Algorithmica (1997) 18: 99–110

Algorithmica

A Series of Approximation Algorithms for the Acyclic Directed Steiner Tree Problem[1]

A. Zelikovsky[2]

Abstract. Given an acyclic directed network, a subset S of nodes (*terminals*), and a root r, the *acyclic directed Steiner tree problem* requires a minimum-cost subnetwork which contains paths from r to each terminal. It is known that unless $NP \subseteq DTIME[n^{\text{polylog}\, n}]$ no polynomial-time algorithm can guarantee better than $(\ln k)/4$-approximation, where k is the number of terminals. In this paper we give an $O(k^{\varepsilon})$-approximation algorithm for any $\varepsilon > 0$. This result improves the previously known k-approximation.

Key Words. Algorithms, Approximations, Steiner tree.

1. Introduction. The general Steiner tree problem in graphs requires a minimum-cost tree spanning a distinguished node set S in a network G. This problem is investigated for different types of networks. We mention below the following cases: usual networks with edge costs (NSP), node-weighted networks (NWSP) where the cost of a tree is the sum of edge costs and prescribed costs of its nodes, acyclic directed networks with edge costs (ADSP), and directed networks (DSP).

We consider the Steiner tree problem for *acyclic* directed graphs, i.e., directed graphs where no directed chain leads from any node to itself.

ACYCLIC DIRECTED STEINER TREE PROBLEM (ADSP). Given an acyclic digraph $G = (V, E, d)$ with edge costs $d: E \to R^+$, a subset $S \subset V$ and a root $r \in V$, find a minimum cost subgraph containing directed paths from r to all elements of S (*minimum-cost Steiner tree*).

For an instance of the general Steiner tree problem, Smt and smt denote the minimum-cost Steiner tree and its cost, respectively. The elements of the set S are called *terminals*. The number of terminals is denoted by k.

ADSP is also known as the *Steiner arborescence problem in acyclic networks* [6]. It has various practical applications. The most important occurs in biology while constructing *phylogenetic trees* [4]. A number of papers are devoted to the case of a digraph embedded in a d-dimensional rectilinear metric. For $d = 2$, a fast and effective heuristic was proposed in [12]; however, this case has not yet been shown to be *NP*-hard. An exact exponential-time algorithm for ADSP based on embedding of a graph in a d-dimensional rectilinear metric was given in [11].

---

[1] This research was supported in part by Volkswagen-Stiftung and Packard Foundation.

[2] Computer Science Department, Thornton Hall, University of Virginia, Charlottesville, VA 22903, USA. alexz@chestnut.cs.virginia.edu.

Received October 15, 1994; revised August 23, 1995. Communicated by M. X. Goemans.

Most of cases of the general Steiner tree problem (NSP, NWSP, ADSP, DSP) are *NP*-hard [7], thus many approximation algorithms have appeared in the last two decades. The quality of an approximation algorithm is measured by its performance ratio: an upper bound on the ratio between the achieved cost and the optimal cost. A worst-case analysis for some approximation algorithms was provided to find its *exact* performance ratio. For the most complicated cases, a performance ratio may depend on the number of terminals. From the other side, significant progress in lower bounds for approximation complexity of *NP*-hard problems has been made in the last few years [16].

The approximation complexity of NSP and NWSP has already been determined. NSP belongs to *MAXSNP*-class [3], so a constant factor approximation algorithm exists [14] and, for some $\varepsilon > 1$, ε-approximation is *NP*-complete [1]. For NWSP, a $(2\ln k)$-approximation algorithm was designed [8]. From the other side, the famous set cover problem may be embedded in NWSP. This implies that NWSP cannot be approximated to within less than $(\frac{1}{4}\ln k)$-factor unless $DTIME[n^{\text{polylog}\, n}] \supseteq NP$ [9]. Therefore, the only question for these problems is still open: what exact constants separate polynomially solvable and *NP*-complete approximations? For NSP, this constant is at most $1 + \ln 2 \approx 1.69$ [18]. For the euclidean and rectilinear subcases of NSP, these constants are at most $1 + \ln(2/\sqrt{3}) \approx 1.1438$ [18] and $\frac{61}{48} \approx 1.271$ [2], respectively.

The approximation complexity of ADSP and DSP is still unknown. The only thing we can say is that the set cover problem can be transformed to ADSP, so these problems are no easier to approximate than NWSP. To determine an upper bound on the approximability of ADSP we may compare it with the next already distinguished approximation complexity class. The famous representative of this class is the chromatic number problem (CNP). This class is characterized by the existence of $\varepsilon > 0$ such that the n^ε-approximation is *NP*-complete [9]. The main result of this paper says that to approximate ADSP is easier than to approximate CNP.

THEOREM 1. *There exists a series of approximation algorithms A_l, $l = 1, 2, \ldots,$ for the acyclic directed Steiner tree problem. The performance ratio of an algorithm A_l is*

$$k^{1/l}(2 + \ln k)^{l-1},$$

where k is the number of terminals. The runtime of algorithm A_l is $O(\alpha + n^{l-1}k^l)$, where n is the number of nodes of the input graph and α means the time complexity of all pairs shortest paths.

REMARK 1. The limit guarantee of a presented series of heuristics

$$\exp[\sqrt{4\ln k \ln(\ln k + 2)} - \ln(\ln k + 2)] = \frac{k^{\sqrt{4\ln(\ln k+2)/\ln k}}}{\ln k + 2}$$

is subpolynomial, i.e., its growth is less than k^ε for any $\varepsilon > 0$.

We believe that the approximation complexity of ADSP is characterized by the presented series of heuristics.

CONJECTURE 1. *ADSP cannot be approximated with a subpolynomial guarantee unless $P = NP$.*

In the next section we describe in terms of contraction several known heuristics for Steiner tree problems and a new level-restricted relative greedy heuristic. In Section 3 we estimate an approximation of optimal Steiner trees for ADSP with level-restricted Steiner trees. A formal definition of heuristics A_l with a runtime analysis is presented in Section 4. The last section is devoted to the proof of the performance ratio claimed in Theorem 1.

2. The Greedy Contraction Framework. We assume that the digraph G is transitive, i.e., for any $u - v$ path, in G there is an edge $(u, v) \in E$. Moreover, the cost of any edge in G coincides with the cost of the minimum-cost path between its ends. G_S denotes the subgraph of G induced by the set $S \cup r$. $Mst(S)$ is the minimum spanning tree of G_S (also called the *minimum spanning arborescence* of G_S) and $M_0 = M_0(S)$ is its cost.

A *full Steiner tree* T is a rooted out-going tree such that

1. the root of T, $r(T)$, belongs to $S \cup r$ and has only one son;
2. all leaves of T belong to S;
3. all other nodes of T do not belong to S.

We can split Smt into edge-disjoint full components. A full tree T is said to be of *level* $l = l(T)$ if every path in T from its root to any leaf has at most l edges.

Contraction of a tree T means reducing to zero the costs of edges of $Mst(S)$ ending at the terminals of T (or edges of G_S between terminals of T for undirected Steiner problems). We denote the result of contraction by S/T. Thus contraction reduces the value $M_0(S)$.

For all the Steiner tree problems, the following greedy contraction framework is successfully used in approximations.

Greedy Contraction Framework (GCF)

(1) `repeat until` $M_0(S) = 0$
  (a) `find a full Steiner tree` T^* `in` *a class* K `which minimizes` *a criterion function* $f(T)$: $T^* \leftarrow arg \min_{T \in K} f(T)$.
  (b) `insert` T^* in *LIST*.
  (c) `contract` T^*, $S \leftarrow S/T^*$.
(2) `reconstruct an output Steiner tree from trees of` *LIST*.

Many famous heuristics can be embedded in this framework considering different definitions of a class K and a criterion function f.

THE MINIMUM SPANNING TREE HEURISTIC (MSTH) [14]. K consists of all paths and $f(T) = d(T)$.

THE RAYWARD-SMITH HEURISTIC (RSH) [13]. K contains all stars and $f(T) = d(T)/(r - 1)$, where r is the number of leaves of T.

THE GENERALIZED GREEDY HEURISTIC (GGH) [17]. K consists of trees with three terminals and $f(T) = d(T) - (M_0(S) - M_0(S/T))$.

THE SIZE-RESTRICTED RELATIVE GREEDY HEURISTIC (SRGH) [18]. $K = K_r$ contains all trees with at most r terminals.

$$f(T) = \frac{d(T)}{M_0(S) - M_0(S/T)}.$$

Let $\tilde{K}$ be the family of all Steiner trees with full components belonging to K. Let $smt_{\tilde{K}}$ be the cost of the minimum Steiner tree in $\tilde{K}$. We denote by $cost_A$ the cost of the output tree of an algorithm A embedded in GCF. To determine a performance guarantee of A we may bound the following two ratios: $a_1 = (smt_{\tilde{K}}/smt)$, and $a_2 = (cost_A/smt_{\tilde{K}})$.

Below we present the known bounds for the ratios a_1 and a_2 for the above heuristics embedded in GCF:

MSTH gives $a_1 \leq 2$ and $a_2 = 1$ for NSP, and $a_1 \leq k$ and $a_2 = 1$ for NWSP, ADSP.
RSH gives $a_1 \leq \frac{5}{3}$ and $a_1 \cdot a_2 \leq 2$ for NSP [15] and $a_1 \cdot a_2 \leq 2 \log k$ for NWSP [8].
GGH gives $a_1 \leq \frac{5}{3}$ and $a_1 \cdot a_2 \leq \frac{11}{6}$ for NSP [17].
SRGH gives $\lim_{r\to\infty} a_1 = 1$ [5] and $\lim_{r\to\infty} a_2 = 1 + \ln 2$ for NSP [18]. In other words, it induces a series of approximation algorithms for NSP with the limit performance ratio $(1 + \ln 2)$.

In this paper we present a

LEVEL-RESTRICTED RELATIVE GREEDY HEURISTIC (LRGH). The class $K = K_l$ consists of full Steiner trees with at most l levels. The criterion function is the same as for SRGH:

$$f(T) = \frac{d(T)}{M_0(S) - M_0(S/T)}. \tag{1}$$

Theorem 2 of the next section says that $a_1 \leq k^{1/l}$ for DSP. The rest of the paper is devoted only to ADSP. In Section 5 we prove that $a_2 \leq (2 + \ln k)$. Unfortunately, we cannot compute exactly $arg \min_{K_l} f(T)$ for $l \geq 3$. Section 4 shows how we avoid this obstacle by restricting the class K_l.

3. Level-Restricted Steiner Trees. A Steiner tree is called *l-restricted* if the level of its full components does not exceed l. Smt_l and smt_l denote the minimum-cost l-restricted Steiner tree and its cost, respectively. The following theorem bounds the approximation of minimal Steiner trees with minimum-cost l-restricted trees.

THEOREM 2. *For any instance of the directed Steiner tree problem,*

$$smt_l/smt \leq k^{1/l}.$$

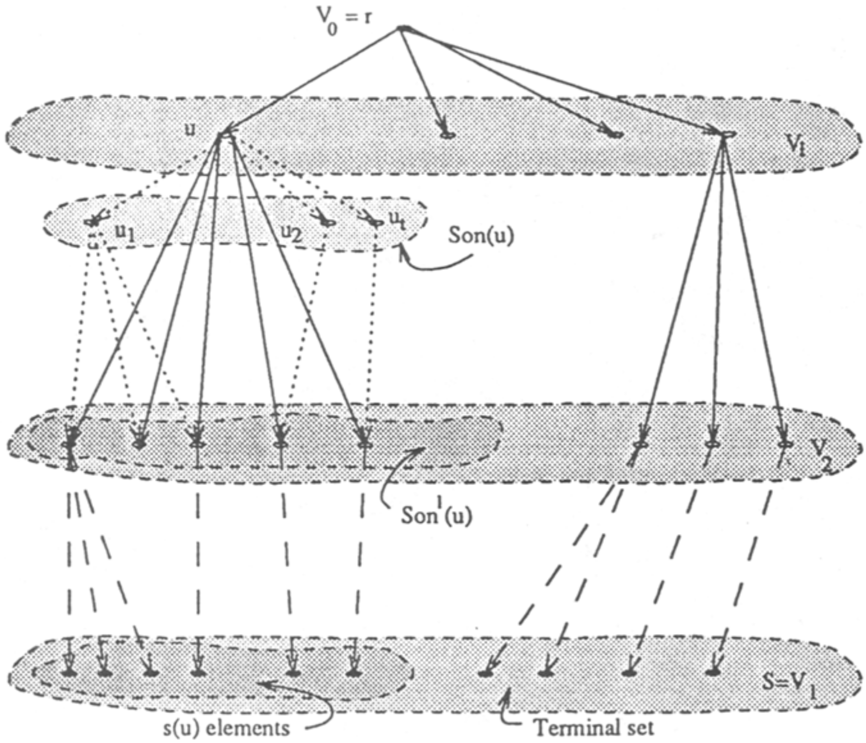

Fig. 1. The l-restricted tree T^l drawn from a full Steiner tree. The Steiner tree edges are dotted.

PROOF. We construct Smt_l for every full component of Smt separately, so we can assume that Smt is a full Steiner tree.

First we introduce some notations. Let $T = Smt$ and let v be a node of T. Let $\bar{v}$ denote the set of all descendants of v and let $s(v)$ denote the number of terminals in $\bar{v}$, e.g., $s(r) = k$. $Son(v)$ is the set of all sons of v in T. Let

$$V_i = \{v \in T, s(v) \geq k^{(l-i)/l} \;\&\; s(v') < k^{(l-i)/l} \text{ for any } v' \in Son(v)\},$$

$i = 1, \ldots, l-1$, $V_l = S$, $V_0 = \{r\}$. For every $v \in V_i$, $i = 0, \ldots, l-1$, let $Son^l(v)$ denote $\bar{v} \cap V_{i+1}$.

Let T^l be a tree with the node set $V^l = \bigcup_{i=0}^{l} V_i$ and the edge set $E^l = \{(u, v)$: $u \in V^l$, $v \in Son^l(u)\}$. The cost of an edge (u, v) in T^l coincides with the cost of the u–v path in the tree T. Note that the tree T^l is an l-restricted Steiner tree, since $u \notin \bar{v}$ for any $u \neq v$, $u, v \in V_i$ (Figure 1).

Let $u \in V_i$ and let $Son(u) = \{u_1, \ldots, u_t\}$. For every $j = 1, \ldots, t$, let U_j denote $Son^l(u) \cap \bar{u}_j$ and let $d(u_j, u_j^*) = arg\max_{v \in U_j} d(u_j, v)$. Then

$$\sum_{v \in Son^l(u)} d(u, v) \leq \sum_{j=1}^{t} |U_j|(d(u_j, u_j^*) + d(u, u_j)) \tag{2}$$

$$= \sum_{j=1}^{t} |U_j| d(u, u_j^*)$$
$$\leq \left(\max_{j=1,\dots,t} |U_j| \right) \sum_{j=1}^{t} d(u, u_j^*).$$

Note that $\sum_{v \in U_j} s(v) = s(u_j)$ yields $|U_j| \min_{v \in V_{i+1}} s(v) \leq s(u_j)$ and

(3) $$\max_{j=1,\dots,t} |U_j| \min_{v \in V_{i+1}} s(v) \leq \max_{j=1,\dots,t} s(u_j).$$

Since $\min_{v \in V_{i+1}} s(v) \geq k^{(l-i-1)/l}$ and $\max_{j=1,\dots,t} s(u_j) \leq k^{(l-i)/l}$, (3) yields

(4) $$\max_{j=1,\dots,t} |U_j| < k^{1/l}.$$

Inequalities (2) and (4) imply

$$\sum_{v \in Son^l(u)} d(u, v) \leq k^{1/l} \sum_{j=1}^{t} d(u, u_j^*).$$

Note that all u–u_j^* paths are edge-disjoint in the tree T. Thus,

$$d(T_l) = \sum_{u \in V^l} \sum_{v \in Son^l(u)} d(u, v) \leq k^{1/l} \sum_{u \in V^l} \sum_{j=1}^{t(u)} d(u, u_j^*) \leq k^{1/l} d(T).$$

4. The Series of Algorithms. In this section we construct recursively the series of algorithms $\{A_l, l = 1, 2, \dots\}$. For any l, Algorithm A_l is LRGH with the restricted subclass of K_l, i.e., it approximates the minimum-cost l-restricted Steiner tree.

A_1 *coincides with MSTH.* Since G_S has no cycles, $Mst(S)$ consists of the cheapest edges ending at S-nodes in G_S. For any $s \in S$, denote the cost of such an edge by $m(s) = \min_{s' \in S} d(s', s)$ (we assume that $d(s, s) = \infty$, since there are no loops in G). So the output cost is $M_0 = \sum_{s \in S} m(s)$.

A_2 *coincides with LRGH.* Our goal is computing Step (a) of GCF for the function (1).

We need the following notations. For any $v \in V - S$, let $d_0 = \min_{s \in S \cup r} d(s, v)$ and $s_0 = arg \min d(s, v)$. $S(v)$ and $t(v)$ denote the set of all S-descendants of v and its size, respectively. For any $s_i \in S(v)$, $d_i = d(v, s_i)$, $m_i = \min_{s \in S} d(s, s_i)$. We assume that the set $S(v)$ is enumerated in such way that $d_i / m_i \leq d_{i+1} / m_{i+1}$.

Every 2-level full Steiner tree T is determined by its root $r(T) \in S \cup r$, the unique internal node $v \in V - S$, and leaves (Figure 2).

The following lemma makes computing $arg \min_{T \in K_2} f(T)$ possible.

LEMMA 1. *For any* $v \in V - S$,

$$\min_{l(T)=2, T \ni v} f(T) = \min_{j=1,\dots,t(v)} \frac{\sum_{i=0}^{j} d_i}{\sum_{i=1}^{j} m_i}.$$

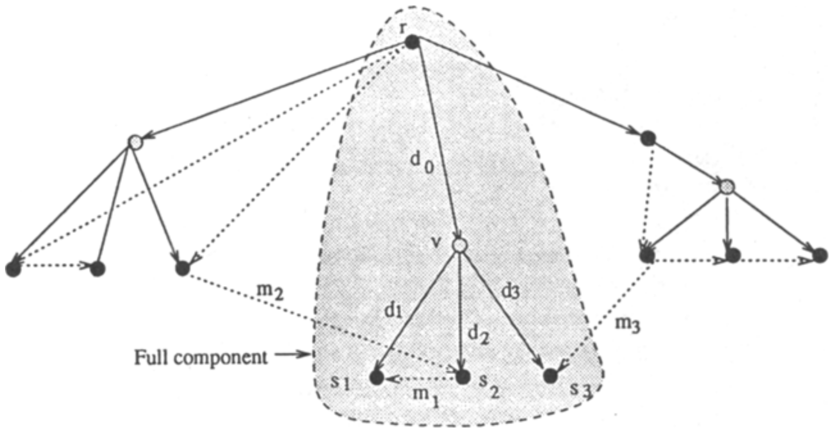

Fig. 2. Minimum spanning and 2-restricted Steiner trees. The MST edges are dotted.

PROOF. Let $T^* = arg\min_{v \in T} f(T)$, and let $\{(s_0^*, v), (v, s_1^*), \ldots, (v, s_{t^*}^*)\}$ be its edges. We can rewrite (1) as follows:

$$f(T^*) \doteq \frac{d(s_0^*, v) + \sum_{i=1}^{t^*} d(v, s_i^*)}{\sum_{i=1}^{t^*} m(s_i^*)}.$$

We may replace the root s_0^* by s_0 in T^* without increasing f since $d(s_0, v) \le d(s_0^*, v)$. Let s^* be the "last" terminal, i.e., $s^* = arg\max_{i=1,\ldots,t^*}[d(v, s_i^*)/m(s_i^*)]$. Assume that, for some $s \in S(v)$,

$$\frac{d(v, s)}{m(s)} \le \frac{d(v, s^*)}{m(s^*)}. \tag{5}$$

To prove the lemma we show that $f(T^* \cup (v, s)) \le f(T^*)$. Indeed,

$$f(T^*) \le f(T^* - (v, s^*)) = \frac{d(T^*) - d(v, s^*)}{\sum_{i=1}^{t^*} m(s_i^*) - m(s^*)}$$

since T^* minimizes f. Therefore, $d(v, s^*)/m(s^*) \le f(T^*)$. Thus, inequality (5) yields

$$f(T^* \cup (v, s)) = \frac{d(T^*) + d(v, s)}{\sum_{i=1}^{t^*} m(s_i^*) + m(s)} \le \frac{d(T)}{\sum_{i=1}^{t^*} m(s_i^*)} = f(T^*). \qquad \square$$

The Algorithms A_i, $i \ge 3$. We cannot find $T_l^* = arg\min_{K_l} f(T)$ exactly even for $l = 3$. So we are looking for a minimum of f in a subclass of K_l defined below.

We define a tree $T_l(u)$, $l \ge 1$, recursively. For any $u \in V$, $T_1(u) = \{(u, s^*)\}$, where $s^* = arg\min_{s \in S}[d(u, s)/m(s)]$. For instance, any edge $(s, s') \in Mst(S)$ may be assigned to $T_1(s)$. Denote $V(u) = \{v \in V - S : d(u, v) \le d(s, v)$ for all $s \in S \cup r\}$. To define $T_l(s)$, $l \ge 2$, we run the following:

Procedure

```
(0) G' ← G;
(1) for each v ∈ V(u) do
    (a) B_v ← (s, v), i ← 1;
    (b) repeat forever
        T^i_{l-1} ← T_{l-1}(v);
        if f(B_v ∪ T^i_{l-1}) ≥ f(B_v), then exit repeat;
        B_v ← B_v ∪ T^i_{l-1};
        contract T^i_{l-1}, S ← S/T^i_{l-1}, i ← i + 1;
    (c) for j = 1, ..., i do
        if f(B_v) < f(T^j_{l-1}) then B_v ← B_v \ T^j_{l-1}
    (d) G ← G';
(2) T_l(u) ← arg min{f(T) : T = T_{l-1}(u) ∨ T = B_v, v ∈ V(u)}
```

In other words, we extend (u, v) with the "best" $(l-1)$-restricted full tree T_{l-1} rooted in v, which we find recursively. We add (T_{l-1})'s as long as this addition decreases the function f. Step (c) further reduces f by discarding (T_{l-1})'s with a positive contribution to $f(B_v)$. Note that step (c) can be omitted for $l = 2$. Then we restore G and find graphs B_v for all other $v \in V(u)$. So each B_v is an l-restricted full tree with the unique son v of the root u. Among all (B_v)'s and $T_{l-1}(u)$ we choose one with the smallest value of f as the tree $T_l(u)$. It is easy to see that Lemma 1 yields $T_2(s) = arg\min_{T \in K_2, r(T)=s} f(T)$.

REMARK 2. For any $s \in S \cup r$, $f(T_l(s)) \leq 1$.

PROOF. Indeed, for a nonzero edge $e \in Mst(S)$, $f(e) = 1$. □

Now we can present the algorithm A_l, $l \geq 2$, as follows:

Algorithm A_l

```
(1) repeat until M_0(S) = 0
    (a) s* ← arg min_{s∈S∪r} f(T_l(s)).
    (b) insert T_l(s*) in LIST.
    (c) contract T_l(s*), S ← S/T_l(s*)
(2) reconstruct an output Steiner tree from trees of LIST.
```

Now we estimate the time complexity of Algorithm A_l. For brevity, the sets and their cardinalities will have the same notations.

Since the graph G is acyclic, we can apply the $O(E + V \log V)$ procedure of finding $MST(S)$ due to Mehlhorn [10]. Thus, A_1 runs in time $O(E + V \log V)$.

For A_2, we need to know all distances between S and $V - S$. This can be done in time $O(E + V \log V)$ by adding an auxiliary node with zero cost edges to $S \cup r$ and finding all distances from this node to all others. Similarly we can find distances between $V - S$

and S. Then in time $O(VS)$ we can find $arg\min_{K_2} f(T)$ (Lemma 1). Thus, the total runtime of A_2 is $O(E + V\log V + S^2V)$.

To find $T_l(s^*)$, $i \geq 3$, we need a runtime $rt_l = O((VS)^{l-1})$, since $rt_l = rt_{l-1}VS$ and $rt_2 = O(VS)$. Thus, A_l has a runtime $O(\alpha + V^{l-1}S^l)$, where α means the time complexity of all pairs shortest paths.

5. The Performance Guarantee. Our first goal is to show that the minimum of the function f in the item (a) of Algorithm A_l is not far from the minimum in the whole class K_l. In other words, we generalize Lemma 1 for arbitrary l.

LEMMA 2. *Let $T_l^* = arg\min_{K_l} f(T)$ and let s^* be the root of T_l^*. Then, for $l \geq 2$,*

$$f(T_l(s^*)) \leq f(T_l^*)(2 + \ln k)^{l-2}. \tag{6}$$

PROOF. Induction on l. The case of $l = 2$ follows from Lemma 1. Denote $c_l = (2 + \log k)^{l-2}$ and $S^* = S \cap T_l^* \backslash \{s^*\}$. Let v^* be the unique son of the root s^* in the tree T_l^* and $d_0 = d(s^*, v^*)$.

First we consider the case of $S \cap T_l(s^*) \backslash \{s^*\} \subseteq S^*$. Consider ADSP for S^* with a root v^*. In the above notations, let Smt_{l-1} and smt_{l-1} be the minimum-cost $(l-1)$-restricted Steiner tree and its cost, respectively. So $T_l^* = (s^*, v^*) \cup Smt_{l-1}$. Denote $s_{l-1} = smt_{l-1}c_{l-1}$. Let $M_0 = M_0(S^*) = \sum_{s \in S^*} m(s)$.

We may prove (6) for B_{v^*} since $f(T_l(s^*)) \leq f(B_{v^*})$. We follow loop (b) of Procedure while it creates B_{v^*}. For brevity, we denote $d_1 = d(T_{l-1}^1)$, $M_1 = M_0(S^*/T_{l-1})$, and $m_1 = M_0 - M_1$. By induction $f(T_{l-1}) \leq f(T_{l-1}^*)c_{l-1}$. By definition of T_{l-1}^*, $f(T_{l-1}^*) \leq f(Smt_{l-1})$. Thus we obtain

$$\frac{d_1}{m_1} \leq \frac{s_{l-1}}{M_0}.$$

After contraction of T_{l-1}^1 the procedure finds T_{l-1}^2, T_{l-1}^3, and so on. Denote their corresponding values by d_i, M_i, and m_i. Similarly, $d_i/m_i \leq s_{l-1}/M_{i-1}$ and, therefore,

$$M_i \leq M_{i-1}\left(1 - \frac{d_i}{s_{l-1}}\right). \tag{7}$$

Unraveling (7), we obtain

$$M_p \leq M_0 \prod_{i=1}^{p} \left(1 - \frac{d_i}{s_{l-1}}\right).$$

Taking the natural logarithm on both sides and simplifying using the approximation $\ln(1+x) \leq x$, we obtain

$$\ln\frac{M_0}{M_p} \geq \frac{\sum_{i=1}^{p} d_i}{s_{l-1}}. \tag{8}$$

Assume that loop (b) interrupts after j iterations, i.e., $f(B_{v^*} \cup T_{l-1}^j) \geq f(B_{v^*})$. If $s_{l-1} \geq M_j$, then let B' be the tree obtained after $p+1$ iterations such that $M_p \geq s_{l-1} \geq M_{p+1}$. Note that $f(B') \geq f(B_{v^*})$, since B' is obtained no later than B_{v^*}.

If $s_{l-1} < M_j$, then $f(B_{v^*}) \le f(T_{l-1}^j) \le s_{l-1}/M_j < 1$. For the purposes of analysis we put $p = j-1$, $d_j = m_j = M_j - s_{l-1}$ and $f(B') = (\sum_{i=0}^{p+1} d_i)/(\sum_{i=1}^{p+1} m_i)$. Note that again $M_p \ge s_{l-1} \ge M_{p+1}$ and $f(B') \ge f(B_{v^*})$.

The inequality $d_{p+1}/m_{p+1} \le s_{l-1}/M_p$ implies $d_{p+1}/s_{l-1} \le m_{p+1}/M_p \le 1$. By Theorem 2, $M_0 \le smt_1 \le k \cdot smt \le k \cdot smt_{l-1}$. So (8) yields

$$\frac{\sum_{i=1}^{p} d_i + M_{p+1} + d_{p+1}}{s_{l-1}} \le 2 + \ln k. \tag{9}$$

Since $f(T_l(s^*)) \le 1$ (Remark 2), (6) is true if $f(T_l^*)c_l \ge 1$. So we may assume that $f(T_l^*)c_l < 1$. Inequality (9) yields

$$1 > f(T_l^*)c_l \ge \frac{d_0 + smt_{l-1}c_l}{M_0} \ge \frac{\sum_{i=0}^{p-1} d_i + M_{p+1}}{M_0}.$$

Since the last ratio is less than 1,

$$f(T_l^*)c_l \ge \frac{\sum_{i=0}^{p+1} d_i}{M_0 - M_{p+1}} = f(B') \ge f(B_{v^*}).$$

Thus, we proved (6) in the case of $S \cap T_l(v^*) \subseteq S^*$.

Now we turn to the case of an arbitrary set $S \cap T_l(v^*)\backslash\{s^*\}$. As above, we prove (6) for the tree B_{v^*}. We partition m_i of the tree T_{l-1}^i into two parts $m_i = m_i^* + \bar{m}_i$, where the first part is the sum of costs of edges ending at the S^*-nodes of T_{l-1}^i and the second is the sum of costs of edges ending at the rest of the S-nodes of T_{l-1}^i in the tree $Mst(S)$. We also partition $d_i = d_i^* + \bar{d}_i$ in the same proportion as m_i, i.e., $d_i^*/m_i^* = \bar{d}_i/\bar{m}_i$. Assign $\bar{d}_i \leftarrow 0$ if $\bar{m}_i = 0$, and $d_i^* \leftarrow 0$ if $m_i^* = 0$.

As above, let loop (b) interrupts after j iterations.

Step (c) of Procedure guarantees that, for any $i = 1, \dots, j-1$,

$$\frac{\bar{d}_i}{\bar{m}_i} = f(T_{l-1}^i) \le f(B_{v^*} - T_{l-1}^i).$$

Thus,

$$f(B_{v^*}) = \frac{d_0 + \sum_{i=1}^{j-1} d_i}{\sum_{i=1}^{j-1} m_i} = \frac{d_0 + \sum_{i=1}^{j-1} d_i^* + \sum_{i=1}^{j-1} \bar{d}_i}{\sum_{i=1}^{j-1} m_i^* + \sum_{i=1}^{j-1} \bar{m}_i} \le \frac{d_0 + \sum_{i=1}^{j-1} d_i^*}{\sum_{i=1}^{j-1} m_i^*}.$$

Note that the previous argument for the case of $S \cap T_l(v^*) \subseteq S^*$ is true for the values d_i^*, m_i^*, and $M_i = M_{i-1} - m_i^*$ if we omit such i's for which $m_i^* = 0$. Therefore,

$$f(B_{v^*}) \le \frac{d_0 + \sum_{i=1}^{j-1} d_i^*}{\sum_{i=1}^{j-1} m_i^*} \le f(T_l^*)c_l. \qquad \square$$

Now we are able to prove the main result of the paper.

PROOF OF THEOREM 1. Let T be the output tree of Algorithm A_l and let $T_l^1, T_l^2, \dots,$ be the trees inserted in *LIST*. Denote $d(T_l^i) = d_i$, $M_i = M_{i-1} - m_i$, where m_i is the sum of costs of edges ending at the S-nodes of T_l^i in the tree $Mst(S)$. As above, $c_l = (2+\ln k)^{l-2}$, $s_l = smt_l c_l$.

Note that $f(T_l^*) \le smt_l/M_0$. By Lemma 2, $d_1/m_1 \le f(T_l^*)c_l \le s_l/M_0$. Inductively,

$$\frac{d_i}{m_i} \le \frac{s_l}{M_{i-1}}.$$

Similarly to (8) we derive

$$\ln \frac{M_0}{M_p} \ge \frac{\sum_{i=1}^{p} d_i}{s_l}.$$

Since, for some j, $M_j = 0$, there is some p such that $M_p \ge s_l \ge M_{p+1}$. Let T' be the Steiner tree formed by the full trees obtained after p iterations of loop (1) and the rest of the Mst-edges. By Remark 2, $d_i/m_i \le 1$ and $d(T) \le d(T')$. Since $d_{p+1}/s_l \le m_{p+1}/M_p \le 1$ and $M_0 \le k\dot{s}mt$,

$$\frac{\sum_{i=1}^{p} d_i + M_{p+1} + d_{p+1}}{s_l} \le 2 + \ln k.$$

This implies that

$$d(T) \le \sum_{i=1}^{p+1} d_i + M_{p+1} \le s_l(2 + \ln k).$$

By Theorem 2, the last value is at most $k^{1/l}(2 + \ln k)^{l-1} smt$. □

PROOF OF REMARK 1. We find the limit performance guarantee of $\{A_l\}$. Denote the performance guarantee of A_l by $f_l(k) = k^{1/l}(2 + \log k)^{l-1}$. We need to find $f(k) = \min_l f_l(k)$. Taking the natural logarithm of $f_l(k)$ and derivative, we obtain

$$-\frac{\ln k}{l^2} + \ln(\ln k + 2) = 0. \tag{10}$$

Substituting the solution of (10) in $\ln f_l(k)$, we obtain

$$\ln f(k) = 2\sqrt{\ln k \ln(\ln k + 2)} - \ln(\ln k + 2).$$ □

Acknowledgments. I am deeply indebted to the referees whose comments considerably improved the final version of this paper. I would also like to thank Professors M. Szegedy and A. Shen for helpful discussions of this work.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. 33rd Annual IEEE Symp. on Foundations of Computer Science*, pp. 14–23, 1992.

[2] P. Berman, U. Fößmeier, M. Karpinski, M. Kaufmann, and A. Zelikovsky. *Approaching the 5/4-Approximations for Rectilinear Steiner Trees*. Lecture Notes in Computer Science, vol. 855. Springer-Verlag, Berlin, 1994, pp. 60–71.

[3] M. Bern and P. Plassmann. The Steiner problems with edge lengths 1 and 2. *Inform. Process. Lett.* 32:171–176, 1989.

[4] J. H. Camin and R. R. Sokal. A method of deducing branching sequences in phylogeny. *Evolution* 19:311–326, 1972.

[5] D. Z. Du, Y. Zhang, and Q. Feng. On better heuristic for Euclidean Steiner minimum trees. In *Proc. 32nd Annual IEEE Symp. on Foundations of Computer Science*, pp. 431–439, 1991.

[6] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*. Annals of Discrete Mathematics, vol. 53. North-Holland, Amsterdam, 1992.

[7] R. M. Karp. Reducibility among combinatorial problems. In Miller and Thatcher (eds.), *Complexity of Computer Computations*, Plenum, New York, 1972, pp. 85–103.

[8] P. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. In *Proc. Third Confer. on Integer Programming and Combinatorial Optimization*, pp. 323–331, 1993.

[9] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. In *Proc. 25th Annual ACM Symp. on Theory of Computing*, pp. 286–293, 1993.

[10] K. Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *Inform. Process. Lett.* 27:125–128, 1988.

[11] L. Nastansky, S. M. Selkow, and N. F. Stewart. Cost minimal trees in directed acyclic graphs. *Z. Oper. Res.* 18:59–67, 1974.

[12] S. K. Rao, P. Sadayappan, F. K. Hwang, and P. W. Shor. The rectilinear Steiner arborescence problem. *Algorithmica* 7:277–288, 1992.

[13] V. J. Rayward-Smith, The computation of nearly minimal Steiner trees in graphs. *Internat. J. Math. Ed. Sci. Tech.* 14:15–23, 1983.

[14] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Japon.* 24:573–577, 1980.

[15] B. M. Waxman and M. Imase. Worst case-performance of Rayward-Smith's Steiner tree heuristic. *Inform. Process. Lett.* 29:283–287, 1988.

[16] M. Yannakakis. Recent developments on the approximability of combinatorial problems. In K. W. Ng *et al.* (eds.), *Algorithms and Computation*. Lecture Notes in Computer Science, vol. 762. Springer-Verlag, Berlin, 1993, pp. 363–368.

[17] A. Z. Zelikovsky. An 11/6-approximation algorithm for the network Steiner problem. *Algorithmica* 9:463–470, 1993.

[18] A. Z. Zelikovsky. Better approximations algorithms for the network and Euclidean Steiner tree problems. Tech. Rep. CS-96-06, University of Virginia, Charlottesville, VA.