Approximation Algorithms for Node-Weighted Directed Steiner Problems*

Gianlorenzo D'Angelo[1][0000−0003−0377−7037] and Esmaeil Delfaraz[2][0000−0002−9642−7652]

[1] Gran Sasso Science Institute (GSSI), L'Aquila, Italy,
gianlorenzo.dangelo@gssi.it
[2] Centre of EXcellence EX-Emerge, University of L'Aquila, Italy,
esmaeil.delfarazpahlevanloo@univaq.it

Abstract. Guha et al. [STOC, 1999] and Moss and Rabani [SIAM J. Comput., 2007] introduced two variants of the Steiner problem in undirected graphs in which the nodes are associated with two values, called costs and prizes. In the *budgeted rooted node-weighted Steiner tree* problem, we are given an undirected graph G with n nodes, a predefined node r, costs and prizes associated to the nodes of G, and a budget B. The aim is to find a tree in G rooted at r such that the total cost of its nodes is at most B and the total prize is maximized. In the *quota rooted node-weighted Steiner tree* problem, we are given a quota Q, instead of the budget, and we aim at minimizing the cost of a tree rooted at r whose overall prize is at least Q.

If the graph is undirected both problems can be approximated within polylogarithmic factors, possibly with a constant-factor budget violation, [Bateni et al. SIAM J. Comput., 2018][Moss and Rabani SIAM J. Comput., 2007]. If the graph is directed, the budgeted problem can be approximated within a $O\left(\frac{\log n'}{\log\log n'}\right)$ factor in quasi-polynomial time, where n' is the number of vertices in an optimal solution [Ghuge and Nagarajan SODA 2020], and within a factor $O(\frac{1}{\epsilon^3}\sqrt{B})$ with a budget violation of $1+\epsilon$, for any $\epsilon \in (0, 1]$, in polynomial time [D'Angelo, Delfaraz, and Gilbert ISAAC 2022].

In this paper, we provide two algorithms for the budgeted and quota problems on directed graphs that achieve, respectively, an $O(\frac{1}{\epsilon^2}n^{2/3}\ln n)$-approximation at the cost of a budget violation of a factor of at most

*This work has been partially funded by the European Union - NextGenerationEU under the Italian Ministry of University and Research (MUR) National Innovation Ecosystem grant ECS00000041 - VITALITY – CUP: D13C21000430001; by PNRR MIUR project GAMING "Graph Algorithms and MinINg for Green agents" (PE0000013, CUP D13C24000430001); by ARS01_00540 - RASTA project, funded by the Italian Ministry of Research PNR 2015-2020; by Project EMERGE: innovation agreement between Ministry of Economical Development, Abruzzo Region, Radiolabs, Elital, Leonardo, Telespazio, University of L'Aquila and Centre of EXcellence EX-Emerge, funded by Italian Government under CIPE n. 70/2017 (Aug. 7, 2017); The first author acknowledges the support of the MUR (Italy) Department of Excellence 2023–2027

$1+\epsilon$, for any $\epsilon \in (0, 1]$, and an approximation factor of $O(n^{2/3} \ln n)$ at the cost of a violation of the quota constraint by a factor of at most 2. We develop a technique resorting on a standard flow-based linear programming relaxation to compute a tree with good trade-off between prize and cost, which allows us to provide polynomial time approximation algorithms for both problems. We provide the first approximation algorithms for these problems that run in polynomial time and guarantee an approximation factor depending only on the number of vertices n.

1 Introduction

Prize Collecting Steiner Tree (**PCST**) refers to a wide class of combinatorial optimization problems, involving variants of the *Steiner tree* problem and *traveling salesperson* problem, with many practical applications in computer and telecommunication networks, VLSI design, computational geometry, wireless mesh networks, and cancer genome studies [5, 10, 16, 20, 25].

In **PCST**, we are given a (directed) graph G, two functions modelling costs and prizes (or penalties) associated to the edges and/or to the nodes of the graph, and we want to find a connected subgraph T of G (usually a tree or an out-tree) which optimizes an objective function defined as a combination of its cost and prize and/or is subject to some constraints on its cost and prize. By considering different constraints and objective functions, one obtains distinct optimization problems. In *budgeted* problems, we are given a budget B and we require that the cost of T is at most B and its prize is maximum. In *quota* problems, we require the prize of T to be at least some quota Q and its cost to be minimum. Additional constraints can be required, for example in *rooted* variants we are given a specific node, called *root*, which has to be part of T and reach all the nodes in T.

While there is a vast literature providing approximation algorithms for many variants of **PCST** on undirected graphs, e.g. [1, 2, 11, 13, 15, 17, 18, 23], the case of directed graphs received less attention [4, 7, 12, 20, 26]. Furthermore, prize collecting Steiner tree problems are usually much harder on directed graphs than on undirected graphs. A well-known example is the *Steiner tree* problem, for which there is a simple polynomial time 2-approximation algorithm for its undirected version, but for its directed version, unless $NP \subseteq \bigcap_{0<\epsilon<1} \text{ZPTIME}(2^{n^\epsilon})$ or the Projection Game Conjecture is false, there is no quasi-polynomial time algorithm that achieves an approximation ratio of $o(\frac{\log^2 k}{\log \log k})$ [14], where k is the number of terminal nodes.

In this paper, we focus on *budgeted* and *quota* problems on *directed* graphs and, motivated by applications in the deployment of wireless relay networks [10, 20] and in the detection of mutated pathways in cancer [16, 25], we study *node-weighted* problems, that is both costs and prizes are associated to the nodes of the graph. We consider the more general *rooted* variant of this problems.

In both budgeted and quota variants, we are given a directed graph $D = (V, A)$ with $|V| = n$, two nonnegative real-valued functions, namely, cost $c(v)$

and prize $p(v)$ that are associated to each vertex $v \in V$, and a root vertex r. In the *Budgeted Directed Rooted Tree problem* (**B-DRT**), we are given a budget B, and we aim at finding an out-tree (a.k.a. out-arborescence) T of D rooted at r such that the sum of the costs of its vertices is at most B and the sum of the prizes of its vertices is maximum. In the *Quota Directed Rooted Tree problem* (**Q-DRT**), we are given a quota Q, and aim at finding an out-tree T of D rooted at r whose total prize is at least Q and total cost is minimum.

Related work. Guha et al. [15] introduced the undirected version of **B-DRT**, called **B-URT**. They gave a polynomial time $O(\log^2 n)$-approximation algorithm that violates the budget constraint by a factor of at most 2. Moss and Rabani [22] improved the approximation factor to $O(\log n)$, with the same budget violation. Their algorithm is based on a Lagrangian multiplier preserving $O(\ln n)$-approximation algorithm, proposed in the same paper, for the problem of minimizing the cost of the nodes in the resulting tree plus the prizes of vertices not spanned by the tree. We call this problem **PC-URT**. However, Könemann et al. [18] pointed out a flaw in their algorithm for **PC-URT** and proposed an alternative Lagrangian multiplier preserving algorithm with the same guarantee. Later, Bateni et al. [2] proposed an $O(\frac{1}{\epsilon^2}\log n)$-approximation algorithm for **B-URT** which requires a budget violation of only $1+\epsilon$, for any $\epsilon \in (0, 1]$. Kortsarz and Nutov [19] showed that the unrooted version of **B-URT**, so does **B-URT**, admits no $o(\log\log n)$-approximation algorithm, unless $NP \subseteq DTIME(n^{\text{polylog}(n)})$, even if the algorithm is allowed to violate the budget constraint by a factor equal to a universal constant. Ghuge and Nagarajan [12] provided a tight quasi-polynomial time $O(\frac{\log n'}{\log\log n'})$-approximation algorithm for the edge-cost version of **B-DRT**, where n' is the number of vertices in an optimal solution, the prize function is a monotone submodular function on subsets of nodes, and the edge costs are positive integers. D'Angelo et al. [7] provided a polynomial time $O(\frac{1}{\epsilon^2}\sqrt{B})$-approximation algorithm for the same problem that violates the budget constraint by a factor of $1+\epsilon$, where $\epsilon \in (0, 1]$. By using a simple reduction, it is easy to see that these results for the directed graphs with edge-costs also hold for their node-cost version. Bateni et al. [2] showed that the integrally gap of the standard flow-based LP for **B-URT**, so is for **B-DRT**, is unbounded.

To the best of our knowledge, the quota problem on directed graphs has not been studied explicitly before. For the node-weighted quota problem on undirected graphs, the algorithm by Moss and Rabani [22] provides an $O(\log n)$-approximation algorithm by using as a black-box the algorithm of Könemann et al. [18] (or the one by Bateni et al. [2]) for **PC-URT**.

It is worth pointing out that by binary searching the budget (resp. quota) space, one can also show that any α-approximation algorithm for a budgeted (resp. quota) problem results in a $(1+\epsilon)$-approximation ($(1-\epsilon)$-approximation) for its quota (budgeted) version, for any $\epsilon > 0$, that violates the quota (budget) constraint by a factor of at most α.

Our results. We introduce a new technique, resorting on flow-based linear programming relaxations, which allows us to find out-trees with a good trade-off between cost and prize. By computing suitable values of quota and budget and applying known tree trimming procedures, we achieve good bicriteria approximations for both **B-DRT** and **Q-DRT**. For **B-DRT**, we introduce a polynomial-time approximation algorithm that violates the budget constraint by a factor of at most $1+\epsilon$, for $\epsilon \in (0,1]$ and achieve an approximation factor of $O(\frac{1}{\epsilon^2} n^{2/3} \ln n)$. For **Q-DRT** we present an $O(n^{2/3} \ln n)$-approximation polynomial-time algorithm that violates the quota constraint by a factor of at most 2.

To the best of our knowledge, our techniques yield the first approximation algorithms for these problems whose approximation ratio depends only on the number of vertices n and whose running time is polynomial in the input size. Indeed, the algorithms in [7] and [12] consider a more general version of our budgeted problem, in which the prize function is monotone and submodular. However, the approximation guarantee of the algorithm in [7] for the budgeted problem depends on the budget B, which can be much higher than n, while the algorithm in [12] requires quasi-polynomial running time. To the best of our knowledge, there are no known approximation algorithms for quota problems in directed graphs or submodular prize functions. Moreover, our algorithms allow the costs to be any non-negative real numbers, whereas the algorithms in [7] and [12] require the costs to be strictly positive integers.

2 Notation and Problem Statement

For an integer s, let $[s] := \{1, \ldots, s\}$. Let $D = (V, A)$ a directed graph with a distinguished vertex $r \in V$ and $c : V \to \mathbb{R}^{\geq 0}$ be a nonnegative cost function on nodes.

A *directed path* is a directed graph made of a sequence of distinct vertices $(v_1, \ldots, v_s)$ and a sequence of directed edges (v_i, v_{i+1}), $i \in [s-1]$. An *out-tree* (a.k.a. out-arborescence) is a directed graph in which there is exactly one directed path from a specific vertex r, called *root*, to each other vertex. If a subgraph T of a directed graph D is an out-tree, then we say that T is an out-tree of D. For simplicity of reading, we may will refer to out-trees simply as trees.

Given two nodes $u, v \in V$, the cost of a path from u to v in D is the sum of the cost of its nodes. A directed path from u to v with the minimum cost is called a *shortest path* and its cost, denoted by $dist(u, v)$, is called the *distance* from u to v in D. Let F be the maximum distance from r to a node in V, $F := \max_{v \in V}\{dist(r, v)\}$. Let $B \in \mathbb{R}^{>0}$, a graph D is called *B-proper* for the vertex r if $dist(r, v) \leq B$ for any v in D.

For any subgraph D' of D, we denote by $V(D')$ and $A(D')$ the set of nodes and edges in D', respectively. Given a subset $S \subseteq V$ of nodes, $D[S]$ denotes the graph induced by set S, i.e., $A(D[S]) = \{(u, v) \in A | u, v \in S\}$.

In what follows, we state the problems we consider in this paper. Let $D = (V, A)$ be a directed graph with n nodes, $c : V \to \mathbb{R}^{\geq 0}$ be a cost function on

nodes, $p : V \to \mathbb{R}^{\geq 0}$ be a prize function on nodes, $r \in V$ be a root vertex. We consider two variants of node-weighted **PCST**.

1. The Quota Directed Rooted Tree problem (**Q-DRT**): Given a quota $Q \in \mathbb{R}^{>0}$, find an out-tree T of D rooted at r that minimizes $c(T) = \Sigma_{v \in V(T)} c(v)$ subject to $p(T) = \Sigma_{v \in V(T)} p(v) \geq Q$.
2. The Budgeted Directed Rooted Tree problem (**B-DRT**): Given a budget $B \in \mathbb{R}^{>0}$, find an out-tree T of D rooted at r that maximizes $p(T) = \Sigma_{v \in V(T)} p(v)$ subject to $c(T) = \Sigma_{v \in V(T)} c(v) \leq B$.

Bicriteria approximation. For $\alpha, \beta \geq 1$, a bicriteria (β, α)-approximation algorithm for **B-DRT** (resp. **Q-DRT**) is one that, for any instance I_B (resp. I_Q) of the problem, returns a solution Sol_{I_B} (resp. Sol_{I_Q}) such that $p(Sol_{I_B}) \geq \frac{OPT_{I_B}}{\alpha}$ (resp. $c(Sol_{I_Q}) \leq \alpha OPT_{I_Q}$) and $c(Sol_{I_B}) \leq \beta B$ (resp. $p(Sol_{I_Q}) \geq \frac{Q}{\beta}$), where OPT_{I_B} (resp. OPT_{I_Q}) is the optimum for I_B (resp. I_Q).

3 Directed Rooted Tree Problems

In this section we present a polynomial time bicriteria $\big(2, O(n^{2/3} \ln n)\big)$-approximation algorithm for **Q-DRT** and a polynomial time bicriteria $\Big(1 + \epsilon, O(\frac{n^{2/3} \ln n}{\epsilon^2})\Big)$-approximation algorithm for **B-DRT**, where ϵ is an arbitrary number in $(0, 1]$. Through the section we let $I_Q =< D = (V, A), c, p, r, Q >$ and $I_B =< D = (V, A), c, p, r, B >$ be two instances of **Q-DRT** and **B-DRT**, respectively, and we let T^*_Q and T^*_B be two optimal solutions for I_Q and I_B, respectively. Both algorithms for **Q-DRT** and **B-DRT** use the same technique and can be summarized in the following three steps:

1. We define a set of linear constraints, denoted as (DRT), over fractional variables, that takes a given quota Π and budget Λ as parameters and admits a feasible solution if there exists a subtree T of D rooted at r such that $p(T) \geq \Pi$ and $c(T) \leq \Lambda$. In particular, an optimal tree T^*_Q for I_Q has prize $p(T^*_Q) \geq Q$ and cost $c(T^*_Q)$, therefore the linear constraints (DRT) where parameters Π and Λ are set to $\Pi = Q$ and $\Lambda = c(T^*_Q)$ admits a feasible solution. It follows that the minimum value OPT_Q of Λ for which the set of linear constraints (DRT) admits a feasible solution when $\Pi = Q$ is a lower bound to $c(T^*_Q)$, i.e. $OPT_Q \leq c(T^*_Q)$. Similarly, the maximum value OPT_B of Π for which the set of linear constraints (DRT) admits a feasible solution when $\Lambda = B$ is an upper bound to $p(T^*_B)$, i.e. $OPT_B \geq p(T^*_B)$.
2. We give a polynomial time algorithm that takes as input a feasible solution to (DRT) and computes a subtree T of D rooted at r such that $c(T) = O((F + \Lambda) n^{2/3} \ln n)$ and $p(T) \geq \Pi/2$, where $F := \max_{v \in V}\{dist(r, v)\}$.
3. **Q-DRT**: We first compute a solution for (DRT) for which $\Pi = Q$ and Λ is minimum, i.e. $\Lambda = OPT_Q$. Then we use such a solution for (DRT) as input to the algorithm in the previous step and obtain a tree T such that $c(T) = O((F + OPT_Q) n^{2/3} \ln n)$ and $p(T) \geq Q/2$. We can show that we can

assume w.l.o.g. that $F \leq (1+\epsilon)c(T_Q^*)$, for any $\epsilon > 0$. Since $OPT_Q \leq c(T_Q^*)$, then the computed tree T satisfies $c(T) = O((1+\epsilon)c(T_Q^*)n^{2/3}\ln n)$ and $p(T) \geq Q/2$.

B-DRT: We compute a solution for (DRT) for which $\Lambda = B$ and Π is maximum, i.e. $\Pi = OPT_B \geq p(T_Q^*)$ and use such a solution as input to the algorithm in the previous step. Since we can assume w.l.o.g. that $F \leq B$, we obtain a tree T such that $c(T) = O(Bn^{2/3}\ln n)$ and $p(T) \geq OPT_B/2 \geq p(T_B^*)/2$.

The tree T may violate the budget constraint by a large factor, however the ratio γ between its prize and its cost is $\Omega\left(p(T_B^*)/(Bn^{2/3}\ln n)\right)$. The bound on the value of γ allows us to apply the trimming process given in [2] to T and obtain another tree $\hat{T}$ with cost $\frac{\epsilon}{2}B \leq c(\hat{T}) \leq (1+\epsilon)B$, for any $\epsilon \in (0,1]$, and prize-to-cost ratio $\frac{p(\hat{T})}{c(\hat{T})} = \epsilon\frac{\gamma}{4}$. Tree $\hat{T}$ achieves an approximation ratio of $O(\frac{n^{2/3}\ln n}{\epsilon^2})$ at the cost of a budget violation of $1+\epsilon$, for any $\epsilon \in (0,1]$.

In the following, we will detail each step of our algorithms.

Step 1: Bounding the optimal cost and prize

Here we define a set of linear constraints that admits a feasible solution if there exists a tree T rooted in r in D such that $c(T) \leq \Lambda$ and $p(T) \geq \Pi$, for given parameters $\Lambda, \Pi \in \mathbb{R}^{>0}$. For each $v \in V$, let $p_v = p(v)$, $c_v = c(v)$, and $\mathcal{P}_v$ be the set of simple paths in D from r to v. Our set of constraints (DRT) is defined as follows.

$$\Sigma_{v\in V} x_v p_v \geq \Pi \tag{D.1}$$

$$\Sigma_{v\in V} x_v c_v \leq \Lambda \tag{D.2}$$

$$\Sigma_{P\in\mathcal{P}_v} f_P^v = x_v, \qquad \forall v \in V \setminus \{r\} \tag{D.3}$$

$$\Sigma_{P\in\mathcal{P}_v : w\in P} f_P^v \leq x_w, \qquad \forall v \in V \setminus \{r\} \text{ and } \forall w \in V \setminus \{v\} \tag{D.4}$$

$$0 \leq x_v \leq 1, \qquad \forall v \in V$$

$$0 \leq f_P^v \leq 1, \qquad \forall v \in V \setminus \{r\}, P \in \mathcal{P}_v$$

We use variables f_P^v and x_v, for each $v \in V$ and $P \in \mathcal{P}_v$, where f_P^v represents the amount of flow sent from r to v using path P and x_v represents both the capacity of node v and the overall amount of flow sent from r to v.

The constraints in (DRT) are as follows. Constraints (D.1) and (D.2) ensure that any feasible (fractional) solution to (DRT) has a prize at least Π and a cost at most Λ. Constraints (D.3) and (D.4) formulate a connectivity constraint through standard flow encoding, that is they ensure that the nodes v with $x_v > 0$ induce subgraph in which all nodes are reachable from r. In particular, constraint (D.3) ensures that the amount of flow that is sent from r to any vertex v must be equal to x_v and constraint (D.4) ensures that the total flow from r to v passing through a vertex w cannot exceed x_w.

Note that (DRT) has an exponential number of variables. However, it can be solved efficiently as we only need to find, independently for any $v \in V \setminus \{r\}$,

a flow from r to v of value x_v that does not exceed the capacity x_w, for each vertex $w \in V \setminus \{r, v\}$.

We can show that a feasible solution to (DRT) can be used to find a lower bound to an optimal solution for **Q-DRT** and an upper bound to an optimal solution for **B-DRT**. The next lemma shows that if there exists a tree T rooted in r in D such that $c(T) \leq \Lambda$ and $p(T) \geq \Pi$, then there exists a feasible solution x for (DRT).

Lemma 1. *Given a directed graph $D = (V, A)$, $r \in V$, $c : V \to \mathbb{R}^{\geq 0}$, $p : V \to \mathbb{R}^{\geq 0}$, $\Lambda \in \mathbb{R}^{>0}$ and $\Pi \in \mathbb{R}^{>0}$, if there exists a tree T rooted in r in D such that $c(T) \leq \Lambda$ and $p(T) \geq \Pi$, then there exists a feasible solution x for* (DRT).

Proof. Let us consider a solution to (DRT) in which $x_v = 1$ for all $v \in V(T)$, while x_v is set to 0 for all $v \notin V(T)$. As $p(T) \geq \Pi$ and $c(T) \leq \Lambda$, then the quota and budget constraints (D.1)–(D.2) are satisfied. Since T is connected and for any $v \in V(T)$, there exists only one path P from r to v in T, constraints (D.3) and (D.4) are satisfied by setting $f_P^v = 1$ and any other flow variable to 0. □

Since T_Q^* is a feasible solution for I_Q, we have that $p(T_Q^*) \geq Q$, which implies that there exists a feasible solution to (DRT) when parameters Λ and Π are set to $\Lambda = c(T_Q^*)$ and $\Pi = Q$. Hence, the optimum OPT_Q to the linear program of minimizing Λ subject to constraints (DRT) in which parameter Π is set to $\Pi = Q$, gives a lower bound to $c(T_Q^*)$, i.e. $OPT_Q \leq c(T_Q^*)$. Similarly, the optimum OPT_B to the linear program of maximizing Π subject to constraints (DRT) with $\Lambda = B$, gives an upper bound to $p(T_B^*)$, $OPT_B \geq p(T_B^*)$. We denote these two linear programs by **Q-DRT-LP** and **B-DRT-LP**, respectively.

Step 2: Finding a good tree from a feasible fractional solution to (DRT)

Here we elaborate the second step and show how to compute a tree with a good trade-off between prize and cost starting from a feasible fractional solution to the set of constraints (DRT).

Theorem 1. *Given a feasible solution x to* (DRT)*, then there exists a polynomial time algorithm that computes a tree T rooted at r such that $c(T) = O((F + \Lambda)n^{2/3} \ln n)$ and $p(T) \geq \Pi/2$.*

We now prove Theorem 1. Let x be a feasible solution for (DRT) and let $S \subseteq V$ be the set of vertices v with $x_v > 0$, i.e., $S = \{v \in V : x_v > 0\}$. We partition S into two subsets $S_1, S_2 \subseteq S$, where $S_1 = \{v \in S | x_v \geq n^{-1/3}\}$ and $S_2 = \{v \in S | x_v < n^{-1/3}\}$. We first focus on nodes in S_1 and, in the following lemma, we show how to compute a tree T rooted at r spanning all vertices in S_1 with cost $c(T) = O((F + \Lambda)n^{2/3} \ln n)$.

Lemma 2. *There exists a polynomial time algorithm that finds a tree T rooted at r spanning all vertices in S_1 with cost $c(T) = O((F + \Lambda)n^{2/3} \ln n)$.*

Proof. Let $U := \{v \in S : x_v \geq n^{-2/3}\}$. We call a vertex $v \in S_1$ a *cheap vertex* if there exists a path from r to v in $D[U]$. We call a vertex $v \in S_1$ an *expensive vertex* otherwise. Note that r belongs to U if $U \neq \emptyset$ since we need to send $n^{-2/3}$ amount of flow from r to any vertex in U by constraint (D.3). Let CH and EX be the sets of cheap and expensive vertices, respectively.

In the following, we first show that we can compute in polynomial time two trees T^{CH} and T^{EX} spanning all nodes in CH and EX, respectively, and having cost $c(T^{CH}) = O(\Lambda n^{2/3})$ and $c(T^{EX}) = O((F+\Lambda)n^{2/3}\ln n)$, then we show how to merge the two trees into a single tree with cost $O((F+\Lambda)n^{2/3}\ln n)$.

We first focus on T^{CH}. By definition, all vertices in CH are reachable from r through paths that contain only vertices in U. Thus, for each $v \in CH$, we compute a shortest path from r to v in $D[U]$ and find a tree T^{CH} rooted at r spanning all and only the vertices in the union of these shortest paths. Since $\Sigma_{u\in U} x_u c_u \leq \Lambda$ (by constraint (D.2)) and $x_u \geq n^{-2/3}$ for any $u \in U$ (by definition of U), then $c(U) = \Sigma_{u\in U} c_u \leq \Lambda n^{2/3}$. Hence $c(T^{CH}) \leq c(U) = O(\Lambda n^{2/3})$.

Now we show that there exists a tree T^{EX} rooted at r spanning all the expensive vertices EX with cost $c(T^{EX}) = O((F+\Lambda)n^{2/3}\ln n)$ and that we can compute T^{EX} in polynomial time. The algorithm to build T^{EX} can be summarized as follows. We first compute, for each $v \in EX$, the set X_v of vertices $w \in S \setminus U$ for which there exists a path from w to v that uses only vertices in $U \cup \{w\}$. Then we compute a small-size hitting set X' of all X_v, i.e. $X' = \arg\min\{|Y| : Y \subseteq \bigcup_{v\in EX} X_v \text{ and } \forall v \in EX,\ Y \cap X_v \neq \emptyset\}$. Finally, we connect r to the vertices of X' and the vertices of X' to those in EX in such a way that each node v in EX is reached from one of the vertices $x \in X'$ that hits X_v, i.e., $x \in X_v$. The bound on the cost of T^{EX} follows from the size of X' and from the cost of nodes in U. We now detail on the construction of T^{EX} and its cost analysis.

Let $U' \subseteq S$ be the set of all vertices w with $x_w < n^{-2/3}$, i.e., $U' = S \setminus U$. For any expensive vertex $v \in EX$, we define X_v as the set of vertices w in U' such that there exists a path from w to v in $D[U \cup \{w\}]$. Note that for any $w \in X_v$, v is reachable from w through a path P such that $V(P)\setminus\{w\} \subseteq U$, i.e., $V(P)\setminus\{w\}$ only contains vertices from U. The following claim gives a lower bound on the size of X_v, for each $v \in EX$.

Claim 1. $|X_v| \geq n^{1/3}$, for each $v \in EX$.

Proof. We know that (i) the amount of flow that each vertex $v \in S_1$ should receive is at least $n^{-1/3}$ (by definition of S_1 and constraint (D.3) of (DRT)), (ii) any path P from r to any $v \in EX$ in the graph $D[S]$ contains at least one vertex $w \in U'$ (by definition of expensive vertices), and (iii) in any path P from r to any $v \in EX$, the node $w \in U'$ in P that is closest to v is a member of X_v, i.e. $w \in X_v$ (by definition of X_v), therefore any flow from r to v should pass through a vertex $w \in X_v$. This implies that the vertices in X_v must send at least $n^{-1/3}$ amount of flow to v in total. Formally, we have

$$\begin{aligned} n^{-1/3} \leq x_v = \Sigma_{P\in\mathcal{P}_v} f_P^v &\leq \Sigma_{w\in X_v} \Sigma_{P\in\mathcal{P}_v : w\in P} f_P^v \\ &\leq \Sigma_{w\in X_v} x_w \leq \Sigma_{w\in X_v} n^{-2/3} = |X_v| n^{-2/3}, \end{aligned}$$

which implies that $|X_v| \geq n^{1/3}$. Note that the first inequality follows from the definition of S_1, the first equality follows from constraint (D.3) of (DRT), the second inequality is due to the fact that, by definition of X_v, any path $P \in \mathcal{P}_v$ contains a vertex $w \in X_v$, the third inequality is due to constraint (D.4) of (DRT), the last inequality is due to $x_w < n^{-2/3}$, for each $w \in U'$, and $X_v \subseteq U'$. This concludes the proof of the claim. □

We use the following well-known result (see e.g. Lemma 3.3 in [3]) to find a small set of vertices that hits all the sets X_t, for all $t \in EX$.

Claim 2. Let V' be a ground set of M elements and $\Sigma = (X'_1, \ldots, X'_N)$ be a collection of subsets of V' such that $|X'_i| \geq R$, for each $i \in [N]$. There is a deterministic algorithm which runs in polynomial time in N and M and finds a subset $X' \subseteq V'$ with $|X'| \leq (M/R) \ln N$ and $X' \cap X'_i \neq \emptyset$ for all $i \in [N]$.

Using the bound of Claim 1 on the size of sets X_v, we can exploit the algorithm of Claim 2, using $\bigcup_{v \in EX} X_v$ as ground set and $\{X_v\}_{v \in EX}$ as collection of subsets, to find a set $X' \subseteq \bigcup_{v \in EX} X_v$ such that $X' \cap X_v \neq \emptyset$, for all $v \in EX$, whose size is at most $|X'| \leq \frac{n \ln n}{n^{1/3}} = n^{2/3} \ln n$. In this case the parameters of Claim 2 are $R = n^{1/3}$, $M = \left|\bigcup_{v \in EX} X_v\right| \leq n$, and $N = |EX| \leq n$. Since for any $v \in EX$ there exists a path from any $w \in X_v$ to v in $D[U \cup \{w\}]$ and X' contains at least one vertex in X_v, then there exists a path from one of the vertices $w \in X'$ to v in $D[U \cup \{w\}]$.

Now we find a shortest path from r to any $w \in X'$ in D. Let $\mathcal{P}_1$ be the set of all these shortest paths. We also find, for each $v \in EX$, a shortest path from an arbitrary vertex $w \in X' \cap X_v$ to v in $D[U \cup \{w\}]$. Let $\mathcal{P}_2$ be the set of all these shortest paths. Let $V(\mathcal{P}_1)$ and $V(\mathcal{P}_2)$ be the union of all the nodes of the paths in $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. Then, we find a tree T^{EX} rooted at r spanning graph $D^{EX} := D[V(\mathcal{P}_1) \cup V(\mathcal{P}_2)]$. Note that such a tree exists as in D^{EX} there exists a path from r to any $w \in X'$ and, for each vertex $v \in EX$, at least a path from one of the vertices in X' to v.

We next move to bounding the cost of T^{EX}, i.e. we bound the total cost of nodes in $V(\mathcal{P}_1) \cup V(\mathcal{P}_2)$. Since $|X'| \leq n^{2/3} \ln n$ and the maximum distance from r to any other node is F, then $c(V(\mathcal{P}_1)) \leq F n^{2/3} \ln n$. As $\Sigma_{u \in U} c(v) \leq \Lambda n^{2/3}$ (by constraint (D.2) of (DRT) and $x_u \geq n^{-2/3}$ for any $u \in U$) and $V(\mathcal{P}_2) \setminus X' \subseteq U$, then $c(V(\mathcal{P}_2) \setminus X') \leq \Lambda n^{2/3}$. Overall, T^{EX} costs at most $O((F + \Lambda) n^{2/3} \ln n)$.

Since both T^{EX} and T^{CH} are rooted at r, we can find a tree T rooted at r that spans all vertices $V(T^{EX}) \cup V(T^{CH})$. Since $c(T^{CH}) + c(T^{EX}) = O((F + \Lambda) n^{2/3} \ln n)$, we have $c(T) = O((F + \Lambda) n^{2/3} \ln n)$, which concludes the proof. □

We are now ready to prove Theorem 1.

Proof (Proof of Theorem 1). As $\Sigma_{v \in S} x_v p_v \geq \Pi$, either $\Sigma_{v \in S_1} x_v p_v \geq \Pi/2$ or $\Sigma_{v \in S_2} x_v p_v \geq \Pi/2$.

If $\Sigma_{v \in S_1} x_v p_v \geq \Pi/2$, then by Lemma 2, we can find in polynomial time a tree T that spans all vertices in S_1 such that $c(T) = O((F + \Lambda) n^{2/3} \ln n)$. Since T spans all vertices of S_1, then $p(T) \geq p(S_1) \geq \Pi/2$.

If $\Sigma_{v\in S_2} x_v p_v \geq \Pi/2$, we have that $p(S_2) = \Sigma_{v\in S_2} p_v \geq n^{1/3} \cdot \Sigma_{v\in S_2} x_v p_v \geq n^{1/3}\Pi/2$, where the first inequality holds since $0 < x_v < n^{-1/3}$ for any $v \in S_2$ and the second inequality holds by the case assumption. We partition S_2 into M groups $U_1, \ldots, U_M$ in such a way that for each $i \in [M-1]$, $|U_i| = 2|S_2|^{2/3}$, and $|U_M| \leq 2|S_2|^{2/3}$. Hence the number of selected groups is at most $M \leq \left\lceil \frac{|S_2|}{2|S_2|^{2/3}} \right\rceil = \left\lceil \frac{|S_2|^{1/3}}{2} \right\rceil \leq \left\lfloor \frac{|S_2|^{1/3}}{2} \right\rfloor + 1 \leq |S_2|^{1/3} \leq n^{1/3}$. Now among $U_1, \ldots, U_M$, we select the group U_z that maximizes the prize, i.e., $z = \arg\max_{i\in[M]} p(U_i)$. We know that $p(U_z) \geq \frac{1}{M}\Sigma_{i=1}^{M} p(U_i) = \frac{p(S_2)}{M} \geq \frac{n^{1/3}\Pi}{2n^{1/3}} = \Pi/2$, where the first inequality is due to an averaging argument, the first equality is due to the additivity of p, and the second inequality is due to $M \leq n^{1/3}$ and $p(S_2) \geq n^{1/3}\Pi/2$.

We now find for each vertex v in U_z a shortest path from r to v and compute a tree T that spans all the vertices in the union of these shortest paths. Clearly, $c(T) \leq 2Fn^{2/3}$ as $|U_z| \leq 2n^{2/3}$ and the cost of a shortest path from r to any $v \in V$ in D is at most F. Furthermore, $p(T) \geq p(U_z) \geq \Pi/2$, by additivity of p. This concludes the proof. □

Step 3: Approximating Q-DRT and B-DRT

We next show how to use Theorem 1 to devise bicriteria approximation algorithms for **Q-DRT** and **B-DRT**. First we prove our result for **Q-DRT**.

Theorem 2. ***Q-DRT*** *admits a polinomial time bicriteria* $\big(2, O(n^{2/3}\ln n)\big)$*-approximation algorithm.*

Proof. We can assume w.l.o.g. that for all nodes v we have $dist(r,v) \leq (1+\epsilon)c(T_Q^*)$, i.e. that $F \leq (1+\epsilon)c(T_Q^*)$, for any $\epsilon > 0$. Indeed, we can guess an $(1+\epsilon)$-approximation of $c(T_Q^*)$ using the following procedure. Let $c_{\min}$ be the minimum positive cost of a vertex and $c_M = \sum_{v\in V} c(v)$, we know that $c(T_Q^*) \leq c_M$. We estimate the value of $c(T_Q^*)$ by guessing N possible values, where N is the smallest integer for which $c_{\min}(1+\epsilon)^{N-1} \geq c_M$. For each guess $i \in [N]$, we remove the nodes v with $dist(r,v) > c_{\min}(1+\epsilon)^{i-1}$, and compute a tree T rooted in r with an algorithm that will be explained later in the proof. Eventually, we output the computed tree T with the smallest cost. Since $c_{\min}(1+\epsilon)^{N-2} < c_M$, the number N of guesses is smaller than $\log_{1+\epsilon}(c_M/c_{\min}) + 2$, which is polynomial in the input size and in $1/\epsilon$.

Let $i \in [N]$ be the smallest value for which $c_{\min}(1+\epsilon)^{i-1} \geq c(T_Q^*)$. Then, $c(T_Q^*) > c_{\min}(1+\epsilon)^{i-2}$ and for all the nodes v in the graph used in guess i, we have $dist(r,v) \leq c_{\min}(1+\epsilon)^{i-1} < (1+\epsilon)c(T_Q^*)$. Since we output a solution with the minimum cost among those computed in the guesses, then the final solution will not be worse than the one computed at guess i. Therefore, from now on we focus on guess i and assume that $F \leq (1+\epsilon)c(T_Q^*)$.

We first find an optimal solution x to **Q-DRT-LP**, let OPT_Q be the optimal value of **Q-DRT-LP**. Observe that x is a feasible solution for the set of constraints (DRT) in which $\Lambda = OPT_Q$ and $\Pi = Q$ and, by Lemma 1, $OPT_Q \leq c(T_Q^*)$. Therefore, we can use x and apply the algorithm in Theorem 1 to obtain

a tree T such that $c(T) = O((F + OPT_Q)n^{2/3} \ln n) = O((1+\epsilon)c(T^*_Q)n^{2/3} \ln n)$ and $p(T) \geq Q/2$, which concludes the proof. □

Next we prove our result for **B-DRT**. Let us assume that for every $v \in V$, $dist(r, v) \leq B$, i.e. $F \leq B$, since otherwise we can remove from D all the nodes v such that $dist(r, v) > B$.

Let x be an optimal solution for **B-DRT-LP** and let T be a tree computed from x by the algorithm in Theorem 1. Since x is a feasible solution to (DRT) when $\Pi = OPT_B \geq p(T^*_B)$ and $\Lambda = B$, then the prize of T is at least $\frac{p(T^*_B)}{2}$ but its cost can exceed the budget B. In this case, however, the cost of T is bounded by $c(T) = O(Bn^{2/3} \ln n)$ and its prize-to-cost ratio is $\gamma = \frac{p(T)}{c(T)} = \Omega\left(\frac{p(T^*_B)}{Bn^{2/3} \ln n}\right)$. Therefore, we can use T and a variant of the trimming process introduced by Bateni et al. [2] for undirected graphs, to compute another tree $\hat{T}$ with cost between $\frac{\epsilon B}{2}$ and $(1+\epsilon)B$ and prize-to-cost ratio $\frac{\epsilon\gamma}{4}$, for any $\epsilon \in (0, 1]$. The following lemma gives extends the trimming process by Bateni et al. to directed graphs. The proof can be found in the full version of the paper [6].

Lemma 3. *Let $D = (V, A)$ be a B-proper graph for a node r. Let T be an out-tree of D rooted at r with the prize-to-cost ratio $\gamma = p(T)/c(T)$. Suppose that, for $\epsilon \in (0, 1]$, $c(T) \geq \epsilon B/2$. One can find an out-tree $\hat{T}$ rooted at r with the prize-to-cost ratio at least $\epsilon\gamma/4$ such that $\epsilon B/2 \leq c'(\hat{T}) \leq (1+\epsilon)B$.*

Theorem 3. ***B-DRT*** *admits a polynomial time bicriteria $\left(1+\epsilon, O(\frac{n^{2/3} \ln n}{\epsilon^2})\right)$-approximation algorithm, for any $\epsilon \in (0, 1]$.*

Proof. We first find an optimal solution x to **B-DRT-LP**. and then we apply the algorithm in Theorem 1 and use x as input to obtain a tree T. Since x is a feasible solution to (DRT) when $\Pi = OPT_B \geq p(T^*_B)$ and $\Lambda = B$, then $c(T) = O(Bn^{2/3} \ln n)$ and $p(T) \geq \frac{OPT_B}{2} \geq \frac{p(T^*_B)}{2}$ as discussed above. The prize-to-cost ratio of T is $\gamma = \frac{p(T)}{c(T)} = \Omega\left(\frac{p(T^*_B)}{Bn^{2/3} \ln n}\right)$. Then, if $c(T) > B$, we can apply the algorithm of Lemma 3 to T and compute another tree $\hat{T}$ with cost $c(\hat{T}) \leq (1+\epsilon)B$ and prize-to-cost ratio at least $\frac{\epsilon\gamma}{4}$. Moreover, $c(\hat{T}) \geq \epsilon B/2$, and therefore we have $p(\hat{T}) = \Omega\left(\frac{\epsilon^2 p(T^*_B)}{n^{2/3} \ln n}\right)$, concluding the proof. □

4 Discussion and Future Work

On budget violation. There is some evidence showing that budget violation is needed to approximate budgeted problems in polynomial time. Kortsarz and Nutov [19] showed that the unrooted version of **B-URT**, admits no $o(\log \log n)$-approximation algorithm, unless $NP \subseteq DTIME(n^{\text{polylog}(n)})$, even if the algorithm is allowed to violate the budget constraint by a factor equal to a universal constant. This lower bound holds for all the budgeted problems in this paper. The only approximation algorithm for the budgeted problems that does not violate the budget constraint is the combinatorial quasi-polynomial time algorithm

given by Ghuge and Nagarajan [12]. Most of the literature on these problems, focuses on approximation algorithms that exploit the standard low-based LP relaxation. Bateni et al. [2] showed that the integrality gap of this LP relaxation is unbounded if no budget violation is allowed, even for **B-URT**. We show that, by violating the budget constraint, one can use this LP and provide an approximation algorithm for the more general **B-DRT**.

On lower bounds. As **Q-DRT** is a more general version of the node-weighted variant of the directed Steiner tree problem, called **DSteinerT**, any lower bound for **DSteinerT** also holds for **Q-DRT**. Recently, Li and Laekhanukit [21] ruled out poly-logarithmic approximation algorithms for the directed Steiner tree problem using the standard flow-based LP. This result holds for **DSteinerT** (and **Q-DRT**) as, in the instance in [21], the incoming edges of each vertex have the same cost. We also know that any α-approximation algorithm for the edge cost version of **B-DRT** results in an $O(\alpha \log n)$-approximation algorithm for the directed Steiner tree problem, which was also pointed out by [12]. This implies that the quasi-polynomial time approximation algorithm of [12] is tight for the edge cost version of **B-DRT**. Furthermore, by performing a binary search on the budget (resp. quota) space, one can use an α-approximation algorithm for a budgeted (resp. quota) problem to obtain a $1+\epsilon$ (resp. $1-\epsilon$)-approximation algorithm for its quota (resp. budgeted) version, for any $\epsilon > 0$, that violates the quota (resp. budget) constraint by a factor of at most α.

Future Works. We believe that finding a polynomial-time lower bound for the problems considered in this paper is an interesting future work as this question for the directed Steiner tree is open for a long-time. The integrality gap of the standard flow-based LP relaxation for **B-URT** is unbounded if no budget violation is allowed [2], and has a polynomial lower bound for the directed Steiner tree problem [21]. Finding an integrality gap for **B-URT** in case of budget violation would be an interesting future work. Extending our results to the edge-cost variants of **B-DRT** and **Q-DRT** would be a very interesting future work. The use of LP-hierarchies for approximation algorithms in directed Steiner trees [24, 9] would be a potential direction to provide approximation algorithms for **B-DRT** and **Q-DRT** and their edge-cost variants. For Directed Steiner Network, it is known that the integrality gap of the Lasserre Hierarchy has a polynomial lower bound [8].

References

1. Aaron Archer, MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Howard J. Karloff. Improved approximation algorithms for prize-collecting steiner tree and TSP. *SIAM J. Comput.*, 40(2):309–332, 2011.
2. Mohammad Hossein Bateni, Mohammad Taghi Hajiaghayi, and Vahid Liaghat. Improved approximation algorithms for (budgeted) node-weighted steiner problems. *SIAM J. Comput.*, 47(4):1275–1293, 2018.
3. Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 590–598. Association for Computing Machinery, 2007.

4. Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed steiner problems. *J. Algorithms*, 33(1):73–91, 1999.
5. Xiuzhen Cheng, Yingshu Li, Ding-Zhu Du, and Hung Q Ngo. Steiner trees in industry. In *Handbook of combinatorial optimization*, pages 193–216. Springer, 2004.
6. Gianlorenzo D'Angelo and Esmaeil Delfaraz. Approximation algorithms for node-weighted steiner problems: digraphs with additive prizes and graphs with submodular prizes. *arXiv preprint arXiv:2211.03653*, 2022.
7. Gianlorenzo D'Angelo, Esmaeil Delfaraz, and Hugo Gilbert. Budgeted out-tree maximization with submodular prizes. In Sang Won Bae and Heejin Park, editors, *33rd International Symposium on Algorithms and Computation, ISAAC 2022, December 19-21, 2022, Seoul, Korea*, volume 248 of *LIPIcs*, pages 9:1–9:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
8. Michael Dinitz, Yasamin Nazari, and Zeyu Zhang. Lasserre integrality gaps for graph spanners and related problems. In Christos Kaklamanis and Asaf Levin, editors, *Approximation and Online Algorithms - 18th International Workshop, WAOA 2020*, volume 12806 of *Lecture Notes in Computer Science*, pages 97–112. Springer, 2020.
9. Zachary Friggstad, Jochen Könemann, Young Kun-Ko, Anand Louis, Mohammad Shadravan, and Madhur Tulsiani. Linear programming hierarchies suffice for directed steiner tree. In Jon Lee and Jens Vygen, editors, *Integer Programming and Combinatorial Optimization - 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings*, volume 8494 of *Lecture Notes in Computer Science*, pages 285–296. Springer, 2014.
10. Xiaofeng Gao, Junwei Lu, Haotian Wang, Fan Wu, and Guihai Chen. Algorithm design and analysis for wireless relay network deployment problem. *IEEE Trans. Mob. Comput.*, 18(10):2257–2269, 2019.
11. Naveen Garg. Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 396–402. ACM, 2005.
12. Rohan Ghuge and Viswanath Nagarajan. Quasi-polynomial algorithms for submodular tree orienteering and other directed network design problems. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1039–1048. SIAM, 2020.
13. Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.
14. Fabrizio Grandoni, Bundit Laekhanukit, and Shi Li. $O(\log^2 k \,/\, \log\log k)$-approximation algorithm for directed steiner tree: a tight quasi-polynomial-time algorithm. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 253–264. ACM, 2019.
15. Sudipto Guha, Anna Moss, Joseph Naor, and Baruch Schieber. Efficient recovery from power outage (extended abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 574–582. ACM, 1999.
16. Dorit S. Hochbaum and Xu Rao. Approximation algorithms for connected maximum coverage problem for the discovery of mutated driver pathways in cancer. *Inf. Process. Lett.*, 158:105940, 2020.
17. David S. Johnson, Maria Minkoff, and Steven Phillips. The prize collecting steiner tree problem: theory and practice. In David B. Shmoys, editor, *Proceedings of the*

Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, pages 760–769. ACM/SIAM, 2000.
18. Jochen Könemann, Sina Sadeghian Sadeghabad, and Laura Sanità. An LMP o(log n)-approximation algorithm for node weighted prize collecting steiner tree. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 568–577. IEEE Computer Society, 2013.
19. Guy Kortsarz and Zeev Nutov. Approximating some network design problems with node costs. *Theor. Comput. Sci.*, 412(35):4482–4492, 2011.
20. Tung-Wei Kuo, Kate Ching-Ju Lin, and Ming-Jer Tsai. Maximizing submodular set function with connectivity constraint: Theory and application to networks. *IEEE/ACM Trans. Netw.*, 23(2):533–546, 2015.
21. Shi Li and Bundit Laekhanukit. Polynomial integrality gap of flow LP for directed steiner tree. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 3230–3236. SIAM, 2022.
22. Anna Moss and Yuval Rabani. Approximation algorithms for constrained node weighted steiner tree problems. *SIAM J. Comput.*, 37(2):460–481, 2007.
23. Alice Paul, Daniel Freund, Aaron M. Ferber, David B. Shmoys, and David P. Williamson. Budgeted prize-collecting traveling salesman and minimum spanning tree problems. *Math. Oper. Res.*, 45(2):576–590, 2020.
24. Thomas Rothvoß. Directed steiner tree and the lasserre hierarchy. *CoRR*, abs/1111.5473, 2011.
25. Fabio Vandin, Eli Upfal, and Benjamin J. Raphael. Algorithms for detecting significantly mutated pathways in cancer. *J. Comput. Biol.*, 18(3):507–522, 2011.
26. Alexander Zelikovsky. A series of approximation algorithms for the acyclic directed steiner tree problem. *Algorithmica*, 18(1):99–110, 1997.