JOURNAL OF ALGORITHMS 17, 381–408 (1994)

Improved Approximations for the Steiner Tree Problem

PIOTR BERMAN*

Department of Computer Science, Pennsylvania State University, University Park, Pennsylvania 16801.

AND

VISWANATHAN RAMAIYER

Tandem Corporation, Santa Clara, California 95051

Received May 1, 1992; revised March 4, 1994

For a set S contained in a metric space, a Steiner tree of S is a tree that connects the points in S. Finding a minimum cost Steiner tree is an NP-hard problem in Euclidean and rectilinear metrics, as well as in graphs. We give an approximation algorithm and show that the worst-case ratio of the cost of our solutions to the optimal cost is better than previously known ratios in graphs and in the rectilinear metric on the plane. Our method offers a trade-off between the running time and the ratio: on one hand it always allows improvement of the ratio, on the other hand it allows previously known ratios to be obtained with greater efficiency. We use properties of optimal rectilinear Steiner trees to obtain significantly better ratio and running time in the rectilinear metric. © 1994 Academic Press, Inc.

1. INTRODUCTION

For a metric space (V, d) and a subset S of V, a tree T is a *Steiner tree* of S if S is contained in the vertex set of T. Given (V, d) and S, the Steiner tree problem is defined as the problem of computing a *Steiner minimal tree* of S, i.e., a Steiner tree of S with minimal cost. In general we assume that an instance of the problem consists of a description of the metric, e.g., in the form of a graph with edge costs, and the set S. While our algorithm can be applied to any metric, we focus mostly on two cases —the metric induced by graphs with edge costs, and the rectilinear metric on the plane.

* Partially supported by NSF Grant CCR-9114545.

It is known that the Steiner tree problem is NP-hard when the metric is given as a graph [14], as well as when the metric is Euclidean [7] or rectilinear [8]. Bern and Plassmann [2] showed that this problem is Max-SNP hard when the metric is given as a graph, thus strongly suggesting the performance ratio attainable by polynomial time algorithms is a constant larger than 1 (we measure the quality of an approximation algorithm by its performance ratio: an upper bound on the ratio between the achieved cost and the optimal cost). It remains an important question however, how good (i.e., close to 1) the performance ratio can be.

A well-known heuristic for this problem uses a minimum spanning tree of S as an approximate solution. This heuristic has a performance ratio of 2 for any metric [15, 18] and $3/2$ for the rectilinear metric on the plane [13]. Recently Du and Hwang [5] showed that the same heuristic achieves a ratio of $2/\sqrt{3}$ for the Euclidean metric on the plane. For many years, the problem of finding a better heuristic remained open.

Recently Zelikovsky [19] gave an approximation algorithm with a ratio of $11/6(\approx 1.84)$ when the metric is specified in the form of a graph, thus improving upon the ratio of 2. This algorithm constructs Steiner minimal trees for 3-element subsets of S and includes them in the solution using a greedy strategy. This achievement raises the following questions:

1. Can the performance ratio be improved even further by considering Steiner minimal trees for subsets of S of size bounded by a constant larger than 3?

2. Can this approach be extended to other metrics such as the Euclidean and rectilinear metrics to improve upon previously known performance ratios for these metrics?

First we address question 1. The natural extension of Zelikovsky's approach to subsets with more than three elements does not improve the performance ratio of $11/6$ for metrics given in the form of a graph. However, in this paper we provide a different way of utilizing Steiner minimal trees for subsets of size limited by a constant $k \geq 3$. For $k = 3$, our algorithm runs faster than Zelikovsky's algorithm and achieves the same ratio of $11/6$. For $k > 3$, we show that our algorithm improves upon this ratio. For example, our algorithm achieves a ratio of $16/9\,(\approx 1.78)$ for $k = 4$. We also show by using a result of Du *et al.* [4] that doubling the parameter k always improves the ratio achieved by our algorithm. Table 1 shows the proven performance ratio of our heuristic (denoted as r) for some values of k (the performance ratio converges to a value close to 1.746).

Next we consider question 2 above. We show that Zelikovsky's algorithm as well as our algorithm achieve improvements over the current

TABLE 1
Performance Ratio in Graphs

k	3	4	8	16	32
r	1.834	1.778	1.754	1.749	1.747

TABLE 2
Performance Ratio in Rectilinear Metric

k	3	4	5	6	7
r	1.375	1.348	1.337	1.332	1.330

best ratio of 3/2 in rectilinear metric on the plane. The ratio achieved by our algorithm (with $k = 3$) as well as Zelikovsky's algorithm is 11/8 (Zelikovsky [20] has independently shown this result) and we save the factor of $\sqrt{|S|}$ in the running time. While Zelikovsky's approach cannot improve upon this ratio, we show that for *every* increase in the value of k the ratio achieved by our algorithm improves by more than $1/k^3$. For example, our algorithm achieves the ratio 97/72 (= 11/8 − 1/36) for $k = 4$. By using properties of the rectilinear metric, we show that the running time of our algorithm is much better in rectilinear metric than in the case of graphs. Table 2 shows the proven performance ratio of our heuristic (denoted as r) in rectilinear metric for some values of k (the performance ratio converges to a value close to 1.323).

This paper is organized as follows. Section 2 presents more background details about the problem and also the main theorems of this paper. Section 3 describes the basic algorithm as applicable to any given metric and proves a characterization of its performance ratio. Section 4 is devoted to our results in rectilinear metric. Conclusions and open problems are discussed in Section 5.

2. Background and Main Theorems

Let T be a Steiner tree of S. T may in general contain points other than the ones from S. We call points from S the *given points* and the additional points in T *Steiner points*. T is called a *full Steiner tree* if every given point is a leaf in T. When we split T at given points with degree more than one, we partition T into edge-disjoint full Steiner trees called *full components* of T [9].

DEFINITION 2.1. A Steiner tree T of S is k-restricted if every full component of T has at most k given points.

A 2-restricted Steiner minimal tree of S is simply a minimum spanning tree of S. Let r_k denote the least upper bound of the ratio of the cost of the k-restricted Steiner minimal tree to the cost of the unrestricted optimum. Observe that the values of r_k's are monotonically decreasing. It is well known that $r_2 \leq 2$ for any metric [18] and therefore computing a minimum spanning tree of S gives us a performance ratio of 2 for all metrics.

It is interesting to ask what is the complexity of computing k-restricted Steiner minimal trees for various metrics. For $k \leq 4$, one may easily verify that this restricted problem is also NP-hard for the metrics considered in this paper. Indeed, the NP-hardness proofs given in [7, 8] for computing the unrestricted Steiner minimal tree in the Euclidean and rectilinear metrics suffice to show the NP-hardness of the case when k is fixed for $k \geq 4$. In the case of graphs, one may reduce the bounded degree independent set problem (with degree at most 3) to this problem (with $k \geq 4$) using the transformation given by Hakimi [11]. However, the complexity of computing 3-restricted Steiner minimal trees is still unresolved.

It is also of interest to ask what are the values of r_k's, $k \geq 2$, for various metrics. While values of r_2 were known for the metrics we consider in this paper [13, 5, 18], no results were known for $k \geq 3$ prior to the work of Zelikovsky [19]. He showed that $r_3 \leq 5/3$ for all metrics. Moreover, he presented a new algorithm that constructs a 3-restricted Steiner tree as an approximate solution for the unrestricted optimum and showed that it achieved a performance ratio of at most

$$r_2 - \frac{r_2 - r_3}{2}, \tag{2.1}$$

which is 11/6. This algorithm runs in polynomial time in metrics for which a Steiner minimal tree for any three points can be found in polynomial time. Although one can construct metrics that do not satisfy this condition, the Steiner minimal tree can usually be found in polynomial time for all subsets of S of size k as it is the case for the rectilinear metric [12] and the metrics defined by graphs [16].

Zelikovsky's algorithm begins with a minimum spanning tree M of S as the initial solution. It then tries to improve M greedily by computing the improvement resulting from inserting Steiner minimal trees of subsets of S

of size 3 into M. The process of improvement discovered by Zelikovsky may be easily extended to subsets of size greater than 3, and is described below.

Let τ denote a subset of S and let t denote its Steiner minimal tree. Adding t to M creates $|\tau| - 1$ basic cycles in M. It is easy to see that one can now remove $|\tau| - 1$ edges from M and obtain a new tree that connects all the points in S. If the cost of this tree is less than the cost of M, then we have obtained an improved solution. In general, there may be several edge sets (of cardinality $|\tau| - 1$) that can be removed from M to obtain the improved solution. The edge set chosen for removal is the one with maximal total cost.

Du *et al.* [4] generalized Zelikovsky's greedy algorithm by considering all subsets of S of size bounded by k, $k \geq 3$. It was shown by extending Zelikovsky's proof ideas that the generalized algorithm has a performance ratio of

$$r_2 - \frac{r_2 - r_k}{k - 1}. \tag{2.2}$$

Note that (2.1) is a special case of (2.2). It was also shown in [4] that

$$r_{2^k} \leq 1 + \frac{1}{k} \tag{2.3}$$

for all metrics. Directly using (2.2) and (2.3) (along with the fact that the values of r_k's are monotonically decreasing) does not allow one to show that the performance ratio of the generalized algorithm is less than 11/6 for all $k > 3$.

We use a different improvement procedure in our algorithm. Although it uses the ideas of Zelikovsky, it is not greedy. When generalized to subsets of S of size bounded by a constant $k > 3$, we are able to show that the performance ratio of our algorithm is better than 11/6. We denote our generalized algorithm as $\mathscr{A}_k$ (where k denotes the largest size of subsets of S considered for improvement) and characterize its performance ratio in the following theorem.

THEOREM 2.1. *For $k \geq 3$, the performance ratio of Algorithm $\mathscr{A}_k$ is at most*

$$r_2 - \sum_{i=3}^{k} \frac{r_{i-1} - r_i}{i - 1}.$$

Note that when $k = 3$, the performance of Algorithm $\mathscr{A}_k$ is identical to that of Zelikovsky's algorithm. It is also clear from (2.3) that the performance ratio of $\mathscr{A}_k$ can always be improved by doubling the value of k.

When the metric is presented in the form of a graph, Algorithm $\mathscr{A}_k$ runs in time needed to compute all-pairs-shortest-paths plus $O(v^{k-2}n^{k-1} + n^{k+1/2})$, where v is the number of vertices in the graph and $n = |S|$.

In the case of the rectilinear metric, however, we show a much better ratio and time for the algorithm. First we present the following theorem.

THEOREM 2.2. *In the rectilinear metric*, $r_k \leq 1 + 1/(2k - 2)$ *for* $k \geq 2$.

The above theorem combined with Theorem 2.1 shows that the performance ratio of Algorithm $\mathscr{A}_k$ in the rectilinear metric improves with every increase of k while it has the performance of Zelikovsky's algorithm for $k = 3$. By using some properties of the rectilinear metric, we reduce the running time to $O(n^{\lfloor k/2 \rfloor + 3/2})$. For $k = 3$, our algorithm runs in time $O(n^{5/2})$ compared to $O(n^3)$ for Zelikovsky's algorithm.

3. ALGORITHM $\mathscr{A}_k$ AND ITS PERFORMANCE

In this section, we present Algorithm $\mathscr{A}_k$ and characterize its performance by proving Theorem 2.1.

An input to Algorithm $\mathscr{A}_k$ describes a metric space (V, d) and a finite set $S \subseteq V$. We begin this section with additional notation and concepts in Section 3.1. The algorithm is presented in Section 3.2. The proof of Theorem 2.1 is presented in Section 3.3. Implementation issues and the running time for graph-induced metrics is discussed in Section 3.4.

3.1. *Removal Set and Add Set*

For all graphs in this paper, we use *costs* (e) to denote the cost of edge e and *cost* (E) to denote the cost of edge set E (the sum of the costs of its edges).

For $\tau \subseteq S$, *Smt*(τ) denotes a Steiner minimal tree of τ in the input metric d, and *scost*(τ) denotes its cost.

As in the case of Zelikovsky's algorithm, our algorithm makes attempts to improve a spanning tree M of S by inserting *Smt*(τ) for subsets $\tau \subseteq S$. Such an insertion is followed by the removal of a set $R \subseteq M$ which has the property that $M' = M - R \cup Smt(\tau)$ is connected and cycle-free; we say that R is a *removal set* for τ in M. If *cost*$(R) >$ *scost*(τ), then M' is a Steiner tree of S with lesser cost than M. To maximize the improvement resulting from the insertion of *Smt*(τ), our algorithm chooses a removal set R with the maximum cost.

Let $g = cost(R) - scost(\tau) = cost(M) - cost(M')$. We call g the *gain* of τ in M; in other words, the improvement to M obtained by discarding R and inserting $Smt(\tau)$.

When our algorithm finds that gain is positive for a subset τ under consideration, it discards R from M and connects the components of $M - R$ with a new set A of edges called the *add set*. The edges of add set allow the algorithm to "mark" the preference to connect the points of τ with $Smt(\tau)$, without any commitment to use $Smt(\tau)$ in the final solution.

We begin with the following lemma, which gives a criterion for including an edge in R, and then describe the procedure *PrepareChange*, which computes the sets of edges R and A.

LEMMA 3.1. *Assume that $|\tau| > 1$. Let e be an edge of maximum cost such that both the connected components of $M - \{e\}$ contain a vertex of τ. Then there exists a maximum cost removal set for τ in M that contains e.*

Proof. Note that $R \subseteq M$ is a removal set for τ in M iff every connected component of $M - R$ contains exactly one vertex from τ (otherwise $M - R \cup Smt(\tau)$ either contains a cycle or is not connected).

Assume now that R has the maximum cost and $e \notin R$. It suffices to show that for some $f \in R$, $cost(f) \leq cost(e)$ and $R - \{f\} \cup \{e\}$ is also a removal set for τ in M.

Since $e \notin R, e$ belongs to one of the connected components of $M - R$. Let C denote the connected component of $M - R$ containing e, and let x denote the vertex of τ contained in C. Let C_1 (resp. C_2) denote the connected component of $C - \{e\}$ that contains x (resp. does not contain x).

Since e lies on the path between some two vertices of τ, it follows that for some $y \in \tau$, e belongs to the path P in M between x and y. Let f be the first edge from R on P as we move along P from x to y. Edge f must exist, otherwise $P \subseteq M - R$ and $y \in \tau$ is contained in C, a contradiction. Moreover, by the selection of e, $cost(f) \leq cost(e)$.

Now note that one endpoint of f must belong to C (in particular, to C_2), and the other endpoint must belong to another connected component of $M - R$, say D. It is easy to see that $M - R$ and $M - (R - \{e\} \cup \{f\})$ have the same connected components except for C and D which are replaced with C_1 and $C_2 \cup D$. It is clear by the definition of C_1 and C_2 that both C_1 and $C_2 \cup D$ contain exactly one vertex from τ. Hence $R - \{e\} \cup \{f\}$ is a removal set for τ in M. ■

We now describe the procedure *PrepareChange* which computes the sets of edges R and A, see Fig. 1 In computing R, the procedure makes use of the fact that the points of τ are connected by $Smt(\tau)$ in $M - R \cup Smt(\tau)$ and therefore there should be no connection between any two

```
procedure PrepareChange ($M, \tau, R, A$)
begin
    if $|\tau| = 1$ then begin
        $R \leftarrow \emptyset$;
        $A \leftarrow \emptyset$;
    end else begin
        Find an edge $e$ of maximum cost such that both the connected
          components of $M - \{e\}$ contain a vertex of $\tau$;
        Obtain the connected components $M_1$ and $M_2$ of $M - \{e\}$;
        Obtain $\tau_i$, the set of vertices of $\tau$ in $M_i$, $i = 1, 2$;
        Create an edge $f$ joining some $u \in \tau_1$ and some $v \in \tau_2$;
        $cost(f) \leftarrow cost(e)$;
        for $i \leftarrow 1$ to 2 do
            PrepareChange ($M_i, \tau_i, R_i, A_i$);
        $R \leftarrow R_1 \cup R_2 \cup \{e\}$;
        $A \leftarrow A_1 \cup A_2 \cup \{f\}$;
    end;
end;
```

FIG. 1. The *PrepareChange* procedure.

points of τ in $M - R$. The algorithm uses a simple greedy strategy to compute R as exemplified by Lemma 3.1. While the set R will be a maximum cost removal set for τ in M, the set A will be a spanning tree of τ.

Now we present the basic properties of the sets R and A computed by the *PrepareChange* procedure.

LEMMA 3.2

1. *R is a removal set for τ in M with the maximum cost.*
2. *A is a spanning tree of τ.*
3. *$cost(A) = cost(R)$.*
4. *If $f = (u, v) \in A$, then $cost(f)$ is equal to the largest cost of an edge on the path in M between u and v.*

Proof. Claim 1. By induction on $|\tau|$. If $|\tau| = 1$, then the only removal set for τ in M is the empty set and $R = \emptyset$ from the code.

Assume that $|\tau| > 1$. By Lemma 3.1, e is correctly included in R. The removal of the remaining edges of R should break M_1 and M_2 into connected components so that each of them contains exactly one element from τ. By the inductive hypothesis, these edges will be properly found by the recursive calls with input parameters M_1, τ_1 and M_2, τ_2.

Claims 2, 3, and 4 follow directly from the code. ■

We obtain the following corollary from claims (1) and (2) of Lemma 3.2.

COROLLARY 3.3. *The edge set $M - R \cup A$ is a spanning tree of S.*

3.2. *The Algorithm*

The computation is divided into two phases, namely, the *evaluation phase* and the *construction phase*. Initially we execute the evaluation phase (see Fig. 2).

This phase maintains a spanning tree M of S which is initialized to a minimum spanning tree of S in metric d, and stacks σ_j, $3 \leq j \leq k$, which are initially empty (lines 1–2).

The phase processes every subset τ of S that has size between 3 and k, progressing from smaller to larger sets (lines 3–4). For every such τ, the removal set R and the add set A are computed by calling the procedure *PrepareChange* (line 5). Then *gain* is computed for the subset τ (line 6). If *gain* is not positive, then the sets R and A are discarded; τ is never considered again by the algorithm. Otherwise, the cost of every edge in A is decreased by *gain* and M is modified by actually removing R and adding A; moreover, the tuple $[\tau, R, A]$ is pushed onto the stack $\sigma_{|\tau|}$(lines 8–10). By Corollary 3.3, M remains a spanning tree of S after this modification.

Note that the edges added to M in line 9 (set A) are artificial edges. Rather than being proposed for the final solution, they merely mark our tentative preference for incorporating $Smt(\tau)$ in the final solution. We have such a preference because we have registered a positive *gain* in line 7. However, we delay the decision concerning the inclusion of $Smt(\tau)$ to the construction phase. Such a decision is taken only after we have considered all possible alternatives. The information stored on the stacks

```
1  $M \leftarrow$ a minimum spanning tree of $S$ in the metric $d$;
2  for $j \leftarrow 3$ to $k$ do $\sigma_j \leftarrow \emptyset$;

3  for $j \leftarrow 3$ to $k$ do
4     for each $j$-element subset $\tau \subseteq S$ do begin
5        PrepareChange $(M, \tau, R, A)$;
6        $gain \leftarrow cost(R) - scost(\tau)$;
7        if $gain > 0$ then begin
8           Decrease the cost of each edge of $A$ by $gain$;
9           $M \leftarrow M - R \cup A$;
10          $push(\sigma_j, [\tau, R, A])$;
11       end;
12    end;
```

FIG. 2. The evaluation phase.

```
1  $N \leftarrow M$;
2  for $j \leftarrow k$ downto 3 do
3     while $\sigma_j \neq \emptyset$ do begin
4        $[\tau, R, A] \leftarrow pop(\sigma_j)$;
5        $M \leftarrow M - A \cup R$;
6        if $A \subseteq N$ then
7           $N \leftarrow N - A \cup Smt(\tau)$
8        else
9           for every edge $e \in A \cap N$ do begin
10             Find $f \in M$ of minimal cost such that $N - \{e\} \cup \{f\}$
                connects $S$;
11             $N \leftarrow N - \{e\} \cup \{f\}$;
12          end;
13    end;

14 return($N$);
```

FIG. 3. The construction phase.

allows us to review all the proposed changes in appropriate order and either "perform" them (replace A with $Smt(\tau)$) or "undo" them (replace A with other edges).

After the evaluation phase is completed, we execute the construction phase. The purpose of this phase (see Fig. 3) is to use the entries stored on the stacks to remove all the artificial edges from M and construct the output which is a Steiner tree of S in the metric d. In particular, this phase will ensure that no edge that belongs to an add set will be present in the final solution. We now explain the actions taken during this phase and also show the correctness of the algorithm.

In addition to the spanning tree M obtained at the end of the evaluation phase, this phase also maintains a Steiner tree N of S. Initially N equals M and at the end of this phase N forms our solution. In this phase, M is simply an auxiliary structure which is used to build the solution tree in N.

The phase considers the entries stored in the stacks in the reverse order in which they were pushed onto them in the evaluation phase. For this purpose, entries of the form $[\tau, R, A]$ are popped from the stacks (line 4), beginning with the stack σ_k and ending with the stack σ_3 (lines 2, 3). Then M and N are suitably modified (lines 5–12) so that, after the modifications, the add set A is not present in both M and N, and M (resp. N) remains a spanning tree (resp. Steiner tree) of S.

We first consider the changes to M and then examine the changes to N. M is set to $M - A \cup R$ in line 5, which is just the reverse of the transformation to M in line 9 of the evaluation phase. Note that the order in which the stacks are considered in this phase is the reverse of the order in which entries were pushed onto the stacks during the evaluation phase. Thus it is clear that in the construction phase M assumes the same values as in the evaluation phase, but in reverse order. In particular, the final value of M here is the same as the initial value of M in the evaluation phase.

We now focus on the changes to N in lines 6–12. There are two cases to consider.

Case 1. $A \subseteq N$

A is replaced with $Smt(\tau)$ in N (line 7). Note that A is the add set for τ and hence is a spanning tree of τ (by Lemma 3.2(2)). Thus after this change N does not contain A and remains a Steiner tree of S.

Case 2. $A \not\subseteq N$

The edges of A present in N are removed one by one and suitable replacement edges from M are added so that N still connects S (lines 9–12). In particular, let $e \in A \cap N$ be removed from N in one iteration of the **for**-loop. Let τ_1 and τ_2 denote the vertex sets of the two connected components of $N - \{e\}$. Since M is a spanning tree of S, there indeed exist one or more edges in M that have one endpoint in τ_1 and the other endpoint in τ_2. The algorithm chooses such an edge f with minimum cost as the replacement edge for e in N. Clearly, $N - \{e\} \cup \{f\}$ connects S.

Also note that A has already been removed from M (in line 5) and hence the replacement edge does not belong to A. Thus after this change N does not contain A and remains a Steiner tree of S.

Since the algorithm considers all the entries stored on the stacks and ensures that the edges of add set present in each entry is removed from N, it follows that at the end of the outer **for**-loop N does not contain any artificial edge. Thus the output returned by the algorithm (line 14) is a Steiner tree of S in metric d.

3.3. *Proof of Theorem* 2.1

First we consider the evolution of the variable M in the evaluation phase.

In an iteration of the evaluation phase, the only change to M occurs in line 9 ($M \leftarrow M - R \cup A$) when *gain* is found to be greater than 0 for the subset τ under consideration; moreover, the tuple $[\tau, R, A]$ is pushed onto a stack.

We have the following lemma.

LEMMA 3.4. *Consider the values of the algorithm variables when $\tau \subseteq S$ is being processed by the evaluation phase. Let g denote the value of gain. If* *the entry* $[\tau, R, A]$ *is pushed onto stack* σ_j, *then*

1. $g = cost(R) - scost(\tau)$,
2. $g > 0$, *and*
3. $cost(R) - cost(A) = (j-1)g$.

Proof. Claims 1 and 2 follow directly from the code.

Claim 3. Observe that $|R| = |A| = |\tau| - 1 = j - 1$. The claim now follows from Lemma 3.2(3) and line 8 of the evaluation phase. ∎

Thus whenever M is modified in the evaluation phase the cost of M decreases by the positive quantity $(j-1)g$ (which equals cost(R) − cost(A)).

We will use g_τ to denote g from Lemma 3.4, and $Q(j)$ to denote the set of τ's stored in σ_j. Note that if $\tau \in Q(j)$, then $|\tau| = j$.

Let M_2 denote the initial value of M and, for $3 \le s \le k$, let M_s denote the value of M following the execution of lines 4–11 with $j = s$. Let $m_j = cost(M_j)$, $j = 2, \dots, k$. As an immediate consequence of Lemma 3.4, we obtain

LEMMA 3.5. *For* $3 \le j \le k$,

$$m_{j-1} - m_j = (j-1) \sum_{\tau \in Q(j)} g_\tau.$$

Now we will focus on the evolution of our tentative solution N in the construction phase. In particular, we will consider the change in the cost of N during one iteration of the **while**-loop.

LEMMA 3.6. *Consider the execution of lines* 4–12 *of the construction phase that processes an entry* $[\tau, R, A]$ *of* σ_j. *As a result of executing these lines the cost of N increases by at most* $(j-2)g_\tau$.

Proof. Note that $|\tau| = j$. We have two cases to consider.

Case 1. $A \subseteq N$. A is replaced with $Smt(\tau)$ in N. From claims 1 and 3 of Lemma 3.4, it follows that the cost of N increases by $(j-2)g_\tau$.

Case 2. $A \not\subseteq N$. Let M_τ be the value of M resulting from the assignment in line 5. It follows from our discussion in Section 3.2 that, when the sets A and R were computed in the evaluation phase by *PrepareChange* (M, τ, R, A), the value of M was equal to M_τ.

Edges from $A \cap N$ are removed from N and replaced with edges from M_τ in a **for**-loop (lines 9–12). Consider the change in the cost of N during one iteration of this loop when an edge $e = (u, v) \in A \cap N$ is replaced with $f \in M_\tau$. Observe that the cost of every edge in the path between u and v in M_τ exceeds the cost of e by at most g_τ (from Lemma 3.2(4) and line 8 of the evaluation phase). Clearly, one of these edges, say f, has the property that $N - \{e\} \cup \{f\}$ connects S. Therefore $cost(f) - cost(e) \leq g_\tau$, and hence the cost of N increases by at most g_τ.

Note that this loop is executed $|A \cap N|$ times, which is at most $|\tau| - 2 = j - 2$, since $A \not\subseteq N$. Therefore the total increase in the cost of N is at most $(j - 2)g_\tau$. ■

Let N_k denote the initial value of N and, for $k > s \geq 2$, let N_s denote the value of N following the execution of lines 3–13 with $j = S + 1$. Let $n_j = cost(N_j)$, $j = 2, \dots, k$.

Note that $N_k = M_k$ and N_2 is the output of the algorithm. From Lemma 3.6, we have the following.

LEMMA 3.7. *For $k \geq j \geq 3$,*

$$n_{j-1} - n_j \leq (j-2) \sum_{\tau \in Q(j)} g_\tau.$$

From Lemma 3.7, the cost of the final solution is given by

$$\begin{aligned} n_2 &\leq n_k + \sum_{j=3}^{k} (j-2) \sum_{\tau \in Q(j)} g_\tau \\ &= m_k + \sum_{j=3}^{k} \frac{j-2}{j-1}(m_{j-1} - m_j) \\ &\qquad\qquad \text{(by Lemma 3.5 and the fact that } N_k = M_k) \\ &= m_2 - \sum_{j=3}^{k} \frac{m_{j-1} - m_j}{j-1} \qquad \text{(by rearranging terms)}. \qquad (*) \end{aligned}$$

Now let T_j, $2 \leq j \leq k$, denote an optimal j-restricted Steiner tree of S in metric d, and let t_j denote its cost. Note that $m_2 = t_2$ since an optimal 2-restricted Steiner tree of S in any metric is simply a minimum spanning tree of S in the metric, and M_2, by definition, is a minimum spanning tree of S in the metric d. Comparing $(*)$ with Theorem 2.1, it is now clear that in order to prove the theorem it suffices to show that

$$m_j \leq t_j, \qquad 3 \leq j \leq k. \qquad (**)$$

The rest of this section is devoted to proving $(**)$.

First we present additional notation.

Let $R(M, \tau)$ denote the removal set for $\tau \subseteq S$ in M as computed by the procedure *PrepareChange*, and let $g(M, \tau) = cost(R(M, \tau)) - scost(\tau)$, i.e., $g(M, \tau)$ is the *gain* obtained by inserting $Smt(\tau)$ into M.

We begin with the following corollary of an *exchange theorem for bases of matroids* (see [10, 3]).

COROLLARY 3.8. *Given two spanning trees M and M' of S and a partition $M = E \cup F$, there is a partition $M' = E' \cup F'$ such that $E \cup F'$ and $E' \cup F$ are both spanning trees of S.*

We use the above corollary to show the following lemma which gives a sufficient condition for inequalities of the same kind as $(**)$.

LEMMA 3.9. *Let T be a Steiner tree of S with full components $Smt(\tau_1), \dots, Smt(\tau_p)$, and let M be a spanning tree of S. Assume that $g(M, \tau_i) \le 0$ for $i = 1, \dots, p$. Then*

$$cost(M) \le cost(T).$$

Proof. We will prove the claim under the somewhat generalized assumption that $T = \bigcup_{i=1}^{p} Smt(\tau_i) \cup A$, where $A \subseteq M$, and $g(M, \tau_i) \le 0$ for $i = 1, \dots, p$. If $p = 0$, then $T = M$ and the claim is trivial. Assume inductively that it is true for $p - 1$.

Let A_i be an arbitrary spanning tree for τ_i, and let $M' = \bigcup_{i=1}^{p} A_i \cup A$. Partition M' into E' and F' where $E' = A_p$. Let E and F be a partition of M that exists by Corollary 3.8. Then $F' \cup E = M' - A_p \cup E$ is a spanning tree of S. By constructing of M' from T, it follows that $T' = T - Smt(\tau_p) \cup E$ is a Steiner tree of S. Obviously, $T' = U_{i=1}^{p-1} Smt(\tau_i) \cup (A \cup E)$ and hence T' satisfies the inductive hypothesis. Therefore,

$$\begin{aligned} cost(M) &\le cost(T') \\ &= cost(T) - scost(\tau_p) + cost(E) \\ &= cost(T) - \big(cost\big(R(M, \tau_p)\big) - g(M, \tau_p)\big) + cost(E) \\ &\le cost(T) - cost\big(R(M, \tau_p)\big) + cost(E) \\ &\qquad \big(\text{since } g(M, \tau_p) \le 0\big). \end{aligned}$$

Now it suffices to show that cost$(R(M, \tau_p)) \ge cost(E)$.

From the way we constructed F, it follows that $F \cup E' = M - E \cup A_p$ is a spanning tree of S, hence $M - E \cup Smt(\tau_p)$ is a Steiner tree of S, and therefore E is a removal set for τ_p in M. By Lemma 3.2(1), $R(M, \tau_p)$ is a removal set for τ_p with maximum cost, hence $cost(R(M, \tau_p)) \ge cost(E)$. ■

Note that since T_j is a j-restricted Steiner tree of S, every full component of T_j has at most j points of S. From Lemma 3.9, it is clear that in order to prove $(**)$ it is sufficient to show the following:

LEMMA 3.10. *For every subset τ of S of size at most j, $g(M_j, \tau) \leq 0$, $3 \leq j \leq k$.*

Proof. Observe that for a subset τ of S such that $|\tau| = 2$, $g(M, \tau) \leq 0$ when M is a minimum spanning tree of S in metric d. Therefore, for all such τ's, $g(M, \tau) \leq 0$ before the execution of the nested loops (lines 3–12) begins in the evaluation phase.

We now use two facts that are proved in the remaining part of this section. First, for every S-element subset τ of S, $3 \leq s \leq k$, there exists a moment during the evaluation phase when $j = s$ and $g(M, \tau) \leq 0$ (Lemma 3.11 below). Second, no modification of M performed during the evaluation phase may increase the value of $g(M, \tau)$ for all $\tau \subseteq S$ (Lemma 3.12 below).

Note that the evaluation phase considers subsets of S in increasing order of their size. Thus it follows that when the outer **for**-loop of the evaluation phase is completed with $j = s$, M equals M_s and $g(M, \tau) \leq 0$ for every subset τ of S such that $|\tau| \leq s$. ■

LEMMA 3.11. *For every s-element subset τ of S, $3 \leq s \leq k$, there exists a moment during the evaluation phase when $j = s$ and $g(M, \tau) \leq 0$.*

Proof. Assume that $|\tau| = S$ and consider the execution of the lines 5–11 when τ is being processed in the evaluation phase. If in line 7 the condition "gain > 0" is false, the lemma clearly holds, since at this moment $g(M, \tau) =$ gain and $j = s$. Therefore assume that gain > 0. Then M is modified to $M' = M - R \cup A$ in line 9. It suffices to show that $g(M', \tau) \leq 0$.

Since A is a spanning tree of τ, A is the only removal set for τ in M'. Therefore $cost(R(M', \tau)) = cost(A)$ and hence

$$g(M', \tau) = cost(A) - scost(\tau).$$

Since $|\tau| = s$, Lemma 3.4 (1) and (3) and the above equation imply that

$$g(M', \tau) = (2 - s)\,gain \leq -gain < 0. \;\blacksquare$$

At this point, to prove $(**)$ (and consequently Theorem 2.1) it remains to show the following lemma.

LEMMA 3.12. *For any $\tau \subseteq S$, the quantity $g(M, \tau)$ never increases during the evaluation phase.*

Proof. Recall that the only change to M in the evaluation phase occurs in line 9 ($M \leftarrow M - R \cup A$). We examine this modification more closely. First we define a *basic exchange* as follows:

Let *M be a spanning tree of S and, for some edge f, let e denote an edge of maximum cost in the basic cycle created by f in M. The process of modifying M to $M' = M - \{e\} \cup \{f\}$ is called a basic exchange.*

By an obvious induction on $|R|$ $(= |A|)$, one can show that the tree $M - R \cup A$ may be obtained from M by a sequence of basic exchanges. The lemma follows trivially from Lemma 3.14 (see below), which shows that $g(M, \tau) \geq g(M', \tau)$ for any $\tau \subseteq S$. ■

Now it only remains to show Lemma 3.14.

Note that $g(M, \tau)$ depends on the cost of $R(M, \tau)$. Let $rcost(M, \tau)$ be a shorthand for cost($R(M, \tau)$), and let $lcost(M, u, v)$ denote the largest cost of an edge on the path in M between u and v.

We use the following characterization of $rcost(M, \tau)$ to prove Lemma 3.14.

LEMMA 3.13. *If $\tau \subseteq S$, $\tau = \beta \cup \{v\}$, $v \notin \beta$ and $\beta \neq \emptyset$, then*

$$rcost(M, \tau) = rcost(M, \beta) + \min_{u \in \beta} lcost(M, u, v).$$

Proof. We prove it by induction on $|\beta|$. For $\beta = \emptyset$, the claim is vacuous. Otherwise, we need to inspect how $R(M, \tau)$ is computed by the procedure *PrepareChange* (see Fig. 1). First we find a maximal cost edge e on the paths in M between vertices of τ. Then we obtain the two connected components M_1 and M_2 of $M - \{e\}$, and form τ_i, the set of vertices of τ in M_i, $i = 1, 2$. Finally, we set

$$R(M, \tau) = R(M_1, \tau_1) \cup R(M_2, \tau) \cup \{e\}.$$

Note that all paths in M between vertices of τ_i are contained in M_i; thus we can write the recurrence

$$rcost(M, \tau) = rcost(M, \tau_1) + rcost(M, \tau_2) + cost(e).$$

Now we consider two cases.

Case 1. $\tau_1 = \{v\}$. Here $\beta = \tau_2$. Since $|\tau_1| = 1$, $rcost(M, \tau_1) = 0$. Moreover, by the choice of e, $\min_{u \in \beta} lcost(M, v, u) = \min_{u \in \beta} cost(e) = cost(e)$. The lemma now follows from the recurrence equation for $rcost$ (M, τ) given above.

Case 2. $\tau_1 = \tau_3 \cup \{v\}$, $v \notin \tau_3$, $\tau_3 \neq \varnothing$. Here $\beta = \tau_3 \cup \tau_2$.

By the choice of e, $lcost(M, v, u) = cost(e)$ for $u \in \tau_2$, and $lcost(M, v, u) \leq cost(e)$ for $u \in \tau_3$. Therefore,

$$\min_{u \in \beta} lcost(M, v, u) = \min_{u \in \tau_3} lcost(M, v, u).$$

Therefore,

$$\begin{aligned} rcost(M, \tau) &= rcost(M, \tau_3 \cup \{u\}) + rcost(M, \tau_2) + cost(e) \\ &= \left(rcost(M, \tau_3) + \min_{u \in \beta} lcost(M, v, u)\right) \\ &\qquad + rcost(M, \tau_2) + cost(e) \end{aligned}$$

by the inductive hypothesis.

On the other hand, $\beta = \tau_3 \cup \tau_2$ and e is a maximum cost edge on the paths between vertices of β. Consequently,

$$rcost(M, \beta) = rcost(M, \tau_3) + rcost(M, \tau_2) + cost(e).$$

The last two equations yield the lemma. ■

LEMMA 3.14. *Assume that e and f are edges and let* $M' = M - \{e\} \cup \{f\}$ *be the tree obtained from M via a basic exchange. Then for any* $\tau \subseteq S$, $g(M, \tau) \geq g(M', \tau)$.

Proof. Consider the definition of gain, namely,

$$g(M, \tau) = rcost(M, \tau) - scost(\tau).$$

Note that $scost(\tau)$ is fixed for a given τ. Thus it suffices to show that $rcost(M, \tau) \geq rcost(M', \tau)$.

We show this by induction on τ. For $|\tau| = 1$, this is equivalent to $0 \leq 0$.

Assume that $|\tau| \geq 2$. For $\tau = \beta \cup \{v\}$ where $v \notin \beta$, by Lemma 3.13 this is equivalent to

$$\begin{aligned} rcost(M, \beta) + \min_{u \in \beta} lcost(M, u, v) \geq rcost(M', \beta) \\ + \min_{u \in \beta} lcost(M', u, v). \end{aligned}$$

By the inductive hypothesis $rcost(M, \beta) \geq rcost(M', \beta)$. Now we will show that for any $u, v \in S$, $lcost(M, u, v) \geq lcost(M', u, v)$ and the lemma clearly follows.

Assume to the contrary: P is the path in M from U to v, g is a maximal cost edge on P, Q is the path in M' from u to v, h is a maximal cost edge on Q, and $cost(h) > (g)$. Since $h \notin P$, h belongs to $P \oplus Q$. The latter set is then a nonempty cycle; hence this is the unique cycle of $M \cup \{f\}$. Since e has the largest cost on this cycle, $cost(h) \leq cost(e)$ and $e \notin P$. However, $e \notin P$ implies that $P \subseteq M'$, therefore $P = Q$ and $cost(h) = cost(g)$, a contradiction.

This completes the proof of Theorem 2.1. ■

3.4. *Running Time in Graph-Induced Metrics*

In this section we discuss the implementation of the algorithm for an arbitrary metric space. We assume that the metric is represented as a graph which forms part of the input. For a specific metric, one may obtain a better efficiency; the next section studies an important example.

Observe that the algorithm needs to generate Steiner minimal trees for all subsets of S of size at most k. Using the method of Zelikovsky [19], one can generate Steiner minimal trees for all triples in a graph in $O(\alpha + vn^2)$ steps where α is the time complexity of the all-pairs-shortest-paths problem, v is the number of vertices in the graph, and $n = |S|$. This method can be easily generalized to k-tuples and requires $O(\alpha + v^{k-2}n^{k-1})$ steps.

The evaluation phase needs to process $O(n^k)$ subsets of S (lines 3–4). For each subset τ, the only time-consuming task is the call of *Prepare Change* procedure (line 5); all the other steps are easily carried out in constant time. We use the data structure of Sleator and Tarjan [17] to store M in the evaluation phase. Using their operations *link*, *cut*, *evert*, *parent*, *root*, *cost* and *maxcost*, one can execute such a call (and the other steps as well) in $O(|\tau|^2 \log n)$ time. Since $|\tau| \leq k$, the running time of the evaluation phase is $O(n^k \log n)$.

In the construction phase, each iteration of the **while**-loop performs updates to both M and N. All steps inside the loop may be performed in constant time except for the one in line 10 which will be our focus now. It is easy to implement this step in $O(n)$ steps: use a tree search to find the connected components N_1 and N_2 of $N - \{e\}$, and then find the least cost edge of M that straddles N_1 and N_2. We now show a more efficient approach.

Let $Smt_\cup$ denote the current value of the union of $Smt(\tau)$'s inserted into the solution tree in line 7 of the construction phase, and let $N' = N - Smt_\cup$.

Note that in line 10 we do not select edge f in a unique manner. As a result, the construction phase may compute N (and implicitly N') in many possible ways. However, one can show that all possible choices of N' can be characterized by the following invariant:

N' is the least cost acyclic subset of M such that $N' \cup Smt_\cup$ is connected.

We now prove this claim informally. After M is changed in line 5 into $M - A \cup R$, the invariant is restored in two possible ways. If $A \subseteq N'$, then we augment $Smt_{\cup}$ by $Smt(\tau)$ and remove A from N'. This restores the invariant, since A is a spanning tree of τ (at this point, we do not need to consider edges of R for possible inclusion in N', since R is the largest cost removal set for τ in the modified M, i.e., in $M - A \cup R$, and $Smt(\tau) \subseteq Smt_{\cup}$). If $A \not\subseteq N'$, we restore the invariant in a greedy manner while removing elements of A from N' one by one.

We will show how to update N' in such a way that it satisfies the above invariant using a data structure to maintain a minimum spanning tree in a dynamic graph.

Let λ be smaller than any cost of an edge introduced in the evaluation phase. Let $Low(\tau)$ be a spanning tree of τ such that every edge in $Low(\tau)$ has cost λ, and let $Low_{\cup}$ be the union of $Low(\tau)$'s corresponding to $Smt(\tau)$'s included in $Smt_{\cup}$. It is clear that the above invariant is equivalent to: N' is the least cost acyclic subset of M such that $N' \cup Low_{\cup}$ is connected. Since $Low_{\cup}$ is contained in the minimum spanning tree of the graph $(S, M \cup Low_{\cup})$, this is equivalent to

N' is a subset of M such that $N' \cup Low_{\cup}$ is a minimum spanning tree of $(S, M \cup Low_{\cup})$.

Consequently, to compute N' it suffices to compute the minimum spanning tree of $M \cup Low_{\cup}$. Note that $M \cup Low_{\cup}$ undergoes only a limited number of changes in a single execution of lines 5–12; in line 5, we change M by $j - 1$ edge insertions and $j - 1$ edge removals: in line 7 we change $Low_{\cup}$ by $j - 1$ edge insertions. Since $|M \cup Low_{\cup}| = O(n)$, we can apply the data structure of Frederickson [6] to update the minimum spanning tree of $M \cup Low_{\cup}$ in time $O(kn^{1/2}) = O(n^{1/2})$.

Since the number of times the **while**-loop is executed is $O(n^k)$, the running time of the construction phase is $O(n^{k+(1/2)})$.

Thus the total running time of the algorithm is $O(\alpha + v^{k-2}n^{k-1} + n^{k+(1/2)})$.

4. Approximations in Rectilinear Metric

4.1. *Structure of Rectilinear Steiner Minimal Tree*

Let $S = \{B_0, \ldots, B_m\}$ be a set of points in the plane, and let $T = Smt(S)$ in rectilinear metric. Assume that T is a full Steiner tree. Then we may assume that T is of one of the two forms given in Fig. 4 see Hwang [13]).

In the figure, $A_1, \ldots, A_m$ are the Steiner points of T; observe that they form a simple path in T. This path is called the *Steiner chain* of T. In Fig. 4a, the Steiner chain is a straight line, whereas in Fig. 4b, all Steiner points *except* A_m lie on a straight line (m is always odd in this case). Note

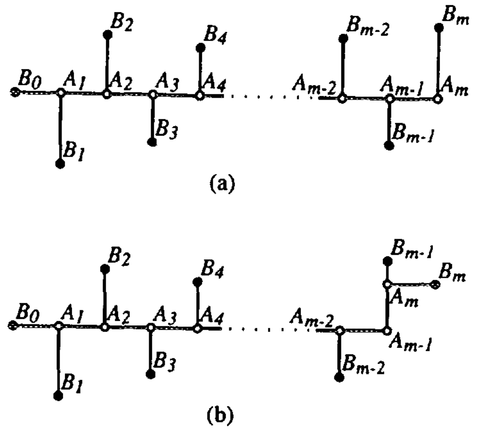


FIG. 4. Forms of T.

how the line segments $A_1B_1, A_2B_2, \ldots$ alternate above and below the Steiner chain.

4.2. *The Value of r_k in Rectilinear Metric*

We will prove Theorem 2.2, which characterizes the value of r_k in rectilinear metric. Note that it suffices to show that given a Steiner tree T of S, there exists a k-restricted Steiner tree of S with cost not exceeding $(1 + 1/(2k - 2))cost(T)$. We will assume that T is a full Steiner tree (otherwise one may split T into smaller full Steiner trees and apply the theorem inductively). Then we may assume w.l.o.g that T is of one of the two forms given in Fig. 4.

Let $A_0 = B_0$, $y_i = |A_{i-1}A_i|$, $1 \le i \le m$, and $x_i = |A_iB_i|$, $0 \le i \le m$, in the figure. For simplicity, we will assume that $x_i \ne x_j$ for $i \ne j$.

First we define the following transformation on T, which we call *doubling at B_j*: add a copy of the edge A_jB_j to T and split T at A_j as shown in Fig. 5

Note that the doubling operation at B_j increases the number of full components in the tree by one and increases the cost of the tree by x_j.

Now we state and prove Lemmas 4.1 and 4.2.

LEMMA 4.1. *Assume that the Steiner chain of T is a straight line. Then there exist* 3*-restricted Steiner trees of S, T_1, T_2, T_3, and T_4, such that the total cost of the* four *trees is at most* five *times the cost of T.*

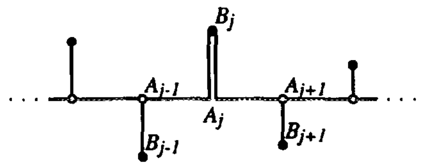


FIG. 5. Doubling at B_j.

Proof. It suffices to show that T contains disjoint sets of edges, E_1, E_2, E_3, and E_4 such that

$$cost(T_i) \leq cost(T) + \Sigma_{e \in E_i} cost(e), \qquad 1 \leq i \leq 4.$$

Now we describe the construction of the four trees. It will be clear from their construction that they satisfy the requirement specified above. For the sake of simplicity, we will assume that $x_m = 0$. Our proof may be easily extended to the case when $x_m > 0$.

Construction of T_1 and T_2. We call B_{2j} an *upper local maximum* if either $x_{2j} > x_{2j-2}, x_{2j+2}$, or $2j = m - 1$ and $x_{m-1} > x_{m-3}$. We obtain T_1' from T by performing the doubling operation at all upper local maxima of T.

It is easy to see that the full components of T_1' of size more than 3 can only have their upper local maxima on (either one or both) the ends of its Steiner chain. We obtain T_1 from T_1' by transforming each full component into a union of full components of size at most 3 as shown in Fig. 6, for example. Note that such transformations do not increase the cost. Thus T_1 is a 3-restricted Steiner tree of S and the cost of T_1 exceeds the cost of T by at most $\Sigma|A_{2j}B_{2j}|$, where B_{2j} is an upper local maximum of T.

The construction of T_2 is similar to that of T_1; double at the *lower local maxima* (which are defined similar to upper local maxima) and transform the resulting full components into full components of size at most 3.

Construction of T_3 and T_4. We obtain T_3' from T by doubling at each B_{2j} which is not an upper local maximum. The cost of T_3' exceeds the cost

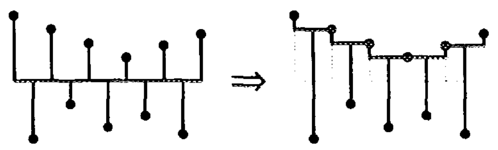


FIG. 6. $T_1' \Rightarrow T_1$.

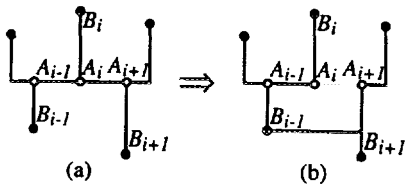


FIG. 7. $T_3' \Rightarrow T_3$.

of T by Σx_{2j} such that B_{2j} is not an upper local maximum. It is easy to see that the full components of T_3' of size more than 3 are only of the form given in Fig. 7a. Observe that B_i is an upper local maximum. We transform each such component of T_3' into the form shown in Fig. 7b (note that in the figure $x_{i-1} < x_{i+1}$, the case when $x_{i-1} > x_{i+1}$ is similar). The above transformation increases the cost of the tree by the length of edge A_iA_j such that $|i-j| = 1$, and B_i is an upper local maximum and B_j is not a lower local maximum. T_3 is the tree that is obtained as a result of these transformations. Clearly, T_3 is 3-restricted.

The construction of T_4 is similar to that of T_3; double at every nonlower local maximum and transform the resulting full components into full components of size at most 3. ■

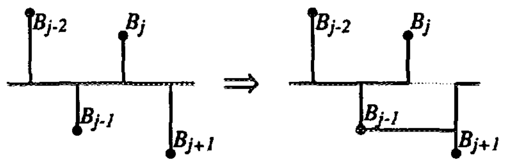
LEMMA 4.2. *Assume that the Steiner chain of T is a straight line. Then, for $k > 3$, there exist k-restricted Steiner trees of S, T_i, T_i', $0 \leq i \leq k-2$, such that the total cost of the $2k-2$ trees is at most $(2k-1)$ times the cost of T.*

Proof. Construction of $T_i, 0 \leq i \leq k-2$. We obtain T_i from T by doubling at each B_j such that $j = i \bmod (k-1)$, $j \neq 0$ and $j \neq m$. It is clear that T_i is k-restricted, and $cost(T_i) \leq cost(T) + \Sigma_j x_j$, where j is as defined above. It immediately follows that $\Sigma_{i=0}^{k-2} cost(T_i) \leq (k-1)cost(T) + \Sigma_{l=1}^{m-1} x_l$.

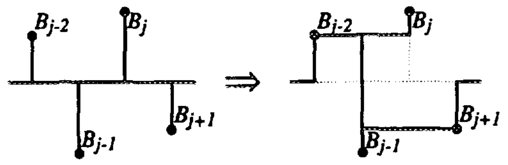
Construction of T_i', $0 \leq i \leq k-2$. We perform transformations on T as shown in Fig. 8 at each B_j such that $j = i \bmod (k-1)$, $j \neq 0$ and $j \neq m$. It is easily verified that the resulting Steiner tree is k-restricted. Note that each of the above transformations increases the cost of T by y_j. Therefore, it follows that $\Sigma_{i=0}^{k-2} cost(T_i') \leq (k-1)cost(T) + \Sigma_{l=1}^{m-1} y_j$.

Note that the cost of T is given by $\Sigma_{l=1}^{m}(x_l + y_l)$. Thus it is clear that the total cost of the trees $T_i, T_i', 0 \leq i \leq k-1$, is at most $(2k-1)$ times the cost of T. ■

Case 1. $x_{j-1} < x_{j+1}$



Case 2. $x_{j-1} > x_{j+1}$ *and* $x_{j-2} < x_j$



Case 3. $x_{j-1} > x_{j+1}$ *and* $x_{j-2} > x_j$

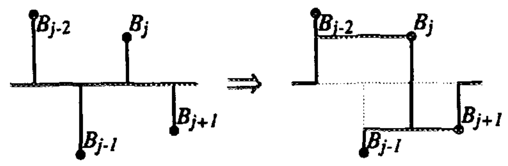


FIG. 8. $T \Rightarrow T_i'$.

Now we prove Theorem 2.2.

Proof. Assume first that the Steiner chain of T is a straight line. For $k = 3$, Lemma 4.1 implies that there exists a 3-restricted Steiner tree of S that has cost at most $5/4$ times the cost of T. For $k > 3$, Lemma 4.2 shows that there exists a k-restricted Steiner tree of S that has cost at most $(2k - 1)/(2k - 2)$ times the cost of T. Thus the theorem clearly holds in this case.

Suppose the Steiner chain of T is not a straight line (Fig. 4b). Obtain tree T' by translating the edge $A_m B_m$ downward so that A_m and A_{m-1} overlap and the points $B_0, A_1, \ldots, A_m, B_m$ lie on a straight line. Let B_m' be the new position of B_m. The Steiner chain of T' is a straight line and therefore one may obtain a k-restricted Steiner tree T'' of $S - \{B_m\} \cup \{B_m'\}$ with the desired cost. For each construction of T'' as specified in the proof, it is clear that the cost of T'' does not increase when B_m' is replaced with B_m. ■

4.3. *Efficiency in the Rectilinear Metric*

Observe that the evaluation phase of Algorithm $\mathcal{A}_k$ examines subsets of S of size at most k, which ordinarily necessitates considering $\Omega(n^k)$ such sets (recall that $n = |S|$). However, one can show that in the rectilinear metric the algorithm may restrict itself to consider *reasonable* subsets only, where the notion of reasonableness is derived from properties of rectilinear Steiner minimal trees [13]. Importantly, all reasonable subsets of size at most k can be generated in $O(n^{\lfloor k/2 \rfloor + 1})$ time (Zelikovsky [20] showed this result for $k = 3$). Since Steiner minimal trees for any subset of S of size at most k can be computed in constant time in rectilinear metric, it follows from our discussion in Section 3.4 that Algorithm $\mathcal{A}_k$ runs in time $O(n^{\lfloor k/2 \rfloor + 3/2})$ in the rectilinear metric. Now we describe properties of rectilinear Steiner minimal trees that allow us to achieve this improvement.

We will assume for reasons of simplicity that no two points in S have the same x or y coordinate. We say that a pair $\{s_1, s_2\} \subseteq S$ is *legal* if no other point in S is enclosed in the smallest oriented rectangle containing s_1 and s_2. We also say that s_2 is a legal NE (north-east) neighbor of s_1 if $\{s_1, s_2\}$ is a legal pair and both coordinates of s_2 are larger than the respective coordinates of s_1. Similarly we define legal SW, SE and NW neighbors. We say that a pair of directions (U, D) is a *zig-zag* if $(\mathrm{U}, \mathrm{D}) \subseteq \{(\mathrm{SE}, \mathrm{NE}), (\mathrm{NE}, \mathrm{SE}), (\mathrm{SE}, \mathrm{SW}), (\mathrm{SW}, \mathrm{SE})\}$. A sequence $\{s_1, \ldots, s_j\}$ is a *reasonable sequence* for a zig-zag (U, D) if the following holds:

$$s_{2i} \text{ is a legal U neighbor of } s_{2i-1} \text{ for } 1 < 2i \leq j,$$

$$s_{2i+1} \text{ is a legal D neighbor of } s_{2i} \text{ for } 1 < 2i + 1 \leq j.$$

A subset W of S is reasonable if the points in W form a reasonable sequence.

LEMMA 4.3. *There exists a k-restricted Steiner minimal tree of S in which the set of given points of every full component is reasonable.*

Proof. Let T be a k-restricted Steiner minimal tree of S. For a contradiction, suppose that T has a full component C such that the set of given points of C is not reasonable. Since C is a Steiner minimal tree of a subset of S, C may be assumed to be of the forms described in Section 4.1. Then there exist given points B_i, B_{i+1} in C such that the rectangle with corners B_i and B_{i+1} contains a given point P which is not in C (Fig 9). We will consider only the case where P lies above the Steiner chain in C; the other case is similar.

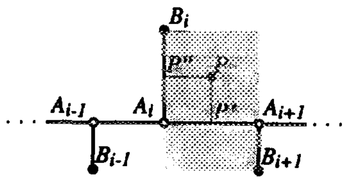


FIG. 9. Full component G.

Let B_j be the first vertex in C on the path in T from P to C. We have three cases to consider:

Case 1. $B_j = B_i$. Add the edge PP' to T and remove the edge A_iB_i to obtain the tree T'. Let $C' = C - \{A_iB_i\} \cup \{PP'\}$. Then C' is a full component in T' and the number of given points in C' is not more than k. Also, every other full component of T' is a full component of T as well. Therefore, T' is k-restricted. Since $|PP'| < |A_iB_i|$, $cost(T') < cost(T)$, a contradiction.

Case 2. B_j lies strictly to the left of B_i in C. Remove the segments A_iA_{i-1}, A_iP', and A_iP'' from T and add the segments PP' and PP'' to obtain T'. It is clear that T' is a Steiner tree of S. It is also easy to see that T' is k-restricted (the full component C in T has been split into two smaller full components in T'). Since $cost(T') = cost(T) - |A_iA_{i-1}|$, we once again have a contradiction.

Case 3. B_j lies strictly to the right of B_i in C. Add the edge PP'' to T and remove the edge A_iA_{i+1} to obtain the tree T'. Again we have a contradiction as in the previous cases. ■

LEMMA 4.4. *Fix $P \in S$ and a zig-zag* (U, D). *Then the number of sequences $\{P, P', P''\}$ of S which are reasonable for* (U, D) *is at most n.*

Proof. Let (U, D) = (NE, SE)(other zig-zags may be handled similarly). We will show that the number of sequences of the form $\{P, P', P''\}$ such that P' is a legal (NE neighbor of P and P'' is a legal SE neighbor of P' is at most n.

We denote the x and y coordinates of a point A as A^x and A^y, respectively.

Let $Q_1, \dots, Q_l$ be the legal NE neighbors of P in S such that $Q_1^x < \cdots < Q_l^x$. Clearly $Q_i^y > Q_j^y$ for $i < j$ (see Fig. 10). Let $Y_i = Z_i \cup \{Q_{i+1}\}$, where $Z_i = \{A \in S | \text{either } Q_i^x < A^x < Q_{i+1}^x \text{ and } A^y < Q_i^y, \text{ or } Q_i^y > A^y > Q_{i+1}^y \text{ and } A^x > Q_i^x, \text{ or both}\}$, $1 \le i \le l - 1$, and $Y_l = \{A \in S | Q_l^x > A^x \text{ and } Q_l^y > A^y\}$ (see shaded region in Fig. 10 showing Y_i). For each Q_i, $1 \le i \le l$, the legal SE neighbors can only be points in Y_i.

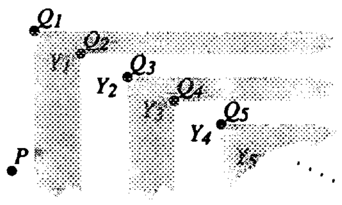


FIG. 10. Reasonable sequences from P.

Therefore, the number of reasonable sequences of the form $\{P, Q_i, P''\}$ is at most $|Y_i|$. Note that $Y_i \cap Y_j = \emptyset$ for $i \neq j$. Therefore, the total number of reasonable sequences of the form $\{P, P', P''\}$ is at most $\Sigma_{i=1}^{l}|Y_i| \leq n$. ∎

LEMMA 4.5. *The number of reasonable sequences of S of size j is $O(n^{\lfloor j/2 \rfloor + 1})$.*

Proof. Fix a zig-zag (U, D). We denote by $t(A, j)$ the number of sequences of length j that begin with A and are reasonable for (U, D)). We let $t'(j) = \max\{t(A, j)|A \in S\}$.

Fix $P \in S$. It suffices to show that $t(P, j)$ is at most $n^{\lfloor j/2 \rfloor}$. The proof proceeds by induction on j.

The claim is trivial when $j = 1, 2$, and it follows from Lemma 4.4 when $j = 3$.

Assume that $j > 3$. We may assume w.l.o.g that a reasonable sequence of the form $\{P, P_1, \ldots, P_{j-1}\}$ is obtained by composing the reasonable subsequences $\{P, P_1, P_2\}$ and $\{P_2, \ldots, P_{j-1}\}$ of length 3 and $j - 2$ respectively. Assume by way of inductive hypothesis that $t'(j-2) \leq n^{\lfloor (j-2)/2 \rfloor}$. It follows that $t(P, j) \leq t(P, 3)t'(j - 2) \leq nn^{\lfloor (j-2)/2 \rfloor}) = n^{\lfloor j/2 \rfloor}$. ∎

LEMMA 4.6. *All reasonable sequences of S of size j can be generated in time $O(n^{\lfloor j/2 \rfloor + 1})$.*

Proof. Assume that the points in S have been sorted according to their x-coordinate as well as for according to their y-coordinate. Then for a zig-zag (U, D), the sets of all the legal U and D neighbors of every point in S can easily be computed in time $O(n^2)$. Once this task is completed, the generation of legal sequences of length at most j can be easily accomplished in time proportional to their number. ∎

5. Conclusions and Open Problems

We presented a new heuristic for the Steiner tree problem and showed that its performance ration is better than previously known ratios for the existing heuristics. We do not know if bounds on performance ratio proven for our heuristics, as well as for that of Zelikovsky [19, 20], are tight; for $k = 3$ we conjecture performance ratios close to r_3. Also, we conjecture that the value of r_k in rectilinear metric for $k \geq 4$ is $1 + 1/(2k - 1)$ rather than $1 + 1/(2k - 2)$ as shown in Theorem 2.2.

It will be interesting to see if there exists a heuristic that considers Steiner minimal trees for subsets of S of size at most k and achieves a better ratio compared to algorithm $\mathscr{A}_k$. Our preliminary results indicate that this might be true for $k = 3$; indeed we believe that one can design a polynomial time approximation scheme for computing 3-restricted Steiner minimal tree, which would automatically improve the performance in the case considered in this paper. It is worth pointing out that this problem is not yet known to be NP-complete.

Very recently, Zelikovsky [21] improved the running time of our algorithm in the rectilinear metric to $O(n^{1.5})$ when $k = 3$.

Acknowledgment

We thank Toshihiro Fukito, Christos Papadimitriou, and Georg Schnitger for helpful discussions and suggestions.

References

1. P. Berman and V. Ramaiyer, "An Approximation Algorithm for the Steiner Tree Problem," Technical Report CS-91-05, The Pennsylvania State University, 1991.
2. M. Bern and P. Plassmann, The Steiner problems with edge lengths 1 and 2, *Inform. Process. Lett.* **32** (1989), 171–176.
3. T. H. Brylawski, Some properties of basic families of subsets, *Discrete Math.* **6** (1973), 333–341.
4. D. Z. Du, Y. Zhang, and Q. Feng, On better heuristic for euclidean Steiner minimum trees *in* "Proceedings, 32nd FOCS, 1991," pp. 431–439.
5. D. Z. Du and F. K. Hwang, An approach for proving lower bounds: Solution of Gilbert–Pollak's conjecture on Steiner ratio, *in* "Proceedings, 31st FOCS," 1990, pp. 76–85.
6. G. N. Frederickson, Data structures for on-line updating of minimum spanning trees, with applications, *SIAM J. Comput.* **14** (1985), 781–798.
7. M. R. Garey, R. L. Graham, and D. S. Johnson, The complexity of computing Steiner minimal trees, *SIAM J. Appl. Math.* **32** (1977), 835–859.
8. M. R. Garey and D. S. Johnson, The rectilinear Steiner problem is NP-complete, *SIAM J. Appl. Math* **32** (1977), 826–834.

9. E. N. Gilbert and H. O. Pollack, Steiner minimal trees, *SIAM J. Appl. Math* **16** (1968), 1–29.
10. C. Greene, A multiple exchange property for bases, *Proc. Amer. Math. Soc.* **39** (1973), 45–50.
11. S. L. Hakimi, Steiner's problem in graphs and its implications *Networks* **1** (1971), 113–133.
12. M. Hanan, On Steiner's problem with rectilinear distance, *SIAM J. Appl. Math.* **14** (1966), 255–265.
13. F. K. Hwang, On Steiner minimal trees with rectilinear distance, *SIAM J. Appl. Math* **30** (1976), 104–114.
14. R. M. Karp, Reducibility among combinatorial problems, "Complexity of Computer Computations," *in* (Miller and Thatcher, Eds), pp. 85–103, Plenum Press, New York, 1972.
15. L. Kou, G. Markowsky, and L. Berman, A fast algorithm for Steiner trees, *Acta Informat.* **15** (1981), 141–145.
16. A. J. Levin, Algorithm for the shortest connection of a group of vertices, *Soviet Math. Dokl.* **12** (1971) 1477–1481.
17. D. D. Sleator and R.E. Tarjan, A data structure for dynamic trees, *J. Comput. System. Sci.* **4** (1975), 362–391.
18. H. Takahashi and A. Matsuyama, An approximate solution for the Steiner problem in graphs, *Math. Japon.* **24** (1980), 573–577.
19. A. Z. Zelikovsky, The 11/6 approximation algorithm for the Steiner problem on networks, submitted for publication.
20. A. Z. Zelikovsky, The 11/8 approximation algorithm for the Steiner problem on networks with rectilinear distance, manuscript.
21. A. Z. Zelikovsky, personal communication.