometry · Optimization

1 Introduction

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

Problem 1 (BST) *Given n points, called* terminals, *in the plane* $\mathbb{R}^2$ *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, where the length of an edge is measured by an underlying metric on* $\mathbb{R}^2$.

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 from the whole plane $\mathbb{R}^2$.

The bottleneck Steiner tree problem (shortly BST) has been studied with many applications, like VLSI layout [7], multi-facility location, and wireless communication network design [26]. Previous works on BST considered the Euclidean plane (L_2) or the rectilinear plane (L_1) as underlying metrics. The problem BST is known to be NP-hard to approximate within ratio $\sqrt{2}$ in the L_2 metric or within ratio 2 in the L_1 metric [26]. For the L_1 metric, the matching upper bound was known by Wang and Du [26] and for the L_2 metric, the best known upper bounds on approximation ratio is 1.866 by Wang and Li [27]. For the special case of this problem where there must be no edge connecting any two Steiner points in the optimal solution, Li et al. [16] presented 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) [12, 22]:

Problem 2 (BST-FT) *Given a set P of n terminals in the plane and a topology* $\mathcal{T}$ *on the terminal set and k Steiner points, find an optimal placement of k Steiner points to obtain a bottleneck Steiner tree that has the same topology as* $\mathcal{T}$.

Once we have an exact algorithm for this problem BST-FT, we can find an exact solution to BST by enumerating all valid tree topologies. For the L_1 (or the L_∞) metric, Ganley and Salowe [12] 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 is much worse. There was no

known exact algorithm for BST in the L_2 metric but few partial results: The first algorithms were heuristics based on nonlinear optimization due to Elzinga et al. [10] and Love et al. [17]. Sarrafzadeh and Wong [22] presented an $O((n+k)\log(n+k))$ time algorithm for the decision version of BST-FT, which can be exploited to devise a simple $(1+\epsilon)$-approximation algorithm for BST-FT, as Ganley and Salowe [12] pointed out. More recently, Bae et al. [4] presented a fixed-parameter algorithm with running time $O(f(k)\cdot n\log n)$ for a special case in which no edge between Steiner points is allowed and two exact algorithms for $k\leq 2$: $O(n\log n)$ time for $k=1$ and $O(n^2)$ time for $k=2$.

Nonetheless, there are some known exact algorithms for the Euclidean Steiner tree problem (under the L_2 metric), where the sum of the edges in the resulting tree is minimized. Enumerating candidate topologies, one of which is of an optimal tree, is seemingly a common approach to obtain exact algorithms for variations of hard Steiner tree problems. Most of them, including the best known one [28], were achieved based on this approach. For more references, see Warme, Winter, and Zachariasen [28].

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 $O(f(k)\cdot n\log^2 n)$-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 for any rational p 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. In particular, the algorithm for the L_1 and the L_∞ metrics has running time of the form $f(k)n^{O(1)}$, which shows that BST in the L_1 or the L_∞ metric is *fixed-parameter tractable* with respect to parameter k, the number of allowed Steiner points. For more information on fixed-parameter tractability or parameterized complexity, see the monograph by Niedermeier [19]. On the other hand, we failed to achieve fixed-parameter tractability for the other cases but it is worth noting that our algorithm for the L_p metric with $1<p<\infty$ is the first exact algorithm. An exact algorithm for the L_2 metric was not known and remained open for a long time after Sarrafzadeh and Wong [22].

The remaining of the paper is organized as follows: Sect. 2 reveals several common properties of optimal solutions to BST in any L_p metric. We present the first fixed-parameter tractable algorithm for the L_1 and the L_∞ metrics in Sect. 3 and the first exact algorithm for the L_p metric with any $1<p<\infty$ in Sect. 4. Finally, Sect. 5 concludes the paper with remarks and open issues.

2 Properties of Bottleneck Steiner Trees

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 $a,b\in\mathbb{R}^2$.

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. A *minimum spanning tree* of P

is a spanning tree of P having minimum sum of the edge lengths over all spanning trees of P. Note that a minimum spanning tree of P is a bottleneck spanning tree of P; one can easily check this using its optimality. In general, there can be more than one minimum spanning tree of P. Throughout this paper, we denote by $MST(P)$ an arbitrary (but fixed) minimum spanning tree of P. We let $e_1, \ldots, e_{n-1}$ be the edges of $MST(P)$ in the order that their lengths are not increasing, and let $|e_i|$ denote the length of e_i with respect to the L_p metric.

A *Steiner tree* with k Steiner points Q for P is a spanning tree of $P \cup Q$. 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 Steiner points that minimizes the bottleneck $b(P \cup Q)$ of $P \cup Q$ over all sets of k points in $\mathbb{R}^2$.

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 *Let P be a set of terminals in the plane. Then, there always exists a bottleneck Steiner tree T^* with k Steiner points for P that satisfies the following conditions **BST1–4**:*

BST1 *Each edge in T^* either belongs to $MST(P)$ or is incident to a Steiner point.*

BST2 *Each Steiner point 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 $1 \le c \le (\Delta - 1)k$ such that T^* excludes $e_1, \ldots, e_c$ but includes $e_{c+1}, \ldots, e_{n-1}$.*

Bae et al. [4] have proved a special case of the theorem for the L_2 metric, and then exploited it to devise exact algorithms for the Euclidean bottleneck Steiner tree problem for $k \le 2$.

Observe that **BST1** easily follows from the optimality of the minimum spanning tree (the proof can be found also in Bae et al. [4]). Furthermore, we can locally rearrange the locations of Steiner points to satisfy **BST2** keeping its maximum edge length and combinatorial structure unchanged. 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 $d(p_i, q) \le d(p_i, p_j)$ for any $1 \le i, j \le m$ with $i \ne j$, then we have $m \le \Delta + 1$ (i.e., $m \le 8$ for the L_1 and the L_∞ metrics; $m \le 6$ for the L_p metric with $1 < p < \infty$). Moreover, the equality $m = \Delta + 1$ holds only if we have equality $d(p_i, q) = d(p_i, p_{i-1}) = d(p_i, p_{i+1})$ for any $2 \le i \le m - 1$, $d(p_1, q) = d(p_1, p_2)$, and $d(p_m, q) = d(p_{m-1}, p_m)$.*

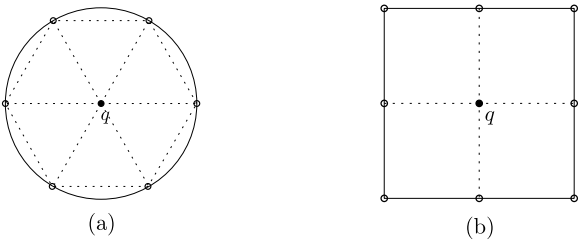

Fig. 1 An illustration to Lemma 1: (**a**) the L_2 metric, (**b**) the L_∞ metric

To illustrate this lemma, one can imagine the Euclidean plane where the 6 points p_i are taken as the vertices of a regular hexagon and q as its center; for the L_∞ metric, the 8 points are taken from each vertex and the midpoint of each side of a square. (See Fig. 1.) Lemma 1 can be shown by the *Hadwiger number* of a convex body in $\mathbb{R}^2$. For smooth flow of discussion, we postpone its proof to Appendix.

Based on Lemma 1, we can prove the following.

Lemma 2 *There exists a bottleneck Steiner tree T^* with k Steiner points for P that satisfies **BST1–3**.*

Proof Suppose, for a contradiction, that there exists no bottleneck Steiner tree with k Steiner points satisfying **BST1–3**. Consider the set of all possible bottleneck Steiner trees with k Steiner points for P with the set Q of k Steiner points satisfying **BST1–2**. Such bottleneck Steiner trees always exist as discussed above. We take a bottleneck Steiner tree T^* among them satisfying the following criteria: (i) the total edge length is minimum and (ii) the number of *heavy points* is minimum among those satisfying (i), where a point of $P \cup Q$ is said to be *heavy* if its degree in T^* is greater than Δ.

Let $q \in P \cup Q$ be a point taking the maximum degree in T^*. Since T^* does not satisfy **BST3**, q is heavy. We consider the following two cases: the degree of q is greater than $\Delta + 1$ or equal to $\Delta + 1$.

We denote by $p_1, \ldots, p_m$ the neighbors of q in T^* in counter-clockwise order around q. If the degree of q is greater than $\Delta + 1$, then there exist two points p_i and p_j such that $d(q, p_i) > d(p_i, p_j)$ by Lemma 1. Hence, if we remove the edge between q and p_i and connect p_i to p_j by a new (shorter) edge, we obtain a new bottleneck Steiner tree with shorter total edge length than T^*, a contradiction to the minimality (i) of T^*.

Thus, the degree m of q must be equal to $\Delta + 1$. If there exist p_i and p_j such that $d(p_i, q) > d(p_i, p_j)$, then we can modify T^* by replacing the edge between p_i and q by the edge connecting p_i and p_j to obtain another bottleneck Steiner tree with smaller total edge length than T^* a contradiction to the minimality (i) of T^*. This implies that q and its neighbors $p_1, \ldots, p_m$ satisfy the assumption of Lemma 1. We further have that $d(q, p_i) = d(p_i, p_{i-1}) = d(p_i, p_{i+1})$ for any $1 \le i \le \Delta + 1$ by Lemma 1 since $m = \Delta + 1$.

We claim that the degree of each p_i is at most $\Delta - 1$. Let N be the set of neighbors of p_i in T^* for any fixed i. Note that $q \in N$. Observe that

$$d(p_i, r) \le d(r, r') \quad \text{for any } r, r' \in N \text{ with } r \ne r',$$

since otherwise replacing the edge rp_i by rr' results in another bottleneck Steiner tree with smaller total edge length, contracting the minimality (i) of T^*. The same argument can apply to show

$$d(p_i, r) \leq d(p_{i-1}, r) \quad \text{and} \quad d(p_i, r) \leq d(p_{i+1}, r) \quad \text{for any } r \in N.$$

Moreover, we also have

$$d(p_i, p_{i-1}) = d(p_i, q) \leq d(p_{i-1}, r),$$

since otherwise removing the edge $p_i q$ and adding edge rp_{i-1} again results in another bottleneck Steiner tree with smaller total edge length (where we used the fact that the path connecting r and p_{i-1} in T^* passes through the edge $p_i q$), contradicting the minimality (i) of T^*. Symmetrically, we have

$$d(p_i, p_{i+1}) = d(p_i, q) \leq d(p_{i+1}, r).$$

Putting it all together, the set $N \cup \{p_{i-1}, p_i, p_{i+1}\}$ fulfills the assumption of Lemma 1 for p_i (as the center point), i.e., for any $r, r' \in N \cup \{p_{i-1}, p_{i+1}\}$ with $r \neq r'$, we have $d(p_i, r) \leq d(r, r')$. This implies $|N| + 2 \leq \Delta + 1$, and thus the degree $|N|$ of p_i in T^* for any $1 \leq i \leq m$ is at most $\Delta - 1$.

By the above claim, we can remove the edge between q and p_{i+1} and connect p_i and p_{i+1} to have another bottleneck Steiner tree with a smaller number of heavy points than that of T^*, keeping the total edge length. This contradicts the minimality (ii) of T^*. Consequently, we conclude that there exists a bottleneck Steiner tree satisfying **BST1–3**, proving the lemma. □

Lemma 3 *There exist a bottleneck Steiner tree T^* for the terminals P and a positive integer c such that T^* fulfills **BST1–3** and excludes $e_1, \ldots, e_c$ but includes $e_{c+1}, \ldots, e_{n-1}$.*

Proof By Lemma 2, there is a bottleneck Steiner tree T^* with k Steiner points satisfying **BST1–3**. Assume that T^* does not satisfy the latter condition of this lemma, i.e., there is an integer i such that T^* excludes e_i but includes e_{i-1}. Let Q be the set of k Steiner points of T^*. Obviously, $b(P \cup Q) \geq |e_{i-1}|$. Let $p, p' \in P$ be the endpoints of e_i. Now, consider the path Π between p and p' in T^*. By **BST1**, Π must contain a Steiner point $q \in Q$ and an edge e incident to q. We modify T^* to obtain another Steiner tree T' with the same set Q of Steiner points by removing e from T^* and adding e_i. Since $|e_i| \leq |e_{i-1}| \leq b(P \cup Q)$, adding e_i does not increase the length of the longest edge of the tree. Note that this operation does not harm the conditions **BST1–3** since e_i is an edge connecting two terminals P. Thus, repeating this process, we obtain the resulting tree T' to satisfy the claimed condition. □

We are now ready to prove Theorem 1.

Proof of Theorem 1 Let T^* be a bottleneck Steiner tree with k Steiner points that fulfills the conditions of Lemma 3, i.e., T^* fulfills **BST1–3** and T^* excludes $e_1, \ldots, e_c$ but includes $e_{c+1}, \ldots, e_{n-1}$ for some integer c. Thus, it suffices to show $c \leq (\Delta - 1)k$.

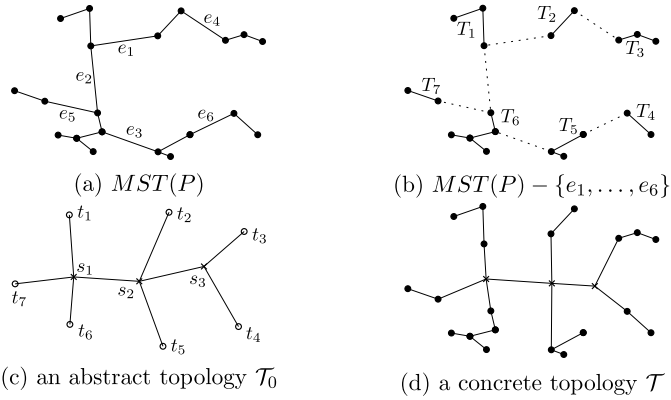

Fig. 2 An illustration of the case when $k = 3$ and $c = 6$. (**a**) From $MST(P)$, (**b**) remove $c = 6$ longest edges $e_1, \ldots, e_6$ to have $c + 1 = 7$ subtrees $T_1, \ldots, T_7$. (**c**) An abstract topology $\mathcal{T}_0$ is a tree on the t_i, representing these subtrees, and the s_i, representing Steiner points. (**d**) A possible concrete topology $\mathcal{T}$ on $P \cup \{s_1, s_2, s_3\}$

Let m_1 be the number of edges of T^* which are contained in $MST(P)$, and m_2 be the number of edges of T^* which are incident to some Steiner point. By **BST1**, we have $m_1 + m_2 = n + k - 1$. Also, by **BST3**, $m_2 \leq \Delta k$ holds. This implies that $m_1 \geq n - 1 - (\Delta - 1)k$. Since T^* includes $e_{c+1}, \ldots, e_{n-1}$, we obtain $c \leq (\Delta - 1)k$. □

Based on Theorem 1, we now describe our strategy for solving BST: For each c with $1 \leq c \leq (\Delta - 1)k$, we remove $e_1, \ldots, e_c$ from $MST(P)$, resulting in $c + 1$ induced subtrees $T_1, \ldots, T_{c+1}$. We then 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.

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

Problem 3 (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 k Steiner points and the $c+1$ subtrees $T_1, \ldots, T_{c+1}$ (which are obtained from $MST(P)$ by removing $e_1, \ldots, e_c$ as above), find an optimal placement of k Steiner points to obtain a bottleneck Steiner tree that has the same topology as $\mathcal{T}_0$.*

Throughout this paper and also earlier works [12, 22], the term "topology" is used to denote a tree whose nodes are labeled by a single or multiple terminals in P or symbolically by each Steiner point without its location. Thus, a topology can be seen as a combinatorial structure that consists of information only about how the terminals and Steiner points are connected, which can be obtained from a Steiner tree. The topology $\mathcal{T}_0$ appeared in the above problem description consists of $c + k + 1$ vertices $V := \{s_1, \ldots, s_k, t_1, \ldots, t_{c+1}\}$, where t_i represents subtree T_i consisting of terminals in P and s_j represents a Steiner point. 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. Also, to distinguish the topology $\mathcal{T}_0$ of the above problem from a topology tree $\mathcal{T}$ used in BST-FT [12, 22], we call $\mathcal{T}_0$ an *abstract topology* while a topology $\mathcal{T}$ on terminals is called a *concrete topology*. Note that a concrete topology is obtained from an abstract topology $\mathcal{T}_0$ by replacing each t_i with its corresponding subtree T_i; Fig. 2 illustrates an example of our problem instance. A Steiner point cannot have degree 1 but may have degree 2 in a bottleneck Steiner tree [22]. Together with **BST3**, we can assume that each s_j has degree between 2 and Δ in $\mathcal{T}_0$.

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 *Let k be the number of allowed Steiner points. Then, the total number of abstract topologies over all $1 \le c \le (\Delta - 1)k$ is bounded by $O((\Delta k + 1)^{\Delta k - 1})$. Moreover, all the abstract topologies can be enumerated in the same time bound.*

Proof Note that each t_i in an abstract topology must be distinguished from any other node since each represents the subtree T_i. But the Steiner vertices s_i are indeed indistinguishable to each other since switching any two Steiner vertices s_i and s_j does not affect the resulting Steiner tree. Thus, for fixed c, an abstract topology can be seen as a tree with $c + 1$ labeled and k non-labeled nodes, where each t_i is labeled while each Steiner vertex s_i is non-labeled.[1] We just give any label to k Steiner vertices so that we regard an abstract topology as a labeled tree with $k + c + 1$ nodes. Then, the number of abstract topologies for fixed c is bounded by $(c + k + 1)^{c+k-1}$ using the Prüfer code [20] (see also [24, Chap. 5]). Summing this number over $1 \le c \le (\Delta - 1)k$, we get the claimed bound.

Enumerating all labeled trees is done in $O(1)$ time per each [23]. □

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

In this section, we present a fixed-parameter tractable algorithm for BST in the L_∞ (and equivalently the L_1) metric with parameter k, the number of allowed Steiner points. Thus, throughout this section, $d(a, b)$ denotes the L_∞ distance between two points $a, b \in \mathbb{R}^2$. Following the strategy explained in the previous section, we devise an exact algorithm for BST-FT-ST, where we are given an abstract topology $\mathcal{T}_0$ on $V = \{s_1, \ldots, s_k, t_1, \ldots, t_{c+1}\}$ for $c \le (\Delta - 1)k = 6k$.

In the subsequent discussion, we focus on the case when 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 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 of the terminals in the other, we can get an optimal placement of Steiner points by combining solutions of

[1] Note that labeled nodes are distinguished from each other while nodes without label are not. In our case, Steiner vertices do not have to be distinguished as discussed.

the subproblems. We thus split the problem at all internal terminal vertices of $\mathcal{T}_0$. Note that the problem is split roughly into at most $\Delta k + 2 = 7k + 2$ subproblems since the number of nodes in $\mathcal{T}_0$ is at most $7k + 1$. Thus, if we can solve the problem for this special case (i.e., no internal terminal node in $\mathcal{T}_0$) in $O(g(n,k))$ time for some function g, then we can solve the problem for general $\mathcal{T}_0$ in $O(kg(n,k))$ time. We hence assume that every internal vertex of $\mathcal{T}_0$ is a Steiner vertex, and we will multiply the time complexity by k at the end.

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 and a corresponding Steiner tree T realizing the abstract topology $\mathcal{T}_0$ such that the length of every edge in T is at most λ. Once we obtain an efficient decision algorithm, we adopt the parametric search technique to find the smallest real value λ^* that yields the positive answer "YES" by the decision algorithm. 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 [12]; 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 associate a region $R_\lambda(v)$ as follows. If $v = t_i$, representing subtree T_i of $MST(P)$, then $R_\lambda(v)$ is defined to be the set of all terminals contained in T_i. Otherwise, if v is a Steiner vertex, then

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

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 of $\mathcal{T}_0$, 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 from definition of the regions $R_\lambda(v)$.

To compute the regions $R_\lambda(v)$, we show that $R_\lambda(v)$ can be simply described in terms of squares centered at the terminals P. We denote by $B(p,r)$ the L_∞-ball of radius r centered at a 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 in $\mathcal{T}_0$ (i.e. the number of edges along the path), and let $L(v)$ 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*

$$R_\lambda(v) = \bigcap_{t_i \in L(v)} \bigcup_{p \in T_i} B(p, \lambda \cdot \delta_{t_i,v}). \tag{1}$$

Proof Let $l(v)$ denote the height of each vertex v in $\mathcal{T}_0$, i.e., $l(v) = 0$ if v is a leaf; otherwise, $l(v) = \max_{u \in C(v)} l(u) + 1$. The proof is done by induction on the height $l(v)$ of v. Note that for any internal vertex v, we have $l(v) \geq 1$.

The base case where $l(v) = 1$ is straightforward. If $l(v) = 1$, then $L(v) = C(v)$ and, by definition of $R_\lambda(v)$, we have

$$R_\lambda(v) = \bigcap_{u \in L(v)} R_\lambda(u) \oplus B_\lambda = \bigcap_{t_i \in L(v)} \bigcup_{p \in T_i} B(p, \lambda \cdot \delta_{t_i,v}),$$

as claimed.

Now, consider the general case where $l(v) > 1$. Observe that $l(v) > l(u)$ for any child $u \in C(v)$ of v. Combining the induction hypothesis with the definition of $R_\lambda(v)$, we thus have

$$R_\lambda(v) = \bigcap_{u \in C(v)} \left(\bigcap_{t_i \in L(u)} \bigcup_{p \in T_i} B(p, \lambda \cdot \delta_{t_i,u}) \right) \oplus B_\lambda. \tag{2}$$

For any two squares $B(p_1, r_1)$ and $B(p_2, r_2)$, it is not difficult to see

$$(B(p_1, r_1) \cup B(p_2, r_2)) \oplus B_\lambda = (B(p_1, r_1) \oplus B_\lambda) \cup (B(p_2, r_2) \oplus B_\lambda)$$

as well as

$$(B(p_1, r_1) \cap B(p_2, r_2)) \oplus B_\lambda = (B(p_1, r_1) \oplus B_\lambda) \cap (B(p_2, r_2) \oplus B_\lambda)$$

if $B(p_1, r_1) \cap B(p_2, r_2) \neq \emptyset$. By the assumption (of the lemma) that $R_\lambda(u) \neq \emptyset$ for all $u \in C(v)$, the right hand side of (2) is simplified as follows:

$$\begin{aligned} R_\lambda(v) &= \bigcap_{u \in C(v)} \bigcap_{t_i \in L(u)} \left(\bigcup_{p \in T_i} B(p, \lambda \cdot \delta_{t_i,u}) \oplus B_\lambda \right) \\ &= \bigcap_{u \in C(v)} \bigcap_{t_i \in L(u)} \bigcup_{p \in T_i} B(p, \lambda \cdot (\delta_{t_i,u} + 1)) \\ &= \bigcap_{t_i \in L(v)} \bigcup_{p \in T_i} B(p, \lambda \cdot \delta_{t_i,v}). \end{aligned}$$

□

Lemma 5 tells us the complexity of $R_\lambda(v)$ as well as how to compute it. Indeed, since equation (1) of Lemma 5 consists of intersections and unions of at most n squares, the complexity of $R_\lambda(v)$ is $O(n^2)$ and it can be constructed in $O(n^2)$ time by computing

the arrangement of the squares. Therefore, the decision version of BST-FT-ST can be solved in $O(kn^2)$ time. A more careful analysis reduces nearly a factor of n for small k.

Lemma 6 *The decision version of BST-FT-ST can be solved in* $O(k^2 n \log n + k^4 n)$ *time.*

Proof We show that the complexity of $R_\lambda(v)$ is $O(k^2 n)$ for any internal (i.e., Steiner) vertex v of $\mathcal{T}_0$, provided that $R_\lambda(u) \neq \emptyset$ for all $u \in C(v)$. For each $t_i \in L(v)$, we denote $\bigcup_{p \in T_i} B(p, \lambda \cdot \delta_{t_i,v})$ by $R_\lambda(v, t_i)$ for convenience. Then, by Lemma 5, $R_\lambda(v)$ is the intersection of $R_\lambda(v, t_i)$ for all $t_i \in L(v)$. Since $R_\lambda(v, t_i)$ is the union of the squares of width $\lambda\delta_{t_i,v}$, any axis-parallel section of $R_\lambda(v, t_i)$ has the length at least $\lambda\delta_{t_i,v}$. This implies that, for any $t_j \in L(v)$ and $p \in T_j$, a side of the square $B(p, \lambda \cdot \delta_{t_j,v})$ can intersect the boundary of $R_\lambda(v, t_i)$ at most $\lambda\delta_{t_i,v}/\lambda\delta_{t_j,v}$ times, which amounts to $O(k)$. Thus, for each $t_j \in L(V)$ and $p \in T_j$, any side of the square $B(p, \lambda \cdot \delta_{t_j,v})$ is chopped into at most $O(k^2)$ smaller pieces by the boundaries of $R_\lambda(v, t_i)$, $t_i \in L(v)$ since $|L(v)| = O(k)$. This finally shows that $R_\lambda(v)$ has complexity at most $O(k^2 n)$ since at most n squares are involved in $R_\lambda(v)$.

Our decision algorithm computes $R_\lambda(v)$ in a bottom-up fashion. Here, the bottom-up fashion means that we handle the vertices in V in nondecreasing order with respect to the label $l(v)$ for vertices $v \in V$, where the label $l(v)$ is as defined in the proof of Lemma 5. The algorithm stops and reports `NO` as soon as it finds a vertex $v \in V$ such that $R_\lambda(v) = \emptyset$; if all the regions $R_\lambda(v)$ turn to be nonempty, the algorithm reports `YES`. The correctness of the algorithm is straightforward as we discussed above. Moreover, if the algorithm is in computing $R_\lambda(v)$ during its execution, then we are sure that $R_\lambda(u) \neq \emptyset$ for all $u \in C(v)$ by the order $l(v)$ of vertices.

In order to compute $R_\lambda(v)$ for vertex $v \in V$, we first compute $R_\lambda(v, t_i)$ for all $t_i \in L(v)$. Recall that $R_\lambda(v, t_i)$ is the union of $|T_i| \leq n$ identical squares. It is known that such a union has linear complexity [14]. Thus, the total complexity of $R_\lambda(v, t_i)$ for all $t_i \in L(v)$ is bounded by $O(n)$. Computing $R_\lambda(v, t_i)$ can be done in total $O(n \log n)$ time using the L_∞ Voronoi diagram for the terminals in T_i since $R_\lambda(v, t_i)$ is the union of the intersection between $B(p, \lambda \cdot \delta_{t_i,v})$ and the Voronoi region of p for every terminal p in T_i. Then, for the boundaries of $R_\lambda(v, t_i)$ for all $t_i \in L(v)$, we run an optimal output-sensitive algorithm [5] that computes the arrangement of line segments in the plane, and then we obtain $R_\lambda(v)$ from the arrangement. Its running time is bounded by $O(n \log n + k^2 n)$ time in our case. Hence, the algorithm takes at most $O(kn \log n + k^3 n)$ time for solving the decision version of BST-FT-ST in the case when each internal vertex of $\mathcal{T}_0$ is a Steiner vertex. As we explained above, this implies an algorithm for solving the decision version of BST-FT-ST for general $\mathcal{T}_0$ in $O(k^2 n \log n + k^4 n)$ time. □

3.2 Optimization Algorithm

In this subsection, we 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 for which the decision algorithm reports* YES. *Then, there exists a Steiner vertex s_i such that the area of $R_{\lambda^*}(s_i)$ is zero.*

Proof Lemma 5 implies that the area of $R_\lambda(s_i)$ is continuously increasing with respect to λ for each Steiner point s_i. Therefore, if the area of $R_{\lambda^*}(s_i)$ is positive for all s_i, then there exists a small positive real ϵ such that $R_{\lambda^*-\epsilon}(s_i) \neq \emptyset$, which contradicts the optimality of λ^*. □

For any Steiner vertex $s_i \in S$ 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 exists $s_i \in S$ 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. Note that there are at most kn^2 critical values; a critical value is identified by one Steiner vertex and two terminals.

A naive method solves BST-FT-ST in the L_∞ metric in $O(kn^2 \log(n+k))$ time; first specify all critical values and sort them, and then do a binary search on critical values using the decision algorithm presented above. We present an improved optimization algorithm by applying the (sorting-based) parametric search technique.

Lemma 8 *BST-FT-ST in the L_∞ metric can be solved in $O(k^2 n \log^2 n + k^4 n \log n)$ time.*

Proof Lemma 7 implies that there are two rectangles $B(p, \lambda^* \cdot \delta_{s_i,t_j})$ and $B(p', \lambda^* \cdot \delta_{s_i,t_{j'}})$ which touch each other at the optimal value λ^*. In other words, there are two terminals $p, p' \in P$ and two integers $1 \leq \delta, \delta' \leq k$ such that $B(p, \delta\lambda^*)$ touches $B(p', \delta'\lambda^*)$. We assume that $B(p, \delta\lambda^*)$ touches $B(p', \delta'\lambda^*)$ along their vertical sides. The other case can be handled in a symmetric way.

Let $x_{p,\delta,\text{left}}(\lambda)$ and $x_{p,\delta,\text{right}}(\lambda)$ be the x-coordinates of the left and the right vertical sides of the rectangle $B(p, \delta\lambda)$, respectively, and let

$$\mathcal{X}(\lambda) := \{x_{p,\delta,\theta}(\lambda) \mid p \in P, 1 \leq \delta \leq k, \theta \in \{\text{left}, \text{right}\}\}.$$

Note that $x_{p,\delta,\text{left}}(\lambda)$ is a linear function with respect to λ on the range $\lambda \geq 0$.

Now, consider sorting the elements of $\mathcal{X}(\lambda^*)$ by a comparison-based parallel sorting algorithm with unknown value of λ^*. Although we do not know the value of λ^*, we can decide whether $x_{p,\delta,\theta}(\lambda^*) \geq x_{p',\delta',\theta'}(\lambda^*)$ holds or not by solving a decision problem at λ such that $x_{p,\delta,\theta}(\lambda) = x_{p',\delta',\theta'}(\lambda)$. Therefore, we can sort $\mathcal{X}(\lambda^*)$ in $O((n + D \log n) \log n)$ time by applying Megiddo's parametric search technique [18] with a parallel sorting algorithm that works in $O(\log n)$ steps with $O(n)$ processors [2], where D is the time taken by a decision algorithm. It is further known for a sorting-based parametric search that Cole's technique [8] can reduce the number of comparisons to trim $O(\log n)$-factor, and the total time complexity can be reduced to $O(n \log n + D \log n)$. (See also [25] for a practical replacement of Cole's technique.)

We label the elements of $\mathcal{X}(\lambda^*)$ in the sorted order by $x_1(\lambda^*), x_2(\lambda^*), \ldots, x_{2n}(\lambda^*)$. Then, since there are $p, p' \in P$ and $1 \leq \delta, \delta' \leq k$ such that $B(p, \delta\lambda^*)$ touches

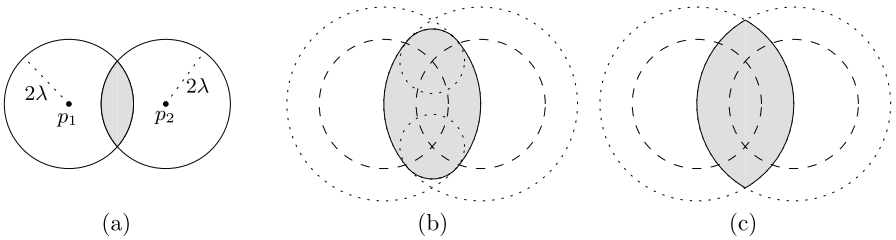

Fig. 3 In the L_p metric for $1 < p < \infty$, it does not hold any more that $(B(p_1, r_1) \cap B(p_2, r_2)) \oplus B_\lambda = (B(p_1, r_1) \oplus B_\lambda) \cap (B(p_2, r_2) \oplus B_\lambda)$, which was a key property to prove Lemma 5. (**a**) $B(p_1, 2\lambda)$ and $B(p_2, 2\lambda)$ for some $\lambda > 0$, where $B(p, r)$ is the L_2-ball centered at p with radius r and B_r is the L_2-ball centered at the origin with radius r. The *shaded region* is their intersection. (**b**) $(B(p_1, 2\lambda) \cap B(p_2, 2\lambda)) \oplus B_\lambda$ is different from (**c**) $(B(p_1, 2\lambda) \oplus B_\lambda) \cap (B(p_2, 2\lambda) \oplus B_\lambda)$

$B(p', \delta'\lambda^*)$ along vertical sides, there must be some index j^* such that $x_{j^*}(\lambda^*) = x_{j^*+1}(\lambda^*)$. This means that we only need to check the critical values among $\{\lambda \mid x_j(\lambda) = x_{j+1}(\lambda), 1 \le j \le 2n-1\}$ as candidates of the optimal λ^*. Namely, it is sufficient to find the smallest one among these $O(n)$ critical values such that the decision algorithm returns YES. This can be done in $O(D \log n)$ time.

The total time complexity is thus $O(n \log n + D \log n)$, which amounts to $O(k^2 n \log^2 n + k^4 n \log n)$ by Lemma 6. □

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_∞ metric. Notice also that the proposed algorithm relies only on the fact that a unit L_∞-ball is a square, and thus all arguments can be applied to the L_1 case.

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 $O((7k+1)^{7k-1} k^4 \cdot n \log^2 n)$ time.*

Remark One can indeed perform the binary search to get the optimal c between 1 and $7k$ based on the following observation: Let c^* be the largest integer of c such that the minimum of the maximum edge length is attained by an edge of $MST(P)$, namely, e_{c+1}. Then, either at c^* or $c^* + 1$, the optimal solution is obtained.

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_p metric for any rational p with $1 < p < \infty$, leading to the first exact algorithm for BST in the Euclidean (L_2) metric. Throughout this section, $d(a, b)$ denotes the L_p distance between two points a and b. Also, we assume that every leaf of the given abstract topology $\mathcal{T}_0$ is not a Steiner vertex as in Sect. 3. If we have an internal terminal vertex in $\mathcal{T}_0$, we split $\mathcal{T}_0$ into two abstract topologies and consider each separately.

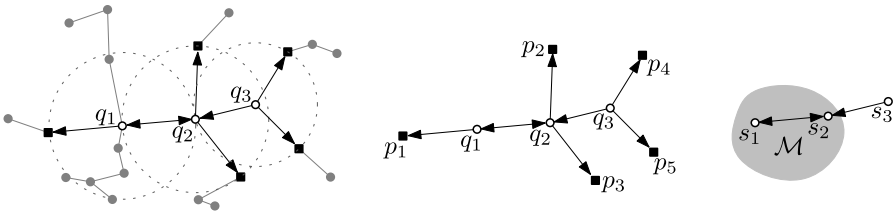

Fig. 4 (**a**) A bottleneck Steiner tree T^* for abstract topology $\mathcal{T}_0$ in Fig. 2: The *arrows* indicate the determinators D_i of each Steiner point q_i, and terminals that are determinators of some q_i are depicted as *small solid squares*. *Dotted circles* indicate B_i whose radius is r_i. (**b**) The subtree of T^* induced by the q_i and their determinators, where we have $D_1 = \{p_1, q_2\}$, $D_2 = \{p_2, p_3, q_1\}$, and $D_3 = \{p_4, p_5, q_2\}$. Observe that this optimal placement $\{q_1, q_2, q_3\}$ of Steiner points is determined only by the subset of terminals $\bigcup_i D_i \cap P = \{p_1, \ldots, p_5\}$. (**c**) The directed graph G on S corresponding to (**b**). Here, we have two equivalence classes $[s_1] = \{s_1, s_2\}$ and $[s_3] = \{s_3\}$ with respect to $(S, \equiv)$, and $[s_3] \preceq [s_1]$ with respect to $([S], \preceq)$. Further, $\mathcal{M} = \{[s_1]\}$ and observe that $r_1 = r_2 \geq r_3$ in (**a**)

As Ganley and Salowe [12] 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 Sect. 3 but would immediately face with a difficulty as a nice description of the region $R_\lambda(v)$ like Lemma 5 is hardly obtained. See Fig. 3. 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 Sect. 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 optimal Steiner tree T^* is supposed to satisfy **BST1–4** since **BST2** is fulfilled by a local modification of each Steiner point and the other three are guaranteed by the choice of abstract topology $\mathcal{T}_0$. Let r_i be the length of the longest edge incident to q_i in T^* and B_i be the L_p-ball centered at q_i with radius r_i, that is, $B_i = \{x \in \mathbb{R}^2 \mid d(q_i, x) \leq r_i\}$. An L_p-ball is determined by three points or two diametral points on its boundary. Thus, by **BST2**, each B_i has two or three points of $P \cup Q$ on its boundary, which are among the neighbors of q_i in T^*. We call the three or two points determining B_i the *determinators* of q_i (or B_i), and denote by D_i the set of determinators of q_i. When there are more than three points on the boundary of B_i, we arbitrarily choose three of them as the members of D_i, so we have that D_i consists of two or three points in $P \cup Q$. We say that the Steiner tree T^* *realizes* the determinator list $\mathcal{D} = (D_1, \ldots, D_k)$. The following is an immediate observation.

Lemma 9 *For any* $1 \leq i, j \leq k$, $q_i \in D_j$ *implies* $r_i \geq r_j$.

Proof $q_i \in D_j$ implies that q_i and q_j are adjacent in T^* and that $r_j = d(q_i, q_j)$. Since q_i is adjacent to q_j, we also know that $r_i \geq d(q_i, q_j) = r_j$. □

In the sequel, we show that one can extract from $\mathcal{D}$ enough information about which of the edges of T^* can be the longest, even without using any geometric knowledge; thus, every element of D_i is handled just in a symbolic level. First, we construct a directed graph G on the set $S = \{s_1, \ldots, s_k\}$ of Steiner vertices, where G has an arc from s_i to s_j if and only if $q_j \in D_i$. Then, the strongly connected components of G induce an equivalent relation $\equiv$ on S. Namely, $s_i \equiv s_j$ if and only if there exists a directed closed walk through s_i and s_j. We denote by $[s_i]$ the equivalence class with respect to $\equiv$ that includes s_i, and let $[S] := \{[s_i] \mid s_i \in S\}$. Also, the strongly connected component decomposition naturally induces a partial ordering $\preceq$ on $[S]$. (See Fig. 4.) These two relations $(S, \equiv)$ and $([S], \preceq)$ are dependent on a determinator list $\mathcal{D}$ only, so they are said to be induced by $\mathcal{D}$. Lemma 9 immediately implies the following.

Lemma 10 *Let T^* be an optimal Steiner tree with k Steiner points $Q = \{q_1, \ldots, q_k\}$ for BST-FT-ST that realizes a determinator list $\mathcal{D}$. Also, for $1 \leq i \leq k$, let r_i be the maximum length among the edges incident to q_i in T^*, and let the relations $(S, \equiv)$ and $([S], \preceq)$ be induced by $\mathcal{D}$ as above. Then, the following conditions are fulfilled*:

- *If $s_j \in [s_i]$, then $r_i = r_j$.*
- *If $[s_i] \preceq [s_j]$, then $r_i \leq r_j$.*

Proof This immediately follows from Lemma 9 and definitions of $(S, \equiv)$ and $([S], \preceq)$. □

Let $\mathcal{M}$ be the set of all the maximal elements in $[S]$ with respect to $\preceq$. More precisely, $\mathcal{M} = \{[s_i] \in [S] \mid \text{there is no other } [s_j] \in [S] \text{ with } [s_i] \preceq [s_j]\}$. We call each subset of Steiner vertices in $\mathcal{M}$ a *primary cluster*. See Fig. 4 for more illustrative explanation. Since the optimal objective value λ^* of BST-FT-ST for a given abstract topology $\mathcal{T}_0$ is determined by $\max_i r_i$, Lemma 10 implies that λ^* is indeed determined by a primary cluster in $\mathcal{M}$.

Lemma 11 *Let T^* be an optimal Steiner tree with k Steiner points $Q = \{q_1, \ldots, q_k\}$ for BST-FT-ST that realizes a determinator list $\mathcal{D}$. Also, for $1 \leq i \leq k$, let r_i be the maximum length among the edges incident to q_i in T^*, and λ^* be the longest edge length of T^*. Then, there always exists a primary cluster $[s_j] \in \mathcal{M}$ induced by $\mathcal{D}$ such that $\lambda^* = r_j$.*

Note that the two relations $(S, \equiv)$ and $([S], \preceq)$ can be inferred from $\mathcal{D}$ even if the placement $Q = \{q_1, \ldots, q_k\}$ of k Steiner points is unknown. In the above discussion, each q_i is treated in a symbolic level and their geometric locations do not matter at all. Hence, given a determinator list $\mathcal{D}$ in which the k Steiner points $q_1, \ldots, q_k$ remain unknown, it is possible to collect possible candidates for the optimal objective value λ^* by Lemmas 10 and 11. In the following, we show how to develop this idea and achieve an optimization algorithm for BST-FT-ST.

4.2 Algorithm for BST-FT-ST

Our algorithm enumerates all possible (combinatorially distinct) determinator lists $\mathcal{D} = (D_1, \ldots, D_k)$ that may be realized by an optimal Steiner tree T^* for a given abstract topology $\mathcal{T}_0$. By a fixed determinator list $\mathcal{D}$, the equivalence relation $(S, \equiv)$ and the partial ordering $([S], \preceq)$ are inferred as above. Also, we obtain the collection $\mathcal{M}$ of primary clusters from $([S], \preceq)$. Our strategy is to enumerate all possible values r_i for each primary cluster $[s_i] \in \mathcal{M}$ that satisfy the first condition of Lemma 10; such r_i shall be called a *critical value*. Then, Lemma 11 ensures that the optimal value λ^* is among those critical values. We will hence collect all critical values by handling each primary cluster and then do a binary search on them with a decision algorithm.

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

1. Enumerate all possible determinator lists $\mathcal{D}$ for the abstract topology $\mathcal{T}_0$.
2. For a fixed determinator list $\mathcal{D}$, build a corresponding concrete subtopology $\mathcal{T}_\mathcal{D}$.
3. For each primary cluster induced by $\mathcal{D}$, 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 detail.

(1) Enumeration of All Possible Determinator Lists For a given abstract topology $\mathcal{T}_0$, a rough calculation on the number of possible combinatorially distinct determinator lists $\mathcal{D}$ gives us $20^k n^{3k}$: for each Steiner vertex $s_i \in S$, (i) we choose two or three neighbors from at most $\Delta = 5$ neighbors of s_i in $\mathcal{T}_0$, resulting in $(\binom{5}{2} + \binom{5}{3})^k = 20^k$ possibilities, and (ii) choose one terminal among at most n terminals in T_j if t_j was chosen for some s_i at (i), resulting in n^{3k} possibilities. Again, keep in mind that we handle Steiner points $D_i \setminus P \subset Q$ in a symbolic way.

In order to prune a number of unnecessary combinations of determinators, we bring a known geometric structure, the *farthest color Voronoi diagram*. The farthest color Voronoi diagram is a generalization of the standard farthest-neighbor Voronoi diagram to colored point sets [1, 13]: Given a collection $\mathcal{C} = \{P_1, \ldots, P_l\}$ of l sets of colored points, define the *distance to a color* i for $1 \le i \le l$ 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 $\{x \in \mathbb{R}^2 \mid d_i(x) > d_j(x), 1 \le j \le l, j \ne i\}$. Each edge of $FCVD(\mathcal{C})$ is the set of points that have the same set of two farthest colors while 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. [13] and Abellanas et al. [1].

For each Steiner vertex $s_i \in S$, let $\mathcal{C}_i := \{V(T_j) \mid t_j \text{ is a neighbor of } s_i \text{ in } \mathcal{T}_0\}$, where $V(T_j) \subseteq P$ denotes the terminal set of subtree T_j (of $MST(P)$). We now introduce an interesting relation between $FCVD(\mathcal{C}_i)$ and the determinators.

Lemma 12 *There exists an optimal Steiner tree T^* with k Steiner points $Q = \{q_1, \ldots, q_k\}$ for an abstract topology $\mathcal{T}_0$ realizing a determinator list $\mathcal{D} = (D_1, \ldots, D_k)$*

such that for any $1 \le i \le k$, *if* $m_i := |D_i \cap P| > 0$, *then the Steiner point* q_i *lies on the* $(3 - m_i)$*-face* ϕ_i *of* $FCVD(\mathcal{C}_i)$ *determined by* $D_i \cap P$.

Proof Pick any $1 \le i \le k$ with $D_i \cap P \neq \emptyset$. Let $p_1, \ldots, p_m$ be m terminals in $D_i \cap P$, where $1 \le m \le 3$. Without loss of generality, we assume that $p_j \in V(T_j)$ for each $1 \le j \le m$.

Note that if Steiner vertex s_i is adjacent to t_j in $\mathcal{T}_0$ we can assume that, in the resulting Steiner tree T^*, q_i is adjacent to the closest terminal of the subtree T_j. Thus, there exists an optimal Steiner tree T^* satisfying the condition that each p_j with $1 \le j \le m$ is the closest terminal in T_j from q_i.

By the definition of determinators, we have $r_i = d(q_i, p_1) = \cdots = d(q_i, p_m) \ge d(q_i, p')$ for any other neighbor p' of q_i in T^* with $p' \notin D_i$. Also, as noted above, p_j is the nearest point from q_i among $V(T_j)$ for each $1 \le l \le m$. Thus, by the definition of $FCVD(\mathcal{C}_i)$, q_i lies on the face ϕ_i determined by $p_1, \ldots, p_m$ and its dimension is $3 - m$. This implies that such an optimal Steiner tree T^* satisfies the condition of the lemma. □

Thus, as a preprocessing, we compute $FCVD(\mathcal{C}_i)$ for each Steiner vertex s_i. Since $\mathcal{C}_i$ consists of at most 5 subsets of P, the total combinatorial complexity of $FCVD(\mathcal{C}_i)$ for all $1 \le i \le k$ is bounded by $O(n)$ and $O(n \log n)$ time is sufficient to build them all for any L_p metric with $1 < p < \infty$ [1, 13].

Lemma 13 *For an abstract topology* $\mathcal{T}_0$, *enumerating at most* $2^{O(k)}n^k$ *combinatorially distinct determinator lists suffices to find one realized by an optimal Steiner tree. Also, the enumeration can be done in the same time bound.*

Proof We calculate the number of possible D_i by splitting it into $D_i \cap P$ and $D_i \setminus P$. By Lemma 12, $D_i \cap P$ is determined by a face ϕ_i (of any dimension) of $FCVD(\mathcal{C}_i)$, and hence there are $O(n)$ possible choices for $D_i \cap P$, including the case of $D_i \cap P = \emptyset$. Thus, we have $O(n)^k$ possibilities for every $D_i \cap P$ with $1 \le i \le k$.

On the other hand, for each edge between two Steiner vertices s_i and s_j in $\mathcal{T}_0$, we have at most 4 possibilities in any optimal Steiner tree T^*; (i) $q_i \in D_j$ and $q_j \in D_i$, (ii) $q_i \in D_j$ and $q_j \notin D_i$, (iii) $q_i \notin D_j$ and $q_j \in D_i$, or (iv) $q_i \notin D_j$ and $q_j \notin D_i$. If we fix one of the four for every edge between Steiner vertices in $\mathcal{T}_0$, then every $D_i \setminus P$ with $1 \le i \le k$ is determined. Since $\mathcal{T}_0$ has at most $k - 1$ such edges between Steiner vertices, we have at most 4^{k-1} possibilities to fix $D_i \setminus P$ for all $1 \le i \le k$.

Thus, we have $O(n)^k \times 4^{k-1} = 2^{O(k)}n^k$ combinatorially distinct determinator lists to test. The enumeration of all such determinator lists can be done easily in the same time by traversing every face of $FCVD(\mathcal{C}_i)$ for each $1 \le i \le k$. □

(2) Building a Corresponding Concrete Subtopology At this step, we are given a determinator list $\mathcal{D} = (D_1, \ldots, D_k)$. Then, $\mathcal{D}$ induces a concrete subtopology $\mathcal{T}_\mathcal{D}$ from the abstract topology $\mathcal{T}_0$ by choosing the terminals appearing in D_i for some $1 \le i \le k$. Namely, $\mathcal{T}_\mathcal{D}$ is a concrete topology on terminals $(D_1 \cup \cdots \cup D_k) \cap P$ and Steiner vertices S. Note that $\mathcal{T}_\mathcal{D}$ consists of at most $3k$ terminals and k Steiner vertices.

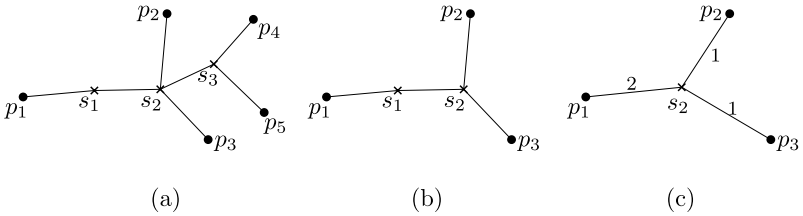

Fig. 5 Concrete subtopologies induced by $\mathcal{D} = \{D_1, D_2, D_3\}$ from abstract topology $\mathcal{T}_0$ of Fig. 2. Here, we take $D_1 = \{p_1, q_2\}$, $D_2 = \{p_2, p_3, q_1\}$, and $D_3 = \{p_4, p_5, q_2\}$ as in Fig. 4(b). (**a**) The induced concrete subtopology $\mathcal{T}_{\mathcal{D}}$, obtained at Step (2). (**b**) For a primary cluster $[s_1] = \{s_1, s_2\} \in \mathcal{M}$, the corresponding subtopology $\mathcal{T}_1$ and (**c**) the weighted topology $\mathcal{T}_1'$. Here, each edge e of $\mathcal{T}_1'$ is labeled with its weight $w(e)$

We should mention that an optimal placement Q of k Steiner points is determined only by the subset of terminals, $\bigcup_i D_i \cap P$, provided that a bottleneck Steiner tree T^* realizes $\mathcal{D}$.

(3) Collecting Critical Values We call a real number λ a *critical value* if there exist a primary cluster $[s_i] \in \mathcal{M}$ and a placement q_j for every $s_j \in [s_i]$ such that the equality $r_j = r_i = \lambda$ holds.

Lemma 14 *The optimal objective value λ^* for BST-FT-ST is a critical value.*

Proof Let $\mathcal{D}$ be the determinator list realized by an optimal Steiner tree T^* with k Steiner points $Q = \{q_1, \ldots, q_k\}$ for a given instance of BST-FT-ST. By Lemma 11, there exists a primary cluster $[s_i] \in \mathcal{M}$ induced by $\mathcal{D}$ such that $\lambda^* = r_i$, where r_i is the maximum length of the edges incident to q_i in T^*. Furthermore, by Lemma 10, the equality $r_j = r_i$ holds for every j such that $s_j \in [s_i]$. Hence, λ^* is a critical value. □

At Step (3), we thus collect all such critical values λ among which the optimal value λ^* is guaranteed to exist by Lemma 14.

Pick a primary cluster $[s_i] \in \mathcal{M}$. We then take a subtopology $\mathcal{T}_i$ of $\mathcal{T}_{\mathcal{D}}$, 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 in $[s_i]$ with degree 2 or 3. (By the assumption discussed above, every internal node corresponds to a Steiner point.) Observe that any degree-2 Steiner point is located at the midpoint of its two neighbors by **BST2**. (For example, see q_1 in Fig. 4(a).) 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: (i) Initially, assign weight 1 to every edge of $\mathcal{T}_i$. (ii) 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. Figure 5 illustrates how to obtain $\mathcal{T}_i$ and $\mathcal{T}_i'$ from $\mathcal{T}_{\mathcal{D}}$.

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 (i.e., Steiner) vertices since all internal nods have degree 3. Without loss of generality (or by index reordering), we can 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

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

By the definition of critical values, there exists a placement $(q_1', \ldots, q_m') \in (\mathbb{R}^2)^m$ of m Steiner points such that the equality $h_e(q_1', \ldots, q_m') = \lambda$ holds for any edge e of $\mathcal{T}_i'$ if and only if λ is a critical value associated with primary cluster $[s_i]$. Thus, our task is to solve the system of $2m + 1$ equations with $2m + 1$ unknowns $q_{1,x}, q_{i,y}, \ldots, q_{m,x}, q_{m,y}, \lambda$:

$$h_e(q_{1,x}, q_{1,y}, \ldots, q_{m,x}, q_{m,y}) = \lambda \quad \text{for each edge } e \text{ of } \mathcal{T}_i', \tag{3}$$

where $q_{i,x}$ and $q_{i,y}$ denote the x- and y-coordinates of q_i.

For the purpose, we transform the system (3) into a system of *polynomials* of degree bounded by a constant. First, observe that for each e of $\mathcal{T}_i'$, h_e is dependent on at most four unknowns: if e is an edge between two Steiner points q_j and q_l, then $h_e = d(q_j, q_l)/w(e)$; otherwise, if e is an edge between a Steiner point q_j and a terminal $p_l \in P$, then $h_e = d(q_j, p_l)/w(e)$. Since our underlying metric is the L_p metric for any fixed rational $1 < p < \infty$, p is represented as $p = \frac{b}{a}$ for some positive integers a and b with $b > a$. Then, for each edge e of $\mathcal{T}_i'$, we have

$$h_e = \begin{cases} \frac{1}{w(e)}(|q_{j,x} - q_{l,x}|^{\frac{b}{a}} + |q_{j,y} - q_{l,y}|^{\frac{b}{a}})^{\frac{a}{b}} & \text{if } e = q_j q_l, \\ \frac{1}{w(e)}(|q_{j,x} - p_{l,x}|^{\frac{b}{a}} + |q_{j,y} - p_{l,y}|^{\frac{b}{a}})^{\frac{a}{b}} & \text{if } e = q_j p_l, \end{cases}$$

by the definition of the L_p distance function. Now, we introduce two new unknowns X_e and Y_e for each e and the following system $\mathcal{P}$ of polynomials: for each edge e of $\mathcal{T}_i'$,

$$\begin{cases} X_e^{2a} - (q_{j,x} - q_{l,x})^{2b} = 0 & \text{if } e = q_j q_l \\ X_e^{2a} - (q_{j,x} - p_{l,x})^{2b} = 0 & \text{if } e = q_j p_l \end{cases} \tag{4}$$

$$\begin{cases} Y_e^{2a} - (q_{j,y} - q_{l,y})^{2b} = 0 & \text{if } e = q_j q_l \\ Y_e^{2a} - (q_{j,y} - p_{l,y})^{2b} = 0 & \text{if } e = q_j p_l \end{cases} \tag{5}$$

$$(X_e + Y_e)^a - w(e)^b \lambda^b = 0 \tag{6}$$

The system $\mathcal{P}$ consists of $6m + 3$ polynomials of degree at most $2b$ with $6m + 3$ unknowns: $q_{1,x}, q_{1,y}, \ldots, q_{m,x}, q_{m,y}, \lambda$ and additionally X_e, Y_e for each edge e of $\mathcal{T}_i'$. Note that the zero set of the system $\mathcal{P}$ includes that of system (3); from (4)–(5), one can easily check that $(X_e + Y_e)^{\frac{a}{b}} = h_e$. Also, since the polynomials in $\mathcal{P}$ contain no common factor, they have finitely many common zeros. More specifically, the system $\mathcal{P}$ is zero-dimensional and the number of zeros of $\mathcal{P}$ is at most $(2b)^{6m+3}$, as known as the Bézout bound [9]. Several single-exponential time algorithms for solving a zero-dimensional system of polynomials are known [11, 15, 21]. (See also a recent survey [3].) In our case, $\mathcal{P}$ can be solved in $b^{O(m)}$ time.

Lemma 15 *For a determinator list $\mathcal{D}$, there are at most $2^{O(k)}$ critical values. All these critical values can be obtained in the same time bound.*

Proof As discussed above, for each primary cluster $[s_i]$, we construct the weighted concrete subtopology $\mathcal{T}_i'$, build the system $\mathcal{P}$ of polynomials (4)–(6) based on $\mathcal{T}_i'$, and solve $\mathcal{P}$. The above procedure can be done in time $b^{O(m)} = 2^{O(m)}$, where b is a constant depending on p as defined above and m is the number of Steiner vertices in $\mathcal{T}_i'$. Summing up m over all primary clusters in $\mathcal{M}$, we get at most k since each primary cluster is an equivalence class of $[S]$ induced by $\mathcal{D}$. It is easy to see that $\sum_i 2^{ca_i} \le 2^{ck}$ for any sequence $\{a_i\}$ of nonnegative integers with $\sum_i a_i \le k$ and any positive real number c. Thus, Step (3) can be performed in time $2^{O(k)}$, and the number of critical values for a determinator list is also bounded by $2^{O(k)}$. □

(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 subtopology $\mathcal{T}_\mathcal{D}$. We make use of a modified version of the decision algorithm by Sarrafzadeh and Wong [22]. We choose an arbitrary internal vertex of $\mathcal{T}_\mathcal{D}$ as a root and consider $\mathcal{T}_\mathcal{D}$ as a rooted tree.

By Lemma 12, every Steiner point q_i should lie in the face ϕ_i chosen at Step (1); if $D_i \cap P = \emptyset$, here we take $\phi_i = \mathbb{R}^2$. We thus redefine the region $R_\lambda(v)$ associated with each vertex v of $\mathcal{T}_\mathcal{D}$ as follows: if v is a leaf, then v corresponds to a terminal p_j and hence let $R_\lambda(v) = \{p_j\}$; otherwise, if v is a Steiner vertex s_i, we let

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

where $C(v)$ is the set of children of v in $\mathcal{T}_\mathcal{D}$ and B_λ is the L_p-ball with radius λ centered at the origin.

If ϕ_i is a segment, a point, or the whole plane, 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 asymptotic 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 $R_\lambda(v)$ is done without additional cost; ϕ_i is either a point, a segment, a triangle, or the whole plane $\mathbb{R}^2$. Since $\mathcal{T}_\mathcal{D}$ consists of $O(k)$ vertices, our modified decision algorithm runs in $O(k \log k)$ time, based on the analysis by Sarrafzadeh and Wong [22]. The algorithm computes $R_\lambda(v)$ in the bottom-up fashion and reports `YES` if $R_\lambda(v) \neq \emptyset$ for all v; otherwise, it reports `NO`.

The following lemma shows the correctness of the above decision algorithm.

Lemma 16 *Let λ be a critical value for a determinator list $\mathcal{D}$. Then, the above algorithm reports* `YES` *for λ on $\mathcal{T}_\mathcal{D}$ if and only if there exists a Steiner tree realizing $\mathcal{D}$ such that its longest edge length is at most λ.*

Proof ($\Rightarrow$) Suppose that the algorithm reports `YES` for λ on $\mathcal{T}_\mathcal{D}$. This means that we have $R_\lambda(v) \neq \emptyset$ for all vertices v of $\mathcal{T}_\mathcal{D}$. Pick any point $q_i \in R_\lambda(s_i)$, and let

$Q := \{q_1, \ldots, q_k\}$ be a placement of k Steiner points; thus, we now fix k unknown points in $\mathcal{D} = \{D_1, \ldots, D_k\}$. By the description of the algorithm together with the analysis of Sarrafzadeh and Wong [22], it is obvious that we have $d(q_i, u) \leq \lambda$ for any $u \in D_i$.

Recall that each edge $s_i t_j$ of the abstract topology $\mathcal{T}_0$ such that $V(T_j) \cap D_i = \emptyset$ is neglected in Step (2) to build $\mathcal{T}_\mathcal{D}$. By definition of $R_\lambda(v)$, we have $q_i \in R_\lambda(v) \subseteq \phi_i$, where ϕ_i is the face of $FCVD(\mathcal{C}_i)$ determined by $D_i \cap P$ unless $D_i \cap P = \emptyset$ (see Lemma 12). This implies that, for each such edge $s_i t_j$ of $\mathcal{T}_0$, there exists a terminal $p_j \in V(T_j)$ such that $d(q_i, p_j) \leq d(q_i, u) \leq \lambda$ for any $u \in D_i$. Therefore, we build a Steiner tree T for P with k Steiner points Q as follows: (i) on $P \cup Q$, put all the edges of $T_1, \ldots, T_{c+1}$. (Recall that the T_i are subtrees corresponding to the vertices t_i of $\mathcal{T}_0$; see Sect. 2.) (ii) for any $u \in D_i$, we put the edge $q_i u$, and also for any edge $s_i t_j$ of $\mathcal{T}_0$ neglected to build $\mathcal{T}_\mathcal{D}$, we put the edge $q_i p_j$. Then, clearly T realizes $\mathcal{D}$ and its longest edge length does not exceed λ.

($\Leftarrow$) Now, let T be a Steiner tree realizing $\mathcal{D}$ with k Steiner points $Q = \{q_1, \ldots, q_k\}$, and suppose that the longest edge length of T is at most λ. Also, let r_i be the maximum edge length among those incident to q_i in T. Then, by **BST2** and Lemma 12, if $D_i \cap P \neq \emptyset$, then q_i lies on a face ϕ_i of $FCVD(\mathcal{C}_i)$. Since $d(q_i, u) \leq r_i \leq \lambda$ for any neighbor u of q_i in T, we have $R_\lambda(s_i) \neq \emptyset$ during the execution of the decision algorithm. In the case of $D_i \cap P = \emptyset$, we similarly have $R_\lambda(s_i) \neq \emptyset$. Consequently, we have $R_\lambda(v) \neq \emptyset$ for all vertices of $\mathcal{T}_\mathcal{D}$ during the execution of the algorithm, and thus the algorithm reports YES. □

For each determinator list $\mathcal{D}$, we can find the minimum value $\lambda^*_\mathcal{D}$ for which the decision algorithm answers YES among the critical values obtained at Step (3) in $O(k \log k \cdot \log(2^{O(k)})) = O(k^2 \log k)$ time. Then, the optimal objective value λ^* is taken to be the minimum of the $\lambda^*_\mathcal{D}$ over all possible determinator lists $\mathcal{D}$. Finally, we have an exact algorithm for BST-FT-ST in the L_p metric for any rational p with $1 < p < \infty$.

Lemma 17 *BST-FT-ST in the L_p metric for any rational p with $1 < p < \infty$ can be solved in $O(2^{O(k)} n^k + n \log n)$ time.*

Proof We first check the correctness. From Lemma 13 and Lemma 14, we know that among all critical values enumerated at Step (3) there is the optimal solution λ^*. Let $\mathcal{D}^*$ be a determinator list realized by an optimal Steiner tree. Then, clearly our decision algorithm returns YES for λ^* on $\mathcal{T}_{\mathcal{D}^*}$ by Lemma 16. On the other hand, for any critical value λ with $\lambda < \lambda^*$, our decision algorithm returns NO on $\mathcal{T}_\mathcal{D}$ induced by any determinator list $\mathcal{D}$ obtained in Step (1), again by Lemma 16. Therefore, λ^* is indeed the minimum value of the $\lambda^*_\mathcal{D}$ over all determinator lists $\mathcal{D}$, and our algorithm correctly finds this value.

To see the time complexity, initially we spend $O(n \log n)$ time to compute $FCVD(\mathcal{C}_i)$ for all $1 \leq i \leq k$. For each determinator list, Steps (3)–(4) take $O(2^{O(k)}) + O(k^2 \log k)$ time. We repeat Steps (3)–(4) $2^{O(k)} n^k$ times (Lemma 13), so $O(n \log n + 2^{O(k)} \cdot n^k)$ time is sufficient to solve BST-FT-ST. □

Combining this with Lemma 4, we obtain the main result of this section:

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

Note that our algorithm only finds an optimal placement Q of k Steiner points with optimal value λ^*. To obtain a bottleneck Steiner tree, it suffices to compute any bottleneck spanning tree or minimum spanning tree for the $n + k$ points $P \cup Q$ in the plane.

5 Concluding Remarks

For the rectilinear cases (the L_1 or the L_∞ metric), we presented an exact algorithm which runs in time $O(f(k) \cdot n \log^2 n)$. Thus, the rectilinear bottleneck Steiner tree problem with parameter k is in FPT, the class of fixed-parameter tractable problems. We remark that by Theorem 1 the algorithms by Bae et al. [4] for $k \leq 2$ can be simply modified for the L_1 metric; thus, the case of $k = 1$ can be solved in $O(n \log n)$ time.

On the other hand, in the L_p metric for $1 < p < \infty$, we failed to achieve fixed-parameter tractability. Nonetheless, it is worth noting that we presented the first exact algorithm that solves the bottleneck Steiner tree problem for the L_p metric for every rational p with $1 < p < \infty$, including the Euclidean (L_2) metric. An interesting open question is: Does the Euclidean bottleneck Steiner tree problem admit an algorithm of running time of the form $f(k) \cdot n^{O(1)}$? Or, is it W[1]-hard? Note that the special case where no edge between Steiner points admits a fixed-parameter algorithm with running time $O(f(k) \cdot n \log n)$ [4].

Acknowledgements This work was partially done when the first author visited Kyoto University in May 2009 for one week. The first author thanks Professor Naoki Katoh for his kind hospitality during the visit. The authors thank anonymous reviewers for invaluable comments to improve the paper.

Appendix: Proof of Lemma 1

We start with some definitions. Let C be a centrally symmetric closed convex body in $\mathbb{R}^2$, i.e., $p \in C$ implies $-p \in C$ (where the center is assumed to be the origin). For any point $p \in \mathbb{R}^2$, let $C(p)$ be the translate of C centered at p, and for any $\lambda \in \mathbb{R}$ let $C(p, \lambda)$ be the scaled translate of C, i.e., $C(p, \lambda) = \{\lambda q + p \mid q \in C\}$. The *Hadwiger number* of C, denoted by $H(C)$, is the largest number of nonoverlapping translates of C that can be brought into contact with C. For more references about the Hadwiger number, see the book by Brass et al. [6] or a survey article by C. Zong [29].

We need the following lemma.

Lemma 18 *For any closed convex body C, any point $x \in C$, and any scalar λ with $0 \leq \lambda \leq 1$, we have $C((1-\lambda)x, \lambda) \subseteq C$. Also, if x is chosen on the boundary, then x remains on the boundary of $C((1-\lambda)x, \lambda)$.*

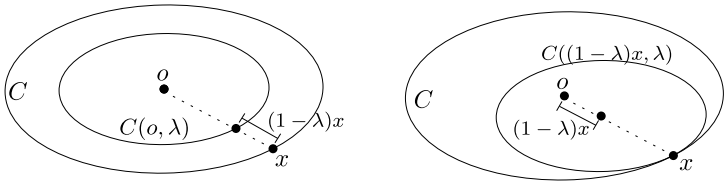

Fig. 6 Illustration to Lemma 18

Proof Recall that $C((1-\lambda)x, \lambda)$ is a convex body obtained by scaling C by λ and then translating it by $(1-\lambda)x$ (Fig. 6). Namely any point in $C((1-\lambda)x, \lambda)$ can be generated by this process from some point $y \in C$; conversely any $y \in C$ is mapped to $\lambda y + (1-\lambda)x$ by this process. Since $\lambda y + (1-\lambda)x$ is a convex combination, we have $\lambda y + (1-\lambda)x \in C$, implying $C((1-\lambda)x, \lambda) \subseteq C$.

Notice that the above process does not move the position of x, which means that x is on the boundary of $C((1-\lambda)x, \lambda)$ if x is on the boundary of C. □

We are ready to prove Lemma 1.

Proof of Lemma 1 Consider a point $q \in \mathbb{R}^2$ and m points $p_1, \ldots, p_m$ in counterclockwise order around q. We denote by $d(a, b)$ the L_p distance between two points a and b for any $1 \leq p \leq \infty$, and define C to be the unit L_p-ball centered at the origin, i.e., $C = \{x \in \mathbb{R}^2 \mid d(x, o) \leq 1\}$. Observe that C is a centrally symmetric convex body in the plane. Recall $d(p_i, q) \leq d(p_i, p_j)$ for any i and j with $i \neq j$ by the lemma assumption.

Consider $C_i := C(p_i, d(p_i, q))$ for any $1 \leq i \leq m$. By the assumption,

$$C_i \text{ touches } q \text{ on its boundary but its interior does not contain any } p_j \text{ with } j \neq i. \tag{7}$$

We now show that, keeping the property (7), $p_1, \ldots, p_m$ can be moved to satisfy $d(p_i, q) = d(p_j, q)$ for any $1 \leq i, j \leq m$. Let us repeatedly apply the following operation: take a farthest point, say p_i, from q and move p_i to q along the straight line passing through p_i and q until $d(p_i, q)$ equals to the distance between q and the second farthest point, say p_j. Then, C_i shrink as shown in Lemma 18 keeping q on its boundary. If there are several farthest points, then we perform the shrinking operation simultaneously. By Lemma 18, after shrinking C_i, it does not contain any other point p_k. Also, since $d(p_i, q) = d(p_j, q) \leq d(p_k, q)$ for any $1 \leq k \leq m$ after the operation, C_k does not contain p_i in its interior. The operation thus preserves the property (7), and hence repeatedly applying the operation ends up with $p_1, \ldots, p_m$ having the desired property.

Therefore we may assume that $d(p_i, q) \leq \min_{j \neq i} d(p_i, p_j)$ and $d(p_i, q) = d(p_j, q)$ for any i, j. Without loss of generality, we can also assume $d(p_i, q) = 2$. Since C is a unit L_p-ball, if $C(p_i)$ overlaps $C(p_j)$, then we have $d(p_i, p_j) < 2$, which is not true. Thus, the $C(p_i)$ do not overlap each other. Also, it is easy to see that $C(p_i)$ touches $C(q)$ in the same sense. This means that $C(p_1), \ldots, C(p_m)$ are m nonoverlapping translates of $C(q)$ that contact with $C(q)$. Hence, $m \leq H(C)$.

It is known that $H(C) = 6$ for every convex body $C \subset \mathbb{R}^2$ except for parallelograms; if C is a parallelogram, $H(C) = 8$ [6, 29]. Note that the L_p-ball is a parallelogram only if $p = 1$ or $p = \infty$. Moreover, the Hadwiger number is realized only when the translates of C and C itself are packed; $C(p_i)$ touches $C(q)$ as well as $C(p_{i+1})$ and $C(p_{i-1})$ for any i, and thus we have the equality $d(p_i, q) = d(p_i, p_{i+1}) = d(p_i, p_{i+1})$, completing the proof. □

References

1. Abellanas, M., Hurtado, F., Icking, C., Klein, R., Langetepe, E., Ma, L., Palop, B., Sacristán, V.: The farthest color Voronoi diagram and related problems. Technical report 002, Rheinische Friedrich–Wilhelms–Universität Bonn (2006)
2. Ajtai, M., Komlos, J., Szemeredi, E.: An $O(n \log n)$ sorting network. Combinatorica **3**, 1–19 (1983)
3. Ayad, A.: A survey on the complexity of solving algebraic systems. Int. Math. Forum **5**(7), 333–353 (2010)
4. Bae, S.W., Lee, C., Choi, S.: On exact solutions to the Euclidean bottleneck Steiner tree problem. Inf. Process. Lett. **110**(16), 672–678 (2010)
5. Balaban, I.J.: An optimal algorithm for finding segments intersections. In: Proceedings of the 11th Annual Symposium on Computational Geometry, pp. 211–219. ACM, New York (1995)
6. Brass, P., Moser, W., Pach, J.: Research Problems in Discrete Geometry. Springer, Berlin (2005)
7. Chiang, C., Sarrafzadeh, M., Wong, C.K.: A powerful global router: based on Steiner min-max trees. In: Proceedings of IEEE International Conference on CAD, pp. 2–5 (1989)
8. Cole, R.: Slowing down sorting networks to obtain faster sorting algorithms. J. ACM **34**(1), 200–208 (1987)
9. Dickenstein, A., Emiris, I.Z.: Solving Polynomial Equations: Foundations, Algorithms, and Applications. Algorithms and Computation in Mathematics, vol. 14. Springer, Berlin (2005)
10. Elzinga, J., Hearn, D., Randolph, W.D.: Minimax multifacility location with Euclidean distances. Transp. Sci. **10**, 321–336 (1976)
11. Faugère, J., Gianni, P., Lazard, D., Mora, F.: Efficient computation of zero-dimensional Gröbner bases by change of ordering. J. Symb. Comput. **16**(4), 329–344 (1993)
12. Ganley, J.L., Salowe, J.S.: Optimal and approximate bottleneck Steiner trees. Oper. Res. Lett. **19**, 217–224 (1996)
13. Huttenlocher, D.P., Kedem, K., Sharir, M.: The upper envelope of Voronoi surfaces and its applications. Discrete Comput. Geom. **9**, 267–291 (1993)
14. Kedem, K., Livne, R., Pach, J., Sharir, M.: On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. Discrete Comput. Geom. **1**(1), 59–71 (1986)
15. Lakshman, Y.N.: A single exponential bound on the complexity of computing Gröbner bases of zero dimensional ideals. In: Effective Methods in Algebraic Geometry. Progress in Mathematics, vol. 94, pp. 227–234. Birkhäuser, Basel (1991)
16. Li, Z.-M., Zhu, D.-M., Ma, S.-H.: Approximation algorithm for bottleneck Steiner tree problem in the Euclidean plane. J. Comput. Sci. Technol. **19**(6), 791–794 (2004)
17. Love, R.F., Wesolowsky, G.O., Kraemer, S.A.: A multifacility minimax location problem with Euclidean distances. Int. J. Prod. Res. **11**, 37–45 (1973)
18. Megiddo, N.: Applying parallel computation algorithms in the design of serial algorithms. J. ACM **30**(4), 852–865 (1983)
19. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and Its Applications, vol. 31. Oxford University Press, Oxford (2006)
20. Prüfer, H.: Neuer Beweis eines Satzes über Permutationen. Arch. Math. Phys. **27**, 742–744 (1918)
21. Rouillier, F.: Solving zero-dimensional systems through the rational univariate representation. Appl. Algebra Eng. Commun. Comput. **9**, 433–461 (1999)
22. Sarrafzadeh, M., Wong, C.K.: Bottleneck Steiner trees in the plane. IEEE Trans. Comput. **41**(3), 370–374 (1992)
23. Shioura, A., Tamura, A., Uno, T.: An optimal algorithm for scanning all spanning trees of undirected graphs. SIAM J. Comput. **26**(3), 678–692 (1997)

24. Stanley, R.P.: Enumerative Combinatorics: Volume 2. Cambridge University Press, Cambridge (2001)
25. van Oostrum, R., Veltkamp, R.C.: Parametric search made practical. Comput. Geom. Theory Appl. **28**(2–3), 75–88 (2004)
26. Wang, L., Du, D.-Z.: Approximations for a bottleneck Steiner tree problem. Algorithmica **32**, 554–561 (2002)
27. Wang, L., Li, Z.: An approximation algorithm for a bottleneck k-Steiner tree problem in the Euclidean plane. Inf. Process. Lett. **81**, 151–156 (2002)
28. Warme, D.M., Winter, P., Zachariasen, M.: Exact algorithms for plane Steiner tree problems: a computational study. In: Du, D.Z., Smith, J.M., Rubinstein, J.H. (eds.) Advances in Steiner Trees, pp. 81–116. Kluwer Academic, Norwell (2000)
29. Zong, C.: The kissing numbers of convex bodies—a brief survey. Bull. Lond. Math. Soc. **30**, 1–30 (1998)