Bicriterial Approximation for the Incremental Prize-Collecting Steiner-Tree Problem

Yann Disser
Department of Mathematics, Technische Universität Darmstadt, Germany

Svenja M. Griesbach
Institute of Mathematics, Technische Universität Berlin, Germany

Max Klimm
Institute of Mathematics, Technische Universität Berlin, Germany

Annette Lutz
Department of Mathematics, Technische Universität Darmstadt, Germany

Abstract

We consider an incremental variant of the rooted prize-collecting Steiner-tree problem with a growing budget constraint. While no incremental solution exists that simultaneously approximates the optimum for all budgets, we show that a bicriterial (α, μ)-approximation is possible, i.e., a solution that with budget $B + \alpha$ for all $B \in \mathbb{R}_{\geq 0}$ is a multiplicative μ-approximation compared to the optimum solution with budget B. For the case that the underlying graph is a tree, we present a polynomial-time density-greedy algorithm that computes a $(\chi, 1)$-approximation, where χ denotes the eccentricity of the root vertex in the underlying graph, and show that this is best possible. An adaptation of the density-greedy algorithm for general graphs is $(\gamma, 2)$-competitive where γ is the maximal length of a vertex-disjoint path starting in the root. While this algorithm does not run in polynomial time, it can be adapted to a $(\gamma, 3)$-competitive algorithm that runs in polynomial time. We further devise a capacity-scaling algorithm that guarantees a $(3\chi, 8)$-approximation and, more generally, a $\left((4\ell - 1)\chi, \frac{2^{\ell+2}}{2^{\ell}-1}\right)$-approximation for every fixed $\ell \in \mathbb{N}$.

2012 ACM Subject Classification Theory of computation → Online algorithms; Mathematics of computing → Graph algorithms; Mathematics of computing → Combinatorial algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases incremental optimization, competitive analysis, prize-collecting Steiner-tree

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.47

Related Version *Full Version*: `https://arxiv.org/abs/2407.04447` [15]

Funding Supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) through subprojects A07 and A09 of CRC/TRR154 and under Germany's Excellence Strategy – The Berlin Mathematics Research Center MATH+ (EXC-2046/1, project IDs: 390685689).

Acknowledgements We thank Felix Hommelsheim, Alexander Lindermayr, and Jens Schlöter for fruitful discussions and three anonymous referees for their comments that improved the presentation of the paper.

1 Introduction

Prize-collecting Steiner-tree problems serve as a model to study the fundamental trade-off between the cost of installing a network and harnessing its benefits in terms of covering vital vertices. They have been used to study, e.g., expanding telecommunication networks (Johnson et al. [32]) or pipeline networks (Salman et al. [41]). Formally, we are given an undirected graph $G = (V, E)$, a positive *edge cost* $c(e) \in \mathbb{R}_{>0}$ for each edge $e \in E$, a non-negative *vertex prize* $p(v) \in \mathbb{R}_{\geq 0}$ for each vertex $v \in V$, and a specified *root vertex* $r \in V$. In the applications above, vertices correspond to geographical locations such as intersections of a road network, the premises of telecommunication customers, or the locations of oil wells. The prize of a vertex represents the estimated revenue associated with connecting the vertex to the core network, represented by the root. The cost of an edge corresponds to, e.g., the monetary cost of laying a fiber-optic cable or a pipeline between the respective end vertices.

Let $\mathcal{T}$ denote the set of all subtrees of G containing r. Then, the *budget version* of the prize-collecting Steiner-tree problem for a given budget $B \in \mathbb{R}_{\geq 0}$ is the optimization

$$\max\bigl\{p(T) : T \in \mathcal{T} \text{ with } c(T) \leq B\bigr\}, \tag{1}$$

where for a subgraph $G' = (V', E') \subseteq G$, we write $p(G') := \sum_{v \in V'} p(v)$ and $c(G') := \sum_{e \in E'} c(e)$. This objective is faced, e.g., by any company interested in building the most profitable telecommunication or pipeline network given limited financial funds. In the following, we arbitrarily fix a rooted subtree $\textsc{Opt}(B)$ attaining the maximum in (1).

In the scenarios mentioned above, it is realistic to assume that the network is expanded over time as the company's budget increases. This motivates us to study an incremental version of the prize-collecting Steiner-tree problem where approximately optimal Steiner-trees have to be maintained under a growing budget constraint. Formally, an *incremental solution* to an instance of the prize-collecting Steiner-tree problem is given by an ordering $\pi = (e_{\pi(1)}, \dots, e_{\pi(l)})$ of a subset of the edges E such that every prefix $(e_{\pi(1)}, \dots, e_{\pi(j)})$ of π with $j \leq l$ induces a rooted subtree $T_j \subseteq G$ and the rooted subtree T_l spans all vertices in $V^* := \{v \in V : p(v) > 0\}$. In the applications above, an incremental solution corresponds to an order in which to build the fiber-optic or pipeline connections.

We evaluate the performance of an incremental solution π in terms of the worst-case approximation guarantee it provides compared to the optimal budgeted Steiner-tree for all budgets $B \in \mathbb{R}_{\geq 0}$. Specifically, for a given budget $B > 0$ and an incremental solution $\pi = (e_{\pi(1)}, \dots, e_{\pi(l)})$ computed by some algorithm $\textsc{Alg}$, let $\textsc{Alg}(B)$ denote the rooted subtree induced by the set of edges $\bigl\{e_{\pi(i)} : i \in \{1, \dots, l\}, \sum_{j=1}^{i} c(e_{\pi(j)}) \leq B\bigr\}$. We aim to maximize the profit $p(\textsc{Alg}(B))$ across all budgets $B \in \mathbb{R}_{\geq 0}$.

Observe that, in general, no incremental solution can stay close to the prizes collected by $\textsc{Opt}(B)$ across all budgets simultaneously (see Figure 1). However, it turns out that it suffices to allow an *additive* slack in the budget constraint to obtain an incremental solution with *multiplicative* approximation guarantee. Formally, we call an algorithm (α, μ)-competitive for $\alpha \geq 0$ and $\mu \geq 1$ if the incremental solution it produces (for any instance of the problem) guarantees that $\mu\, p(\textsc{Alg}(B + \alpha)) \geq p(\textsc{Opt}(B))$ for all $B \in \mathbb{R}_{\geq 0}$.

1.1 Our Results

We first consider the incremental prize-collecting Steiner-tree problem on trees and analyze the density-greedy algorithm introduced by Alpern and Lidbetter [4] in a different context. Roughly, the algorithm repeatedly considers subtrees of maximum density, adds them to

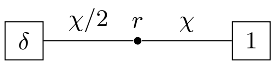

Figure 1 Instance of the prize-collecting Steiner-tree problem with $\delta \ll 1$ and no $(0, \mu)$-competitive incremental solution: Staying competitive for budget $B = \chi/2$ requires to build the left edge first, but then we are not competitive for the budget $B = \chi$.

the incremental solution and contracts the corresponding subgraphs. We show that this algorithm constructs an incremental solution that lags behind the optimum solution by some additive slack in the budget, but otherwise collects the same prize. Figure 1 shows that we need to allow a slack depending on the *eccentricity* of the root vertex $\chi := \max\{\ell(v) : v \in V\}$ of G, where $\ell(v)$ denotes the minimum cost among all r-v-paths in G. We show that the density-greedy algorithm requires slack exactly χ.

▶ **Theorem 1.** *On trees, the density-greedy algorithm can be implemented in polynomial time and is (α, μ)-competitive for any finite $\mu \geq 1$ if and only if $\alpha \geq \chi$.*

In particular, the density-greedy algorithm is $(\chi, 1)$-competitive on trees. We show that this is best possible.

▶ **Theorem 2.** *There exists no (α, μ)-competitive algorithm for $\alpha < \chi$ and $\mu < \chi/\alpha$ or for $\alpha < \chi/2$ and $\mu \geq 1$, even on trees.*

We proceed to generalize the density-greedy algorithm in order to apply it to non-tree networks. Here, an additive slack in budget equal to the eccentricity of the root no longer suffices for it to stay close to the optimum solution. We obtain, however, an approximation guarantee when granting an additive slack equal to the maximum cost γ of any vertex-disjoint r-v-path in G.

▶ **Theorem 3.** *The density-greedy algorithm is $(\gamma, 2)$-competitive, but not (χ, μ)-competitive for any $\mu \geq 1$.*

While the density-greedy algorithm cannot be implemented in polynomial time, unless $\mathsf{P} = \mathsf{NP}$, we give an approximate variant of the algorithm that runs in polynomial time at the expense of increasing the approximation factor from 2 to 3.

▶ **Corollary 4.** *An approximate variant of the density-greedy algorithm runs in polynomial time and is $(\gamma, 3)$-competitive.*

We further prove that no algorithm can reach the optimum on general graphs if the additional slack in budget only depends on the length of the longest vertex-disjoint paths from the root.

▶ **Theorem 5.** *For every (α, μ)-competitive algorithm with α only depending on γ, it holds that $\mu \geq 17/16$.*

Note that since $\chi \leq \gamma$, this result also precludes the existence of an (α, μ)-competitive algorithm with α depending only on χ and γ and $\mu < 17/16$.

Finally, we design and analyze a capacity-scaling algorithm that is inspired by the incremental maximization algorithm of Bernstein et al. [8] and combines this approach with the density-greedy algorithm. Instead of relying on subtrees of maximum density, the algorithm uses optimum solutions for the budget-constrained prize-collecting Steiner-tree problem for exponentially growing budgets. These solutions are concatenated to give a solution for the incremental prize-collecting Steiner-tree problem. We obtain the following result.

▶ **Theorem 6.** *The capacity-scaling algorithm is $\big((4\ell-1)\chi, \frac{2^{\ell+2}}{2^{\ell}-1}\big)$-competitive for every $\ell \in \mathbb{N}$.*

In particular, our algorithm guarantees an 8-approximation when allowed an additional 3χ in budget and its approximation guarantee approaches 4 for increasing additive slack.

1.2 Related Work

Our work is based on the cardinality-constrained incremental maximization framework of Bernstein et al. [8]. They considered monotone augmentable objective functions (a class of functions containing monotone submodular functions) subject to a growing cardinality constraint and devise a cardinality-scaling algorithm that is 2.618-competitive. Disser et al. [17] expanded on this result by giving an improved lower bound of 2.246 and considering a randomized scaling approach for the problem that achieves a randomized competitive ratio of 1.772. The framework was extended to a budget-constrained variant by Disser et al. [16]. A detailed treatment of incremental maximization was given by Weckbecker [43].

The Steiner-tree problem is known to be NP-complete since Karp [33]. In fact, it is NP-hard to approximate the problem within a factor of 96/95, as shown by Chlebík and Chlebíková [14]. The current best approximation algorithm with an approximation guarantee of $\ln(4)+\epsilon$ is due to Byrka et al. [13] improving upon previous approximation algorithms (e.g., Hougardy and Prömel [29]; Robins and Zelikovsky [40]). Based on earlier work of Balas [7], the prize-collecting variant of the problem was first studied by Bienstock et al. [9] who gave a 3-approximation for the objective to minimize the sum of the edge costs and the weights of the non-collected vertices. This was later improved to a $(2-\frac{1}{n})$-approximation by Goemans and Williamson [23], where n is the number of vertices of the graph, and to 1.9672 by Archer et al. [5]. In a recent paper by Ahmadi et al. [3] the approximation ratio was further reduced to 1.7994. Johnson et al. [32] discussed further natural variants of the problem: net-worth, budget, and quota. The *net-worth problem* where the task is to maximize the total collected prize minus the total cost is NP-hard to approximate by any constant factor (Feigenbaum et al. [18]). In the *budget problem*, the task is to find a tree that maximizes the total collected prize collected subject to a budget constraint on the total cost of the tree. Constant-factor approximations are only known for the unrooted variant of the problem [32, 35, 39]; see also the discussion by Paul et al. [38]. For a problem more general than the rooted version, Ghuge and Nagarajan [21] gave a quasi-polynomial poly-logarithmic approximation. In the *quota problem*, the task is to find a minimum cost tree that collects at least a given total prize. The quota version where all vertices have a prize of 1 is known as the k-MST problem. The best known approximation for the k-MST problem is a 2-approximation due to Garg [20] improving upon previous approximation algorithms (e.g., Arya and Ramesh [6]; Blum et al. [11]; Garg [19]). As discussed by Johnson et al. [32], algorithms for the k-MST problem that rely on the primal-dual algorithm of Goemans and Williamson [23] (such as the one by Garg [20]) extend to the general quota problem so that the best known approximation factor for the quota problem is 2 as well; see also the discussion by Griesbach et al. [24]. An incremental version of the k-MST problem has been studied by Lin et al. [36]. Here, an incremental solution is a sequence of edges such that the cost of the smallest prefix of edges that connects at least k vertices approximates the value of the optimal k-MST solution for all $k \in \mathbb{N}$. They noted that a deterministic 8-competitive and a randomized $2e$-competitive incremental solution are implicit in the works on the minimum latency problem [10, 22] and gave improved incremental solutions with competitive ratios 4 and e for the deterministic and randomized case, respectively.

The more general Steiner forest problem asks for given pairs of vertices to be connected at minimal cost. This problem admits a $(2-\frac{1}{n})$-approximation (Agrawal et al. [1]; Goemans and Williamson [23]). For a prize-collecting version of the problem, Ahmadi et al. [2] recently gave a 2-approximation improving the 2.54-approximation by Hajiaghayi and Jain [27] whose approach has been generalized by Sharma et al. [42] towards more general submodular penalty functions.

Further related are dynamic and universal variants of the Steiner-tree problem. In the dynamic Steiner-tree problem, vertices are added or deleted in an online fashion and a network spanning the active vertices has to be maintained with approximately optimal costs at all times. If only vertex additions are allowed, the natural greedy algorithm yields a $\mathcal{O}(\log n)$-approximation under n additions as shown by Imase and Waxman [30]. Gu et al. [25] gave an algorithm that adds an edge and swaps an edge per addition and maintains a constant approximation. Naor et al. [37] studied a further variant of the problem where also vertices have costs. Xu and Moseley [44] gave a learning-augmented algorithm that uses a prediction about which terminals will be added eventually. When only deletions are allowed, Gupta and Kumar [26] showed how to maintain a constant approximation while changing a constant number of edges per deletion. Jia et al. [31] introduced the notion of universal Steiner-trees which are rooted spanning trees containing an approximately minimal Steiner-tree for any subset of vertices. Universal Steiner-trees with poly-logarithmic approximation factor have been devised by Busch et al. [12].

2 Incremental Prize-Collecting Steiner-Tree on Trees

In this section we introduce the *density-greedy algorithm* and prove that it constructs a $(\chi, 1)$-competitive algorithm for the incremental prize-collecting Steiner-tree problem on trees. We further prove that this is the best possible approximation guarantee, i.e., there is no $(\alpha, 1)$-competitive algorithm for trees where $\alpha < \chi$. Throughout this section, we assume that $G = (V, E)$ is a tree with root vertex r and $p(r) = 0$.

The density-greedy algorithm uses the concept of a *min-max subtree* introduced by Alpern and Lidbetter [4] in the context of the expanding search problem. For a non-empty rooted subtree $T \in \mathcal{T}$ of G, its density $d(T)$ is defined as the ratio of the prizes that are collected by T and the cost of T, i.e., $d_G(T) := p(T)/c(T)$. When the graph G is clear from context, we may drop the index G. For the empty rooted subtree $T_\emptyset := (\{r\}, \emptyset)$, we set $d(T_\emptyset) := 0$. We further say that $T \in \mathcal{T}$ has *maximum density* if $d(T) \geq d(T')$ for all $T' \in \mathcal{T}$. Let $\mathcal{M} := \{T \in \mathcal{T} : d(T) \geq d(T') \text{ for all } T' \in \mathcal{T}\}$ be the set of all trees with maximum density d^*. The inclusion-wise minimal elements of $\mathcal{M}$ are called the *min-max-subtrees* of the graph. The following properties of the set $\mathcal{M}$ are due to Alpern and Lidbetter [4].

▶ **Lemma 7** (cf. [4], Lemma 2)**.** *The set $\mathcal{M} \cup (\{r\}, \emptyset)$ is closed under union and intersection. Furthermore, the following holds:*

(i) *Distinct min-max subtrees are (edge-)disjoint.*
(ii) *Every branch at r of $T \in \mathcal{M}$ has density d^*.*
(iii) *Every min-max subtree T of G has only one branch at r.*

We are now in position to explain the basic idea of the *density-greedy algorithm* (cf. Algorithm 1). At the beginning, we start with the empty solution $\pi = ()$. As long as the tree spanned by the edges in π does not collect the total prize of G, all edges contained in the current solution are contracted and we compute a min-max subtree in the contracted subgraph. By Lemma 7, this min-max subtree has a unique edge that is adjacent to the root. This edge is added to π and the whole procedure is repeated until π collects the total prize of G. The density of the root vertex is defined to be zero. Hence, Algorithm 1 always terminates and returns a solution π that spans all vertices in V^*. In the remaining part of this section, we denote by π the incremental solution returned by Algorithm 1. Recall that, for some cost budget $B \in \mathbb{R}_{\geq 0}$, we denote by $\textsc{Alg}(B)$ the tree that is induced by the maximum prefix of π that still obeys the cost budget B.

Algorithm 1 The density-greedy algorithm for trees.

```
1: $\pi \leftarrow ()$
2: $\overline{T} \leftarrow (\{r\}, \emptyset)$
3: while $p_{G/\overline{T}}(G/\overline{T}) \neq 0$ do
4:     fix arbitrary min-max subtree $T^*$ of $G/\overline{T}$
5:     denote by $e^*$ the unique edge in $T^*$ incident to $r$
6:     denote by $e \in E$ the edge in $G$ that corresponds to $e^*$ in $G/\overline{T}$
7:     append $e$ to $\pi$
8:     $\overline{T} \leftarrow \overline{T} \cup \{e\}$
9: return $\pi$
```

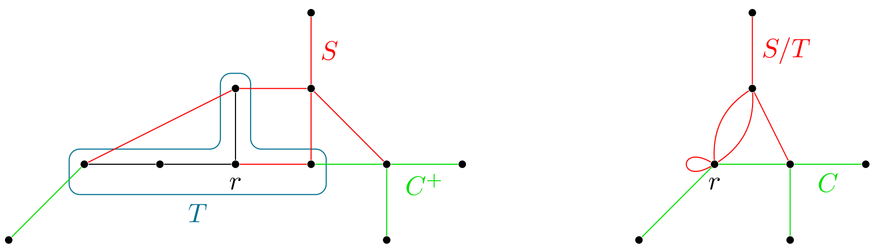

Figure 2 The left figure shows graph G with tree $T \in \mathcal{T}$ and the right figure shows the contracted graph G/T. The contracted subgraph S/T contains parallel edges and loops. The extension C^+ of the connected subgraph $C \subset G/T$ is not connected.

The algorithm maintains a graph where the edges of a rooted tree are contracted. For a formal analysis of the algorithm, we introduce the following notion regarding contractions. For some $T \in \mathcal{T}$, we obtain the *contracted graph* $G/T = (V_{G/T},\ E_{G/T})$ by contracting all edges of T into the root. The graph G/T then consists of $|V| - |E_T|$ vertices including the root vertex and $|E| - |E_T|$ edges, possibly including parallel edges and self-loops. The vertex prizes in G/T remain the same as the ones in the original graph G where the prize of the root is set to 0, i.e., $p_{G/T}(v) \coloneqq p(v)$ for all $v \in V_{G/T} \setminus \{r\}$ and $p_{G/T}(r) \coloneqq 0$. Also any edge that is not contracted keeps its original cost, i.e., $c_{G/T}(e) \coloneqq c(e)$ for all $e \in E_{G/T}$. By keeping parallel edges and self-loops, we obtain the property that each edge in G is in a one-to-one correspondence to either an edge in T or to an edge in the contracted graph G/T. This property will be of particular use for the following definition. Let $S = (V_S, E_S)$ be a subgraph of G. For a rooted subtree $T \in \mathcal{T}$ of G, we denote by $E_{S/T}$ the edges of S that have a one-to-one correspondence to an edge in the contracted graph G/T, i.e., $E_{S/T} = E_S \setminus E_T$. Then the *contracted subgraph* S/T is the subgraph of G/T that is induced by the edge set $E_{S/T} \subseteq E_{G/T}$. In particular, we obtain $p_{G/T}(S/T) = p(S) - p(V_S \cap V_T)$ and $c_{G/T}(S/T) = c(S) - c(E_S \cap E_T)$.

To reverse the contraction, we introduce the notion of an extended subgraph. To this end, let $T \in \mathcal{T}$ and let $C = (V_C, E_C)$ be a subgraph of the contracted graph G/T. We denote by E_{C^+} the edges of E, that are in one-to-one correspondence to an edge in E_C. Note that $|E_C| = |E_{C^+}|$. Then, the *extended subgraph* $C^+ = (V_{C^+}, E_{C^+})$ (or *extension of* C) is the subgraph of G that is induced by the edge set E_{C^+}. Observe, that C^+ is not necessarily connected in G, even if C is connected in G/T. See Figure 2 for an illustration. However, if C is connected and has at most one vertex adjacent to the root vertex, then C^+ is also connected. We emphasise that in general, extending a contracted subgraph does not necessarily yield the original subgraph, i.e., for a subgraph S of G and some $T \in \mathcal{T}$ we have $(S/T)^+ \subseteq S$ and equality only holds when $E_S \cap E_T = \emptyset$. However, repeating the contraction of T, then yields the same subgraph of G/T, i.e., $(S/T)^+/T = S/T$.

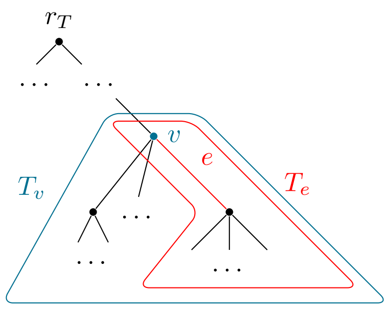

Figure 3 For the tree T with root r_T, the branch T_e rooted in edge e is shown in red and the branch T_v rooted in vertex v is shown in blue.

For the analysis of the density-greedy algorithm, we need to generalize the notion of density to subtrees that are not necessarily rooted. To this end, let $T = (V_T, E_T)$ be a subtree of G and let $r_T \in V_T$ be the unique vertex of T that has the shortest distance to r in G. Then, the *density of* T *over* G is defined as $d_G(T) := \big(p(T) - p(r_T)\big)/c(T)$. Similarly, for a collection of disjoint subtrees $F = \{T_1, \dots, T_k\}$, we set $d_G(F) := \big(\sum_{i=1}^{k} p(T_i) - p(r_{T_i})\big)/\big(\sum_{i=1}^{k} c(T_i)\big)$ where again, for $i \in \{1, \dots, k\}$, the vertex r_{T_i} is the unique vertex in T_i with the smallest distance to r in G. In order to avoid division by 0, we set the density of a tree or forest to 0 whenever the edge set is empty.

Let $T = (V_T, E_T)$ be a subtree of G with root vertex r_T. For an edge $e \in E_T$ we define the *branch* T_e rooted in edge e as the subtree of T that is induced by the endpoints of e and all vertices $u \in V_T$ such that e lies on the unique r_T-u-path in T, see Figure 3 for an illustration. Analogously, for a vertex $v \in V_T$ we define the *branch* T_v rooted at v as the subtree of T that is induced by vertex v and all vertices $u \in V_T$ such that v lies on the unique r_T-u-path in T, see Figure 3 for an illustration.

We proceed to show that for a min-max subtree, every branch T_e rooted at some edge e has at least the same density as T. The proof of the following lemma and further proofs are deferred to the full version of the paper [15].

▶ **Lemma 8.** *Let T be a min-max subtree of G. Then, for every edge e in T we have for the branch T_e of T rooted at e that $d_G(T_e) \geq d(T)$. Strict inequality holds, whenever $T \neq T_e$.*

The next lemma shows an important property about the order in which the density-greedy algorithm adds edges to the solution. In particular, it implies that all edges of the extension T^+ of a min-max tree computed by the density-greedy algorithm are added to the solution π consecutively.

▶ **Lemma 9.** *Let T^* be a min-max subtree of $G/\overline{T}$ for some $\overline{T} \in \mathcal{T}$ as computed in an iteration of the density-greedy algorithm for trees. Then, until all edges of the extension T^+ of T^* are added to π, the extended subgraphs of computed min-max subtrees are subtrees of T^+ and have a larger density than T^+.*

Next, we give the first lower bound for the total prize that is collected by the density-greedy algorithm for trees for some specific cost budgets. To do so, we denote by T_1 the min-max subtree of G that is computed and fixed in the first iteration of Algorithm 1. Recall that $\textsc{Alg}(B)$ denotes the tree induced by the maximum prefix of the solution computed by the density-greedy algorithm obeying the cost budget B.

▶ **Lemma 10.** *For every cost budget $B \leq c(T_1)$, we have $p\big(\textsc{Alg}(B + \chi)\big) \geq d(T_1)B$. Furthermore, if $B = c(T_1)$, we have $p\big(\textsc{Alg}(B)\big) = d(T_1)B$.*

Lemma 9 states that the first $|T_1|$ edges of π are the edges of T_1. Let T_2^* be the ($|T_1|+1$)-st min-max subtree that is computed by the density-greedy algorithm. In particular, T_2^* is the first min-max subtree whose extension T_2 contains no edge of T_1. This inductively defines all pairwise disjoint min-max subtrees $T_1^*, \dots, T_k^*$ and their extensions $T_1, \dots, T_k$ such that $V^* \subseteq \bigcup_{i=1}^k V_{T_i}$. We define $\overline{T}_i \coloneqq \bigcup_{j=1}^i T_j$ for $1 \leq i \leq k$ and set $\overline{T}_0 \coloneqq (\{r\}, \emptyset)$. Note that, since T_i^* is a min-max subtree of $G/\overline{T}_{i-1}$, we have for the densities over G that $d_G(T_i \cup T_{i+1}) = d_{G/\overline{T}_{i-1}}\big((T_i^* \cup T_{i+1}^*)\big) \leq d_{G/\overline{T}_{i-1}}(T_i^*) = d_G(T_i)$. In particular, this yields $d_G(T_{i+1}) \leq d_G(T_i)$ for all $0 < i < k$. By applying Lemma 10 iteratively on $G/\overline{T}_i$ for all $i \in \{0, \dots, k-2\}$, we obtain the following Corollary.

▶ **Corollary 11.** *Let $B \in \mathbb{R}_{\geq 0}$ be a cost budget such that $\sum_{i=1}^{j-1} c(T_i) \leq B \leq \sum_{i=1}^{j} c(T_i)$ for some $1 \leq j \leq k$. Then, we have*

$$p\big(\textsc{Alg}(B+\chi)\big) \geq \sum_{i=1}^{j-1} d(T_i)c(T_i) + d(T_j)\left(B - \sum_{i=1}^{j-1} c(T_i)\right).$$

To give an upper bound on the prize collected by the optimum solution, we show the following lemma.

▶ **Lemma 12.** *Let $T \in \mathcal{T}$ and $j \in \{1, \dots, k\}$. Then, we have*

$$p(T) \leq \sum_{i=1}^{j-1} d(T_i)c(T_i) + d(T_j)\left(c(T) - \sum_{i=1}^{j-1} c(T_i)\right).$$

From Corollary 11 and Lemma 12 we obtain that the density-greedy algorithm is $(\chi, 1)$-competitive. Together with a suitable lower bound, we obtain the following result.

▶ **Theorem 1.** *On trees, the density-greedy algorithm can be implemented in polynomial time and is (α, μ)-competitive for any finite $\mu \geq 1$ if and only if $\alpha \geq \chi$.*

We further prove that that there is no $(\alpha, 1)$-competitive algorithm for trees for any $\alpha < \chi$, which implies that the density-greedy algorithm gives the best possible approximation guarantee. In fact, we show the following stronger result.

▶ **Theorem 2.** *There exists no (α, μ)-competitive algorithm for $\alpha < \chi$ and $\mu < \chi/\alpha$ or for $\alpha < \chi/2$ and $\mu \geq 1$, even on trees.*

3 Incremental Prize-Collecting Steiner-Tree on General Graphs

In this section we study the incremental prize-collecting Steiner-tree problem on general graphs. As a first step, we extend the density-greedy algorithm from the previous section such that we can apply it on general graphs, as well. Afterwards we introduce a new algorithm based on capacity scaling and, finally, we give general lower bounds for the approximation factors of the incremental prize-collecting Steiner-tree problem. Throughout this section, we assume that $G = (V, E)$ is a graph with root vertex r and $p(r) = 0$.

3.1 Adapting the Density-Greedy Algorithm

We start by generalizing statement (ii) and (iii) of Lemma 7 to general graphs.

▶ **Lemma 13.** *Let $T \in \mathcal{M}$ be a rooted subtree of G with maximum density d^*. Then the following holds:*

(i) *Every branch of T at r has density d^*.*
(ii) *If T is a min-max subtree of G, T has only one branch at r.*

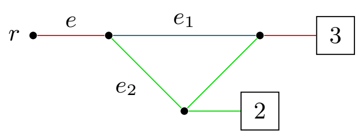

Figure 4 Assume all edges have unit cost. Then, G has two min-max subtrees: T_1 consists of the blue and red edges, T_2 consists of the green and red edges. Both have maximum density $d^* = 1$. Note that neither the union nor the intersection of T_1 and T_2 is a min-max subtree.
If we slightly reduce the cost of edge e_2 to $3/4$, tree T_2 is the only min-max subtree. Hence, in the first iteration of Algorithm 1, we contract edge e. However, in the second iteration, the tree spanned by the blue and red edges is the only min-max subtree and thus, Lemma 9 does not hold for general graphs.

A significant difference to the setting on trees, however, is that on general graphs the min-max subtrees are not closed under intersection and union, i.e., statement (i) of Lemma 7 does not hold anymore, as illustrated in Figure 4. As a consequence we do not obtain Lemma 9, that is, the edges of a fixed min-max subtree are not necessarily added to the solution consecutively. Since the analysis of the density-greedy algorithm crucially depends on this property, we need to make some adjustments to the algorithm when generalizing it in order to recover this property. The *density-greedy algorithm for general graphs* is then defined as follows.

Algorithm 2 The density-greedy algorithm for general graphs.

```
1: $\pi \leftarrow ()$
2: $\overline{T} \leftarrow (\{r\}, \emptyset)$
3: while $p_{G/\overline{T}}(G/\overline{T}) \neq 0$ do
4:     fix arbitrary min-max subtree $T^*$ of $G/\overline{T}$
5:     denote by $T^+$ the extension of $T^*$ in $G$
6:     run Algorithm 1 on $T^+/\overline{T}$ and obtain $\pi^+$
7:     append $\pi^+$ to $\pi$ and add edges of $T^+$ to $\overline{T}$
8: return $\pi$
```

First, we show that we do not obtain the same competitive ratios for the density-greedy algorithm for general graphs as we did for trees.

▶ **Proposition 14.** *The density-greedy algorithm is not (χ, μ)-competitive for any $\mu \geq 1$.*

We work towards showing that the density-greedy algorithm for general graphs is $(\gamma, 2)$-competitive, where γ denotes the maximum cost of a (vertex-disjoint) path in G starting in the root r. To do so, we again denote by T_1 the first min-max subtree that is computed in Step 4 of Algorithm 2. The following lemma is the counterpart to Lemma 10.

▶ **Lemma 15.** *For any cost budget $B \leq c(T_1)$, we have $p\big(\textsc{Alg}(B + \gamma)\big) \geq d(T_1)B$. Furthermore, if $B = c(T_1)$, we have $p\big(\textsc{Alg}(B)\big) = d(T_1)B$.*

We continue by generalizing the notion of the density to subtrees of general graphs. To this end, let T be a (not necessarily rooted) subtree of G. Since the choice of the vertex r_T in T that has the shortest distance to r is not necessarily unique when G is a graph, we define the density of T over an underlying rooted subtree $T' \supset T$. Then, the choice of the vertex r_T of T that has the shortest distance to r in T' is again unique and we define the density of T over T' as $d_{T'}(T) := \big(p(T) - p(r_T)\big)/c(T)$. Thereby, this notion is consistent with the density used in Section 2. We are now ready to prove the upper bound of Theorem 3.

▶ **Theorem 3.** *The density-greedy algorithm is $(\gamma, 2)$-competitive, but not (χ, μ)-competitive for any $\mu \geq 1$.*

Proof. The fact that the density-greedy algorithm is not (χ, μ)-competitive for any $\mu \geq 1$ follows from Proposition 14.

We proceed to show that $2p\big(\textsc{Alg}(B+\gamma)\big) \geq p\big(\textsc{Opt}(B)\big)$ for every cost budget $B \in \mathbb{R}_{\geq 0}$. We denote by $c(\pi)$ the total cost of all edges in the solution π. If $c(\pi) \leq B$, we obtain that $p(\textsc{Alg}(B+\gamma)) = p(G) \geq p\big(\textsc{Opt}(B)\big)$ and thus, the claim holds.

It remains to consider the case where $c(\pi) > B$. We denote by $T_1, \dots, T_n$ the extensions of the min-max subtrees computed in Step 5 of Algorithm 2 and let $k \in \{1, \dots, n\}$ be such that $\sum_{i=1}^{k-1} c(T_i) \leq B < \sum_{i=1}^{k} c(T_i)$. Note that by construction, T_i and T_j are edge disjoint for all $i \neq j$. Also, every T_i is a tree since the min-max subtrees all have only one branch at the root. We denote by $\overline{T}_i = \bigcup_{j=1}^{i} T_j$ the union of all trees $T_1, \dots, T_i$ and set $\overline{T}_0 = (\{r\}, \emptyset)$. We claim that

$$d_{\overline{T}_k}(T_1) \geq d_{\overline{T}_k}(T_2) \geq \cdots \geq d_{\overline{T}_k}(T_k). \tag{2}$$

To prove this, we assume for contradiction that there is some $1 \leq i < k$ such that $d_{\overline{T}_k}(T_i) < d_{\overline{T}_k}(T_{i+1})$. However, since $(T_i \cup T_{i+1})/\overline{T}_{i-1}$ is a rooted tree in $G/\overline{T}_{i-1}$, this assumption yields

$$d_{G/\overline{T}_{i-1}}((T_i \cup T_{i+1})/\overline{T}_{i-1}) = d_{\overline{T}_k}(T_i \cup T_{i+1}) > d_{\overline{T}_k}(T_i) = d_{G/\overline{T}_{i-1}}(T_i/\overline{T}_{i-1}),$$

contradicting the choice of $T_i/\overline{T}_{i-1}$ as a min-max subtree in $G/\overline{T}_{i-1}$. Thus, the claim holds. If we assume that $\textsc{Opt}(B) \subseteq \overline{T}_{k-1}$, using $\overline{T}_{k-1} \subseteq \textsc{Alg}(B)$ gives $p(\textsc{Opt}(B)) \leq p(\textsc{Alg}(B))$ and the theorem follows immediately. Hence, we assume for the remaining part of this proof that $\textsc{Opt}(B) \nsubseteq \overline{T}_{k-1}$. Using the facts that $T_k/\overline{T}_{k-1}$ is a min-max subtree of $G/\overline{T}_{k-1}$ and that $\overline{T}_{k-1} \subseteq \textsc{Alg}(B)$, we obtain

$$\begin{aligned} d_{\overline{T}_k}(T_k) = d_{G/\overline{T}_{k-1}}(T_k/\overline{T}_{k-1}) &\geq \frac{p_{G/\overline{T}_{k-1}}(\textsc{Opt}(B)/\overline{T}_{k-1})}{c_{G/\overline{T}_{k-1}}(\textsc{Opt}(B)/\overline{T}_{k-1})} \qquad (3)\\ &\geq \frac{p_{G/\textsc{Alg}(B)}(\textsc{Opt}(B)/\textsc{Alg}(B))}{B}. \qquad (4)\end{aligned}$$

Applying Lemma 15 iteratively on $G/\overline{T}_{i-1}$ with cost budget $c(T_i)$ for all $1 \leq i \leq k-1$ and for $G/\overline{T}_{k-1}$ with cost budget $B - c(\overline{T}_{k-1}) < c(T_k)$, we obtain

$$p\big(\textsc{Alg}(B+\gamma)\big) \geq \sum_{i=1}^{k-1} d_{\overline{T}_k}(T_i)c(T_i) + d_{\overline{T}_k}(T_k)\big(B - c(\overline{T}_{k-1})\big) \geq d_{\overline{T}_k}(T_k)B,$$

where we used $d_{\overline{T}_k}(T_i) \geq d_{\overline{T}_k}(T_k)$ for all $i \leq k$. Finally, we conclude that

$$\begin{aligned} p\big(\textsc{Opt}(B)\big) &\leq p\big(\textsc{Alg}(B)\big) + p_{G/\textsc{Alg}(B)}\big(\textsc{Opt}(B)/\textsc{Alg}(B)\big)\\ &\leq p\big(\textsc{Alg}(B)\big) + d_{\overline{T}_k}(T_k)B\\ &\leq p\big(\textsc{Alg}(B)\big) + p\big(\textsc{Alg}(B+\gamma)\big) \qquad (5)\\ &\leq 2p\big(\textsc{Alg}(B+\gamma)\big). \end{aligned}$$ ◀

The computational bottleneck of the density-greedy algorithm is the computation of a min-max subtree in Step 4 of Algorithm 2. As the computation of a subtree of maximum density is NP-hard (Lau et al. [34]), there is no polynomial implementation of this step, unless $\mathsf{P} = \mathsf{NP}$. However, Hermans et al. [28] give a polynomial 2-approximation for the computation of a subtree with maximal density. In order to obtain a polynomial algorithm for the incremental prize-collecting Steiner-tree problem on general graphs, we need to do the following adaptations to Algorithm 2:

1. In Step 4, we use the approximation algorithm of Hermans et al. [28] to obtain a tree T^* of $G/\overline{T}$ with density $d_{G/\overline{T}}(T^*) \geq d^*/2$ where d^* is the maximum density of a rooted subtree in $G/\overline{T}$.
2. Let T_i^* and T_{i+1}^* be two such trees computed in two consecutive iterations of the while loop such that the density of T_{i+1}^* is larger than the density of T_i^*. If both trees are rooted in the same vertex, we switch their order. Otherwise, we merge both trees.

The second point of these adjustments makes sure that the densities of the subtrees computed by the algorithm are decreasing, i.e., that inequality (2) in the proof of Theorem 3 still holds. We observe that when running Algorithm 2 with these adjustments, the proof of Theorem 3 goes through except for the fact that the right hand sides of inequalities (3) and (4) are multiplied with $1/2$. This ultimately leads to an additional loss of $p\big(\textsc{Alg}(B+\gamma)\big)$ in (5). We obtain the following result.

▶ **Corollary 4.** *An approximate variant of the density-greedy algorithm runs in polynomial time and is* $(\gamma, 3)$*-competitive.*

3.2 A Capacity-Scaling Algorithm

The density-greedy algorithm turns out to be a $(\gamma, 2)$-competitive algorithm for the prize-collecting Steiner-tree problem on general graphs. This result is particularly strong when the given instance G is sparse and hence, the maximum cost of a (vertex-disjoint) path starting in the root is small compared to the cost of a rooted spanning subtree of G. However, in general, the value γ can be large. This applies for example, when G is Hamiltonian. In such cases, it is more desirable to obtain an (α, μ)-competitive algorithm where α depends on χ rather than on γ. To this end, we introduce the *capacity-scaling algorithm* inspired by the incremental maximization algorithm of Bernstein et al. [8]. The basic idea is as follows. At the beginning the solution π is set to the empty sequence $()$. Then, in each iteration $i = 0, 1, \ldots$ the algorithm computes a maximum total prize rooted subtree T_i of G that obeys the cost budget $2^i\chi$. Afterwards, we run Algorithm 1 on T_i and append the returned solution to π. This is done in such a way, that the solution π contains neither double or parallel edges nor cycles.

The *capacity-scaling algorithm* is given in more detail in Algorithm 3.

Algorithm 3 The capacity-scaling algorithm.

```
1: $i \leftarrow 0$
2: $\pi \leftarrow ()$
3: **while** $\pi$ does not span all vertices in $V^*$ **do**
4:     $T_i \in \arg\max\{p(T) : T \in \mathcal{T}, c(T) \leq 2^i\chi\}$
5:     **run** Algorithm 1 on $T_i$ with root $r$ and obtain $\pi_i$
6:     append $\pi_i$ to $\pi$ (skip double edges, parallel edges, and edges that close cycles)
7:     $i \leftarrow i + 1$
8: **return** $\pi$
```

In order to analyse the capacity-scaling algorithm properly, we give the following lemmata for general trees (not necessary subtrees of G). Similar to the concept of min-max subtrees, we introduce the definition of min-max trees. In particular, we call a tree T with root vertex r_T

a *min-max tree with root* r_T if $p(r_T) = 0$ and for all rooted subtrees $T' = (V_{T'}, E_{T'}) \subset T$ with $r_T \in V_{T'}$ and $c(T') > 0$, we have $d(T) > d(T')$. First, we generalize Lemma 8 to min-max trees. The only adaption in the proof is that we use that T is a min-max tree instead of a min-max subtree. Specifically, we obtain the following result.

▶ **Lemma 16.** *Let T be a min-max tree with root r_T. Then, for every edge e in T we have $d_T(T_e) \geq d(T)$, where T_e is the branch of T rooted at e. Strict inequality holds, whenever $T \neq T_e$.*

The following lemma is crucial for the proof of Lemma 18, where Lemma 17 is used for $\delta = \left(1 - 2^{-k+1}\right)$.

▶ **Lemma 17.** *Let T be a tree with root r_T and let $\delta \in [0,1]$ and $k \in \mathbb{N}$. Assume that for every $\lambda' \in [0,1]$ and for every min-max tree T' with root $r_{T'}$, there exists a forest $F' = (V_{F'}, E_{F'}) \subseteq T'$ with $r_{T'} \in V_{F'}$ and at most k components such that $c(F') \leq \lambda' c(T')$ and $p(F') \geq \delta\lambda' p(T')$. Then, for all $\lambda \in [0,1]$, there exists a forest $F = (V_F, E_F) \subseteq T$ with $r_T \in V_F$ and at most k components such that $c(F) \leq \lambda c(T)$ and $p(F) \geq \delta\lambda p(T)$.*

Using the results of Lemma 17, we can show the following.

▶ **Lemma 18.** *Let T be a tree with root r_T and let $\lambda \in [0,1]$ and $k \in \mathbb{N}$. Then there exists a forest $F = (V_F, E_F) \subseteq T$ with $r_T \in V_F$ and at most k components such that*

$$c(F) \leq \lambda c(T) \quad \textit{and} \quad p(F) \geq \left(1 - 2^{-k+1}\right) \lambda p(T).$$

The following corollary will be useful for the analysis of the capacity-scaling algorithm.

▶ **Corollary 19.** *For every $\delta \geq 1$, $h \in \mathbb{N}$, and every cost budget $B \in \mathbb{R}_{\geq 0}$, we have that $p(\textsc{Opt}(B + h\chi)) \geq (1 - 2^{-h})\delta^{-1} p(\textsc{Opt}(\delta B))$.*

We are now ready to prove the main theorem of this subsection.

▶ **Theorem 6.** *The capacity-scaling algorithm is $\left((4\ell - 1)\chi, \frac{2^{\ell+2}}{2^\ell - 1}\right)$-competitive for every $\ell \in \mathbb{N}$.*

Proof. For cost budget $B \in \mathbb{R}_{\geq 0}$, let $k \in \mathbb{N}$ be such that $B \in \left[\left(2^{k+1} - 4\ell\right)\chi, \left(2^{k+2} - 4\ell\right)\chi\right)$. We denote by $\textsc{Alg}(B)$ the solution of the capacity-scaling algorithm for cost budget B. Since $\textsc{Alg}$ grows monotonously, it holds that

$$p(\textsc{Alg}(B + (4\ell - 1)\chi)) \geq p(\textsc{Alg}((2^{k+1} - 1)\chi)). \tag{6}$$

Let $T_0, T_1, \ldots$ be the subtrees of G computed by the capacity-scaling algorithm (Algorithm 3). Note that

$$\sum_{i=0}^{k} c(T_i) \leq \sum_{i=0}^{k} 2^i \chi = (2^{k+1} - 1)\chi.$$

Thus, $\textsc{Alg}((2^{k+1} - 1)\chi)$ contains the tree T_k, i.e.,

$$p(\textsc{Alg}((2^{k+1} - 1)\chi)) \geq p(T_k). \tag{7}$$

Let $\textsc{Opt}(2^k\chi)$ be the optimal solution in G for cost budget $2^k\chi$. Then, a spanning tree of $\textsc{Opt}(2^k\chi)$ is a potential candidate for the capacity-scaling algorithm when computing tree T_k, which yields $p(\textsc{Opt}(2^k\chi)) = p(T_k)$. Finally, we use Corollary 19 with $\delta = 4$, $h = \ell$

and cost budget $B_k := (2^k - \ell)\chi \geq \frac{B}{4}$ and together with inequalities (6) and (7) we obtain the claim as follows

$$\begin{aligned} p(\textsc{Alg}(B + (4\ell - 1)\chi)) &\geq p(\textsc{Alg}((2^{k+1} - 1)\chi)) \\ &\geq p(T_k) \\ &= p(\textsc{Opt}(2^k\chi)) \\ &= p(\textsc{Opt}(B_k + \ell\chi)) \\ &\geq \frac{1 - 2^{-\ell}}{4} p\big(\textsc{Opt}(4B_k)\big) \\ &\geq \frac{2^\ell - 1}{2^{\ell+2}} p\big(\textsc{Opt}(B)\big). \end{aligned} \tag{8}$$ ◀

The computational bottleneck of the capacity-scaling algorithm is the solution of the prize-collecting Steiner tree problem with budget $2^i\chi$ in Step 4 of Algorithm 3. The NP-hardness of the Steiner tree problem implies the hardness of the budget variant of the prize-collecting Steiner tree problem, so there is no polynomial implementation of Step 4, unless $\mathsf{P} = \mathsf{NP}$. In terms of approximations, only a quasi-polynomial algorithm with a poly-logarithmic approximation guarantee is known (Ghuge and Nagarajan [21]); see also the discussion by Paul et al. [38]. We leave it as an open problem to devise a polynomial algorithm that is $(O(1)\chi, O(1))$-competitive.

3.3 Lower Bound for General Graphs

We complement our algorithmic results with a lower bound for general graphs. For the proof, we first construct a small graph for which there is a trade-off between the maximal prize that can be collected at budgets 3 and 5, respectively. A careful analysis of the graph that has an arbitrary large number of copies of this graph joined at the root then yields the following result.

▶ **Theorem 5.** *For every (α, μ)-competitive algorithm with α only depending on γ, it holds that $\mu \geq 17/16$.*

References

1 Ajit Agrawal, Philip N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995. doi:10.1137/S0097539792236237.

2 Ali Ahmadi, Iman Gholami, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Mohammad Mahdavi. 2-approximation for prize-collecting Steiner forest. In David P. Woodruff, editor, *Proceedings of the 35th ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, pages 669–693, 2024. doi:10.1137/1.9781611977912.25.

3 Ali Ahmadi, Iman Gholami, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Mohammad Mahdavi. Prize-collecting Steiner tree: A 1.79 approximation. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1641–1652, 2024. doi:10.1145/3618260.3649789.

4 Steve Alpern and Thomas Lidbetter. Mining coal or finding terrorists: The expanding search paradigm. *Oper. Res.*, 61(2):265–279, 2013. doi:10.1287/OPRE.1120.1134.

5 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. doi:10.1137/090771429.

6 Sunil Arya and H. Ramesh. A 2.5-factor approximation algorithm for the k-MST problem. *Inf. Process. Lett.*, 65(3):117–118, 1998. doi:10.1016/S0020-0190(98)00010-6.

7 Egon Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989. doi:10.1002/NET.3230190602.

8 Aaron Bernstein, Yann Disser, Martin Groß, and Sandra Himburg. General bounds for incremental maximization. *Math. Program.*, 191(2):953–979, 2022. doi:10.1007/S10107-020-01576-0.

9 Daniel Bienstock, Michel X. Goemans, David Simchi-Levi, and David P. Williamson. A note on the prize collecting traveling salesman problem. *Math. Program.*, 59:413–420, 1993. doi:10.1007/BF01581256.

10 Avrim Blum, Prasad Chalasani, Don Coppersmith, William R. Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 163–171, 1994. doi:10.1145/195058.195125.

11 Avrim Blum, R. Ravi, and Santosh S. Vempala. A constant-factor approximation algorithm for the k-MST problem. *J. Comput. Syst. Sci.*, 58(1):101–108, 1999. doi:10.1006/JCSS.1997.1542.

12 Costas Busch, Da Qi Chen, Arnold Filtser, Daniel Hathcock, D. Ellis Hershkowitz, and Rajmohan Rajaraman. One tree to rule them all: Poly-logarithmic universal Steiner tree. In *Proceedings of the 64th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 60–76, 2023. doi:10.1109/FOCS57990.2023.00012.

13 Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, 2013. doi:10.1145/2432622.2432628.

14 Miroslav Chlebík and Janka Chlebíková. The Steiner tree problem on graphs: Inapproximability results. *Theor. Comput. Sci.*, 406(3):207–214, 2008. doi:10.1016/J.TCS.2008.06.046.

15 Yann Disser, Svenja M Griesbach, Max Klimm, and Annette Lutz. Bicriterial approximation for the incremental prize-collecting steiner-tree problem. *arXiv preprint*, 2024. arXiv:2407.04447.

16 Yann Disser, Max Klimm, Annette Lutz, and David Weckbecker. Fractionally subadditive maximization under an incremental knapsack constraint with applications to incremental flows. *SIAM J. on Discret. Math.*, 8:764–789, 2024. doi:10.1137/23M156926.

17 Yann Disser, Max Klimm, Kevin Schewior, and David Weckbecker. Incremental maximization via continuization. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *Proceedings of the 50th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 47:1–47:17, 2023. doi:10.4230/LIPICS.ICALP.2023.47.

18 Joan Feigenbaum, Christos H. Papadimitriou, and Scott Shenker. Sharing the cost of multicast transmissions. *J. Comput. Syst. Sci.*, 63(1):21–41, 2001. doi:10.1006/JCSS.2001.1754.

19 Naveen Garg. A 3-approximation for the minimum tree spanning k vertices. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 302–309, 1996. doi:10.1109/SFCS.1996.548489.

20 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 (STOC)*, pages 396–402, 2005. doi:10.1145/1060590.1060650.

21 Rohan Ghuge and Viswanath Nagarajan. Quasi-polynomial algorithms for submodular tree orienteering and directed network design problems. *Math. Oper. Res.*, 47(2):1612–1630, 2022. doi:10.1287/MOOR.2021.1181.

22 Michel X. Goemans and Jon M. Kleinberg. An improved approximation ratio for the minimum latency problem. *Math. Program.*, 82:111–124, 1998. doi:10.1007/BF01585867.

23 Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995. doi:10.1137/S0097539793242618.

24 Svenja M. Griesbach, Felix Hommelsheim, Max Klimm, and Kevin Schewior. Improved approximation algorithms for the expanding search problem. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *Proceedings of the 31st Annual European Symposium on Algorithms (ESA)*, pages 54:1–54:15, 2023. `doi:10.4230/LIPICS.ESA.2023.54`.

25 Albert Gu, Anupam Gupta, and Amit Kumar. The power of deferral: Maintaining a constant-competitive Steiner tree online. *SIAM J. Comput.*, 45(1):1–28, 2016. `doi:10.1137/140955276`.

26 Anupam Gupta and Amit Kumar. Online Steiner tree with deletions. In Chandra Chekuri, editor, *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 455–467, 2014. `doi:10.1137/1.9781611973402.34`.

27 Mohammad Taghi Hajiaghayi and Kamal Jain. The prize-collecting generalized Steiner tree problem via a new approach of primal-dual schema. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 631–640, 2006.

28 Ben Hermans, Roel Leus, and Jannik Matuschke. Exact and approximation algorithms for the expanding search problem. *INFORMS J. Comput.*, 34(1):281–296, 2022. `doi:10.1287/IJOC.2020.1047`.

29 Stefan Hougardy and Hans Jürgen Prömel. A 1.598 approximation algorithm for the Steiner problem in graphs. In Robert Endre Tarjan and Tandy J. Warnow, editors, *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1999.

30 Makoto Imase and Bernard M. Waxman. Dynamic Steiner tree problem. *SIAM J. Discret. Math.*, 4(3):369–384, 1991. `doi:10.1137/0404033`.

31 Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for TSP, Steiner tree, and set cover. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 386–395, 2005. `doi:10.1145/1060590.1060649`.

32 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 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 760–769, 2000.

33 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–103, 1972. `doi:10.1007/978-1-4684-2001-2_9`.

34 Hoong Chuin Lau, Trung Hieu Ngo, and Bao Nguyen Nguyen. Finding a length-constrained maximum-sum or maximum-density subtree and its application to logistics. *Discret. Optim.*, 3(4):385–391, 2006. `doi:10.1016/J.DISOPT.2006.06.002`.

35 Asaf Levin. A better approximation algorithm for the budget prize collecting tree problem. *Oper. Res. Lett.*, 32(4):316–319, 2004. `doi:10.1016/J.ORL.2003.11.002`.

36 Guolong Lin, Chandrashekhar Nagarajan, Rajmohan Rajaraman, and David P. Williamson. A general approach for incremental approximation and hierarchical clustering. *SIAM J. Comput.*, 39(8):3633–3669, 2010. `doi:10.1137/070698257`.

37 Joseph Naor, Debmalya Panigrahi, and Mohit Singh. Online node-weighted Steiner tree and related problems. In Rafail Ostrovsky, editor, *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 210–219, 2011. `doi:10.1109/FOCS.2011.65`.

38 Alice Paul, Daniel Freund, Aaron Ferber, David B. Shmoys, and David P. Williamson. Erratum to "Budgeted prize-collecting traveling salesman and minimum spanning tree problems"'. *Math. Oper. Res.*, 48(4):2304–2307, 2023. `doi:10.1287/moor.2022.1340`.

39 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. `doi:10.1287/MOOR.2019.1002`.

40 Gabriel Robins and Alexander Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM J. Discret. Math.*, 19(1):122–134, 2005. `doi:10.1137/S0895480101393155`.

41 F. Sibel Salman, Joseph Cheriyan, R. Ravi, and S. Subramanian. Approximating the single-sink link-installation problem in network design. *SIAM J. Optim.*, 11(3):595–610, 2001. doi:10.1137/S1052623497321432.

42 Yogeshwer Sharma, Chaitanya Swamy, and David P. Williamson. Approximation algorithms for prize collecting forest problems with submodular penalty functions. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1275–1284, 2007.

43 David Weckbecker. *Competitive Analysis for Incremental Maximization.* PhD thesis, TU Darmstadt, 2023.

44 Chenyang Xu and Benjamin Moseley. Learning-augmented algorithms for online Steiner tree. In *Proceedings of the 36th Conference on Artificial Intelligence (AAAI)*, pages 8744–8752, 2022. doi:10.1609/AAAI.V36I8.20854.