Faster Approximation Algorithms for the Rectilinear Steiner Tree Problem[*]

U. Fößmeier,[1] M. Kaufmann,[1] and A. Zelikovsky[2]

[1]Wilhelm-Schickard-Institut für Informatik, Universität Tübingen,
Sand 13, 72076 Tübingen, Germany
{foessmei,mk}@informatik.uni-tuebingen.de

[2]Department of Computer Science, University of Virginia,
Charlottesville, VA 22903-2242, USA

Abstract. The classical Steiner tree problem requires a shortest tree spanning a given vertex subset within a graph $G = (V, E)$. An important variant is the Steiner tree problem in rectilinear metric. Only recently two algorithms were found which achieve better approximations than the "traditional" one with a factor of 3/2. These algorithms with an approximation ratio of 11/8 are quite slow and run in time $O(n^3)$ and $O(n^{5/2})$. A new simple implementation reduces the time to $O(n^{3/2})$. As our main result we present efficient parametrized algorithms which reach a performance ratio of $11/8 + \varepsilon$ for any $\varepsilon > 0$ in time $O(n \cdot \log^2 n)$, and a ratio of $11/8 + \log \log n / \log n$ in time $O(n \cdot \log^3 n)$.

1. Introduction

Let $S = \{v_1, \ldots, v_n\}$ be a set of points in the plane which is called a *terminal*. A *Steiner tree* is a tree in the plane which contains the set S. The *Steiner problem* is to find a Steiner tree of minimal length. There are several versions of the problem, extensively described in the literature [3], [6]–[8], [14], [15], [17].

We focus our attention on the *rectilinear Steiner problem*; this is the version where the distance between two points is the sum of the differences of their x- and y-coordinates. Recently this problem obtained new importance in the development of techniques for VLSI routing [11], [12] and has been known to be NP-hard for a long time [10], [6]. Therefore polynomial-time algorithms for approximate solutions have been investigated.

---

[*] Parts of this work have been done at the Max-Planck-Institut für Informatik, Saarbrücken and at the Fakultät für Mathematik und Informatik, Universität Passau, Passau.

The quality of an approximation is measured by its performance ratio: an upper bound on the ratio between the achieved length and the optimal length.

The well-known *MST-heuristic* for the Steiner problem approximates a Steiner minimum tree with a minimum length spanning tree of a complete graph G_S, which has a vertex set S and edge lengths equal to the shortest path lengths in the graph G.

The MST-heuristic for the rectilinear case is due to Hwang [9]. He proved that the ratio between the length of the minimum spanning tree and the length of the Steiner minimum tree is $3/2$, and gave an $O(n \cdot \log n)$-time implementation for this approximation algorithm.

Surprisingly two better approximations have recently been given [1], [19], which can guarantee a performance ratio of $11/8$. The algorithm of Zelikovsky runs in time $O(n^3)$ and has been improved by the $O(n^{5/2})$ algorithm of Berman and Ramaiyer. The main results of this paper are:

- An acceleration of the second algorithm to $O(n^{3/2})$.
- A new parametrized $O(r \cdot m \cdot n \cdot \log^2 n)$ algorithm with a performance ratio of $11/8 + 1/2m + 1/(8 \cdot 32^{r-2})$ for any parameters $m, r > 0$.

First we introduce some notation: $ST(S)$ and $st(S)$ are a Steiner tree of S and its length, respectively, $SMT(S)$ denotes a Steiner minimum tree with length $smt(S)$. For a complete graph with a vertex set S, $M(S)$ is a minimum length spanning tree of S, with length $m(S)$.

In general, a Steiner tree for a set S may contain as vertices also other vertices than the terminals. These additional vertices are called the *Steiner vertices*. $ST(S)$ is called a *full Steiner tree* if S coincides with the set of leaves of $ST(S)$. If $ST(S)$ is not full, we can split it into edge-disjoint full Steiner subtrees, the *full components* of $ST(S)$. $ST(S)$ is called *k-restricted* if every full component has at most k terminals. Let the shortest k-restricted Steiner tree for the set S, denoted by $SMT_k(S)$, have the length $smt_k(S)$. Notice that $SMT_2(S) = M(S)$.

We review in Section 2 the basic algorithm due to Berman and Ramaiyer [1]. In Section 3 we prove some facts that lead to a reduction of the necessary iteration steps. Then we show how to precompute the data for each iteration efficiently, proving the time bound of $O(n^{3/2})$. Finally, in Section 5 the new fast algorithm is presented, which runs in $O(n \cdot \text{polylog}(n))$ time.

2.   The Basic Algorithm

The idea of the algorithm is, starting with $M(S)$, iteratively to compute optimal Steiner trees for small subsets z of the terminals (e.g., $|z| = 3$) and insert these small Steiner trees into the current tree. Let (s_1, s_2, s_3) be a triple of terminals with coordinates (x_1, y_1), (x_2, y_2), and (x_3, y_3), respectively. Such a triple is called a *star*, if $x_1 < x_2 < x_3$ and $(y_1 - y_2)(y_2 - y_3) < 0$. There are four types of stars corresponding to the four possible orders of the y-coordinates. In the following, we consider the case of $y_2 < y_1 < y_3$ (similar arguments can be used for the other types of stars), and denote s_1 as the *left point*, s_2 as the *bottom point*, and s_3 as the *top point* of the star.

A *center* of a star is the point $c = (x_2, y_1)$, which is the Steiner vertex of $SMT(s_1, s_2, s_3)$. We use $z = (c; s_1, s_2, s_3)$ to denote such a star.

More specifically the idea is to start with a minimum spanning tree for S, then to add iteratively some new points (centers of stars) and to construct new minimum spanning trees for the new point set. At the end we get a 3-restricted Steiner tree for the point set S reaching the claimed approximation ratio. Berman and Ramaiyer did that also for general $k \geq 3$, considering not only triples but k-tuples. However, the complexity of the algorithm increases very fast for higher k. Therefore we restrict ourselves to $k = 3$.

- For $u, v \in S$, $Bridge(u, v)$ is the longest edge on the path between u and v in the actual tree.
- $Bridge(z) = \{Bridge(u, v); u, v \in z\}$ for every star z.

The criterion for the insertion of the center of some star in Zelikovsky's algorithm is the *gain*(z) which is defined to be the difference of the cost of $Bridge(z)$ and $smt(z)$. Zelikovsky proposes always to add the star with the maximum gain as long as there are stars with positive gain. Thus the length of the actual minimum spanning tree is decreased and the approximation ratio of $11/8$ can be proven.

The algorithm of Berman and Ramaiyer also operates with stars and their gains but it is more involved. We need the following notations:

For $e = Bridge(u, v)$ with cost c, $alt(e) = (u, v)$ with cost $= (c - gain(z))$.

$$Altbridge(z) = \{alt(u, v); (u, v) \in Bridge(z)\}.$$

Intuitively, we simulate an $SMT(z) = M(z \cup c)$ where c is the center of z by inserting two new edges between the terminals of z (namely, the $Altbridge(z)$). Thus two cycles appear. $Bridge(z)$ denotes the set of edges which have to be removed from these cycles in order to get a new minimum tree.

Note that the definitions of gain, Bridge, alt, and Altbridge depend on an actual edge set E; we mark this set by a lower index in the following.

The algorithm of Berman and Ramaiyer consists of three phases: the evaluation phase, the selection phase, and the construction phase. In the evaluation phase we first compute $M(S)$. Then, for all stars z, the values $gain_E(z)$ are computed. E is initialized as the set of edges occurring in $M(S)$. If $gain_E \leq 0$, then z is discarded, otherwise B_{old} is removed from E and B_{new} is inserted instead, with $B_{\text{old}} = Bridge_E(z)$, $B_{\text{new}} = Altbridge_E(z)$. Then the triple $(z, B_{\text{old}}, B_{\text{new}})$ is stored in a stack. We call such star to be *treated*. In the selection phase the stars z are removed one by one from the stack, and the actual z is either discarded or *accepted* and then added to a list L if it might be included in the final tree. In more detail, (z, B_{new}) is inserted in the list if $B_{\text{new}} \subseteq E$, i.e., the $Altbridge(z)$ were not used as bridges by another star at a later moment in the evaluation phase; if not, the set B_{new} is removed from E and a new (actualized) minimum tree is recomputed. In the construction phase the stars stored in L are *realized* and so the output tree is built up from the entries of the list L.

During the algorithm we have to compute bridges, altbridges, and gains of all stars as well as some modifications on the actual set E and the corresponding MST. There are several methods which support manipulations on MSTs. Frederickson [5] gave a data structure for maintaining MSTs in dynamic graphs. Insertions and deletions are allowed

as are manipulations on the edge lengths. This method can also be used to compute
(alt)bridges and gains. The update time per step is $O(\sqrt{e})$, where e is the number of
edges. Another method [4] that only works on dynamic planar graphs takes only $O(\log n)$
time per operation.

Berman and Ramaiyer apply Frederickson's data structure and state that the number of
edges to consider is only $O(n)$, namely, the set of edges in the actual minimum spanning
tree + the set of edges in $SMT(z)$. Thus each basic step in Berman and Ramaiyer's
algorithm takes time $O(\sqrt{n})$. Trivially there are $\binom{n}{3} = \Theta(n^3)$ stars, so the algorithm
needs $\Theta(n^{3.5})$ time. In both papers [19] and [1], it has been observed that only $O(n^2)$
stars have to be considered, improving the time bound to $\Theta(n^3)$ [19] and $\Theta(n^{5/2})$ [1],
respectively.

In the next two sections we show that only $O(n)$ stars have to be considered and how
this set of stars is constructed efficiently.

3. $O(n)$ Stars Are Enough

In the following we prove that it is sufficient to consider only a linear number of stars
and this set can be constructed in time $O(n \cdot \log^2 n)$. This immediately implies a drastic
improvement of the running time of the 11/8-approximation algorithm down to $O(n^{3/2})$.

For brevity, we assume that coordinates of all terminals and subtractions between
them are distinct.

Consider the three terminals $s_1 = (x_1, y_1)$, $s_2 = (x_2, y_2)$, $s_3 = (x_3, y_3) \in S$. We again
assume without loss of generality that $x_1 < x_2 < x_3$ and $y_2 < y_1 < y_3$. The terminals
defining a star z also define a rectangle R where they lie on the boundary. R is empty if
there is no terminal (x, y) such that $x_1 < x < x_3$ and $y_2 < y < y_3$.

Berman and Ramaiyer show that it is sufficient to consider a family of stars for which
we know that there is a 3-restricted SMT with stars from this family. They also prove

Lemma 1.   *For any terminal set S there is a 3-restricted Steiner minimum tree $SMT_3(S)$
using only stars defining empty rectangles.*

A star $z = (c; s_1, s_2, s_3)$ is called a *tree star* if $M(S \cup c)$ contains the edges of the set
$\{(c, s_1), (c, s_2), (c, s_3)\}$.

Lemma 2.   *For any set S of terminals there is an $SMT_3(S)$ using only tree stars.*

Proof.   Let $z = (c; s_1, s_2, s_3)$ be a star of size 3 in an $SMT_3(S)$ T with the center c, E_z is
the set of edges between the points of z. $T \setminus E_z$ consists of three disjoint and nonconnected
terminal sets S_1, S_2, S_3, each of them containing a terminal $s_i \in S_i$, $i = 1, \ldots, 3$, that
is adjacent to c. Let e_1 and e_2 be the shortest edges that connect the components; thus
$T' := T \setminus E_z \cup \{e_1, e_2\}$ is a minimum spanning tree for the point set $S \cup S^* \setminus \{c\}$, where
S^* is the set of all Steiner points of T; this is because T is a minimum spanning tree
for the point set $S \cup S^*$. Assume that an edge $(u, c) \in E_z$ ($u \in \{s_1, s_2, s_3\}$) does not
belong to $T_{\min} := M(S \cup \{c\})$; let v be the terminal being adjacent to u lying on the path
between u and c in $T_{\min}$; the path P_1 between u and v in $T_{\min}$ and the path P_2 between

u and v in T' form a cycle in the point set $S \cup S^* \backslash \{c\}$; the longest edge on this cycle, say e, lies on P_1, since P_2 is a part of a minimum spanning tree of $S \cup S^* \backslash \{c\}$. (u, c) is longer than e, because e lies on the unique cycle of $T_{\min} \cup \{(u, c)\}$; thus (u, c) is longer than the longest edge on P_2 since e is longer than any edge on P_2. Therefore adding (u, c) to T and deleting the longest edge on the path P_2 would decrease the length of T, a contradiction since T is an $SMT_3(S)$. $\qquad\square$

Lemmas 1 and 2 imply

Lemma 3. *For any terminal set S there is an $SMT_3(S)$ using only tree stars defining empty rectangles.*

We call such stars *proper stars*.

A star is *positive (negative)* if $(y_2 - y_0) - (x_2 - x_1) > 0 \, (< 0)$. This means that the top point lies above (below) the diagonal through the center of the star. Yao [18] introduced a graph associated with the set S: every vertex is connected to the nearest vertices in all eight angles defined by axes and bisectors. The Yao graph contains $M(S \cup c)$. Since in $M(S \cup c)$ only one terminal point from each sector might be connected to c it implies directly the following:

Lemma 4. *For any terminal set S there are at most two distinct tree stars with the same center c, namely the positive and the negative with the shortest length.*

Let c be the center of a star z. If the bottom point b is closer to the center c than the left point l, then the star has a *bottom root*, otherwise it has a *left root*. Accordingly, a star is either named *bottom* or *left*.

**Lemma 5.** *For any terminal set S the number of left stars is at most $4n$.*

Proof. Let $s = (x, y)$ be a left root of two stars $z = (c = (x_1, y); s, s_1 = (x_1, y_1), s_2)$ and $z' = (c' = (x_1', y); s, s_1' = (x_1', y_1'), s_2')$ and $x_1 < x_1'$. Then $y_1 < y_1'$, since both stars are empty, and $y - y_1' > x_1' - x$, since z' is a left star. Thus s_1' lies below the diagonal drawn in Fig. 1 and so the rectilinear distance between s_1 and s_1' is smaller than the distance between s_1 and c. Since the edge (c, s_1) belongs to $M(S \cup c)$, the distance between c and s_1' has to be larger than (c, s_1) (otherwise (c, s_1) would be the longest edge in a cycle), that means $y_1' - y_1 < x_1' - x_1$. Thus

$$y - y_1' > y_1' - y_1. \tag{1}$$

Claim 1. *There is no other left star with the bottom point s_1.*

Proof. Indeed, let $(d = (x_1, y'); s' = (x', y'), s_1, s_3)$ be such a star. This star together with z and z' are empty, so $x' < x$ and

$$y_1' > y'. \tag{2}$$
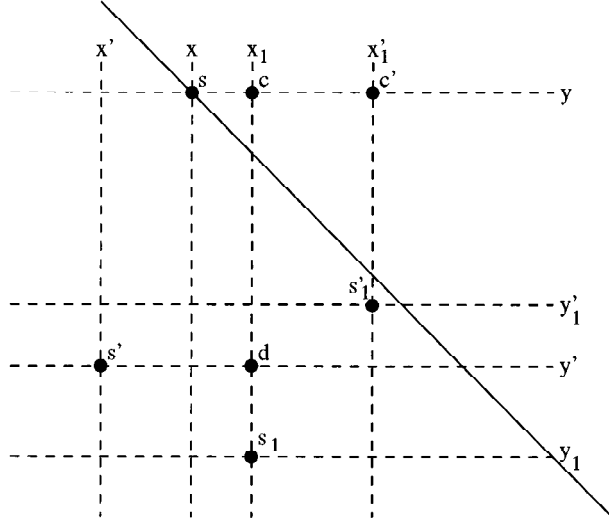
Fig. 1.   Two proper left stars with the same root.

Since c and d are both centers of left stars,

$$|s, c| \ < \ |c, s_1|, \tag{3}$$

$$|s, s'| \ < \ |c, d| + |d, s'| \ < \ |c, d| + |d, s_1| = |c, s_1|. \tag{4}$$

Inequalities (1) and (2) imply

$$|s', s_1| = |s', d| + |d, s_1| < |s_1, d| + |c, d| = |c, s_1|. \tag{5}$$

Inequalities (3)–(5) imply that the edge (c, s_1) does not belong to $M(S \cup c)$, because it is the longest edge in the cycle c, s, s', s_1, c.   □

We count the number of possible centers for left stars: Imagine a grid having one row and one column through every terminal, thus n rows and n columns. The n^2 intersections of this grid are possible locations for such centers. If there are r ($r \geq 2$) centers for left stars in the same row (i.e., the terminal in this row is the left root of k tree stars), then by Claim 1 in $r - 1$ corresponding columns (all but the rightmost) there are no other centers. Thus r ($r \geq 1$) centers in a row imply $(r - 1) \cdot (n - 1) = nr - n - r + 1$ intersections that are not centers. Thus for $R = \sum_i r_i$ centers there are $\sum_i (nr_i - n - r_i + 1) = nR - n^2 - R + n$ noncenter intersections. The total number of intersections (centers and noncenter intersections) is n^2. Hence, $nR - n^2 - R + n + R \leq n^2$ and $R \leq (2n^2 - n)/n$ or $R < 2n$. Now, Lemma 5 follows from Lemma 4.   □

Similarly, the number of bottom stars is at most $4n$. Thus, for all possible four types of stars, Lemma 5 implies

Lemma 6.   *The number of proper stars is at most* $32n$.

It follows directly from Lemmas 2 and 6, that we only need a linear number of stars during the construction of a 3-restricted Steiner tree. In the next section we show how to determine these stars.

4.   Computing the Stars

Like in the previous section we concentrate on the computation of left stars, the other kinds of stars can be determined analogously. From the proof of Lemma 5 we can extract the following conditions to get a linear number of left stars; Fig. 2 illustrates the method described in this section: For each left root $s_l = (x_l, y_l)$ let $s_r = (x_r, y_r)$ be the rightmost candidate for a bottom point such that the rectangle defined by s_l and s_r is empty and s_r lies below the diagonal $y = -x + x_l + y_l$ (otherwise s_r would lie closer to the center than s_l and thus s_l is not the root of the star). It is clear that simultanously s_r has maximal y-coordinate among all candidates for bottom points.

If s_r does not exist, then s_l cannot be the left root of any star. Otherwise we continue our search for more bottom points following the proof of Lemma 5: If there is another bottom point $s' = (x', y')$ of a left star with left root s_l, then the following three conditions hold: $x_l < x' < x_r$, $y' < y_r$, and $y_r - y' < x_r - x'$. The last condition means that s' lies above the diagonal $y = x + y_r - x_r$, which intersects s_r, and is necessary to fulfill the
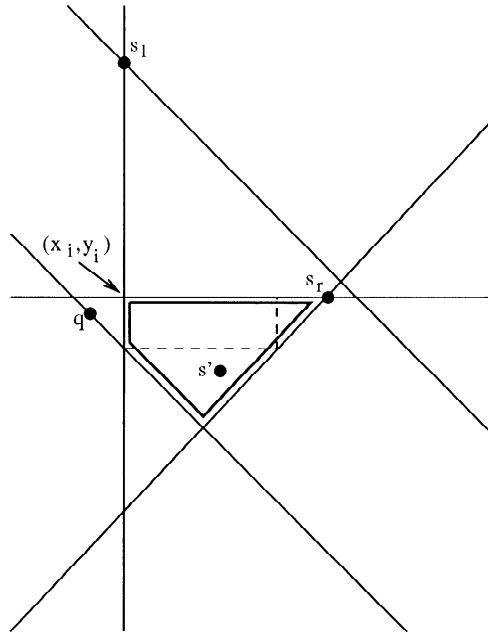
Fig. 2.   Computing left stars.

tree star condition for s'. The proof of Lemma 5 states that if there is another terminal $p = (x_p, y_p)$ with $y' < y_p < y_r$, $x_p < x_l$, and p also lies closer to the center defined by p and s' than s', then s' is not a bottom point (in Fig. 2 q plays the role of our p, but q does not fulfill the left root condition; consequently, a star having q as the left point and s' as the bottom point would be a bottom star, not a left star). This means that we have to search for a terminal s' with maximal y-coordinate, to the right of s_l, to the left of and below s_r which has no other terminal in the triangle described by the two diagonals through s' and the horizontal line through s_r (in Fig. 2 q lies outside this triangle). This terminal can be taken to be a bottom point of a left star with left root s_l, it should be deleted from the data structure of the bottom points such that it will not be taken later as a bottom point for other left roots. Then we iterate our search where s' plays the role of s_r. This is correct since there is no other terminal to the right of s', which might be a bottom point candidate for a star with s_l. More formally, we get the following:

Algorithm

(1) sort the terminals in S in decreasing order with respect to their y-coordinates
(2) **for** all $s_l = (x_l, y_l) \in S$ (from top to bottom)
(3)   find the rightmost terminal $s_r = (x_r, y_r)$ with $x_l < x_r$, $y_l > y_r$, and $y_l - y_r > x_r - x_l$;
(4)   test the tree star condition;
(5)   determine terminal $s' = (x', y')$ with the following four properties:
         (a) $x_l < x' < x_r$,
         (b) the triangle determined by the diagonals $y = -x + x' + y'$ and $y = x + y' - x'$ and by the horizontal line $y = y_r$ is empty,
         (c) y' is maximal;
         (d) s' is unmarked,
(6)   **if** s' exists **then** mark it, test the tree star condition,
(7) $s_r := s'$; **goto** (5);

The second loop (steps (5)–(7)) can be simplified: Determine the intersection (x_i, y_i) of the vertical line through s_l and the horizontal line through s_r and the terminal $q = (x_q, y_q)$ with $x_i > x_q$, $y_i > y_q$ and $y_i - y_q + x_i - x_q$ is minimal. q is the closest terminal lying left and below the intersection of the two lines. Instead of computing different triangles for the different terminals below the horizontal line through s_r, we can replace step (5) by

(5′) determine terminal $s' = (x', y')$ with the following three properties:
         (a) s' lies in the polygon determined by the vertical line through x_l, the diagonals $y = x + y_r - x_r$ and $y = -x + y_q + x_q$ and $y = y_r$;
         (b) y' is maximal;
         (c) s' is unmarked,

The following figure indicates what has to be done in step (5′). Marking the terminals is done by deleting them from the data structure for the bottom points.

   To simplify step (5′) we can partition the query polygon into two triangles and one rectangle, solve the problem in these simpler query polygons, and maximize over the

resulting terminals. In the following we show how to solve one of the query problems efficiently. The other problems can be done similarly.

We start with an abstract description: For real numbers a, b, and h, where $a < b$, let $T(a, b, h)$ denote the triangle that is bounded by the lines $y = h$, $y = x + h - b$, and $y = -x + h + a$. Hence, the upper side of this triangle is horizontal and its other two sides have equal length and are parallel to the diagonals $y = x$ and $y = -x$.

Let S be a set of n points in the plane. We want to maintain the points of S under insertions and deletions such that, for any query triangle $T = T(a, b, h)$, we can find a point of $S \cap T$ having maximal y-coordinate.

First note that a point (x, y) is contained in T iff $y \geq x + h - b$, $y \geq -x + h + a$, and $y \leq h$. We transform our problem as follows:

Transform each point $(x, y) \in S$ to the point $(\alpha, \beta, \gamma) \in \mathbb{R}^3$, where $\alpha = y + x$, $\beta = y - x$, and $\gamma = y$. Let S' be the resulting set of points. A query triangle $T(a, b, h)$ is transformed to the point (A, B, C) in $\mathbb{R}^3$: $A = h + a$, $B = h - b$, and $C = h$.

In the transformed problem we want to find a point (α, β, γ) of S such that $\alpha \geq A$, $\beta \geq B$, $\gamma \leq C$, and having a maximal third coordinate. We first solve a simpler problem: Maintain a set S'' of planar points in a data structure such that for any query point (B, C) we can find a point $(\beta, \gamma) \in S$, such that $\beta \geq B$, $\gamma \leq C$, and γ is maximal. This problem can be solved by means of priority search trees. The solution uses $O(n)$ space and has query and update times of $O(\log n)$.

Back to our problem for S'. We store the points of this set in the leaves of a balanced binary search tree, sorted by their first coordinates. Each node u of this tree contains a pointer to an associated structure: Let S'_u be the set of points of S' that are stored in the subtree of u. Then the associated structure of u is a priority search tree for the set S'_u taking only the second and third coordinates into account.

A query is solved as follows. Let (A, B, C) be a query point. We search in the tree for A. During this search, each time we move from a node u to its left son, we query the associated structure of the right son v of u. That is, we find a point $(\beta, \gamma) \in S'_v$ such that $\beta \geq B$, $\gamma \leq C$, and γ is maximal. At the end of the search, we have found $O(\log n)$ candidates. The point with the maximal third coordinate among them is the solution. That is, this point satisisfies $\alpha \geq A$, $\beta \geq B$, $\gamma \leq C$, and has a maximal third coordinate among such points.

It is clear that the query time is $O(\log^2 n)$. Moreover, the data structure uses $O(n \log n)$ space. Using standard dynamization techniques, the data structure can be maintained under insertions and deletions in $O(\log^2 n)$ amortized time [16].

In our case we only need deletions. The other queries can be performed analogously. This proves the following theorem:

Theorem 1. *In time $O(n \log^2 n)$ we can determine the $O(n)$ stars which are sufficient for the execution of the approximation algorithms. Hence for any rectilinear Steiner problem an $11/8$-approximation can be found in time $O(n^{3/2})$.*

5. A Fast Parametrized Version of the Algorithm

In this section we present a variant of the algorithm which saves a factor of nearly $\sqrt{n}$ in the time bound. In time $O(n \cdot \text{polylog}(n))$ a performance ratio of $11/8 + \varepsilon$ can be reached

for any constant $\varepsilon > 0$. The O in the running time of the algorithm is proportional to the parameters r and m, the performance ratio will be $11/8 + 1/2m + 1/(8 \cdot 32^{r-2})$. The new algorithm combines the merits of the algorithms in [1] and [19]: While Zelikovsky always computes the actual best star, Berman and Ramaiyer allow the order of treating the stars in the evaluation phase to be arbitrary; therefore they need the selection phase. Our idea is not always to compute the best star (with the highest gain), but yet one with a gain of at least $m/(m+1)$ times the optimum. So we save the selection phase (like Zelikovsky does) and are not forced to compute an optimal star $O(n)$ times.

Definition 1.

(a) g_i is the gain of star z_i at the beginning of the repeat loop.
(b) a_i is the gain of star z_i at the time of its treatment (actual gain).
(c) a_i' is the gain of star z_i at the time of its treatment, where no artificial edges are used, i.e., $\mathrm{bridges}(z_i) :=$ edges of maximal length on the paths between the terminals of the star which are original, i.e., $\subseteq \mathrm{M}(S)$.

In the algorithm we always compute a_i' instead of a_i, therefore a new edge (a B_{new} of an older star) will never be deleted and all stars once stored in the list *RES* will be constructed. We have to show that the cases where a star has a large a_i but small a_i' and is therefore discarded do not weaken the result too much.

Furthermore, we only consider "planar" stars, i.e. stars where the *altbridges* do not cross an existing edge. Here we also have to prove that nonplanar stars would not achieve too much further gain, i.e., the performance ratio can be proved even without treating nonplanar stars. The planarity of the structure is necessary to ensure the efficient computing of the gains in time $O(\log n)$ [4].

<div align="center">Algorithm</div>

(1) compute the stars z_i $(i = 1, \ldots, n)$ and store them in a list L;
(2) **repeat** $r \cdot m \cdot \log n$ times:
(3) $g_i := gain(z_i)$ $(i = 1, \ldots, n)$;
(4)   sort L according to g_i (in decreasing order);
(5)   let j be maximal such that $g_j \geq g_1(m/(m+1))$
(6)   **for** $i := 1$ **to** j **do**
(7) $a_i' :=$ actual $gain(z_i)$ with $bridges(z_i) \subseteq E$;
(8)   **if** $a_i' \geq g_1(m/(m+1))$ **and** z_i is planar **and** $B_{\mathrm{old}}(z_i) \subseteq E$
(9)   **then**
(10)   store z_i in a list *RES*;
(11)   delete z_i from L;
(12) $M := M \setminus B_{\mathrm{old}}(z_i) \cup B_{\mathrm{new}}(z_i)$;
(13)   **endif**
(14)   **endfor**
(15) **endrepeat**.

Note that if a star does not fulfill the condition $a_i' \geq g_1(m/(m+1))$ in line 8 it is not automatically refused; it remains in the list L and will be considered again in subsequent rounds of the repeat loop.

In the next subsections we prove

Theorem 2. *For any given set of terminals S, $|S| = n$, the algorithm computes a Steiner tree $ST(S)$ in time $O(r \cdot m \cdot n \cdot \log^2 n)$ for any given parameters $r, m > 0$. Its performance ratio is at most $11/8 + 1/2m + 1/(8 \cdot 32^{r-2})$.*

5.1. *Performance Ratio*

In this section we prove the performance ratio of the algorithm. We compare the list of our accepted stars with the set of treated stars of the algorithm of Berman and Ramaiyer where the performance ratio of 11/8 is known. Recall the definition of *treated* in Berman and Ramaiyer's algorithm. Our aim is to reach any constant fraction of the gain they can realize.

Every star z having positive gain at the end of the evaluation phase has (at least) one of the following characteristics:

  (a) *gain(z)* (i.e., a_z) $\leq g_1(m/(m+1))$, where g_1 is the gain of the best star at the beginning of the final round of the repeat loop.
  (b) z is not planar.
  (c) *bridges(z)* $\nsubseteq E$, i.e., z would delete an artificial edge.

For the following analysis let x be the sum of the gains of the treated stars in the algorithm of Berman and Ramaiyer.

Lemma 7. *After the $(m \cdot r \cdot \log n)$th execution of the repeat loop (line 2), the remaining gain of all planar stars z for which the condition bridges(z) $\subseteq E$ holds sums to at most $x/32^{r-2}$.*

Proof. First we consider the case $m = 1$. Let *Max* be the maximal gain of a star at the beginning of the first round. Then after i rounds every star has gain $\leq Max/2^i$ (in every round the "better half" of the stars is executed), thus after $r \cdot \log n$ rounds $\leq Max/n^r$. Since there are at most $32n$ stars the whole gain left is $\leq 32nMax/n^r = 32Max/n^{r-1}$, or (for $n \geq 32$) $\leq Max/32^{r-2} \leq x/32^{r-2}$.

In the case $m > 1$ we simulate one round for $m = 1$ by several rounds. How many rounds are necessary, such that the gain of any remaining star is bounded by $Max/2$? After i rounds every star has gain $\leq Max \cdot (m/(m+1))^i$. Setting $i = m$, the factor becomes $(m/(m+1))^m$ which converges for large m to $1/e < \frac{1}{2}$. Therefore after m rounds the gain of every remaining star is at most $Max/2$. After $m \cdot r \cdot \log n$ rounds, the total gain of all nontreated stars is bounded by $x/32^{r-2}$. $\qquad\square$

Lemma 8.

$$\max_{\hat{S}\ independent\ set\ of\ stars} \sum_{z \in \hat{S}} (a_z - a'_z) \leq \frac{2x}{m},$$

i.e., the sum of all gains we lose by refusing to delete artificial edges (edges B_{new} of older stars) is bounded by $2x/m$.

Proof. When a star z_{i_1} is treated it uses an edge $e_{i_1}^*$ as B_{old} and creates an edge e_{i_1} as B_{new}. Let $\bar{S}$ be the set of stars z with positive gain and $e_{i_1} \in bridges(z)$. Let z_{i_2} be the star for which $a_{z_{i_2}} - a'_{z_{i_2}}$ is maximal. It follows that at the time of the treatment of z_{i_1}, z_{i_2} had a larger gain than z_{i_1}, since otherwise z_{i_2} would have had $e_{i_1}^*$ as a bridge instead of e_{i_1}. So the gain we lose by the wrong choice of z_{i_1} instead of z_{i_2} is at most a_{i_2}, namely, the $gain(z_{i_2})$ at the time when z_{i_2} was treated; for only *one* star $\in \bar{S}$ could be realized using the edge e_{i_1} as a bridge. Thus the total loss is at most a_{i_2}. z_{i_2} had a gain of at least $a_{i_1} + a_{i_2}$ before the treatment of z_{i_1} ($a_{i_1} = |e_{i_1}^*| - |e_{i_1}|$). Thus $g_{i_2} \geq a_{i_1} + a_{i_2}$ or $g_{i_2} - a_{i_1} \geq a_{i_2}$.

We conclude from the acceptation of z_{i_1}:

$$a_{i_1} \geq a'_{i_1} \geq \frac{m}{m+1} g_1 \geq \frac{m}{m+1} g_{i_1}$$

$$\Rightarrow \quad a_{i_2} \leq g_{i_2} - a_{i_1} \leq g_{i_1} - a_{i_1} \leq \frac{m+1}{m} a_{i_1} - a_{i_1} = \frac{a_{i_1}}{m}.$$

z_{i_1} produces *two* new edges that could be used as B_{old} by other stars. Thus the loss resulting from the wrong choice of a_{i_1} can be bounded by $2a_{i_1}/m$.

Summing over all stars, the total loss resulting from the discard of stars with artificial edges can be bounded by $\leq 2x/m$. $\qquad\square$

Next we give the following property on intersecting stars:

Lemma 9. *Let T be a 3-restricted SMT with two stars a,b intersecting each other. Then we can replace these stars by another star z without increasing the total length of the SMT.*

Proof. Let $a = (a_1, a_2, a_3)$ and $b = (b_1, b_2, b_3)$ be two intersecting stars of a 3-restricted SMT T. These stars define empty rectangles; thus the height of one of them should be larger than the height of the other. The former star, say a, and its induced rectangle is called *vertical* and the latter star, say b, is *horizontal*. Then the two rectangles form a cross with the regions A, B, C, D, E (see Fig. 3). Note that some of the external regions (A, B, C, D) could have expansion 0, then the cross degenerates. We will look at this case later. The total length of both stars a and b is equal to half of the perimeter of the minimal rectangle which contains this cross, say p ($= p_x + p_y$), plus the height of the horizontal rectangle, say h, plus the width of the vertical one, say w. Two of the external regions contain two terminals each, the other two regions contain one terminal each; and we know that the two regions containing two terminals do not lie at opposite sides of E. Without loss of generality regions A and D contain two terminals each (otherwise we have to turn the cross). We now have the situation that a_2 and a_3 lie in region D, b_2 and b_3 in region A, a_1 lies in region B, and b_1 in C. Let P be the shortest path which connects the stars a and b in T. We distinguish two cases:

(I) *P connects terminals from A and D (see Fig. 4).* Consider the star $z = \{a_1, b_1, a_j\}$ where a_j has the larger y-coordinate among $\{a_2, a_3\}$. Then z together with the edges (a_2, a_3) and (b_2, b_3) and P connects the six terminals and is not longer than a, b, and P:

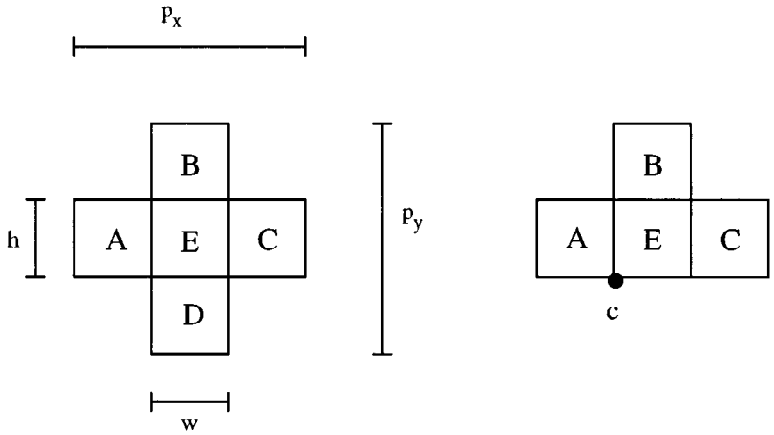$$d(z) + d(a_2, a_3) + d(b_2, b_3) \leq p + h + w = d(a) + d(b).$$

P_x

h

P_y

c

w

Fig. 3.   Two intersecting stars.

(II) *P connects, e.g., B and A*. Consider the star $z = \{b_1, a_j, b_j\}$ where a_j has the larger y-coordinate among $\{a_2, a_3\}$ and b_j has the larger x-coordinate among $\{b_2, b_3\}$. As in case I the cost of z plus the lengths of the edges (a_2, a_3) and (b_2, b_3) is not higher than the cost of a and b together. The same holds for the star $z = \{a_1, b_1, a_j\}$ and the edges (a_2, a_3) and (b_2, b_3).

We now consider the case where the stars a and b have a common end c such that one of the external regions, say D, has expansion 0, because c would be the only terminal
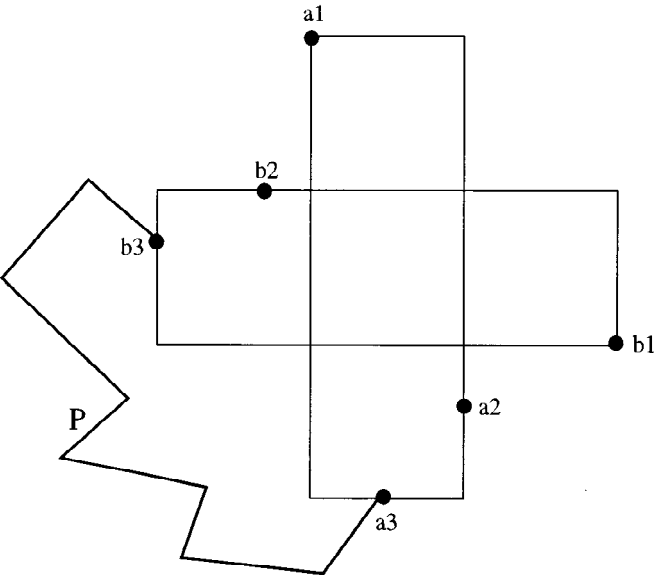
a1

b2

b3

b1

a2

P

a3

Fig. 4.   Replacing intersecting stars.

in D (right-hand part of Fig. 3); now c lies at the border of D and A. We double the terminal c such that both A and D can be regarded as regions with one terminal (c_1 or c_2, respectively) and can apply the constructions above. $\square$

Corollary 1.   *In the situation of Lemma 9 there is always a better star $\hat{z}_h$ that contains two terminals of the horizontal star, namely, those that lie closely at the left and right of the intersecting vertical star. Analogously the existence of a better star $\hat{z}_v$ can be proven, containing the two corresponding terminals of the vertical star.*

Proof.   Follows directly from the constructions in Lemma 9. $\square$

Now we are able to prove

Lemma 10.   *At the end of the evaluation phase all stars with positive gain that were refused because of nonplanarity together have a gain of at most $2x/m$.*

Proof.   Let z_{i_2} be a star with positive actual gain that intersects an accepted star z_{i_1}. Without loss of generality among all accepted stars producing an intersection with z_{i_2}, z_{i_1} is the "oldest," i.e., the earliest to be accepted. From Lemma 9 we know that there exists a star $\hat{z}$ which is not accepted and for which $a_{\hat{z}} \geq a_{z_{i_1}} + a_{z_{i_2}}$ holds. There can be four reasons why $\hat{z}$ has not been accepted:

(a) $\hat{z}$ was treated before z_{i_1} but was not planar. From the construction of $\hat{z}$ it follows that one of the stars z_{i_1} or z_{i_2} is not planar in this case, too. This star cannot be z_{i_1} because it is already accepted, therefore it has to be z_{i_2}. However, this means that z_{i_2} intersects another star that was treated before $\hat{z}$ which contradicts the conditions of z_{i_1}.

(b) $\hat{z}$ was treated before z_{i_1} but uses an artificial edge. $\hat{z}$ would be accepted using an artificial edge but refused when not using artificial edges. That means that the loss originating from discarding $\hat{z}$ is already counted earlier (see Lemma 8), even if z_{i_1} would be refused too.

(c) $\hat{z}$ was treated before z_{i_1} but its actual gain $a_{\hat{z}}$ was too small for an acceptation. In this case $a_{z_{i_1}}$ would also be too small and we have a contradiction to the acceptation of z_{i_1}.

(d) $\hat{z}$ was treated after z_{i_1}, $\Rightarrow g_{\hat{z}} \leq g_{z_{i_1}}$. Now we have

$$a_{z_{i_1}} + a_{z_{i_2}} \leq a_{\hat{z}} \leq g_{\hat{z}} \leq g_{z_{i_1}} \leq a_{z_{i_1}} \frac{m+1}{m}$$

$$\Rightarrow a_{z_{i_2}} \leq a_{z_{i_1}} \frac{m+1}{m} - a_{z_{i_1}} = a_{z_{i_1}} \frac{1}{m}.$$

So we lose very little.

Finally we consider the question, How many stars intersecting the same star z_{i_1} do we lose? For the notation see Fig. 5. Let $\hat{S}_l$ and $\hat{S}_r$ be the better stars instead of H and V_t, or H and V_{t+1}, each containing the terminals a and b, or b and c, respectively (the existence
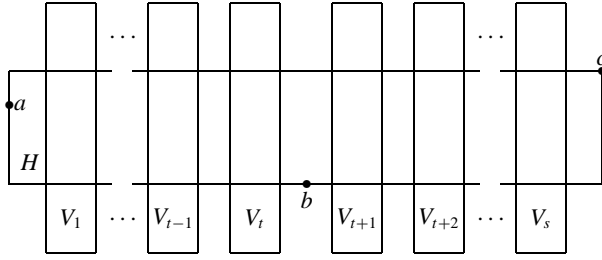
Fig. 5. One star intersects several other stars.

of the stars $\hat{S}_l$ and $\hat{S}_r$ follows from Corollary 1). With similar arguments as in the case where one intersection holds: $\hat{S}_l$ and $\hat{S}_r$ were treated *after* H, therefore $g_{\hat{S}_l} \le g_H$ and $g_{\hat{S}_r} \le g_H$. $\hat{S}_l$ is better than H and V_t together and intersects V_{t-1}; $\Rightarrow$ there is a star $\hat{S}'_l$ being better than $\hat{S}_l$ and V_{t-1} together, therefore better than H, V_t, and V_{t-1} together and $\hat{S}'_l$ intersects V_{t-2} and so on. The same construction is valid for the right side, i.e., the stars $V_{t+1}, \ldots, V_s$. Thus there are stars S^*_l and S^*_r with

$$a_{S^*_l} + a_{S^*_r} \ge 2a_H + \sum_{i=1}^{s} a_{V_i}.$$

Now, similar to the inequality above:

$$2a_H + \sum_{i=1}^{s} a_{V_i} \le a_{\hat{S}_l} + a_{\hat{S}_r} \le g_{\hat{S}_l} + g_{\hat{S}_r} \le 2g_H \le 2a_H \frac{m+1}{m} \quad \Rightarrow \quad \sum_{i=1}^{s} a_{V_i} \le \frac{2a_H}{m}.$$

Thus by discarding a family of nonplanar stars we lose at most $2/m$ gain of a star already accepted. This completes the proof of Lemma 10. $\square$

We conclude that the total gain g that could be realized is thus at least

$$\frac{smt_2(S) - smt_3(S)}{2} \left(1 - \left(\frac{1}{32^{r-2}} + \frac{2}{m} + \frac{2}{m} \right) \right).$$

This leads to a performance ratio of

$$\frac{3}{2} - \frac{\frac{3}{2} - \frac{5}{4}}{2} \left(1 - \frac{4}{m} - \frac{1}{32^{r-2}} \right) = \frac{11}{8} + \frac{1}{2m} + \frac{1}{8 \cdot 32^{r-2}}.$$

5.2. *Running Time*

Here we want to analyse the algorithm to prove the time bound of $O(r \cdot m \cdot n \cdot \log^2 n)$. As shown in Section 4, line 1 can be executed in time $O(n \cdot \log^2 n)$.

Lines 3–5 obviously need $O(n \cdot \log n)$ time for sorting $O(n)$ objects.

We cannot keep j small $(j \in \Theta(n))$; but since we ensure that the stars do not intersect (line 8) we can apply the methods described in [4], such that the recomputations of the gains can be done in time $O(\log n)$. Thus lines 7–12 can be executed in time $O(\log n)$. This is the reason why we ask the stars to be planar in line 8.

In total, the algorithm needs $O(r \cdot m \cdot n \cdot \log^2 n)$ time which is $O(n \cdot \log^2 n)$ if we keep the parameters r and m constant. If we choose $r = \log \log n$ and $m = \log n / \log \log n$, we yield an $O(n \cdot \log^3 n)$ algorithm with a performance ratio of $11/8 + O(\log \log n / \log n)$ which is asymptotically equal to $11/8$.

6.   Conclusion

In this paper refined versions of the new approximation algorithms for rectilinear Steiner trees of Zelikovsky and Berman and Ramaiyer were presented. Aiming for faster algorithms, not for better approximations, we improved the previous best time bound of $O(n^{5/2})$ considerably. For the $11/8$ approximation we showed a running time of $n^{3/2}$. An alternative algorithm for a parametrized approximation gave an $11/8 + \varepsilon$ approximation for any $\varepsilon > 0$ within time $O(n \cdot \log^2 n)$, which is close to optimal. Allowing time $O(n \cdot \log^3 n)$, this algorithm provided an $11/8 + \log \log n / \log n$ approximation. To improve the running time and the approximation further, new methods seem to be necessary.

Acknowledgement

References

1. P. Berman, V. Ramaiyer, Improved approximations for the Steiner tree problem. *Proceedings*, 3*rd ACM–SIAM Symposium on Discrete Algorithms*, pp. 325–334, 1992.
2. M.W. Bern, R.L. Graham, The shortest-network problem. *Sci. Amer.*, **1**, 66–71, 1989.
3. Ding-Zhu Du, Yanjun Zhang, Qing Feng, On better heuristic for Euclidean Steiner minimum trees. *Proceedings*, 32*nd Foundations of Computer Science*, pp. 431–439, 1991.
4. D. Eppstein, G.F. Italiano *et al.*, Maintenance of a minimum spanning forest in a dynamic planar graph. *Proceedings*, 1*st ACM–SIAM Symposium on Discrete Algorithms*, pp. 1–11, 1990.
5. G. Frederickson, Data structures for on-line updating of minimum spanning trees, with applications. *SIAM J. Comput.*, **14**, 781–789, 1985.
6. M.R. Garey, D.S. Johnson, The rectilinear Steiner problem is NP-complete. *SIAM J. Appl. Math.*, **32**, 826–834, 1977.
7. E.N. Gilbert, H.O. Pollak, Steiner minimal trees. *SIAM J. Appl. Math.*, **16**, 1–29, 1968.
8. M. Hanan, On Steiner's problem with rectilinear distance. *SIAM J. Appl. Math.*, **14**, 255–265, 1966.
9. F.K. Hwang, On Steiner minimal trees with rectilinear distance. *SIAM J. Appl. Math.*, **30**, 104–114, 1976.
10. R.M. Karp, Reducibility among combinatorial problems. In Miller and Thatcher (eds.), *Complexity of Computer Computations*, Plenum, New York, pp. 85–103, 1972.
11. B. Korte, H.J. Prömel, A. Steger, Steiner trees in VLSI-layout. In B. Korte *et al.* (eds.), *Paths, Flows and VLSI-Layout*, Springer-Verlag, Berlin, pp. 185–214, 1990.

12. Th. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*. Wiley, New York, 1990.
13. K. Mehlhorn, *Data Structures and Algorithms III*: *Multidimensional Data Structures and Computational Geometry*. Springer-Verlag, Berlin, 1985.
14. D. Richards, Fast heuristic algorithms for rectilinear Steiner trees. *Algorithmica*, **4**, 191–207, 1989.
15. H. Takahashi, A. Matsuyama, An approximate solution for the Steiner problem in graphs. *Math. Japon.*, **24**, 573–577, 1980.
16. D.E. Willard, G.S. Lueker, Adding range restriction capability to dynamic data structures. *J. Assoc. Comput. Mach.*, **32**, 597–617, 1985.
17. Y.F. Wu, P. Widmayer, C.K. Wong, A faster approximation algorithm for the Steiner problem in graphs. *Acta Inform.*, **23**, 223–229, 1986.
18. A.C. Yao, On constructing minimum spanning trees in k-dimensional space and related problems. *SIAM J. Comput.*, **11**, 721–736, 1982.
19. A.Z. Zelikovsky, The 11/8-approximation algorithm for the Steiner problem on Networks with rectilinear distance. In *Sets*, *Graphs and Numbers*, Colloq. Math. Soc. Janos Bolyai, Vol. 60, North-Holland, Amsterdam, pp. 733–745, 1992.
20. A.Z. Zelikovsky, An 11/6-approximation algorithm for the Steiner problem on graphs. *Algorithmica*, **9**, 463–470, 1993.