A Polynomial-Time Subpolynomial-Approximation Scheme for the Acyclic Directed Steiner Tree Problem

Alexander Zelikovsky*
Institute of Mathematics
Akademiei 5, Kishinev, 277028, Moldova
Email: 17azz@mathem.moldova.su

April 1994

Abstract

The acyclic directed Steiner tree problem (ADSP) requires a minimal outward tree within an acyclic digraph with edge costs $G = (V, E, d)$ which connects a root $r$ with a distinguished subset $S \subseteq V$, $\#S = k$. The best possible performance guarantee of any polynomial approximation algorithm for ADSP cannot be less than $\frac{1}{4} \log k$ unless $\tilde{P} \supseteq NP$. The presented series of heuristics $A_n$ has a performance guarantee $k^{\frac{1}{n}}(1 + \ln k)^{n-1}$. This implies that the approximation scheme for ADSP.

Keywords: Algorithms, approximations, Steiner tree

*Research partially supported by the Volkswagen-Stiftung.
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 will 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), 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 $d : E \to \mathbb{R}^+$, $S \subseteq V$ and $r \in V$, find a minimum cost outward-directed tree from a root $r$ containing $S$ (minimal Steiner tree).

For an instance of the general Steiner tree problem, $Smt$ and $smt$ denote the minimal 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 [5]. It has various practical applications. The most important occurs in biology while constructing phylogenetic trees [3]. 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 [11], 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 [10].

The most of cases of the general Steiner tree problem (NSP, NWSP, ADSP, DSP) are $NP$-hard [6], so many approximation algorithms 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 low bounds for approximation complexity of $NP$-hard problems has been made in the last few years [15].

The approximation complexity of NSP and NWSP is already determined. NSP belongs to MAX SNP-class [2], so a constant factor approximation algorithm exists [13] and for some $\epsilon > 1$, $\epsilon$-approximation is $NP$-complete [1]. For NWSP, a $2 \log k$-approximation algorithm was designed [7]. 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 $\frac{1}{4}$ unless $\tilde{P} \subseteq NP$ [9],\(^1\) Therefore, the only question for these problems is still open: what exact

\(^1\)Here we use $\tilde{P}$ to mean the complexity class $\text{DTIME}[n^{\log \log n}]$
constants separate polynomially solvable and \( NP \)-complete approximations? For NSP, this constant is at most \( 1 + \log 2 \approx 1.69 \) [17]. For the euclidean and rectilinear subcases of NSP, these constants are at most \( 1 + \frac{\log 1}{\sqrt{3}} \approx 1.1438 \) [17] and \( \frac{61}{38} \approx 1.271 \) [18], respectively.

The approximation complexity of ADSP and DSP is still unknown, the only we can say that the set cover problem can be transformed to ADSP, so these problems are not easier to approximate than NWSP. To determine an upper bound of 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 \( \epsilon > 0 \) such that the \( n^\epsilon \)-approximation is \( NP \)-complete [9]. The main result of the paper says that to approximate ADSP is easier than to approximate CNP.

**Theorem 1** There is a polynomial-time approximation scheme \( A_l, l = 1, 2, \ldots \), for the acyclic directed Steiner tree problem. The performance ratio of an algorithm \( A_l \) is

\[
k^l(1 + \log k)^{l-1},
\]

where \( k \) is the number of terminals. The runtime of an algorithm \( A_l \) is \( O(n^2 + n^{l-1}k^l) \), where \( n \) is the number of vertices of the input graph.

**Remark 1** The function

\[
\exp\left[\frac{4\ln k \ln (\ln k + 1)}{\ln k} - \ln (\ln k + 1)\right] = \frac{k^{\frac{4\ln k \ln (\ln k + 1)}{\ln k}}}{\ln k + 1}
\]

is a subpolynom, i.e. its growth is less than \( k^\epsilon \) for any \( \epsilon > 0 \).

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

**Conjecture 1** ADSP cannot be approximated with subpolynom 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

At first 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 minimal path between its ends. $G_S$ denotes a subgraph of $G$ induced by the set $S \cup r$. $Mst(S)$ is the minimum spanning tree of $G_S$ and $M_0 = M_0(S)$ is its cost.

A full Steiner tree does not contain internal terminal nodes and it has only one edge from its root. We can split $Smt$ into edge-disjoint full components. A full tree has a level $l$ if every path from its root to any leaf has at most $l$ edges.

Contraction of a tree $T$ means reducing to 0 the costs of edges of $Mst(S)$ coming to the terminals of $T$ (or edges of $G_S$ between terminals of $T$ for undirected Steiner problems). We denote the result of contraction by $S/T$. So 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 $T^\ast$ in a class $K$ which minimizes
       a criterion function $f(T) = \arg \min_{T \in K} f(T)$.
   (b) insert $T^\ast$ in $LIST$.
   (c) contract $T^\ast$, $S \leftarrow S/T^\ast$.
(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) [13].
  $K$ consists of all paths and $f(T) = d(T)$.

- The Rayward-Smith’s heuristic (RSH) [12].
  $K$ contains all stars and $f(T) = \frac{d(T)}{r-1}$, where $r$ is the number of leaves of $T$.

- The generalized greedy heuristic (GGH) [16].
  $K$ consists of trees with 3 terminals and $f(T) = d(T) - (M_0(S) - M_0(S/T))$.

- The size-restricted relative greedy heuristic (SRGH) [17].
  $K = K_r$ contains all trees with at most $r$ terminals. $f(T) = \frac{d(T)}{M_0(S) - M_0(S/T)}$.

To determine a performance guarantee of an algorithm $A$ embedded in GCF we may bound the following two ratios:

$$a_1 = \frac{smt_K}{smt}.$$
where \( \text{smt}_K \) is the minimum tree cost in the family \( \tilde{K} \) containing all Steiner trees with full components belonging to \( K \):

\[
a_2 = \frac{\text{cost}_A}{\text{smt}_K},
\]

where \( \text{cost}_A \) is the cost of the output tree of the greedy algorithm \( A \).

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 \( a_1 \leq 5/3 \) and \( a_1 \cdot a_2 \leq 2 \) for NSP [14] and \( a_1 \cdot a_2 \leq 2 \log k \) for NWSP [7].

GGH gives \( a_1 \leq 5/3 \) and \( a_1 \cdot a_2 \leq 11/6 \) for NSP [16].

SRGH gives \( \lim_{r \to \infty} a_1 = 1 \) [4] and \( \lim_{r \to \infty} a_2 = 1 + \log 2 \) for NSP [17]. In other words, it induces a polynomial \((1 + \log 2)\)-approximation scheme for NSP.

In this paper we present a level-restricted relative greedy heuristic (LRGH).

The class \( K = K_1 \) 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 \( a_1 \leq k^{1/4} \) for DSP. The rest of the paper is devoted only to ADSP. In Section 5 we prove that \( a_2 \leq (1 + \log k) \).

Unfortunately, we cannot exactly compute \( \arg \min_{K_1} f(T) \) for \( l \geq 3 \). Section 4 shows how we avoid this obstacle restricting the class \( K_1 \).

3 Level-restricted Steiner trees

A Steiner tree is called \( l \)-restricted if the level of its full components does not exceed \( l \). \( \text{smt}_l \) and \( \text{smt}_l \) denote the minimal \( l \)-restricted Steiner tree and its cost, respectively. The following theorem bounds the approximation of minimal Steiner trees with minimal \( l \)-restricted trees.

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

\[
\text{smt}_l / \text{smt} \leq k^{1/4}.
\]
Proof. We will construct $Smt_i$ for every full component of $Smt$ separately, so we can assume, that $Smt$ is a full Steiner tree.

At first we introduce some denotations. Let $T = Smt$ and $v$ be its node. $\bar{v}$ denotes the set of all descendants of $v$ and $s(v)$ denotes 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

$$V_i = \{ v \in T, s(v) \geq \frac{1}{k} & s(v') < \frac{1}{l} \text{ for any } v' \in Son(v) \},$$

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

Let $T^l$ be a tree with the node set $V^l = \cup_{i=0}^l V_i$ 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 $u \neq v$, $u, v \in V_i$ (Fig. 1, Steiner tree edges are dotted).

![Figure 1: The $l$-restricted tree $T^l$ drawn from a full Steiner tree](image)

Let $u \in V_i$, $Son(u) = \{u_1, ..., u_{l=t(u)}\}$. For every $j = 1, ..., t$, denote $U_j = Son^l(u) \cap \overline{a_j}$, $d(u_j, u_j^*) = \max_{v \in U_j} d(u_j, v)$. Then

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

In this section we construct recursively the series of algorithms \{A_l, l = 1, 2, \ldots\}. For any \( l \), the algorithm \( A_1 \) is LRGH with the restricted subclass of \( K_l \), i.e., it approximates the minimal \( l \)-restricted Steiner tree. \( A_1 \) coincides with MSTH.

Since \( G_S \) has no cycles, \( \operatorname{Mst}(S) \) consists of the cheapest edges coming to \( S \)-nodes in \( G_S \). For any \( s \in S \), denote the cost of such edge by \( m(s) = \min_{s' \in S} d(s', s) \). So the output cost is \( M_0 = \sum_{s \in S} m(s) \).

\( A_2 \) coincides with LRGH.

Our goal is computing of Step (a) of GCF for the function (1).

We need the following denotations. Let \( v \in V - S \), \( d_0 = \min_{s \in S} d(s, v) = d(s_0, v) \). \( S(v) \) 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 \( \frac{d_i}{m_i} \leq \frac{d_{i+1}}{m_{i+1}} \).

The class \( K = K_2 \) consists of 2-level full Steiner trees. Every such tree is determined by its root, unique internal node \( v \in V - S \) and leaves (Fig. 2, MST-edges are dotted).

The following lemma makes possible computing of \( \min_{T \in K_2} f(T) \).
Figure 2: Minimum spanning and 2-restricted Steiner trees

Lemma 1 For any \( v \in V - S \),

\[
\min_{T \ni v} f(T) = \min \frac{\sum_{j=0}^{d} d_j}{\sum_{i=1}^{m} m_i}
\]

Proof. Let \( T^* = \arg \min_{v \in T} f(T) \), and \( \{(s_0^*, v), (v, s_1), \ldots, (v, s_t^*)\} \) be its edges. We can rewrite (1) as follows

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

We may replace \( s_0^* \) by \( s_0 \) in \( T^* \) without increasing \( f \) since \( d(s_0, v) \leq d(s_0^*, v) \). Let \( s \in S(v) \) and

\[
\frac{d(v, s)}{m(s)} \leq \frac{d(v, s^*)}{m(s^*)} = \max_{i=1, \ldots, t^*} \frac{d(v, s_i^*)}{m(s_i^*)}
\]

To prove Lemma we will show that \( f(T^* \cup (v, s)) \leq f(T^*) \). Indeed,

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

since \( T^* \) minimizes \( f \). Therefore, \( \frac{d(v, s^*)}{m(s^*)} \leq f(T^*) \). Thus, inequality (5) yields

\[
f(T^* \cup (v, s)) = \frac{d(T^*) + d(v, s)}{\sum_{i=1}^{t^*} m(s_i^*) + m(s)} \leq \frac{d(T)}{\sum_{i=1}^{t^*} m(s_i^*)} = f(T^*)
\]

The algorithms \( A_i \), \( i \geq 3 \).
As mentioned above, we cannot exactly find $\arg\min_{K_i} f(T)$. So we are looking for a minimum of $f$ in a subclass of $K_i$ defined below.

We define a tree $T_i(u), l \geq 2$, recursively. For any $u \in V$, $T_1(u) = \arg\min f(T)$. For any $s \in S$, denote by $V(s)$ the Voronoi region of $s$, i.e. $V_S(s) = \{v \in V, s = \arg\min_{s \in S} d(s,v)\}$. To determine $T_i(u), l \geq 3$, we use the following

**Procedure**

$T_i(u) \leftarrow (s_0, u) = \arg\min_{s \in S} d(s,u)$;

$f(T_i(u)) \leftarrow \infty$;

$S \leftarrow S \cup u$;

repeat forever

$v^* \leftarrow \arg\min_{v \in V(u)} f(T_{i-1}(v))$

if $f(T_i(u) \cup T_{i-1}(v^*)) \geq f(T_i(u))$, then exit repeat

$T_i(u) \leftarrow T_i(u) \cup T_{i-1}(v^*)$

contract $T_{i-1}(v^*)$

**Remark 2** $f(T_i(v^*)) \leq 1$.

Indeed, for a non-zero 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) $u^* \leftarrow \arg\min_{u \in V} f(T_1(u))$.

(b) insert $T_l(u^*)$ in LIST.

(c) contract $T_l(u^*)$, $S \leftarrow S/T_l(u^*)$

(2) reconstruct an output Steiner tree from trees of LIST.

Now we will estimate time complexity of algorithms $A_l$. For brevity, the sets and its cardinalities will have the same denotations.

To compute the graph $G_S$ we need an $O(E)$-breadth-first-search. $Mst(S)$ can be obviously found in time $O(E)$. Thus, $A_1$ runs in time $O(E)$.

For $A_2$, we need to know all distances between $V - S$ and $S$ ($O(V^2)$), then in time $O(S)$ we can find $T_2(u)$ for any $u \in V - S$ (Lemma 1). Thus, the total runtime of $A_2$ is $O(V^2 + S^2V)$.

For the case $i \geq 3$, we may find all pairs shortest paths for the input graph $G$ ($O(V^2)$). A runtime of Procedure $T_l(u)$ is $rt_l = O((VS)^{l-1})$, since $rt_1 = rt_{l-1}VS$ and $rt_2 = O(VS)$. Thus, $A_l$ has a runtime $O(V^{l-1}S^l)$. 
5 The Performance Guarantee

Our first goal is to show that the minimum of the function \( f \) in the item \( (a) \) of Algorithm \( A_i \) is not far from the minimum in the whole class \( K_i \). In other words, we generalize Lemma 1 for arbitrary \( l \).

Lemma 2 Let \( T_i^* = \text{arg min}_{K_i} f(T) \) and \( v^* \) be the unique son of a root of \( T_i^* \). Then

\[
f(T_i(v^*)) \leq f(T_i^*)(2 + \log k)^{l-2}
\]

(6)

Proof. Induction on \( l \). The case of \( l = 2 \) follows from Lemma 1. Denote \( c_l = (2 + \log k)^{l-2} \) and \( S^* = S \cap T_i^* \).

At first we consider the case of \( S \cap T_i(v^*) \neq S^* \). Consider ADSP for \( S^* \) with a root \( v^* \). In above denotations, let \( S_{\text{mt}}_{l-1} \) and \( S_{\text{mt}}_{l-1} \) be the minimal \( l - 1 \)-restricted Steiner tree and its cost, \( s_{l-1} = S_{\text{mt}}_{l-1} c_{l-1} \) and \( M_0 = M_0(S^*) \) be the cost of the minimal spanning tree \( \text{Mst}(S^*) \). Let \( d_1 = d(T_{l-1}) \), \( M_{l-1} = M_0(S^*/T_{l-1}) \) and \( m_1 = M_0 - M_1 \). By induction \( f(T_{l-1}) \leq f(T_{l-1}) c_{l-1} \). Since \( f(T_{l-1}) \leq f(S_{\text{mt}}_{l-1}) \), we obtain \( \frac{d_{l-1}}{m_1} \leq \frac{s_{l-1}}{M_0 - M_{l-1}} \). After contraction of \( T_{l-1} = T_{l-1}^* \) the procedure finds \( T_{l-1}^2, T_{l-1}^3 \) and so on. Denote their corresponding values by \( d_i, M_i \) and \( m_i \). By induction \( \frac{d_i}{m_i} \leq \frac{s_{l-1}}{M_0 - M_{l-1}} \) and, therefore

\[
M_i \leq M_{i-1}(1 - \frac{d_i}{s_{l-1}})
\]

(7)

Now we apply an analysis technique due to Leighton and Rao \[8\] to prove two following inequalities. Unraveling (7), we obtain

\[
M_r \leq M_0 \prod_{i=1}^{r} (1 - \frac{d_i}{s_{l-1}}).
\]

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

\[
\ln \frac{M_0}{M_r} \geq \frac{\sum_{i=1}^{r} d_i}{s_{l-1}}
\]

The procedure interrupts when \( f \) begins to increase. But we will continue it until \( M_r \geq s_{l-1} \geq M_{r+1} \). So \( M_{r+1} \leq s_{l-1} \). The inequality \( \frac{d_{r+1}}{s_{l-1}} \leq \frac{s_{l-1}}{M_{r+1}} \) implies \( \frac{s_{l-1}}{M_{r+1}} \leq \frac{m_{r+1}}{M_{r+1}} \leq 1 \). Therefore, the inequality \( M_0 \leq k \cdot s_{l-1} \) yields

\[
\frac{\sum_{i=1}^{r} d_i + M_{r+1} + d_{r+1}}{s_{l-1}} \leq 2 + \ln k
\]

(8)

Inequality (8) holds, since Remark 2 allows us to assume that \( f(T_i^*) c_i < 1 \). The inequality (6) follows from the following series of inequalities:

\[
f(T_i(v^*)) = \frac{d(T_i(v^*))}{m(T_i(v^*))} \leq \frac{d(T_i(v^*))}{m(T_i(v^*))} \leq (2 + \log k)^{l-2}
\]
\[
\frac{d_0 + \sum_{i=1}^{r+1} d_i}{M_0 - M_{r+1}} \leq \frac{d_0 + \sum_{i=1}^{r+1} d_i + M_{r+1}}{M_0} \leq \frac{d_0 + \text{smt}_{i-1} c_i}{M_0} \leq \frac{(d_0 + \text{smt}_{i-1}) c_i}{M_0} = f(T_i^*) c_i.
\]

Inequality (9) holds, since both its sides are less than 1. Thus, we proved (6) in the case of \( S \cap T_i(v^*) \subseteq S^* \).

Now we turn to the case of an arbitrary set \( S \cap T_i(v^*) \). We partition \( m_i \) of the tree \( T_{i-1}^* \) into two parts \( m_i = m_i^* + \overline{m}_i \), where the first part is the sum of costs of edges coming to \( S^* \)-vertices of \( T_{i-1}^* \) and the second is the sum of costs of edges coming to the rest of \( S \)-vertices of \( T_{i-1}^* \) in the tree \( \text{Mst}(S) \). We also partition \( d_i = d_i^* + \overline{d}_i \) in the same proportion as \( m_i \), i.e. \( \frac{d_i^*}{m_i^*} = \frac{\overline{d}_i}{\overline{m}_i} \). Assign \( \overline{d}_i \leftarrow 0 \) if \( \overline{m}_i = 0 \), and \( d_i^* \leftarrow 0 \) if \( m_i^* = 0 \).

The condition of the procedure interruption implies
\[
f(T_i(v^*)) \leq f(T_i(v^*) - T_i^{r+1}(v^*)),
\]
\[
f(T_i(v^*)) \geq f(T_i^{r+1}(v^*)) = \frac{\overline{d}_i + 1}{m_{r+1}} \text{Thus,}
\]
\[
f(T_i(v^*)) = \frac{d(T_i(v^*))}{m(T_i(v^*))} \leq \frac{d_0 + \sum_{i=1}^{r+1} d_i^* + \sum_{i=1}^{r+1} \overline{d}_i}{\sum_{i=1}^{r+1} m_i^* + \sum_{i=1}^{r+1} \overline{m}_i} \leq \frac{d_0 + \sum_{i=1}^{r+1} d_i^*}{\sum_{i=1}^{r+1} m_i^*} \leq \ldots \leq \frac{d_0 + \sum_{i=1}^{r+1} d_i^*}{\sum_{i=1}^{r+1} m_i^*} \leq f(T_i^*) c_i \diamond
\]

Note, that the previous argument for the case of \( S \cap T_i(v^*) \subseteq S^* \) 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,
\[
f(T_i(v^*)) \leq \frac{d_0 + \sum_{i=1}^{r+1} d_i^*}{\sum_{i=1}^{r+1} m_i^*} \leq f(T_i^*) c_i \diamond
\]

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

**Proof of Theorem 1.**

Let \( T \) be the output tree of Algorithm \( A_i \) and \( T_1^1, T_1^2, T_1^3 \) be the trees inserted in \( \text{LIST} \). Denote \( d(T_1^i) = d_i \), \( M_i = M_{i-1} - m_i \), where \( m_i \) is the sum of costs of edges coming to \( S \)-vertices of \( T_1^i \) in the tree \( \text{Mst}(S) \). As above, \( c_i = (2 + \ln k)^{l-2} \), \( s_i = \text{smt}(c_i) \).

Note that \( f(T_i^*) \leq \text{smt}(c_i) / M_0 \). By Lemma 2, \( d_i / m_i \leq f(T_i^*) c_i \leq s_i / M_0 \). Inductively,
\[
\frac{d_i}{m_i} \leq \frac{s_i}{M_{i-1}}.
\]
Now we are ready to use the same argument as for the proof of Inequality (8) to obtain for $M_r \geq s_l \geq M_{r+1}$ the following inequality:

$$\sum_{i=1}^{r+1} d_i + M_{r+1} + d_{r+1} \leq 2 + \ln k$$

This implies that

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

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

We omit here a slight improvement of the previous analysis which leads to the bound claimed in Theorem 1.

Now we will find the limit performance guarantee of $\{A_i\}$. Denote the performance guarantee of $A_i$ by $f_i(k) = k^\frac{1}{r}(1 + \log k)^{l-1}$. We need to find $f(k) = \min_i f_i(k)$. Taking natural logarithm of $f_i(k)$ and derivative, we obtain

$$\ln f(k) = \frac{-\ln k}{r} + \ln (\ln k + 1) = 0 \quad (10)$$

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

$$\ln f(k) = 1 \sqrt{\ln k \ln (\ln k + 1)} - \ln (\ln k + 1) \quad \Box$$

References


