CCIT Report #435
July 2003

Approximation Schemes for Multicast QoS Routing with Guaranteed Performance

Ariel Orda *Alexander Sprintson*
Department of Electrical Engineering
Technion—Israel Institute of Technology
Haifa 32000, Israel

Abstract

Emerging group applications require efficient multicast schemes that provide Quality of Service (QoS) guarantees. QoS can be achieved by provisioning multicast trees that satisfy QoS constraints. Since the efficient usage of network resources is an important requirement, the cost of the constructed multicast tree should be as small as possible. Accordingly, in this study we investigate the fundamental problem of finding a multicast tree that satisfies end-to-end QoS constraints at minimum cost.

This problem has been extensively studied. However, existing solutions have either relied on heuristic approaches or considered special cases, such as the case where the delay and cost of each link are identical. Moreover, many of the heuristic approaches are based on restricting assumptions, such as symmetric link delays. In this study we propose a novel algorithmic scheme, with proven performance guarantees, for this fundamental multicast problem. Effectively, this scheme allows to obtain an approximate solution to this problem out of any given approximate scheme of its (simpler) unconstrained directed version, with about identical (ε-close) performance guarantees.

Keywords– Multicast, Routing, Quality of Service, Approximation Algorithms.

Multicast is an important network mechanism that allows simultaneous transmission of data to multiple destinations with minimal bandwidth consumption. In order to support new applications such as multimedia streaming and video conferencing, multicast mechanisms are expected to provide a certain degree of Quality of Service. A fundamental problem in this context is to identify multicast trees that satisfy end-to-end QoS constraints at minimum cost. Since *bottleneck* QoS constraints, such as bandwidth, can be efficiently handled by pruning infeasible links, we focus on *additive* QoS constraints, such as delay or jitter, which are much more difficult to handle.

Finding multicast trees that support additive QoS constraints is an intractable problem, as it contains the Minimum Steiner Tree (MST) and Restricted Shortest Path (RSP) problems, each known to be $\mathcal{NP}$-hard [9]. Essentially, MST is a special case of our problem with no QoS constraints, whereas RSP is the special case of unicast. The first problem, MST, has been extensively investigated for undirected networks, and several efficient solutions, of constant approximation ratios, have been established (see, *e.g.*, [12]). For *directed* networks, the only general solution was recently established in [5]. The second problem, RSP, has been the subject of several studies [6, 10, 11, 15], which proposed efficient approximation schemes. In particular, several efficient algorithms have been proposed for computing a path that satisfies the delay constraint and whose cost is at most $(1 + \varepsilon)$ times higher than the optimum.

The problem that we consider, namely establishment of efficient QoS multicast routing schemes, has attracted a large body of research (see *e.g.*, [1, 7, 13, 19–21, 23, 25] and references therein). A good survey of multicast routing protocols and their QoS extensions can be found in [22]. Many of these studies employed heuristic approaches [1, 7, 13, 19, 21, 25]. Moreover, these heuristics were often based on restricting assumptions, such as a symmetry of link delays [7, 13, 19, 25]. *Provable* approximate solutions have been proposed, however they either considered restricted special cases, or else incurred a potentially large violation of the QoS constraint. For example, [14] effectively deals with our problem, however in the special case of *identical link delays*. As another example, [23] also dealt with our problem, however in the special case where the *delay* and *cost* of each link are *identical*. Obviously, such simplified assumptions do not hold in many practical settings. In [16], a provable approximation to our problem has been presented, however it allows a *violation* of the delay constraint by a factor as large as $\log N$, where N is the number of nodes; moreover, it assumes *symmetric* links. Hence, to the best of our knowledge, no solution of provable performance has been established to this fundamental multicast problem for *general* networks and *no violation* of the QoS constraints. Accordingly, in this study, we propose novel schemes for general directed networks that achieve, for any fixed $i > 0$ and $\varepsilon > 0$, an approximation ratio of $(1 + \varepsilon)i(i - 1)K^{1/i}$, where K is the number of terminals, *i.e.*, our schemes find a tree that satisfies the QoS constraints and whose cost is at most $(1 + \varepsilon)i(i - 1)K^{1/i}$ times higher than the optimum. For example, for a typical value $K = 100$, choosing $i = 2$ and $\varepsilon = 0.05$ yields a worst-case guaranteed ratio of 21. For an asymptotically large number of terminals, the approximation ratio is upper-bounded by $\log^2 K$. A proven lower bound for this problem is $\ln K$ [2]. Although the ideas that led to development of the schemes as well as performance proofs are rather involved, the schemes itself are relatively simple and easy to implement.

Due to the fundamental nature of the considered problem, our results can by used in a variety of practical applications. Indeed, any multicast architecture that provides a certain degree of Quality of Service requires efficient schemes for identification of QoS trees. For example, the ATM PNNI [17] protocol supports point-to-multipoint calls that satisfy QoS constraints specified by connection requests. Due to the connection-oriented nature of ATM, bandwidth is reserved along the tree's links for a long period of time. This implies the need to select "good" trees in terms of resource utilization. The IntServ/RSVP framework relies on underlying IP routing protocols for selecting paths and trees. Since the selection of an unsuitable tree results incurs major overhead, it would be desirable to provide a tree that satisfies the QoS constraints in the first place. In the Differential Services framework [3, 4], a *bandwidth broker* is responsible for establishing suitable trees that satisfy service level agreements (SLA). Here two, it is desirable that bandwidth brokers be capable of computing the QoS trees. Finally, the MPLS architecture [18] is expected to supports data forwarding for multicast traffic, which requires establishing suitable QoS trees.

The rest of this paper is organized as follows. In Section 1, we formally state the considered problems. In Section 2, we briefly describe the algorithm of [5] for the MST problem in directed networks, which serves as a basic building block in our solution. Next, in Section 3, we present the first approximation scheme, which, while conceptually simple, incurs a high computational complexity. Accordingly, in Sections 4 and 5 we present two additional schemes, whose computational complexity is reasonable. Finally, we discuss our results in Section 6.

1 Model and Problem Formulation

This section formulates the general model and main problems addressed in this paper.

1.1 Basic definitions

A *network* is represented by a directed graph $G(V, E)$, where V is the set of nodes and E is the set of links. Let $N = |V|$ and $M = |E|$. A *path* is a finite sequence of nodes $\mathcal{P} = \{v_0, v_1, \cdots, v_h\}$, such that, for $0 \leq i \leq h-1$, $(v_i, v_{i+1}) \in E$; $h = |\mathcal{P}|$ is then said to be the *number of hops* (*or hop count)* of $\mathcal{P}$. A *cycle* is a path whose source and destination nodes are identical.

A *directed tree* is a subgraph $\mathcal{T}$ of $G(V, E)$ with a source node s such that every node is reached from s by a unique path. A *multicast connection* uses a tree $\mathcal{T}$ to interconnect the source s and the members of a multicast group $X = \{t_1, t_2, \cdots, t_K\}$. Given a tree $\mathcal{T}$, a path between the source s and a node $v \in \mathcal{T}$ on links that belong to $\mathcal{T}$ is denoted by $\mathcal{P}_{(\mathcal{T},v)}$.

As mentioned in the Introduction, bottleneck QoS constraints can be easily handled by pruning all links that do not satisfy the constraints. Hence we focus on additive QoS constraints such as delay or jitter. For clarity of presentation and without loss of generality, we describe our model and problems in terms of end-to-end *delay* requirements.

We assume that each link offers a delay guarantee d_l. The delay $D(\mathcal{P})$ of a path $\mathcal{P}$ is the sum of delays of its links, *i.e.*, $D(\mathcal{P}) = \sum_{l \in \mathcal{P}} d_l$. Each link is also associated with a nonnegative cost c_l, which estimates the amount of network resources consumed in order to support delay constraint d_l. The link cost may depend on various factors, *e.g.*, the link available bandwidth and its location. The cost $C(\mathcal{P})$ of a path $\mathcal{P}$ is defined to be the sum of the costs of its links, *i.e.*, $C(\mathcal{P}) = \sum_{l \in \mathcal{P}} c_l$. Similarly, the cost $C(\mathcal{T})$ of a tree $\mathcal{T}$ is $C(\mathcal{T}) = \sum_{l \in \mathcal{T}} c_l$. We assume that all parameters (cost and delays) are (non-negative) integers.

Definition 1 (Transitive closure) *The* transitive closure $\hat{G}$ *of* G *is a graph that includes, for each pair of nodes* u *and* v *in* G*, a link* (u, v) *such that* $c_{(u,v)}$ *is the minimum cost of a path between* u *and* v *in* G*.*

Definition 2 (i-level tree) *Let* $\mathcal{T}$ *be a tree with source* s*.* $\mathcal{T}$ *is said to be an* i-level tree *if for, each node* $v \in \mathcal{T}$*, it holds that the path between* s *and* v *in* $\mathcal{T}$ *includes at most* i *links.*

1.2 Problem formulation

We are now ready to formulate the problem considered in this study.

Problem RST (Restricted Steiner Tree) *Given a graph G, a source s, a set of K terminals $X = \{t_1, \cdots, t_K\}$ and a delay constraint D, find a minimum cost tree $\mathcal{T}$ that connects s to each terminal $t_j \in X$ and satisfies the delay constraint D,* i.e.*, for each $t_j \in X$ it holds that $D(\mathcal{P}_{(\mathcal{T},t_j)}) \leq D$.*

An instance (G, s, X, D) of Problem DST is denoted by I_X. We denote by $OPT(I_X)$ the cost of an optimal solution for an instance I_X.

Clearly, Problem RST is intractable, as it contains the Minimum Steiner Tree (MST) and Restricted Shortest Path (RSP) problems, which are known to be $\mathcal{NP}$-hard [9]. Moreover, it is hard to approximate Problem RST to a factor better than $\ln K$ [2]. The bound holds for both directed and undirected graphs.

1.3 Related problems

Our approximation scheme for Problem RST employs a reduction to the following problem of finding an (unconstrained) Steiner tree in directed graphs.

Problem DST (Directed Steiner Tree) *Given a directed graph G, a source s and a set of K terminals $X = \{t_1, \cdots, t_K\}$, find a minimum cost tree $\mathcal{T} \in G$ that connects s and each terminal $t_j \in X$.*

An instance (G, s, X) of Problem DST is denoted by I'_X. We denote by $OPT(I'_X)$ the cost of an optimal solution for instance I'_X. We also denote by $OPT^{(i)}(I'_X)$ the cost of the optimum i-level tree that solves instance I'_X of Problem DST.

An approximation scheme for Problem DST was presented in [5]. That scheme computes, for any fixed i, a tree $\mathcal{T}$ that connects source s and terminals X and satisfies $C(\mathcal{T}) \leq i(i-1)K^{1/i}OPT(I'_X)$. The computational complexity of that scheme is $\mathcal{O}(N^{i-1}K^{2i-1})$.

Another related problem is to find a minimum cost (unicast) path that satisfies a given delay constraint. In effect, this is the *Restricted Shortest Path* problem, defined as follows.

Problem RSP (Restricted Shortest Path) *Given a source node s, a destination node t and a delay constraint D, find a minimum cost path $\mathcal{P}$ between s and t such that $D(\mathcal{P}) \leq D$.*

Several efficient approximation schemes have been proposed for this problem. In particular, [15] presented an algorithm that computes, in $\mathcal{O}(MN(\frac{1}{\varepsilon} + \log\log N))$ time, a path that satisfies the delay constraint D and whose cost is at most $(1+\varepsilon)$ times higher than the optimum; we refer to this solution as Algorithm RSP-1. Taking a somewhat different approach, [10] proposed to alleviate the delay constraints and presented an algorithm that computes, for each node $v \in V$, a path between s and v such that $D(\mathcal{P}) \leq (1+\varepsilon)D$ and $C(\mathcal{P}) \leq OPT$, where OPT is the cost of an optimal path between s and v that satisfies the delay constraint D. The computation complexity of this algorithm is $\mathcal{O}(\frac{1}{\varepsilon}(M + N\log N)N)$; we refer to this solution as Algorithm RSP-2.

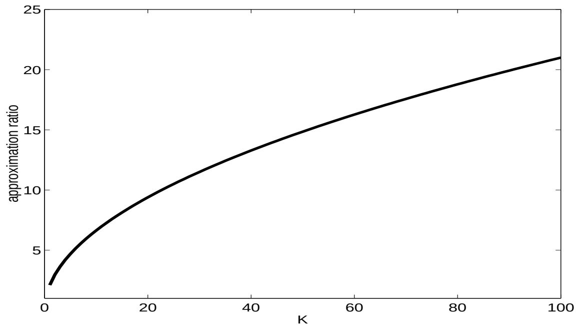


Figure 1: Approximation ratio as a function of the number of terminals.

1.4 Our results

Since the considered problem is $\mathcal{NP}$-hard, we focus on (provable) approximate solutions. We present three approximation schemes for Problem RST. The first scheme provides a solution whose cost is at most $i(i-1)K^{1/i}$ times higher than the optimum, for any integer $i > 0$. The computational complexity of the scheme depends on the values of the link delays, hence can be prohibitively high. The complexities of the second and third schemes are much lower, and do not depend on delay values. The second scheme allows a small violation of the delay constraint. More specifically, it computes a tree $\mathcal{T}$ such that, for each $t_j \in X$, it holds that $D(\mathcal{P}_{(\mathcal{T},t_j)}) \leq (1+\varepsilon)D$ and $C(\mathcal{T}) \leq (1+\varepsilon)i(i-1)K^{1/i}OPT(I_X)$, for any $\varepsilon > 0$ and integer $i > 0$. The third scheme provides a solution that does not violate the delay constraint and whose cost is at most $(1+\varepsilon)^2 i(i-1)K^{1/i}$ times higher than the optimum, for any $0 < \varepsilon \leq 1$ and integer $i > 0$. The computational complexity of the second and third schemes are $\mathcal{O}((\frac{i\cdot N}{\varepsilon})^{i-1} K^{2i-1})$ and $\mathcal{O}((\log\log N + \log\frac{1}{\varepsilon})(\frac{N}{\varepsilon})^{i-1} K^{3i-2})$, respectively.

Fig. 1 shows the approximation ratios that can be achieved by our schemes for typical values of K, namely $2 \leq K \leq 100$. For these values, choosing $i = 2$ and $\varepsilon = 0.05$ yields a worst-case guaranteed ratio of 21. For large values of K, namely up to $K = 10^4$, an optimal value of i is at most 4 and the approximation ratio is close to $\log^2 K$. We note that for this range, a choice of $i = 3$ allows to achieve a relatively close approximation ratio at significantly lower computational complexity. Finally, we note that, for $K \to \infty$, one can achieve an approximation ratio of $\log^2 K$.

Our results are summarized in the following table.

Scheme	Delay Violaltion	Approximation Ratio	Complexity
1	none	$i(i-1)K^{1/i}$	(high)
2	$1+\varepsilon$	$(1+\varepsilon)i(i-1)K^{1/i}$	$\mathcal{O}((\frac{i\cdot N}{\varepsilon})^{i-1} K^{2i-1})$
3	none	$(1+\varepsilon)i(i-1)K^{1/i}$	$\mathcal{O}((\log\log N + \log\frac{1}{\varepsilon})\cdot (\frac{N}{\varepsilon})^{i-1} K^{3i-2})$

2 Preliminaries: Approximation Scheme for Problem DST

Previous studies [5, 14] have pointed out that problems RST and DST are closely related. We exploit this relation by constructing a reduction from Problem RST to Problem DST. Then, we solve Problem DST by using the algorithm presented in [5]. In this section we briefly describe the algorithm of [5].

The algorithm, referred to as Algorithm DST, uses the notion of *density* of a multicast tree, which is defined to be the ratio of the tree cost to the number of terminals.

Algorithm DST comprises of a recursive procedure $\mathrm{A}_i(K, r, Y)$, which identifies an i-level tree $\mathcal{T}$ that connects node r with at least K terminals $Y' \subseteq Y$. More specifically, Procedure $\mathrm{A}_1(K, r, Y)$ finds the K terminals which are closest to the root and connects them to the root by using shortest paths; for $i > 1$, Procedure $\mathrm{A}_i(K, r, Y)$ repeatedly finds a node v and a number $K', 1 \leq K' \leq K$ such that the density of the tree $\mathcal{T}_{i-1}(K', v, Y) \cup \{(r, v)\}$ is the minimal among all trees of this form, where $\mathcal{T}_{i-1}(K', v, Y)$ is the tree returned by the invocation of Procedure A_{i-1} for (K', v, Y). The detailed description of Algorithm DST appears in Fig. 2.

Algorithm DST ($I'_X(G, s, X), i$**)**
input:
I'_X - an instance of Problem DST
G - the graph,
s- source node.
$X = \{t_1, \cdots, t_K\}$ - the set of K terminals.
i- the level of the returned tree.
output:
$\mathcal{T}$- A solution to I_X.

$\mathcal{T} \leftarrow \mathrm{A}_i(K, s, X)$
return $\mathcal{T}$

Procedure $\mathrm{A}_i(K, r, Y)$
if there do not exist K terminals reachable from r **then**
  return $\emptyset$
$\mathcal{T} \leftarrow \emptyset$
while $K > 0$
  $\mathcal{T}_{\mathrm{BEST}} \leftarrow \emptyset$; $K_{\mathrm{BEST}} \leftarrow 0$
  for each link $(r, v) \in G$ and each $K', 1 \leq K' \leq K$ **do**
    $\mathcal{T}' \leftarrow \mathrm{A}_{i-1}(K', v, Y) \cup \{(r, v)\}$
    if $\frac{C(\mathcal{T}_{\mathrm{BEST}})}{K_{\mathrm{BEST}}} \geq \frac{C(\mathcal{T}')}{K'}$ **then**
      $\mathcal{T}_{\mathrm{BEST}} \leftarrow \mathcal{T}'$; $K_{\mathrm{BEST}} \leftarrow K'$
  $Y' \leftarrow \{t_j \mid t_j \in Y \wedge t_j \in \mathcal{T}_{\mathrm{BEST}}\}$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_{\mathrm{BEST}}$
  $Y \leftarrow Y \setminus Y'$
  $K \leftarrow |Y|$
return $\mathcal{T}$

Figure 2: Algorithm DST

The following theorem was proven in [5].

Theorem 1 [5] *Given an instance* $I'_X = (G, s, X)$ *of Problem DST and an integer* $i, 1 \leq i \leq \log |X|$*, Algorithm* DST *returns a tree* $\mathcal{T}$ *that satisfies* $C(\mathcal{T}) \leq i(i-1)K^{1/i} OPT(I'_X)$.

By Theorem 1, Algorithm DST returns a tree whose cost is at most $i(i-1)K^{1/i}$ times more than the optimum. For our purposes, we shall need a more general version of Theorem 1. In particular, we prove that the algorithm returns a tree whose cost is at most $i(i-1)K^{1/i}\hat{C}$, provided that $\hat{C}$ satisfies the following condition: for each subset Y of X, it holds that $OPT^{(i)}(I'_Y(G, s, Y)) \leq |Y|^{1/i}\hat{C}$. We note that, for certain instances I'_X of Problem DST, it might be the case that $\hat{C} \leq OPT(I'_X)$.

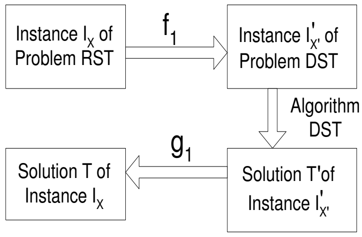


Figure 3: Reduction from Problem RST to DST

Theorem 2 *Given an instance* $I'_X = (G, s, X)$ *of Problem DST and an integer* $i, 1 \le i \le \log|X|$. *Let* $\hat{C}$ *be the minimum cost such that for each subset* Y *of* X *holds* $OPT^{(i)}(I'_Y(G, s, Y)) \le |Y|^{1/i}\hat{C}$, i.e.,

$$\hat{C} = \max_{Y \subseteq X} \left\{ \frac{OPT^{(i)}(I'_Y(G, s, Y))}{|Y|^{1/i}} \right\}.$$

Then, Algorithm DST *returns a tree* $\mathcal{T}$ *that satisfies* $C(\mathcal{T}) \le i(i-1)K^{1/i}\hat{C}$.

Proof: See Appendix. ■

Theorem 3 *Let* n *be the maximum number of links that originate from a node in* G. *Then, the computation complexity of Algorithm* DST *is* $\mathcal{O}(n^{i-1}K^{2i-1})$.

Proof: The computational complexity of procedure A_i for $i = 1$ is $\mathcal{O}(K)$. Note also that Procedure A_i invokes Procedure A_{i-1} at most nK^2 times. Hence, the computational complexity of Procedure A_i and, in turn, Algorithm DST is $\mathcal{O}(n^{i-1}K^{2i-1})$. ■

3 First Approximation Scheme: simple but inefficient

In this section we present the first approximation scheme, which, while conceptually simple, incurs a high computational complexity.

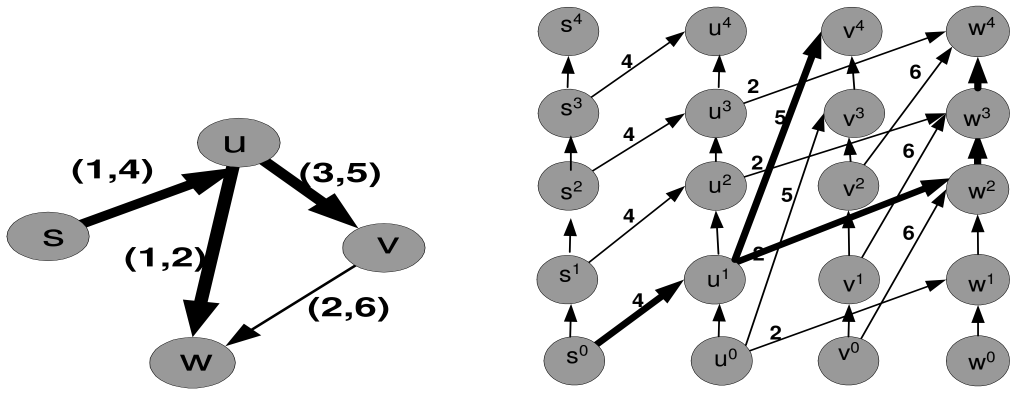
3.1 T-Reductions

We begin by introducing the concept of T-*Reductions*, which allow to establish an approximation scheme for Problem RST out of any given approximation scheme for Problem DST. In this section we describe T_1-reductions, while T_2- and T_3-reductions are introduced in sections 4 and 5, respectively.

Definition 3 (T_1-reduction) *A* T_1-reduction *from Problem RST to Problem DST is a duple* (f_1, g_1) *that satisfies the following:*

- *The function* f_1 *maps an instance* $I_X(G, s, X, D)$ *of Problem RST to an instance* $I'_{X'}(G', s', X')$ *of Problem DST such that* $OPT(I'_{X'}) \le OPT(I_X)$*;*
- *The function* g_1 *maps a solution* $\mathcal{T}'$ *of* $I'_{X'}$ *to solution* $\mathcal{T}$ *of* I_X *such that* $C(\mathcal{T}) = C(\mathcal{T}')$.

As we show below, a T_1-reduction (f_1, g_1) gives rise to an approximation scheme for Problem RST.



(a) Original graph G. Associated with each link are its delay and cost

(b) Layers Graph $\hat{L}_1$ (some of the links are omitted)

Figure 4: Construction of Layers Graph $\hat{L}_1$

3.2 Layers Graph

We proceed by presenting a structure termed *Layers Graph*, which allows to establish a T_1-reduction (f_1, g_1) from Problem RST to Problem DST.

The purpose of the Layers Graph, denoted by $\hat{L}_1$, is to distinguish between trees that connect the source s to the terminals X and also satisfy the QoS constraint D and all other trees in G. Specifically, the layer graph $\hat{L}_1$ is constructed as follows. First, we compute, for each two nodes u and v and each delay constraint d, $1 \leq d \leq D$, a minimum cost path $\mathcal{P}^d_{(u,v)}$ between u and v whose delay is at most d. Next, for each node $v \in G$, we add $D+1$ nodes $v^0, \cdots, v^D$ to $\hat{L}_1$. For each $v \in G$ and each d, $0 \leq d \leq D-1$, we add to $\hat{L}_1$ a link (v^d, v^{d+1}) whose cost is 0. Next, for each two nodes, u^d and v^j, such that $j > d$, we add to $\hat{L}_1$ a link (u^d, v^j) whose cost is $c_{(u^d,v^j)} = C(\mathcal{P}^{j-d}_{(u,v)})$. Fig. 4 depicts an example of a Layers Graph.

Consider a tree $\mathcal{T}$ in G that connects s and the terminals $X = \{t_1, \cdots, t_K\}$, and, in addition, satisfies the delay constraint D. We show that there exists a corresponding tree $\mathcal{T}' \in \hat{L}_1$ that connects s^0 and $X' = \{t_1^D, \cdots, t_K^D\}$, such that $C(\mathcal{T}) = C(\mathcal{T}')$. The tree $\mathcal{T}'$ is defined recursively, starting with node s^0. First, for each node $v \in \mathcal{T}$, compute the delay d_v of the path that between s and v in $\mathcal{T}$, *i.e.*, $d_v = D(\mathcal{P}_{(\mathcal{T},v)})$. Next, each link $l(s,v) \in \mathcal{T}$, we add to $\mathcal{T}'$ a link (s^0, v^{d_v}). Next, we grow the tree from each node v^{d_v}: for each link $(v,u) \in \mathcal{T}$ we add to $\mathcal{T}'$ a link (v^{d_v}, u^{d_u}). Next, we proceed to grow the tree from node u^{d_u} an so on. The process ends with a tree $\mathcal{T}'$ that connects s and nodes $\{t_1^{d_{t_1}}, \cdots, t_K^{d_{t_K}}\}$. Note that, for each $t_j \in X$, it holds that $d_{t_j} \leq D$. We then use links (t_j^d, t_j^{d+1}) of zero cost in order to construct a tree $\mathcal{T}'$ that connects s and the terminals $X' = \{t_1^D, \cdots, t_K^D\}$. For example, consider the tree $\mathcal{T} = \{(s,u),(u,v),(u,w)\}$ in Fig. 4(a). The corresponding tree $\mathcal{T}' = \{(s^0,u^1),(u^1,v^4),(u^1,w^2),(w^2,w^3),(w^3,w^4)\}$ in the Layers Graph $\hat{L}_1$ is marked by bold lines in Fig. 4(b).

Similarly, it can be shown that, for each tree $\mathcal{T}'$ in $\hat{L}_1$ that connects s^0 and $X' = \{t_1^D, \cdots, t_K^D\}$, there exists a tree $\mathcal{T}$ in G that connect s and the terminals X, such that $\mathcal{T}$ satisfies the delay constraint D and $C(\mathcal{T}) = C(\mathcal{T}')$.

The reduction (f_1, g_1) is then defined as follows.

Definition 4 (Reduction(f_1, g_1)) *A* reduction(f_1, g_1) *is a pair of functions* f_1, g_1 *such that:*

1. *The function* f_1 *gets as input an instance* $I_X = (G, s, X, D)$ *of Problem RST and returns an instance* $I'_{X'} = (\hat{L}_1, s^0, X')$ *of Problem DST, where* $\hat{L}_1$ *is a Layer graph and* $X' = \{t_1^D, \cdots, t_K^D\}$.
2. *The function* g_1 *gets as input a solution* $\mathcal{T}'$ *of* $I'_{X'}$ *and returns a tree*

$$\mathcal{T} = \bigcup_{(u^d, v^j) \in \mathcal{T}'} \mathcal{P}_{(u,v)}^{j-d}, \tag{1}$$

where $\{\mathcal{P}_{(u,v)}^d\}$ *are paths computed during the construction of the Layers Graph* $\hat{L}_1$.

In the following lemmas we prove that the functions f_1 and g_1 constitute a valid T_1-reduction.

Lemma 1 *If* $I'_{X'} = f_1(I_X)$ *then* $OPT(I'_{X'}) \leq OPT(I_X)$.

Proof: Let $\mathcal{T}^{\text{opt}}$ be an optimal solution for instance I_X of Problem RST, *i.e.*, $C(\mathcal{T}^{\text{opt}}) = OPT(I_X)$. For each node $v \in \mathcal{T}^{\text{opt}}$, we denote by d_v the delay of the path between s and v in $\mathcal{T}^{\text{opt}}$. Let $\hat{\mathcal{T}} = \{(u^{d_u}, v^{d_v}) \mid (u, v) \in \mathcal{T}^{\text{opt}}\} \cup \{(t_j^d, t_j^{d+1}) \mid t_j \in \mathcal{T}^{\text{opt}},\ d_{t_j} \leq d \leq D - 1\}$. It is easy to verify that $\hat{\mathcal{T}}$ is a tree in $\hat{L}_1$ that connects source s_0 with terminals $X' = \{t_j^D \mid t_j \in X\}$ and it holds that $C(\mathcal{T}^{\text{opt}}) = C(\hat{\mathcal{T}})$. We conclude that $OPT(I'_{X'}) \leq OPT(I_X)$ and the lemma follows. ■

Lemma 2 *Let* $\mathcal{T}'$ *be a solution of instance* $I'_{X'}$ *of Problem DST. Then,* $\mathcal{T} = g_1(\mathcal{T}')$ *is a solution of instance* I_X *of Problem DST and it holds that* $C(\mathcal{T}) = C(\mathcal{T}')$.

Proof: According to the definition of g_1, $\mathcal{T}$ includes, for each link $l = (u^d, v^j)$ in $\mathcal{T}'$, a path $\mathcal{P}_{(u,v)}^{j-d}$ in G that connects nodes u and v and whose delay is at most $(j - d)$. Clearly, for each $t_j \in X$ it holds that $D(\mathcal{P}_{(\mathcal{T}, t_j)}) \leq D$, which implies that $\mathcal{T}$ is a solution of instance I_X. Since $C(\mathcal{P}_{(u,v)}^{j-d}) = c_l$ it follows that $C(\mathcal{T}) = C(\mathcal{T}')$. ■

3.3 Approximation Scheme

The T_1-reduction (f_1, g_1), gives rise to the corresponding approximation scheme for Problem RST. Specifically, given an instance I_X of Problem RST we compute an instance $I'_{X'}$ of Problem DST by invoking function f_1. Next, we find a solution $\mathcal{T}'$ of $I'_{X'}$ by applying Algorithm DST. Finally, we identify a solution $\mathcal{T}$ of I_X by invoking function g_1 on $\mathcal{T}'$. The detailed description of the scheme, implemented by Algorithm RST-1, appears in Fig. 5.

Theorem 4 *Algorithm* RST-1 *returns a solution* $\mathcal{T}$ *to instance* I_X *of Problem RST such that* $C(\mathcal{T}) \leq i(i-1)K^{1/i}OPT(I_X)$.

Proof: Lemmas 1 and 2 imply that (f_1, g_1) is a valid T_1-reduction. Hence, the instance $I'_{X'}$ of Problem DST, computed in line 13, satisfies $OPT(I'_{X'}) \leq OPT(I_X)$. By Theorem 1, Algorithm DST$(I'_{X'}, i)$ returns a tree $\mathcal{T}'$ that satisfies $C(\mathcal{T}') \leq i(i-1)K^{1/i}OPT(I'_{X'})$. Since g_1 maps $\mathcal{T}'$ to a solution $\mathcal{T}$ of I_X such that $C(\mathcal{T}) = C(\mathcal{T}')$, we have $C(\mathcal{T}) \leq i(i-1)K^{1/i}OPT(I'_{X'}) \leq i(i-1)K^{1/i}OPT(I_X)$ and the theorem follows. ■

Algorithm RST-1 ($I_X(G, s, X, D), i$)

input:

I_X - an instance of Problem RST

G - the graph,

s- source node.

$X = \{t_1, \cdots, t_K\}$ - the set of K terminals.

D- the delay constraint.

i- the level of the returned tree.

variables:

$\hat{L}_1(\hat{V}, \hat{E})$- The Layers Graph.

output:

$\mathcal{T}$- A solution to I_X.

for each pair of nodes $(u, v) \in G$ **do**
    for each $d \leftarrow 1$ to D **do**
        $\mathcal{P}^d_{(u,v)} \leftarrow$ a minimum cost path between v and u whose delay is at most d
$\hat{V} \leftarrow \emptyset$, $\hat{E} \leftarrow \emptyset$
for each node $v \in G$ **do**
    $\hat{V} \leftarrow \hat{V} \cup \{v^0, \cdots, v^D\}$
    for each d, $0 \le d \le D - 1$ **do**
        $\hat{E} \leftarrow \hat{E} \cup \{(v^d, v^{d+1})\}$
        $c_{(v^d, v^{d+1})} \leftarrow 0$
for each pair of nodes $(u^d, v^j) \in \hat{V}$, $j > d$ **do**
    $\hat{E} \leftarrow \hat{E} \cup \{(u^d, v^j)\}$
    $c_{(u^d, v^j)} \leftarrow C(\mathcal{P}^{j-d}_{(u,v)})$
$I'_{X'} \leftarrow (\hat{L}_1, s_0, \{t^D_1, \cdots, t^D_K\}$
$\mathcal{T}' \leftarrow \mathrm{DST}(I'_{X'}, i)$
$\mathcal{T} \leftarrow \bigcup_{(u^d, v^j) \in \mathcal{T}'} \mathcal{P}^{j-d}_{(u,v)}$
return $\mathcal{T}$

Figure 5: Algorithm RST-1

Note 1 In Algorithm RST-1, we can substitute Algorithm DST with any approximation scheme for Problem DST and obtain, through a T_1 reduction (f_1, g_1), a solution to Problem RST with the same approximation ratio as for Problem DST. For example, for the special case of a small number of terminals, [8] presents an algorithm that identifies an exact (***i.e.***, *optimal*) solution to Problem DST within the computational complexity of $\mathcal{O}(MN^{4K-2} + N^{4K-1}\log N)$. By employing this algorithm, we can identify an exact (optimal) solution for Problem RST in that special case. ■

Due to the large size of the Layers Graph $\hat{L}_1$, the computational complexity of Algorithm RST-1 is too high. Indeed, since $\hat{L}_1$ has $\mathcal{O}(N \cdot D)$ nodes, the running time of Algorithm DST is $\mathcal{O}((N \cdot D)^{i-1}K^{2i-1})$ (by Theorem 3). In the following sections we show how to construct Layers Graphs of smaller size which result in more efficient approximation schemes.

4 Second Approximation Scheme: efficient, but violates the delay constraint

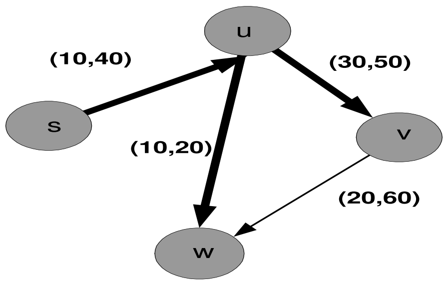
In the previous section we showed that the Layers Graph concept can be employed in order to construct an approximation scheme for Problem RST. However, the computational complexity of the resulting scheme depends on the value of the delay constraint D, which can be large. In this section, we present an approximation scheme whose computational complexity is much lower.

4.1 Layers Graph $\hat{L}_2$

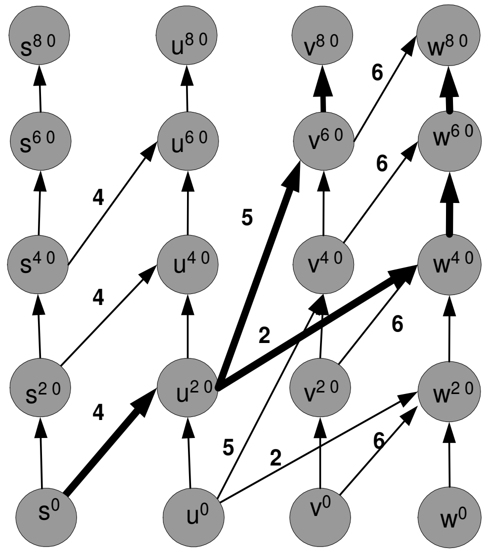
We begin by presenting a Layers Graph $\hat{L}_2$, which is similar to $\hat{L}_1$, but has a much smaller size. The idea is to use the technique of *linear scaling* in order to build a Layers Graph $\hat{L}_2$ with a much smaller number of layers than in $\hat{L}_1$. Specifically, the layers of $\hat{L}_2$ correspond to delay values $\{0, \Delta, 2\Delta, \cdots, \hat{D}\}$, where $\Delta = \frac{\varepsilon \cdot D}{i}$ and $\hat{D} = \Delta \cdot \frac{i(1+\varepsilon)}{\varepsilon} = D(1+\varepsilon)$. We begin by computing, for each pair of nodes $u, v \in G$ and for each $d \in \{\Delta, 2\Delta, \cdots, \hat{D}\}$, a path $\mathcal{P}^d_{(u,v)}$ between u and v such that $D(\mathcal{P}^d_{(u,v)}) \leq d$ and $C(\mathcal{P}^d_{(u,v)}) \leq (1+\varepsilon)C^d_{(u,v)}$, where $C^d_{(u,v)}$ is the minimum cost of a path between u and v whose delay is at most d. For this purpose we use the algorithm presented in [15], which we refer to as Algorithm RSP-1.

The Layers Graph $\hat{L}_2$ is then constructed as follows. For each node $v \in G$, we add $n = \frac{\hat{D}}{\Delta} = \frac{i(1+\varepsilon)}{\varepsilon}$ nodes $\{v^0, v^{\Delta}, v^{2\Delta}, \cdots, v^{\hat{D}}\}$ to $\hat{L}_2$. For each $v \in G$ and each $d, 0 \leq d \leq n-1$, we add to $\hat{L}_2$ a link $(v^{d \cdot \Delta}, v^{(d+1)\Delta})$ with zero cost. Next, for each two nodes $u^{d \cdot \Delta} \in \hat{L}_2$ and $v^{j \cdot \Delta} \in \hat{L}_2$, $j > d$, we add a link $(u^{d \cdot \Delta}, v^{j \cdot \Delta})$ whose cost is set to $C(\mathcal{P}^{(j-d)\Delta}_{(u,v)})$. Fig. 6 depicts an example of original network G and the corresponding Layers Graph $\hat{L}_2$ for $D = 40$, $\varepsilon = 1$ and $i = 2$. In this example we have $\Delta = 20$, $\hat{D} = 80$ and $n = 4$. Note that the number of nodes in Layers Graph $\hat{L}_2$ is just 20, compared to $(D+1)4 = 164$ nodes in the Layers Graph $\hat{L}_1$ that corresponds to G.

Far each $\mathcal{T}$ in G that connects s and the terminals $X = \{t_1, \cdots, t_K\}$, and, in addition, satisfies the delay constraint D there exists a corresponding tree $\mathcal{T}' \in \hat{L}_2$ that connects s^0 and $X' = \{t_1^{\hat{D}}, \cdots, t_K^{\hat{D}}\}$, such that $C(\mathcal{T}) \leq (1+\varepsilon)C(\mathcal{T}')$. For example, consider the tree $\mathcal{T} = \{(s,u), (u,v), (u,w)\}$ in Fig. 6(a). The corresponding tree $\mathcal{T}' = \{(s^0, u^{20}), (u^{20}, v^{60}), (u^{20}, w^{40}), (w^{40}, w^{60}), (v^{60}, v^{80}), (w^{60}, w^{80})\}$ in the Layers Graph $\hat{L}_2$ is marked by bold lines in Fig. 6(b). Recall that in Layers Graph $\hat{L}_1$ each node $v \in G$ is mapped to $v^{d_v} \in \hat{L}_1$, where d_v is the delay of the path that between s and v in $\mathcal{T}$. Thus node u is mapped to node $u^{10} \in \hat{L}_1$.



(a) Original graph G. Associated with each link are its delay and cost



(b) Layers Graph $\hat{L}_2$ (for a small value of ε, some of the links are omitted)

Figure 6: Construction of Layers Graph $\hat{L}_2$.

Since there is no such node in $\hat{L}_2$, node u is mapped to the nearest node of higher layer, *i.e.*, u^{20}. We continue to grow the tree from node u^{20}: link $(u, v) \in \mathcal{T}$ is mapped to link $(u^{20}, v^{60}) \in \mathcal{T}'$, while link $(u, w) \in \mathcal{T}$ is mapped to link (u^{20}, w^{40}).

4.2 T_2-reductions

We proceed by introducing the concept of a T_2*-reduction*, that allows to obtain an efficient approximation scheme for Problem RST.

Definition 5 (T_2-reduction) *A T_2-reduction from Problem RST to Problem DST is a triple (f_2, g_2, ε) that satisfies the following:*

- *f_2 maps an instance $I_X(G, s, X, D)$ of Problem RST to an instance $I'_{X'}(G', s', X')$ of Problem DST such that:*
  1. *$|X'| = |X|$;*
  2. *$OPT^{(i)}(I'_{X'}) \leq (1 + \varepsilon)|X|^{1/i} OPT(I_X)$;*
  3. *for each $Y' \subseteq X'$ it holds that $OPT^{(i)}(I'_{Y'}) \leq (1 + \varepsilon)|Y'|^{1/i} OPT(I_X)$, where $I'_{Y'} = (G', s', Y')$.*
- *g_2 maps a solution $\mathcal{T}'$ of $I'_{X'}$ to a tree $\mathcal{T} \in G$ such that*
  1. *$C(\mathcal{T}) \leq C(\mathcal{T}')$;*
  2. *for each $t_j \in X$ it holds that $D(\mathcal{P}_{(\mathcal{T}, t_j)}) \leq (1 + \varepsilon)D$.*

As we show below, a T_2-reduction (f_2, g_2, ε) gives rise to an approximation scheme for Problem RST that allows a small violation (by a factor of $(1+\varepsilon)$) of the delay constraint.

We proceed to define a T_2-reduction (f_2, g_2, ε).

Definition 6 (Reduction(f_2, g_2, ε)) *A* reduction(f_2, g_2, ε) *is a pair of functions f_2, g_2 and an approximation ratio ε, such that:*

- *The function f_2 receives as input an instance $I_X(G, s, X, D)$ of Problem RST and an approximate ratio ε, and returns an instance $I'_{X'}(\hat{L}_2, s^0, X')$ of Problem DST, where $\hat{L}_2$ is the Layers Graph and $X' = \{t_j^{\hat{D}} \mid t_j \in X\}$.*
- *The function g_2 receives as input a solution $\mathcal{T}'$ of $I'_{X'}$. The function returns a tree*

$$\mathcal{T} = \bigcup_{(u^d, v^j) \in \mathcal{T}'} \mathcal{P}^{(j-d)}_{(u,v)}, \tag{2}$$

where $\left\{\mathcal{P}^{(d)}_{(u,v)}\right\}$ are paths computed during construction of the Layers Graph $\hat{L}_2$.

We proceed to show that (f_2, g_2, ε) is a valid T_2-reduction (as per Definition 5). We will use the following lemma, taken from [24].

Lemma 3 [24] *Let $\hat{G}$ be a transitive closure of graph G. Then, for each tree $\mathcal{T} \in \hat{G}$ that connects source s with a group X of terminals and for each i, $1 \le i \le \log|X|$ there exists an i-level tree $\hat{\mathcal{T}}$ in $\hat{G}$ that connects s with X such that $C(\hat{\mathcal{T}}) \le |X|^{1/i} \cdot C(\mathcal{T})$.*

Lemma 4 *If $I'_{X'} = f_2(I_X)$ then $OPT^{(i)}(I'_{X'}) \le (1+\varepsilon)|X|^{1/i} OPT(I_X)$.*

Proof: Let $\hat{I}_{\hat{X}}(\hat{L}_1, s^0, \hat{X}) = f_1(I_X)$ and let $\mathcal{T}$ be a solution to instance $\hat{I}_{\hat{X}}$ of Problem DST. By Lemma 1, $C(\mathcal{T}) \le OPT(I_X)$. We note that the Layers Graph $\hat{L}_1$ is a transitive closure *per se*. Hence, by Lemma 3, there exists a tree an i-level tree $\hat{\mathcal{T}}$ in $\hat{L}_1$ such that $C(\hat{\mathcal{T}}) \le |X|^{1/i} C(\mathcal{T}) \le |X|^{1/i} OPT(I_X)$.

We round the delay value d_l of each link $l \in \hat{\mathcal{T}}$, replacing it by d'_l, as follows:

$$d'_l = \left\lceil \frac{d_l}{\Delta} \right\rceil \cdot \Delta,$$

where $\Delta = \frac{\varepsilon \cdot D}{i}$. Note that after the rounding delay values of each link increase by at most Δ, *i.e.*, $d'_l \le d_l + \Delta$.

For each node $v \in \hat{\mathcal{T}}$, we denote by d_v and d'_v the delay of the path between s and v in $\hat{\mathcal{T}}$ with respect to the original and rounded delay values, respectively, *i.e.*, $d_v = \sum_{l \in \mathcal{P}_{(\hat{\mathcal{T}}, v)}} d_l$ and $d'_v = \sum_{l \in \mathcal{P}_{(\hat{\mathcal{T}}, v)}} d'_l$.

For each node $v \in \hat{\mathcal{T}}$, we define $\mathcal{F}(v) = v^{d'_v}$. Note that the delay of the path $\mathcal{P}_{(\hat{\mathcal{T}}, v)}$ with respect to the original link delays is at most D, *i.e.*, $\sum_{l \in \mathcal{P}_{(\hat{\mathcal{T}}, v)}} d_l \le D$. It follows that the delay of the path $\mathcal{P}_{(\hat{\mathcal{T}}, v)}$ with respect to the rounded link costs is at most $D + i\Delta = (1+\varepsilon)D = \hat{D}$. We conclude that, for each $v \in V$, it holds that $\mathcal{F}(v) \in \hat{L}_2$.

For each link $l = (u, v) \in \hat{\mathcal{T}}$, we define $\mathcal{F}(l) = (\mathcal{F}(u), \mathcal{F}(v))$. As shown above, $\mathcal{F}(u) \in \hat{V}$ and $\mathcal{F}(v) \in \hat{V}$. Moreover, there is a link between $\mathcal{F}(u) = u^{d'_u}$ and $\mathcal{F}(v) = v^{d'_v}$ in $\hat{L}_2$, whose cost is set to $C(\mathcal{P}^{d'_v - d'_u}_{(u,v)})$. Since $d'_v - d'_u > d_v - d_u$, it holds that $C(\mathcal{P}^{d'_v - d'_u}_{(u,v)}) \le (1+\varepsilon) c_l$.

Let $\hat{\mathcal{T}}' = \{\mathcal{F}(l) \mid l \in \hat{\mathcal{T}}\} \cup \{(t_j^d, t_j^{d+1}) \mid t_j \in \hat{\mathcal{T}}, d_{t_j} \leq d \leq \hat{D} - 1\}$. From the above discussion it follows that $\hat{\mathcal{T}}'$ is an i-level tree in $\hat{L}_2$ that connects source s_0 with nodes $t_1^{\hat{D}}, \cdots, t_K^{\hat{D}}$. Moreover, since the cost of each link in $\hat{\mathcal{T}}'$ is at most $(1+\varepsilon)$ higher than the cost of the corresponding link in $\hat{\mathcal{T}}$, it follows that $C(\hat{\mathcal{T}}') \leq (1+\varepsilon)C(\hat{\mathcal{T}})$. Thus, $OPT^{(i)}(I'_{X'}) \leq C(\hat{\mathcal{T}}') \leq (1+\varepsilon)|X|^{1/i}OPT(I_X)$ and the lemma follows. ■

Lemma 5 *Let $I'_{X'} = f_2(I_X)$. For each subset $Y' \subseteq X'$ it holds that $OPT^{(i)}(I'_{Y'}) \leq (1+\varepsilon)|Y'|^{1/i}OPT(I_X)$, where $I'_{Y'} = (\hat{L}_2, s^0, Y')$;*

Proof: Let Y' be a subset of X', we denote $Y = \{t_j \mid t_j^{\hat{D}} \in Y'\}$. Next, we denote by I_Y the instance (G, s, Y, D) of Problem RST. Note that $Y \subseteq X$ and $I'_{Y'} = f_2(I_Y)$. Hence, by Lemma 4, $OPT^{(i)}(I'_{Y'}) \leq (1+\varepsilon)|Y'|^{1/i}OPT(I_Y)$. Since $Y \subseteq X$ it holds that $OPT(I_Y) \leq OPT(I_X)$. We conclude that $OPT(I'_{Y'})^{(i)} \leq (1+\varepsilon)|Y'|^{1/i}OPT(I_X)$ and the lemma follows. ■

Lemma 6 *Let $\mathcal{T}'$ be a solution of instance $I'_{X'}$ of Problem DST. Then, $\mathcal{T} = g_2(\mathcal{T}')$ is a tree that connects the source s to the terminals X in G and satisfies $C(\mathcal{T}) = C(\mathcal{T}')$ as well as $D(\mathcal{P}_{(\mathcal{T},t_j)}) \leq (1+\varepsilon)D$ for each $t_j \in X$.*

Proof: According to the definition of g_2, $\mathcal{T}$ includes, for each link $l = (u^d, v^j)$ in $\mathcal{T}'$, the path $\mathcal{P}_{(u,v)}^{(j-d)}$, which was computed during the construction of the Layers Graph $\hat{L}_2$. Since $D(\mathcal{P}_{(u,v)}^{(j-d)}) \leq (j-d)$ and $C(\mathcal{P}_{(u,v)}^{(j-d)}) \leq c_l$, we conclude that $C(\mathcal{T}) = C(\mathcal{T}')$, and for each terminal $t \in X$, it holds that $D(\mathcal{P}_{(\mathcal{T}',t)}) \leq \hat{D} = (1+\varepsilon)D$. ■

4.3 Approximation Scheme

The T_2-reduction (f_2, g_2, ε), gives rise to the corresponding approximation scheme for Problem RST. Specifically, given an instance I_X of Problem RST we compute an instance $I'_{X'}$ of Problem DST by invoking function f_2. Next, we find a solution $\mathcal{T}'$ of $I'_{X'}$ by applying Algorithm DST. Finally, we identify a solution $\mathcal{T}$ of I_X by invoking function g_2 on $\mathcal{T}'$. The detailed description of the scheme, implemented by Algorithm RST-2, appears in Fig. 7.

Theorem 5 *Given an instance I_X of Problem RST, Algorithm* RST-2 *identifies, in $\mathcal{O}((\frac{N}{\varepsilon})^{i-1}K^{2i-1})$ time, a tree $\mathcal{T} \in G$ such that $C(\mathcal{T}) \leq (1+\varepsilon)i(i-1)K^{1/i}OPT(I_X)$ and $D(\mathcal{P}_{(\mathcal{T},t_j)}) \leq (1+\varepsilon)D$ for each $t_j \in X$.*

Proof: Lemmas 4, 5 and 6 imply that (f_2, g_2, ε) is a valid T_2-reduction. Let $I'_{X'}$ be an instance of Problem DST computed in line 16. Since (f_2, g_2, ε) is a valid T_2-reduction, for each subset Y' of X', it holds that $OPT^{(i)}(I'_{Y'}) \leq (1+\varepsilon)|Y'|^{1/i}OPT(I_X)$. Thus, the condition of Theorem 2 holds for $\hat{C} = (1+\varepsilon)OPT(I_X)$. Hence, Algorithm DST returns a tree $\hat{\mathcal{T}}$ such that $C(\hat{\mathcal{T}}) \leq (1+\varepsilon)i(i-1)K^{1/i}OPT(I_X)$. Since g_2 maps $\hat{\mathcal{T}}$ to a tree $\mathcal{T} \in G$ such that $C(\mathcal{T}) = C(\hat{\mathcal{T}})$ and $D(\mathcal{P}_{(\mathcal{T},t_j)}) \leq (1+\varepsilon)D$ for each $t_j \in X$, we have $C(\mathcal{T}) \leq (1+\varepsilon)i(i-1)K^{1/i}OPT(I_X)$.We conclude that Algorithm RST-2 identifies a tree $\mathcal{T} \in G$ such that $C(\mathcal{T}) \leq (1+\varepsilon)i(i-1)K^{1/i}OPT(I_X)$ and $D(\mathcal{P}_{(\mathcal{T},t_j)}) \leq (1+\varepsilon)D$ for each $t_j \in X$.

The computational complexity of Algorithm RST-2 is dominated by the time required for executing Algorithm DST for $\hat{L}_2$. Since the number of nodes in $\hat{L}_2$ is $N \cdot \frac{i(1+\varepsilon)}{\varepsilon} = \mathcal{O}(\frac{i \cdot N}{\varepsilon})$, the running time of the algorithm is $\mathcal{O}((\frac{i \cdot N}{\varepsilon})^{i-1}K^{2i-1})$. ■

Algorithm RST-2 ($I_X(G, s, X, D), i, \varepsilon$)

input:

I_X - an instance of Problem RST

G - the graph

s- source node

$X = \{t_1, \cdots, t_K\}$ - the set of K terminals

D- the delay constraint

i- the level of the returned tree

ε- the approximation ratio

variables:

$\hat{L}_2(\hat{V}, \hat{E})$- The Layers Graph.

output:

$\mathcal{T}$- A solution to I_X.

$\Delta \leftarrow \frac{\varepsilon \cdot D}{i}$

for each pair of nodes (u, v), $u, v \in G$ **do**

  for each d, $0 \leq d \leq n$ **do**

    $\mathcal{P}^{d \cdot \Delta}_{(u,v)} \leftarrow$RSP-1$(G, u, v, d \cdot \Delta, \varepsilon)$

$\hat{V} \leftarrow \emptyset$, $\hat{E} \leftarrow \emptyset$

$n \leftarrow \frac{i(1+\varepsilon)}{\varepsilon}$

$\hat{D} \leftarrow \Delta \cdot n$

for each node $v \in G$ **do**

  $\hat{V} \leftarrow \hat{V} \cup \{v^0, v^{\Delta}, v^{2\Delta}, \cdots, v^{n \cdot \Delta}\}$

  for each $d \leftarrow 0$ to n **do**

    $\hat{E} \leftarrow \hat{E} \cup \{(v^{d \cdot \Delta}, v^{(d+1) \cdot \Delta})\}$

    $c_{(v^{d \cdot \Delta}, v^{(d+1) \cdot \Delta})} \leftarrow 0$

  for each pair of nodes (u^d, v^j), $u^d, v^j \in G$, $d > i$ **do**

    $\hat{E} \leftarrow \hat{E} \cup \{(u^d, v^j)\}$

    $c_{(u^d, v^j)} = C(\mathcal{P}^{(j-d)\Delta}_{(u,v)})$

$I'_{X'} \leftarrow (\hat{L}_2, s^0, \{t_1^{\hat{D}}, \cdots, t_K^{\hat{D}}\})$

$\mathcal{T}' \leftarrow \mathrm{DST}(I'_{X'}, i)$

$\mathcal{T} = \bigcup_{(u^d, v^j) \in \mathcal{T}'} \mathcal{P}^{j-d}_{(u,v)}$

return $\mathcal{T}$

Figure 7: Algorithm RST-2

5 Third Approximation Scheme: efficient and with no delay violation

In this section we present an approximation scheme for Problem RST that has low computational complexity and does not violate the delay constraint. The idea is to use a new Layers Graph $\hat{L}_3$ that is similar to $\hat{L}_1$, but contains much less links and nodes.

In order to construct $\hat{L}_3$ we need to have an estimate B on the value of $OPT(I_X)$. We assume for the moment that such an estimate is given, while later, in Section 5.5, we shall show how to identify a sufficiently good estimate.

5.1 Path Aggregation

Recall that Algorithm RST-1 begins by computing the set S that includes, for each two nodes u and v and each delay constraint d, $1 \leq d \leq D$, a minimum cost path $\mathcal{P}^d_{(u,v)}$ between u and v whose delay is at most d. The tree returned by the algorithm comprises of paths that belong to S. Note that S contains a large number of paths ($\mathcal{O}(N^2 D)$). Moreover, the computation of each $\mathcal{P}^d_{(u,v)} \in S$ incurs high complexity. Accordingly, we use an alternative set of paths, S' of much smaller size. In addition, the set S' comprises of suboptimal paths, whose computation requires much less time. Specifically, we set $\Delta = \frac{\varepsilon B}{4K-2}$ and compute, for each $u, v \in G$ and for each $c = \Delta, 2 \cdot \Delta, \cdots, B$, a path $\hat{\mathcal{P}}^c_{(u,v)}$, such that:

1. $C(\hat{\mathcal{P}}^c_{(u,v)}) \leq c + \Delta$;
2. $D(\hat{\mathcal{P}}^c_{(u,v)}) \leq D(\mathcal{P}')$ for each path $\mathcal{P}'$ between u and v that satisfies $C(\mathcal{P}') \leq c$.

Note that S' is a path set that represents much bigger path set S. Thus, we say that S' *aggregates* path set S.

For example, Fig. 8 demonstrates the paths that belong to sets $S = \{\mathcal{P}_1, \cdots, \mathcal{P}_8\}$ and $S' = \{\hat{\mathcal{P}}_1, \cdots, \hat{\mathcal{P}}_3\}$ in the delay-cost plane. A path $\mathcal{P}$ is represented by a point $(D(\mathcal{P}), C(\mathcal{P}))$. Note that the delay of $\hat{\mathcal{P}}_1$ is no higher than that of $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$, while the cost of $\hat{\mathcal{P}}_1$ is higher than that of $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ by at most 2Δ. Thus, we can use $\hat{\mathcal{P}}_1$ instead of $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$. We use $\hat{\mathcal{P}}_2$ instead of $\mathcal{P}_3$, $\mathcal{P}_4$, $\mathcal{P}_5$ and $\hat{\mathcal{P}}_6$ instead of $\mathcal{P}_7$, $\mathcal{P}_8$ and $\mathcal{P}_9$.

We compute set S' by interchanging delays and costs in G and invoking Algorithm RSP-2, presented in [10], on the resulting graph for delay constraint c and $\varepsilon = \frac{\Delta}{c}$. Finally, we insert all paths $\hat{\mathcal{P}}^c_{(u,v)}$ to S', *i.e.*, $S' = \{\hat{\mathcal{P}}^c_{(u,v)} \mid u, v \in G, c = \Delta, 2\Delta, \cdots, B\}$.

Lemma 7 *For each path $\mathcal{P}^d_{(u,v)} \in S$ there exists a path $\hat{\mathcal{P}}^c_{(u,v)} \in S'$ such that $D(\hat{\mathcal{P}}^c_{(u,v)}) \leq D(\mathcal{P}^d_{(u,v)})$ and $C(\hat{\mathcal{P}}^c_{(u,v)}) \leq C(\mathcal{P}^d_{(u,v)}) + 2\Delta$.*

Proof: Let $\mathcal{P}^d_{(u,v)}$ be a path in S. Let $c = \Delta \left\lceil \frac{C(\mathcal{P}^d_{(u,v)})}{\Delta} \right\rceil$. Note that since $C(\mathcal{P}^d_{(u,v)}) \leq B$ it holds that $c \leq B$, hence there exist path $\hat{\mathcal{P}}^c_{(u,v)}$ in S'. We show that $\hat{\mathcal{P}}^c_{(u,v)}$ satisfies both conditions stated in the lemma. Recall that $\hat{\mathcal{P}}^c_{(u,v)}$ is computed by AlgorithmRSP-2 applied for u, v, c, and Δ. Thus, since $C(\mathcal{P}^d_{(u,v)}) \leq c$, we have $D(\hat{\mathcal{P}}^c_{(u,v)}) \leq D(\mathcal{P}^d_{(u,v)})$. In addition, the cost $C(\hat{\mathcal{P}}^c_{(u,v)})$ of $\hat{\mathcal{P}}^c_{(u,v)}$ is at most $c + \Delta \leq C(\mathcal{P}^d_{(u,v)}) + 2\Delta$. ■

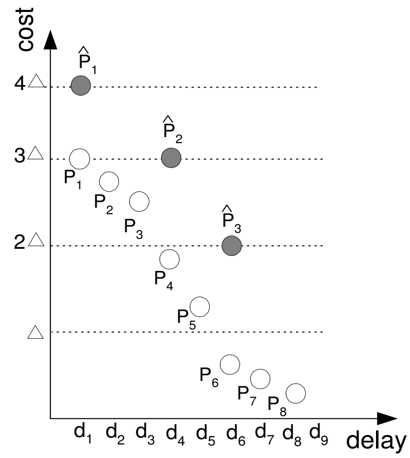


Figure 8: Paths that belong to sets $S = \{\mathcal{P}_1, \cdots, \mathcal{P}_8\}$ and $S' = \{\hat{\mathcal{P}}_1, \cdots, \hat{\mathcal{P}}_3\}$ are depicted in the delay-cost plane.

5.2 Layers Graph $\hat{L}_3$

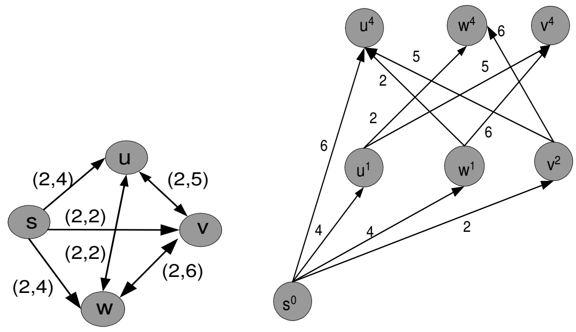
As mentioned, our purpose is to build a Layers Graph $\hat{L}_3$ that is similar to $\hat{L}_1$, but has smaller size.

In order to reduce the size of $\hat{L}_3$ we restrict ourself to i-label trees, which, by Lemma 3, provide a good approximation of the optimum solution. Thus, all links of $\hat{L}_1$ do not belong to i-level trees are omitted from $\hat{L}_3$. We construct $\hat{L}_3$ in i phases, as follows: in the first phase we add links that originate from s and the corresponding nodes, in the second phase we add links that originate from the nodes added in the second phase, *etc.*. Fig. 9 depicts an example of Layers Graph $\hat{L}_3$, which comprises of several 2-level trees. In order to further reduce the size of $\hat{L}_3$, we use path aggregation. More specifically, links of $\hat{L}_3$ represent paths in S', whose size is smaller than that of S. Thus, each node $v^i \in \hat{L}_3$ has only $\mathcal{O}(\frac{B}{\Delta}N) = \mathcal{O}(\frac{KN}{\varepsilon})$ links that originate from it. Hence, the number of nodes that we add to $\hat{L}_3$ in the first phase is $\mathcal{O}(\frac{KN}{\varepsilon})$, in the second phase we add $\mathcal{O}((\frac{KN}{\varepsilon})^2)$ nodes, *etc.*, and the total number of nodes and links is $\mathcal{O}((\frac{KN}{\varepsilon})^i)$. The important property of Layers Graph $\hat{L}_3$ is that the maximum number n of links that originate from a node in $\hat{L}_3$ is at most $\mathcal{O}(\frac{KN}{\varepsilon})$, compared to $\mathcal{O}(D)$ in $\hat{L}_1$. Since n determines the running time of Algorithm DST applied to $\hat{L}_3$, this results in a significant reduction in the computational complexity of the overall scheme.

We proceed to describe the construction of $\hat{L}_3$ in more details. $\hat{L}_3$ is constructed through the following iterative process. We maintain a set A_h that records the nodes added to $\hat{L}_3$ at iteration h. We begin with $\hat{L}_3 = \{s^0\}$ and $A_0 = \{s^0\}$. At iteration h, we execute the following loop. For each node $u^d \in A_{h-1}$ and for each path $\hat{\mathcal{P}}^c_{(u,v)} \in S'$, such that $D(\hat{\mathcal{P}}^c_{(u,v)}) \leq D - d$, we add a node u^j to $\hat{L}_3$ and A_h, where $j = d + D(\hat{\mathcal{P}}^c_{(u,v)})$. In addition, we add a link (u^d, v^j) to $\hat{L}_3$ whose cost is set to $C(\hat{\mathcal{P}}^c_{(u,v)})$. The process terminates after i iterations. Finally, for each terminal $t_j \in X$ we add a node t^D_j to $\hat{L}_3$, and a zero-cost link that connects each node $t^d_j \in \hat{L}_3$ to t^D_j.

5.3 T_3-reductions

We define the concept of a T_3-*reduction*, which is similar to a T_2-reduction, but with no violation of the delay constraint.



(a) Original graph G. Associated with each link are its delay and cost

(b) Layers Graph $\hat{L}_3$ (for a small value of ε)

Figure 9: Construction of Layers Graph $\hat{L}_3$.

Definition 7 (T_3-reduction) *A* T_3-reduction *from Problem RST to Problem DST is a quadruple* $(f_3, g_3, B, \varepsilon)$ *that satisfies the following:*

- *The function* f_3 *maps an instance* $I_X(G, s, X, D)$ *of Problem RST to an instance* $I'_{X'}(G', s', X')$ *of Problem DST, such that:*
  1. $|X'| = |X|$;
  2. $OPT^{(i)}(I'_{X'}) \leq |X|^{1/i} OPT(I_X) + \varepsilon B$;
  3. *for each* $Y' \subseteq X'$ *it holds that* $OPT^{(i)}(I'_{Y'}) \leq |Y'|^{1/i} OPT(I_X) + \varepsilon B$, *where* $I'_{Y'} = (G', s', Y')$;
- *The function* g_3 *maps a solution* $\mathcal{T}'$ *of* $I'_{X'}$ *to solution* $\mathcal{T}$ *of* I_X *such that* $C(\mathcal{T}) \leq C(\mathcal{T}')$.

As it is the case for a T_1-reduction, a T_3-reduction $(f_3, g_3, B, \varepsilon)$ gives rise to an approximation scheme for Problem RST.

We proceed to define a T_3-reduction$(f_3, g_3, B, \varepsilon)$.

Definition 8 (Reduction$(f_3, g_3, B, \varepsilon)$) *A* reduction $(f_3, g_3, B, \varepsilon)$ *is a pair of functions* f_3, g_3, *an estimate* B *on* $OPT(I_X)$, *and an approximation ratio* ε, *such that:*

- *The function* f_3 *receives as input an instance* $I_X(G, s, X, D)$ *of Problem RST, and returns an instance* $I'_{X'}(\hat{L}_3, s^0, X')$ *of Problem DST, where* $\hat{L}_3$ *is the Layers Graph for* G, B *and* ε, *and* $X' = \{t_j^D \mid t_j \in X\}$.
- *The function* g_3 *receives as input a solution* $\mathcal{T}'$ *of* $I'_{X'}$. *The function returns a tree*

$$\mathcal{T} = \bigcup_{l(u^d, v^j) \in \mathcal{T}'} \hat{\mathcal{P}}^{c_l}_{(u,v)}.$$

In order to prove that $(f_3, g_3, B, \varepsilon)$ is a valid T_3-reduction, we need the following lemma.

Lemma 8 *Let $\hat{G}$ be a transitive closure of the graph G and let $I_X(\hat{G}, s, X)$ be an instance of Problem DST. Then there exists an i-level tree $\hat{\mathcal{T}} \in \hat{G}$ that connects s and terminals X such that $C(\mathcal{T}) \leq |X|^{1/i} \cdot OPT(I_X)$ and the number of links in $\hat{\mathcal{T}}$ is at most $2|X| - 1$.*

Proof: By Lemma 3, there exists a tree $\mathcal{T}$ that connects s with X such that $C(\mathcal{T}) \leq |X|^{1/i} \cdot OPT(I_X)$. Let $\hat{\mathcal{T}}$ be such a tree with minimum number of links.

We prove that $\hat{\mathcal{T}}$ has at most $2|X| - 1$ links. Suppose, by way of contradiction, that $\hat{\mathcal{T}}$ has more than $2|X| - 1$ links. Then, there exists a node $u^d \in \hat{\mathcal{T}} \neq s^0$ that has only one child v^j that belongs to $\hat{\mathcal{T}}$. We then substitute the links $(p(u^d), u^d)$ and (u^d, v^j) by a link $(p(u^d), v^j)$, where $p(u^d) \in \hat{\mathcal{T}}$ is a parent node of node u^d. The cost of the resulted tree is identical to $C(\hat{\mathcal{T}})$, but the number of links is fewer than in $\hat{\mathcal{T}}$, which contradicts the fact the number of links in $\hat{\mathcal{T}}$ is minimal. ■

We proceed to show that the function f_3 satisfies the conditions of a T_3-reduction.

Lemma 9 *Let $I'_{X'} = f_3(I_X)$. If $OPT(I_X) \leq B$ then $OPT^{(i)}(I'_{X'}) \leq |X|^{1/i} OPT(I_X) + \varepsilon B$.*

Proof: Let $\hat{I}_{\hat{X}}(\hat{L}_1, s^0, \hat{X}) = f_1(I_X)$ and let $\mathcal{T}_{\hat{X}}$ be a solution to instance $\hat{I}_{\hat{X}}$. By Lemma 1, it holds that $C(\mathcal{T}_{\hat{X}}) \leq OPT(I_X)$. Lemma 8 implies that there exists an i-level tree $\hat{\mathcal{T}}_{\hat{X}}$ in $\hat{L}_1$ such that $C(\hat{\mathcal{T}}_{\hat{X}}) \leq |X|^{1/i} C(\mathcal{T}_{\hat{X}}) \leq |X|^{1/i} OPT(I_X)$ and the number of links in $C(\hat{\mathcal{T}}_{\hat{X}})$ is at most $2N - 1$.

We show that there exists an i-level tree $\hat{\mathcal{T}}'_{X'}$ in $\hat{L}_3$ that connects s^0 and the terminals $X' = \{t_j^D \mid t_j \in X\}$ such that $C(\hat{\mathcal{T}}'_{X'}) \leq C(\hat{\mathcal{T}}_{\hat{X}}) + \varepsilon \cdot B$. We construct $\hat{\mathcal{T}}'_{X'}$ through the following iterative process. For each node $v_j \in \hat{\mathcal{T}}_{\hat{X}}$, there is a corresponding node $v_{j'} \in \hat{\mathcal{T}}'_{X'}$, such that $j' \leq j$. We maintain a set A_h, which keeps each node added to $\hat{\mathcal{T}}'_{X'}$ at iteration h and the corresponding node in $\hat{\mathcal{T}}_{\hat{X}}$. We begin by setting $\hat{\mathcal{T}}'_{X'} = \{s^0\}$ and $A_0 = \{(s^0, s^0)\}$. At iteration h we perform the following loop. For each pair of nodes $(u^{d'}, u^d) \in A_{h-1}$, and for each link $l(u^d, v^j) \in \hat{\mathcal{T}}_{\hat{X}}$ we set $c'_l = \left\lceil \frac{c_l}{\Delta} \right\rceil \Delta$. Since $c_l \leq OPT(I_X) \leq B$, it holds that $c'_l \in \{\Delta, 2\Delta, \cdots, B\}$, which implies that there exists $\hat{\mathcal{P}}^{c'_l}_{(u,v)} \in S'$. Next, we set $j' = d' + D(\hat{\mathcal{P}}^{c'_l}_{(u,v)})$. Note that $c_{(u^{d'}, v^{j'})} \leq c_l + 2\Delta = c_l + \frac{\varepsilon B}{2K-1}$ and $d_{(u^{d'}, v^{j'})} \leq d_l$. Next, we add a link $(u^{d'}, v^{j'})$ to $\hat{\mathcal{T}}'_{X'}$ and pair of nodes $(u^{j'}, u^j)$ to A_h. The process terminates after i iterations. Finally, we augment $\hat{\mathcal{T}}'_{X'}$ by zero-cost links in order to obtain a tree that connects source s^0 to terminals X'.

Note that $C(\hat{\mathcal{T}}'_{X'}) \leq \sum_{l' \in \hat{\mathcal{T}}'_{X'}} c_{l'} \leq \sum_{l \in \hat{\mathcal{T}}_{\hat{X}}} (c_l + \frac{\varepsilon B}{2K-1}) \leq C(\hat{\mathcal{T}}_{\hat{X}}) + \varepsilon B$, where the last inequality holds because tree $\hat{\mathcal{T}}_{\hat{X}}$ has at most $2K - 1$ links. We conclude that $OPT^{(i)}(I'_{X'}) \leq C(\hat{\mathcal{T}}_{\hat{X}}) + (2K-1)\Delta \leq |X|^{1/i} OPT(I_X) + \varepsilon B$ and the lemma follows. ■

Lemma 10 *Let $I'_{X'} = f_3(I_X)$. If $OPT(I_X) \leq B$ then, for each subset $Y' \subseteq X'$, it holds that $OPT^{(i)}(I'_{Y'}) \leq |Y'|^{1/i} OPT(I_X) + \varepsilon B$, where $I'_{Y'} = (\hat{L}_3, s^0, Y')$;*

Proof: Let Y' be a subset of X', we denote by $I'_{Y'}$ the instance $(\hat{L}_3, s^0, Y')$ of Problem DST. Let $Y = \{t_j \mid t_j^D \in Y'\}$ and let I_Y be the instance (G, s, Y, D) of Problem RST. Note that $I'_{Y'} = f_3(I_Y)$. Lemma 9 implies that $OPT^{(i)}(I'_{Y'}) \leq |Y|^{1/i} OPT(I_Y) + \varepsilon B$. Since $Y \subseteq X$, we have $OPT(I_Y) \leq OPT(I_X)$. We conclude that $OPT^{(i)}(I'_{Y'}) \leq |Y|^{1/i} OPT(I_X) + \varepsilon B$ and the lemma follows. ■

We proceed to show that the function g_3 satisfies the conditions of T_3-reduction.

Lemma 11 *Let $I'_{X'} = f_3(I_X)$ and let $\mathcal{T}'$ be a solution of instance $I'_{X'}$. Then, $\mathcal{T} = g_3(\mathcal{T}')$ is a solution of instance I_X and $C(\mathcal{T}) = C(\mathcal{T}')$.*

Proof: By the definition of g_3, $\mathcal{T}$ includes, for each link $l(u^d, v^j)$ in $\mathcal{T}'$, a path $\hat{\mathcal{P}}^{c_l}_{(u,v)} \in S'$. We note that $C(\hat{\mathcal{P}}^{c_l}_{(u,v)}) \leq c_l$ and $D(\hat{\mathcal{P}}^{c_l}_{(u,v)}) \leq j-1$. We conclude that $\mathcal{T}$ is a solution of the instance I_X and $C(\mathcal{T}) = C(\mathcal{T}')$. ∎

5.4 Algorithm SCALE

The T_3-reduction $(f_3, g_3, B, \varepsilon)$, gives rise to the corresponding approximation scheme for Problem RST. Specifically, given an instance I_X of Problem RST we compute an instance $I'_{X'}$ of Problem DST by invoking function f_3. Next, we find a solution $\mathcal{T}'$ of $I'_{X'}$ by applying Algorithm DST. Finally, we identify a solution $\mathcal{T}$ of I_X by invoking function g_3 on $\mathcal{T}'$. The detailed description of the scheme, implemented by Algorithm SCALE, appears in Fig. 10. Note that this algorithm is not a complete approximation scheme because it assumes that an estimate B on $OPT(I_X)$ is known.

Lemma 12 *If $OPT(I_X) \leq B$ then Algorithm* SCALE *returns a solution $\mathcal{T}$ to I_X such that $C(\mathcal{T}) \leq i(i-1)K^{1/i}(OPT(I_X) + \varepsilon B)$.*

Proof: Let $I'_{X'}$ be an instance of Problem DST computed in line 22. By Lemmas 9 and 10, for each $Y' \subseteq X'$ it holds that $OPT^{(i)}(I'_{Y'}) \leq |Y'|^{1/i} OPT(I_X) + \varepsilon B$. Thus, the condition of Lemma 2 holds for $\hat{C} = OPT(I_X) + \varepsilon B$. Hence, it follows that Algorithm DST returns a tree $\hat{\mathcal{T}}$ such that $C(\hat{\mathcal{T}}) \leq i(i-1)K^{1/i}(OPT(I_X) + \varepsilon B)$. By Lemma 11, g_3 satisfies the conditions of a T_3-reduction. Thus, g_3 maps $\hat{\mathcal{T}}$ to a tree $\mathcal{T} \in G$ such that $C(\mathcal{T}) = C(\hat{\mathcal{T}})$. We conclude that $C(\mathcal{T}) \leq i(i-1)K^{1/i}(OPT(I_X) + \varepsilon B)$ and the lemma follows. ∎

Lemma 13 *The computational complexity of Algorithm* SCALE *is $\mathcal{O}(\left(\frac{4N}{\varepsilon}\right)^{i-1} K^{3i-2})$.*

Proof: The Layers Graph $\hat{L}_3$ is constructed in i iterations. At iteration j, we invoke Algorithm RSP-2 $\mathcal{O}(\frac{B}{\Delta}) = \mathcal{O}(\frac{K}{\varepsilon})$ times, for each $c \in \{\Delta, 2\Delta, \cdots, B\}$. Since the running time of Algorithm RSP-2 is $\mathcal{O}(\frac{(M+N\log N)\cdot N\cdot B}{\Delta})$ $=\mathcal{O}(\frac{(M+N\log N)\cdot N\cdot K}{\varepsilon})$, the total running time of all invocations of Algorithm RSP-2 is $\mathcal{O}(\frac{(M+N\log N)iK^2N}{\varepsilon^2})$.

Each node $v^j \in \hat{L}_3$ has at most $\frac{N\cdot B}{\Delta}$ links originated from it. Thus, by Theorem 3, the execution time of Algorithm DST is $\mathcal{O}(\left(\frac{N\cdot B}{\Delta}\right)^{i-1} K^{2i-1}) = \mathcal{O}(\left(\frac{KN}{\varepsilon}\right)^{i-1} K^{2i-1}) = \mathcal{O}(\left(\frac{N}{\varepsilon}\right)^{i-1} K^{3i-2})$.

We conclude that the total running time of the algorithm is dominated by the time required to execute Algorithm DST, and the lemma follows. ∎

5.5 Lower and Upper Bounds

Algorithm SCALE, presented in the previous section, requires an estimate B on the cost $OPT(I_X)$ of the optimal solution to I_X. In this section we show how to obtain a good estimate B. For this purpose, we maintain lower and upper bounds, L and U, on $OPT(I_X)$. We begin with some initial bounds, and proceed to iteratively improve them, until they become sufficiently tight. The technique we use is similar to the one employed in [15] for finding restricted shortest (unicast) paths.

The initial upper and lower bounds, L and U, are identified by Procedure BOUND. We denote by $c^1 < c^2 < \cdots < c^r$ the distinct cost values of the link in G. Our goal is to identify the maximum cost value $c^* \in \{c^j\}$ such that if all links whose cost is higher than c^* are omitted from G, the resulted graph G' has no tree that connects s and terminal X and satisfies the QoS

Algorithm SCALE **(**$I_X(G, s, X, D), i, B, \varepsilon$**)**

input:

I_X - an instance of Problem RST

G - the graph

s- source node

$X = \{t_1, \cdots, t_K\}$ - the set of K terminals

D- the delay constraint

i- the level of the returned tree

B- an estimate on $OPT(I_X)$

ε- approximation ratio

variables:

$\hat{L}_3(\hat{V}, \hat{E})$- The Layers Graph.

output:

$\mathcal{T}$- A solution to I_X.

$\Delta \leftarrow \frac{\varepsilon B}{4K-2}$
$S' \leftarrow \emptyset$
for each pair of nodes (u, v), $u, v \in G$ **do**
    for $c = \Delta, 2\Delta, \cdots, B$ **do**
        $\hat{\mathcal{P}}^c_{(u,v)} \leftarrow$RSP-2$(G, u, v, c, \frac{\Delta}{c})$
        $S' \leftarrow S' \cup \hat{\mathcal{P}}^c_{(u,v)}$
$\hat{L}_3 \leftarrow \{s^0\}$
$A_0 \leftarrow \{s^0\}$
for $h \leftarrow 1$ to i **do**
    for each node $u^d \in A_{h-1}$ **do**
        if $u \in X$ **then**
            $\hat{L}_3 \leftarrow \hat{L}_3 \cup \{(u^d, u^D)\}$
            $c_{(u^d, u^D)} \leftarrow 0$
        for each path $\hat{\mathcal{P}}^c_{(u,v)} \in S'$ that originates from
          u such that $D(\hat{\mathcal{P}}^c_{(u,v)}) \leq D - d$ **do**
            $j \leftarrow d + D(\hat{\mathcal{P}}^c_{(u,v)})$
            $A_h \leftarrow A_h \cup \{v^j\}$
            $\hat{L}_3 \leftarrow \hat{L}_3 \cup \{(u^d, v^j)\}$
            $c_{(u^d, v^j)} \leftarrow C(\hat{\mathcal{P}}^c_{(u,v)})$
$X' \leftarrow \{t_1^D, \cdots, t_K^D\}$
if terminals X' are not reachable from s^0 in $\hat{L}_3$ **then**
    return FAIL
$I'_{X'} \leftarrow (\hat{L}_3, s^0, X')$
$\mathcal{T}' \leftarrow$ DST$(I'_{X'}, i)$
$\mathcal{T} \leftarrow \bigcup_{l(u^d, v^j) \in \mathcal{T}'} \mathcal{P}^{c_l}_{(u,v)}$
return $\mathcal{T}$

Figure 10: Algorithm SCALE

constraint D. Clearly, any such tree contains at least one link whose cost is c^* or more, hence c^* is a lower bound on $OPT(I_X)$. In addition, there exists a tree $\mathcal{T}$ that comprises of links whose cost is c^* or less and satisfies the constraint D. Since the number of links in $\mathcal{T}$ is at most N we conclude that $c^* \cdot N$ is an upper bound on $OPT(I_X)$.

Procedure BOUND performs a binary search on the values $c^1, c^2, \cdots, c^r$. At each iteration we check whether $c \leq c^*$, where c is the current estimate of c^*. For this purpose we remove from G all links whose cost is more than c and find a minimum delay path between s and each terminal in X . If all paths satisfy the delay constraint D, then $c \geq c^*$; otherwise $c < c^*$. The total number of iterations is $\mathcal{O}(\log r) = \mathcal{O}(\log N)$. At each iteration we execute a shortest path algorithm, namely Dijkstra's, whose complexity is $\mathcal{O}(M + N \log N)$. Thus, the total computational complexity of the procedure is $\mathcal{O}(M + N \log N) \log N$.

In order to find a better estimate, we use Procedure TEST, which gets B and ε as input and returns either a solution $\mathcal{T}$ to instance I_X or FAIL. If the procedure returns FAIL, then $OPT(I_X) > B$; otherwise, it is the case that $C(\mathcal{T}) \leq (1+\varepsilon)i(i-1)K^{1/i}B$. Procedure TEST invokes Algorithm SCALE for (I_X, i, B, ε). If Algorithm SCALE returns a tree $\mathcal{T}$ whose cost is at most $(1+\varepsilon)i(i-1)K^{1/i}B$, then Procedure TEST returns $\mathcal{T}$; otherwise Procedure TEST returns FAIL.

Lemma 14 *If Procedure* TEST *returns FAIL then $OPT(I_X) > B$; otherwise Procedure* TEST *returns a tree $\mathcal{T}$ such that $C(\mathcal{T}) \leq (1+\varepsilon)i(i-1)K^{1/i}B$.*

Proof: Suppose, by way of contradiction, that Algorithm SCALE returns FAIL and $OPT(I_X) \leq B$. Then, Algorithm SCALE is invoked with $B \geq OPT(I_X)$ and, by Lemma 12, it returns a solution $\mathcal{T}$ to I_X such that $C(\mathcal{T}) \leq i(i-1)K^{1/i}(OPT(I_X) + \varepsilon B) \leq (1+\varepsilon)i(i-1)K^{1/i}B$. Thus, Procedure TEST must return $\mathcal{T}$, which results in a contradiction. ■

We tighten the lower and upper bounds L, U by performing a binary search on the interval (L, U) on a logarithmic scale. In each iteration we invoke Procedure TEST with $B = \sqrt{\frac{U \cdot L}{(1+\varepsilon)i(i-1)K^{1/i}}}$. If Procedure TEST returns FAIL, then it is the case that $OPT(I_X) > B$, hence L is set to B. Otherwise, Algorithm SCALE returns a tree $\mathcal{T}$ whose cost is at most $(1+\varepsilon)i(i-1)K^{1/i}B$, hence we set $U = C(\mathcal{T})$. We also keep $\mathcal{T}$ as a possible solution for instance I_X.

Note that, if the ratio U/L is equal to β_j at iteration j, then at iteration $j+1$ we have

$$\beta_{j+1} = \frac{(1+\varepsilon)i(i-1)K^{1/i}B}{L} = \frac{U}{B} = \sqrt{(1+\varepsilon)i(i-1)K^{1/i}\beta_j}.$$

Thus, since $\beta_1 = N$, after $\mathcal{O}(\log\log N + \log\frac{1}{\varepsilon})$ iterations we have $\beta_j \leq (1+\varepsilon)^2 i(i-1)K^{1/i}$. Finally, we return the solution $\mathcal{T}$ to instance I_X such that $C(\mathcal{T}) = U$. Since $U \leq \beta_j L \leq (1+\varepsilon)^2 i(i-1)K^{1/i}OPT(I_X)$, we conclude that the cost of $\mathcal{T}$ is at most $(1+\varepsilon)^2 i(i-1)K^{1/i}$ times higher than the optimum.

5.6 Approximation Scheme

The above is summarized through a detailed description of the approximation scheme, namely Algorithm RST-3, specified in Fig. 11.

The following theorem establishes the complexity and performance guarantees of Algorithm RST-3.

Theorem 6 *Given an instance I_X of Problem RST, Algorithm* RST-3 *identifies, in $\mathcal{O}((\log\log N + \log\frac{1}{\varepsilon})\left(\frac{4N}{\varepsilon}\right)^{i-1}K^{3i-2})$ time, a solution tree $\mathcal{T}$ to I_X such that $C(\mathcal{T}) \leq (1+\varepsilon)i(i-1)K^{1/i}OPT(I_X)$.*

Algorithm RST-3 ($I_X(G, s, X, D), i, \varepsilon$**)**

input:

I_X - an instance of Problem RST

G - the graph

s- source node

$X = \{t_1, \cdots, t_K\}$ - the set of K terminals

D- the delay constraint

i- the level of the returned tree

ε- the approximation ratio

output:

$\mathcal{T}$- A solution to I_X.

$L, U, \hat{\mathcal{T}} \leftarrow$ BOUND$(I_X(G, s, X, D))$

do

$B \leftarrow \sqrt{\frac{U \cdot L}{(1+\varepsilon)i(i-1)|X|^{1/i}}}$

$\mathcal{T} \leftarrow$ TEST$(I_X(G, s, X, D), B, \varepsilon)$

if Procedure TEST returned FAIL **then** $L \leftarrow B$

else $U \leftarrow C(\mathcal{T})$, $\hat{\mathcal{T}} \leftarrow \mathcal{T}$

until $U/L \leq (1+\varepsilon)^2 i(i-1)|X|^{1/i}$.

return $\mathcal{T}$.

Procedure TEST$(I_X(G, s, X, D), B)$

$\mathcal{T} \leftarrow$ SCALE$(I_X(G, s, X, D), i, B)$

if Algorithm SCALE returned FAIL or $C(\mathcal{T}) \geq 2i(i-1)K^{1/i}B$ **then**

return FAIL

else

return $\mathcal{T}$

Procedure BOUND$(I_X(G(V, E), s, X, D))$

let $c^1 < c^2 < \cdots < c^r$ the distinct costs values of the links.

$low \leftarrow 1$; $high \leftarrow r$

while $low < high - 1$

$h \leftarrow \lfloor (high + low)/2 \rfloor$

$E' \leftarrow \{l | c_l \leq c^h\}$

Use Dijkstra's algorithm to compute a minimum delay path $\mathcal{P}_{(s,t_j)}$ in $G(V, E')$ between s and each $t_j \in X$

if for each $t_j \in X$ it holds that $\mathcal{P}_{(s,t_j} \leq D$ **then**

$high \leftarrow h$

$\hat{\mathcal{T}} = \cup_{t_j \in X} \mathcal{P}_{(s,t_j)}$

else

$low \leftarrow h$

$U \leftarrow N \cdot c^{high}$; $L \leftarrow c^{high}$;

return $L, U, \hat{\mathcal{T}}$;

Figure 11: Algorithm RST-3

Proof: Procedure BOUND computes obvious lower and upper bounds L and U on $OPT(I_X)$. As discussed above, the bounds remain valid during the execution of the loop that begins at line 2, and after the execution of this loop it holds that $\frac{U}{L} \leq (1+\varepsilon)^2 i(i-1)|X|^{1/i}$. The algorithm returns a tree $\mathcal{T}$ that satisfies $C(\mathcal{T}) \leq (1+\varepsilon)^2 i(i-1)|X|^{1/i} L \leq (1+\varepsilon)^2 i(i-1)|X|^{1/i} OPT(I_X)$. By invoking Algorithm RST-3 for $\frac{\varepsilon}{3}$ we can achieve an approximation ratio of $(1+\varepsilon)i(i-1)|X|^{1/i}OPT(I_X)$.

We proceed to analyze the computational complexity of Algorithm RST-3. As discussed above, the loop that begins at line 2 is executed $\mathcal{O}(\log\log N + \log\frac{1}{\varepsilon})$ times. At each iteration we invoke Procedure TEST. Procedure TEST, in turn, comprises of a single invocation of Algorithm SCALE for (I_X, i, B, ε), thus its running time is $\mathcal{O}((\log\log N+\log\frac{1}{\varepsilon})\left(\frac{(4K-2)N}{\varepsilon}\right)^{i-1} K^{2i-1})$. We conclude that the total running time of Algorithm RST-3 is

$$\mathcal{O}((\log\log N + \log\frac{1}{\varepsilon})\left(\frac{(4K-2)N}{\varepsilon}\right)^{i-1} K^{2i-1}) = \mathcal{O}((\log\log N + \log\frac{1}{\varepsilon})\left(\frac{4N}{\varepsilon}\right)^{i-1} K^{3i-2}).$$

■

Note 2 Again, in Algorithm RST-3, by substituting Algorithm DST with any approximation scheme for Problem DST (with some approximation ratio α), we obtain, through a T_3 reduction, a solution to Problem RST with an approximation ratio of $(1+\varepsilon)\alpha$. For example, by using the solution of [8], we can obtain a $(1+\varepsilon)$-optimal solution to Problem RST for a small number of terminals. ■

6 Discussion

In this paper, we have investigated the fundamental problem of finding minimum cost multicast trees under additive QoS constraints. Our major contributions are two efficient approximation schemes that identify, for any fixed $i > 0$ and $\varepsilon > 0$, a tree whose cost is at most $(1+\varepsilon)i(i-1)K^{1/i}$ times higher than the optimum, where K is the number of terminals.

The first scheme, implemented by Algorithm RST-2, is more efficient, but allows a small violation (by a factor of $(1+\varepsilon)$) of the QoS constraint. Specifically, it identifies, in $\mathcal{O}(\left(\frac{i\cdot N}{\varepsilon}\right)^{i-1} K^{2i-1})$ time, a tree such that the delay of every path between the source and any terminal is at most $(1+\varepsilon)D$, where D is the delay constraint. The second scheme, implemented by Algorithm RST-3, finds a tree that fully satisfies the QoS constraint and incurs a computational complexity of $\mathcal{O}((\log\log N+\log\frac{1}{\varepsilon})\left(\frac{N}{\varepsilon}\right)^{i-1} K^{3i-2})$, which is by a factor of $\mathcal{O}((\log\log N+\log\frac{1}{\varepsilon})K^{i-1})$ higher than that of Algorithm RST-2. To the best of our knowledge, the proposed schemes are the first solutions of provable performance to this fundamental multicast problem. Our schemes work in general network settings and topologies and they allow to find solutions with either no violation or at most a small violation of the QoS constraint.

Our major contribution is the concept of T-reductions, which allow to use *any* solution to the Directed Steiner Tree problem in order to obtain a corresponding solution to its RST version. For example, by using a T_3-reduction and the DST algorithm of [8], we obtain a polynomial, ε-optimal solution to Problem RST in the special case of a small number of terminals.

Appendix A: Proof of Theorem 2

In the appendix we present a detailed proof of Theorem 2. The proof follows [5] almost verbatim. We begin by defining a variant of Problem DST, which seeks a minimum cost tree that connects *part* of the terminals.

Problem (D-STEINER(K, s, X)) *Given a root $s \in V$, an integer K and a set $X \subseteq V$ of terminals with $|X| \geq K$, construct a tree rooted at s, spanning any K terminals in X and of minimum possible cost.*

Recall that the *density* of a tree $\mathcal{T}$ is the ratio of the cost of the tree to the number of terminals in $\mathcal{T}$. We denote the density of $\mathcal{T}$ by $\zeta(\mathcal{T})$. In addition, we denote by $\hat{C}$ the minimum cost such that, for each subset Y of X, holds $OPT^{(i)}(I'_Y(G,s,Y)) \leq |Y|^{1/i}\hat{C}$, *i.e.*,

$$\hat{C} = \max_{Y \subseteq X} \left\{ \frac{OPT^{(i)}(I'_Y(G,s,Y))}{|Y|^{1/i}} \right\}, \tag{A1}$$

where $OPT^{(i)}(I'_X)$ is the cost of an optimum i-level tree that solves instance I'_X of Problem DST.

Definition 9 (Partial Approximation) *An $f(K)$-partial approximation procedure for* D-STEINER(K, s, X) *is a procedure that constructs a tree $\mathcal{T}'$ rooted in s, spanning $1 \leq K' \leq K$ terminals in X such that $\zeta(\mathcal{T}') \leq f(K)\frac{\hat{C}}{K}$.*

Let A(K, s, X) be a partial approximation procedure for D-STEINER(K, s, X), we define the Algorithm B(K, s, X) for D-STEINER(K, s, X), as follows.

Definition 10 (Algorithm B) *Algorithm B(K, s, X) begins by invoking Algorithm A for (K, s, X). Let $\mathcal{T}_1$ be a tree returned by Algorithm A and let K_1 be the number of terminals in $\mathcal{T}_1$. If $K_1 = K$, Algorithm B terminates and returns a tree $\mathcal{T}_1$. Otherwise, B(K, s, X) returns the union of $\mathcal{T}_1$ and the tree returned by a recursive call to B$(K - K_1, s, X - X_1)$, where X_1 is the set of terminals spanned by $\mathcal{T}_1$.*

Lemma 15 *Given* A(K, s, X)*, an $f(K)$-partial approximation for* D-STEINER(K, s, X) *where $f(x)/x$ is a decreasing function of x, the algorithm* B(K, s, X) *returns a solution $\mathcal{T}$ for* D-STEINER(K, s, X) *of cost $C(\mathcal{T}) \leq g(K)\hat{C}$, where $g(K) = \int_0^K \frac{f(x)}{x}dx$.*

Proof: We will prove the claim by induction on K. The base case, $K = 1$, follows as $f(1) \leq \int_0^1 \frac{f(x)}{x}dx$ (by the decreasing property of $\frac{f(x)}{x}$). Suppose it is true for all values less than K. Suppose the call to A(K, s, X) returns a tree $\mathcal{T}_1$ rooted at s that spans K_1 terminals. Since A(K, s, X) is an $f(K)$-partial approximation solution, it holds that

$$\zeta(\mathcal{T}_1) = \frac{C(\mathcal{T}_1)}{K_1} \leq f(K)\frac{\hat{C}}{K} \tag{A2}$$

$$C(\mathcal{T}_1) \leq K_1 \frac{f(K)}{K}\hat{C} \leq \tag{A3}$$

$$\leq \left(\int_{K-K_1}^{K} \frac{f(x)}{x}dx \right) \hat{C}, \tag{A4}$$

where the last inequality follows from the decreasing property of $\frac{f(x)}{x}$. If $K_1 = K$, the algorithm returns $\mathcal{T}_1$. For this case, $C(\mathcal{T}_1) \leq g(K)\hat{C}$.

Suppose $K_1 \leq K$. Let X_1 be the set of terminals spanned by $\mathcal{T}_1$ and let $\mathcal{T}_2$ be the tree returned by the recursive call to $\mathrm{B}(K - K_1, s, X \setminus X_1)$. By the inductive hypothesis, $C(\mathcal{T}_2) \leq g(K - K_1)\hat{C}$, *i.e.*,

$$C(\mathcal{T}_2) \leq \left(\int_0^{K-K_1} \frac{f(x)}{x} dx \right) \hat{C} \tag{A5}$$

Adding (A4) and (A5), we get

$$C(\mathcal{T}_1) + C(\mathcal{T}_2) \leq g(K)\hat{C}$$

This proves that, for this case too, the algorithm returns a tree $\mathcal{T}$ whose cost is at most $g(K)\hat{C}$.

■

We denote by $\mathcal{T}^{(i)}_{OPT}(K, s, X)$, the optimum i-level tree that solves D-STEINER(K, s, X). We denote the cost and density of $\mathcal{T}^{(i)}_{OPT}(K, s, X)$ by $C^{(i)}_{OPT}(K, s, X)$ and $\zeta^{(i)}_{OPT}(K, s, X)$, respectively.

The following lemma is taken from [5].

Lemma 16 *The trees $\mathcal{T}_{\mathrm{BEST}}$ chosen by the Algorithm A_i, $i \geq 2$ in line 11 (Fig. 2) have the following property: $\zeta(\mathcal{T}_{\mathrm{BEST}}) \leq (i-1)\zeta^{(i)}_{OPT}(K, s, Y)$, where K and Y refer to the current values being used by the Algorithm A_i.*

We are no ready to prove Theorem 2.

Theorem 2 Given are an instance $I'_X = (G, s, X)$ of Problem DST, $K = |X|$ and an integer $i, 1 \leq i \leq \log |X|$. Let $\hat{C}$ be the minimum cost such that for each subset Y of X holds $OPT^{(i)}(I'_Y(G, s, Y)) \leq |Y|^{1/i}\hat{C}$, *i.e.*,

$$\hat{C} = \max_{Y \subseteq X} \left\{ \frac{OPT^{(i)}(I'_Y(G, s, Y))}{|Y|^{1/i}} \right\}.$$

Then, Algorithm DST returns a tree $\mathcal{T}$ that satisfies $C(\mathcal{T}) \leq i(i-1)K^{1/i}\hat{C}$.

Proof: We divide the execution of $A_i(K, s, X)$ into stages, each stage corresponds to one iteration of the outer loop (line 4, Fig. 2). Let X_j be the set of unsatisfied terminals, *i.e.*, terminals that have not been yet connected by the tree. We denote $K_j = |X_j|$. Lemma 16 implies that, at stage j, Algorithm A_i identifies a tree with density no worse than $(i-1)\frac{C^{(i)}_{OPT}(K_j, s, X_j)}{K_j}$. Since Problem D-STEINER$(|Y|, s, Y)$ is a generalization of Problem DST for $I'_Y(G, s, Y)$, it holds that $C^{(i)}_{OPT}(|Y|, s, Y) \leq OPT^{(i)}(I'_Y(G, s, Y))$. Hence, the density of the tree identified at stage j by Algorithm A_i is no worse than

$$(i-1)\frac{C^{(i)}_{OPT}(K_j, s, X_j)}{K_j} \leq (i-1)\frac{OPT^{(i)}(I'_{X_j}(G, s, X_j))}{K_j} \leq (i-1)\frac{K_j^{1/i}}{K_j}\hat{C}.$$

Hence, each stage behaves like an $(i-1)K_j^{1/i}$-partial approximation to D-STEINER(K_j, s, X_j). Using Lemma 15 we obtain the following bound on the cost $C(\mathcal{T})$ of tree $\mathcal{T}$ identified by $A_i(K, s, X)$.

$$C(\mathcal{T}) \leq (i-1)\hat{C} \int_0^K \frac{y^{1/i} dy}{y} = i(i-1)K^{1/i}\hat{C}.$$

■

References

[1] T. Alrabiah and T. F. Znati. Low-Cost, Bounded-Delay Multicast Routing for QoS-Based Networks. In *Proceedings of IEEE ICCCN'98*, Lafayette, Lousiana, October 1998.

[2] J. Bar-Ilan, G. Kortsarz, and D. Peleg. Generalized Submodular Cover Problems and Applications. In *Proceedings of the fourth Israel Symposium on Theory of Computing and Systems*, Jerusalem, Israel, June 1996.

[3] Y. Bernet et al. A Framework for Differentiated Services. Internet draft, Internet Engineering Task Force, February 1999.

[4] S. Blake et al. An Architecture for Differentiated Services. - RFC No. 2475. Internet Engineering Task Force, December 1998.

[5] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation Algorithms for Directed Steiner Problems. *Journal of Algorithms*, 33(1):73–91, October 1999.

[6] F. Ergun, R. Sinha, and L. Zhang. An Improved FPTAS for Restricted Shortest Path. *Information Processing Letters*, 83(5):237–293, September 2002.

[7] A. Erzin. Polynomial Algorithm for Bandwidth-Delay-Constrained Multicast Routing Problem. In *Proceedings of International Conference on Integer Programming and Combinatorial Optimization*, Houston, Texas, USA, 1998.

[8] J. Feldman and M. Ruhl. The Directed Steiner Network Problem is Tractable for a Constant Number of Terminals. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS '99)*, New York, New York, October 1999.

[9] M.R. Garey and D.S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1979.

[10] A. Goel, K.G. Ramakrishnan, D. Kataria, and D. Logothetis. Efficient Computation of Delay-Sensitive Routes from One Source to All Destinations. In *Proceedings of IEEE INFOCOM'01*, Anchorage, Alaska, April 2001.

[11] R. Hassin. Approximation Schemes for the Restricted Shortest Path Problem. *Mathematics of Operations Research*, 17(1):36–42, February 1992.

[12] S. Hougardy and H. J. Prömel. A 1.598 Approximation Algorithm for the Steiner Problem in Graphs. In *Proceedings of Symposium on Discrete Algorithms (SODA)*, 1999.

[13] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicast Routing for Multimedia Communication. *IEEE/ACM Transactions on Networking*, 1(3):286–292, 1993.

[14] G. Kortsarz and D. Peleg. Approximating the Weight of Shallow Steiner Trees. *Discrete Applied Mathematics*, 93:265–285, 1999.

[15] D.H. Lorenz and D. Raz. A Simple Efficient Approximation Scheme for the Restricted Shortest Path Problem. *Operations Research Letters*, 28(5):213–219, June 2001.

[16] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III. Bicriteria Network Design Problems. *Journal of Algorithms*, 28(1):142–171, 1998.

[17] Private Network-Network Interface Specification v1.0 (PNNI). ATM Forum Technical Committee, March 1996.

[18] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC No. 3031. Internet Engineering Task Force, January 2001.

[19] R. Sriram, G. Manimaran, and C. Ram. Algorithms for Delay-constrained Low-cost Multicast Tree Construction. *Computer Communications*, 21(18):1693–1706, November 1998.

[20] A. Striegel and G. Manimaran. A Survey of QoS Multicasting Issues. *IEEE Communications*, 40(6):82–87, June 2002.

[21] Q. Sun and H. Langendörfer. An Efficient Delay-Constrained Multicast Routing Algorithm. *Journal of High Speed Networks*, 7(1):43–55, 1998.

[22] B. Wang and J. Hou. Multicast Routing and its QoS Extension: Problems, Algorithms, and Protocols. *IEEE Network*.

[23] G.L. Xue. Provably Good Approximations to Minimum Cost Delay-Constrained Multicast Trees. In *Proceedings of IEEE International Conference on Computer Communications and Networks, ICCCN'99*, Natick-Boston, MA, October 1999.

[24] A. Zelikovsky. A Series of Approximation Algorithms for the Acyclic Directed Steiner Tree Problem. *Algorithmica*, 18(1):99–110, 1997.

[25] Q. Zhu, M. Parsa, and J. J. Garcia-Luna-Aceves. A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting. In *Proceedings of IEEE INFOCOM'95*, Boston, Massachusetts, April 1995.