Parameterized Analysis of the Online Priority and Node-Weighted Steiner Tree Problems

Spyros Angelopoulos

Parameterized Analysis of the Online Priority and Node-Weighted Steiner Tree Problems*

Spyros Angelopoulos
CNRS and Sorbonne Université
spyros.angelopoulos@lip6.fr

November 10, 2020

Abstract

In this paper we study the online variant of two well-known Steiner tree problems. In the online setting, the input consists of a sequence of terminals; upon arrival of a terminal, the online algorithm must irrevocably buy a subset of edges and vertices of the graph so as to guarantee the connectivity of the currently revealed part of the input. More precisely, we first study the *online node-weighted Steiner tree problem*, in which both edges and vertices are weighted, and the objective is to minimize the total cost of edges and vertices in the solution. We then address the *online priority Steiner tree problem*, in which each edge and each request are associated with a priority value, which corresponds to their bandwidth support and requirement, respectively. Both problems have applications in the domain of multicast network communications and have been studied from the point of view of approximation algorithms.

Motivated by the observation that competitive analysis gives very pessimistic and unsatisfactory results when the only relevant parameter is the number of terminals, we introduce an approach based on *parameterized* analysis of online algorithms. In particular, we base the analysis on additional parameters that help reveal the true complexity of the underlying problem, and allow a much finer classification of online algorithms based on their performance. More specifically, for the online node-weighted Steiner tree problem, we show a tight bound of $\Theta(\max\{\min\{\alpha, k\}, \log k\})$ on the competitive ratio, where α is the ratio of the maximum node weight to the minimum node weight and k is the number of terminals. For the online priority Steiner tree problem, we show corresponding tight bounds of $\Theta(b \log \frac{k}{b})$, when $k > b$ and $\Theta(k)$, when $k \leq b$, where b is the number of priority levels and k is the number of terminals. Our main results apply to both deterministic and randomized algorithms, as well as to generalized versions of the problems (i.e., to Steiner forest variants).

1 Introduction

1.1 Problem statements and motivation

Steiner tree problems occupy a central place in optimization theory. In its standard formulation, the input to the Steiner tree problem consists of an undirected graph $G = (V, E)$, with cost function $c : E \rightarrow \mathbb{R}^+$, as well as a set of vertices $K \subseteq V$, also called *terminals*. The objective is to to find a minimum-cost subgraph

*This paper is an extended combined version of two conference papers: *The Node-Weighted Steiner Problem in Graphs of Restricted Node Weights*, Proceedings of the 10th Scandinavian Workshop on Algorithm Theory, pp 208–219, 2006 and *Online Priority Steiner Tree Problems*, Proceedings of the 11th Symposium on Algorithms and Data Structures, pp 27–48, 2009.

which spans all vertices in K (clearly, such a subgraph is always a tree). The cost of the Steiner tree is defined as the sum of the costs of its edges. A generalization of this fundamental problem associates the set K with of k *pairs* of vertices $(s_1, t_1), \ldots, (s_k, t_k)$. Once again, the objective is to identify a subgraph of minimum cost, such that for all $i \in [1, k]$ there is a path connecting s_i to t_i in the subgraph. The problem is known as the *Generalized Steiner Problem* (GS), or the *Steiner Forest Problem* since the solution is a forest of G.

In the online version of the problem, the set of terminals K is not known in advance. Instead, the terminals are revealed as a sequence of *requests*. Upon arrival of a request $t \in K$, the online algorithm must guarantee connectivity of all terminals revealed up to this point in time; this implies that the algorithm may need to update its current solution by buying new edges. This action is irrevocable, in the sense that any edge that is bought contributes to the final solution cost. We make the usual assumption in the field that the graph G is known to the algorithm. In the standard framework of competitive analysis (see, e.g. [11]), the goal is to design online algorithms of small competitive ratio. In the context of Steiner tree problems, in particular, the competitive ratio is defined as the supremum of the ratio of the cost of the solution produced by the algorithm over the optimal off-line cost (namely the cost of an offline algorithm which has complete knowledge of the request set K).

Both the offline and online variants of Steiner tree problems have been studied extensively in the literature (see Section 1.2 for some representative results) and are often encountered in the context of several combinatorial optimization problems. In addition to their appeal from the point of view of theoretical analysis, they offer useful formulations of real-life applications. For instance, they are useful in abstracting the design of multicast protocols in computer networks, which involves distribution of the same information stream to the members of the multicast group over an existing network. Indeed, multicasting can be modeled as the problem of selecting communication links (i.e, edges of the underlying graph) so as to minimize the cost (in terms of either bandwidth and/or the physical cost of links) for supporting traffic routing through a tree. For the interplay between the Steiner Tree problem and multicast applications, the interested reader is referred to the thesis [17] and the survey [32].

In this work we study the online variants of the following Steiner tree problems:

- The *node-weighted Steiner tree* problem (abbr. NWST). Here, not only the edges, but also the vertices of the graph G are associated with a weight (cost). More formally, the cost function is defined as $c : V \cup E \rightarrow \mathbb{R}^+$. The solution cost is the sum of the cost of all edges and vertices in the tree solution.

- The *priority Steiner tree* problem (abbr. PST). Here, each edge $e \in E$ is associated with a *priority* $p(e) \in \{1, 2, \ldots, b\}$, where $b \in \mathbb{N}^+$ is some specified maximum priority value. In addition, every terminal $t \in K$ is associated with a priority *demand* $p(t) \in \{1, 2, \ldots, b\}$. Each edge has a cost function $c : V \rightarrow \mathbb{R}^+$ (but nodes do not have weights). Last, a vertex $r \in V$ is specified as the *source* or *root*. A feasible solution is a subgraph of G in which there is a path p_t from the root to each terminal $t \in K$ such that each edge in p_t has priority at least $p(t)$.

The *generalized node-weighted Steiner problem* (GNWS) and the *generalized priority Steiner tree problem* (GPS) are defined as extensions of the tree variants in the natural way, i.e, by substituting the set of terminals K with a set of pairs $\{(s_i, t_i) | i \in [1, k]\}$. For GPS, in particular, each pair (s_i, t_i) requests priority p_i, and the solution stipulates a path from s_i to t_i in which each edge has priority at least p_i. Throughout the paper we will always denote $|K|$ by k.

As with the standard version, both variants we consider in this work arise often in interesting applications. For instance, node-weighted Steiner tree formulations have been used in multicasting under multi-domain and hierarchical constraints [8] as well as in environmental engineering, specifically in the design

of wildlife corridors for multiple species [30]. Moreover, the priority Steiner tree problem is a formulation of multicasting with Quality of Service (QoS) guarantees in networks with heterogeneous users. More precisely, each priority level corresponds to a bandwidth (or QoS) level, the priority of a terminal reflects the bandwidth requirement of the end user, and the solution guarantees that information can be efficiently disseminated from the source to all users of the multicast group in accordance to QoS provisions [14].

Note that neither online node-weighted Steiner tree problems nor online priority Steiner tree problems have been studied prior to the conference versions of this paper. We believe that this is largely due to the fact that, at first sight, there are straightforward matching upper and lower bounds on the worst-case competitive ratio of the two problems. It is very easy, indeed, to show that every deterministic/randomized algorithm is $\Omega(k)$-competitive, using an adversarial argument along the lines of [38]. Moreover, even extremely naive algorithms are $O(k)$-competitive, and thus optimal. Consider, for instance, the naive algorithm for PST which buys a new path from the root to the currently requested terminal; in particular, the algorithm does not set the weight of each bought edge to zero. This algorithm achieves the same competitive ratio as the greedy algorithm, which considers, in a very natural way, all previously bought edges as available for free when serving future requests. The same holds for the naive algorithm for NWST which buys a new path to the current tree, without considering previously bought edges and vertices. It is immediately obvious that the greedy algorithm should have better performance than the naive algorithm, however competitive analysis does not reflect this separation, and deems both algorithms optimal. In a sense, this deficiency of competitive analysis has parallels with the well-studied *paging* problem, in which extremely naive algorithms (such as Flush-When-Full) attain the optimal competitive ratio [11]. This drawback has already been observed by Ramanthan [33] and Faloutsos *et al.* [18] in the context of the online *Directed Steiner tree* problem, who followed the approach of *parameterized analysis*. More precisely, their analysis incorporates not only the number of terminals k, but also the *edge asymmetry* of the underlying directed graph, which informally is a measure of the "directedness" of the input graph G. Once the asymmetry of the graph is taken into consideration as a parameter in the analysis, it is indeed possible to separate algorithms based on their anticipated performance, and in [4, 5, 6] we were able to establish optimal competitive ratios. More significantly, this parameterized analysis helps reveal the true complexity and richness of the problem which is lost when the competitive ratio is a function only of the number of terminals k.

The above discussion motivates our approach in the context of the online node-weighted and priority Steiner tree problems. For the former, we define the *node asymmetry* (or simply *asymmetry* when clear from context) as the quantity $\alpha = \max_{v,u \in V} c(v)/c(u)$, i.e., the ratio of the maximum node weight to the minimum node weight in the graph. For the latter, the relevant parameter is the number of priority levels b. Note that if $\alpha = 1$ or $b = 1$, the problems reduce to the classic Steiner tree problem; in contrast, when α or b attain relatively large values, it is easy to show that the competitive ratios are (unavoidably) linear in k. Our goal in this work is to obtain tight bounds on the competitive ratios for all possible ranges of the parameters (as opposed to only the extreme values).

We note that for a directed graph $G = (V, E)$, its edge asymmetry is defined as follows [33]. Let A denote the set of pairs of vertices in V such that if the pair u, v is in A, then either $(v, u) \in E$ or $(u, v) \in E$ (i.e, there is an edge from u to v or an edge from v to u or both). Then the edge asymmetry of G is defined as $\gamma = \max_{\{v,u\} \in A} \frac{c(v,u)}{c(u,v)}$.

1.2 Related work

The standard Steiner Tree problem (in which $c(v) = 0$, for all $v \in V$), was one of the earliest problems shown to be NP-hard by Karp [28]; in fact, the problem is APX-hard [10] [35]. Currently, the best up-

per bound for general graphs is 1.55 due to Byrka *et al.* [12]. For the Generalized Steiner Problem, the corresponding bound is 2 [21],[1]. In terms of online algorithms, Imase and Waxman [27] showed that a simple greedy algorithm achieves optimal competitive ratio $\Theta(\log k)$ for the online Steiner Tree. Berman and Coulston [9] extended the result to the Generalized Steiner problem by providing a more sophisticated algorithm. The performance of the greedy algorithm for the Generalized Steiner Problem has also been studied by Westbrook and Yan [39] and Awerbuch *et al.* [7]; the latter showed that the greedy algorithm is $O(\log |V|^2)$-competitive. For the online Steiner Tree in the Euclidean plane, the best known lower bound on the competitive ratio is $\Omega(\log k/ \log \log k)$ due to Alon and Azar [3]. Westbrook and Yan [38] showed that in directed graphs, the competitive ratio can be as bad as $\Omega(k)$, which is trivially matched by a naive algorithm that serves each request by buying a least-cost path from the root to the requested terminal. More recently, several other settings for the online Steiner tree problem have been studied (e.g. the setting in which a request may correspond to a deletion rather than insertion of a terminal [25], algorithms with limited revocable decisions [22], and stochastic requests [20]). Recent work on online Steiner tree problems builds heavily on online primal-dual approaches and linear-programming techniques (see [2] for a framework for online algorithms in the context of generalized connectivity and cut problems).

Berman (see reference in [29]) showed an approximation-preserving reduction from Set Cover to NWST, which implies, using results of [19] and [34], that NWST cannot be approximated within a factor of $o(\log k)$ unless P=NP. We emphasize that the hardness result holds only when α is unbounded, in particular, it is required that α is as big as $\Omega(n)$. Klein and Ravi [29] presented an asymptotically optimal algorithm which guarantees an approximation ratio of $2 \ln k$. Guha and Khuller [24] improved the upper bound to $1.35 \ln k$ (approximately). Concerning the online variant, Naor *et al.* [31] showed a $O(\log n \log^2 k)$ upper bound for online NWST a result which was extended to GNWS by Hajiaghayi *et al.* [26]; for both problems a lower bound of $\Omega(\log n \log k)$ applies. We emphasize that n denotes the number of vertices in the graph.

The priority Steiner tree problem was introduced by Charikar, Naor and Schieber [14]. In their work, they provided $O(\min\{\log k, b\})$ approximation algorithms for both PST and GPS. The question whether PST could be approximated within a constant factor was left open in [14], and sparked considerable interest in this problem, until Chuzhoy *et al.* [15] showed a lower bound of $\Omega(\log \log n)$ (under the complexity assumption that NP does not have slightly superpolynomial time deterministic algorithms). Interestingly, this is one of few problems for which a $\log \log$-inapproximability result is currently the best known.

To the best of our knowledge, the first application of parameterized analysis in the context of Steiner tree problems is due to Ramanathan [33] who studied approximations of the Steiner tree problem in directed graphs of bounded *edge asymmetry*. Informally, the edge asymmetry captures the "directedness" of the underlying graph. [33] proposed three different metrics of edge asymmetry, with the metric γ (defined earlier) being the most natural one. According to this measure, undirected graphs are graphs of asymmetry $\gamma = 1$, whereas directed graphs in which there is at least one pair of vertices v, u such that $(v, u) \in E$, but $(u, v) \notin E$ are graphs with unbounded asymmetry ($\gamma = \infty$). Between these extreme cases, graphs of small asymmetry are useful, for instance, in modeling networks which are relatively homogeneous networks in terms of the cost of antiparallel links. In [33] it was shown that for two of the proposed asymmetry metrics, there is an algorithm with approximation ratio at most twice the asymmetry metric (when this metric is defined by γ, the result is straightforward). [33] also gave an experimental evaluation of this algorithm in the setting of randomly generated graphs.

The first study of the online Steiner tree problem in graphs of bounded edge asymmetry γ is due to Faloutsos *et al.* [18]. The bounds on the competitive ratio were improved in [4, 5], which showed that the competitive ratio of the simple greedy algorithm is bounded by $O\left(\min\left\{\max\left\{\gamma\frac{\log k}{\log \gamma}, \gamma\frac{\log k}{\log\log k}\right\}, k\right\}\right)$, and

that every online algorithm has competitive ratio $\Omega\left(\min\left\{\max\left\{\gamma\frac{\log k}{\log\gamma}, \gamma\frac{\log k}{\log\log k}\right\}, k^{1-\epsilon}\right\}\right)$, for arbitrarily small constant ϵ. The small gap between these bounds was finally closed in [6] which showed that for all γ in the range of values for which [5] is not tight, the competitive ratio of the greedy algorithm is $\Theta\left(\frac{\log(k/\gamma)}{\log\log(k/\gamma)}\right)$, and this bound is tight for any deterministic algorithm. It is worth emphasizing that the introduction of the edge asymmetry as an additional parameter yields a far more challenging and interesting analysis than the trivial $\Theta(k)$ bound (which is optimal even for a naive algorithm that serves a new request by ignoring edges that have already been bought [38]).

1.3 Contribution of this paper

Concerning NWST, we begin with a study of the approximability of its offline variant. Our motivation stems from the results of [14], who gave parameterized approximation algorithms for PST and GPS; this demonstrates that parameterized analysis has been useful in the study of approximability of these problems, so one should naturally investigate whether similar benefits can be attained in the context of their online variants. More precisely, we first show that unless P=NP, offline NWST cannot be approximated within a factor better than $O(\min\{\log\alpha, \log k\})$ (Theorem 6). The proof uses ideas behind the reduction of Klein and Ravi (see section 1.2), however we reduce from Set Cover of Bounded Set Size; the result then follows from Trevisan's inapproximability result [36]. Theorem 7 shows that the hardness bound is asymptotically tight, by presenting an algorithm which is a combination of the Klein-Ravi algorithm for GNWS and the constant-factor approximation algorithm of Goemans and Williamson for GS.

We then move to the onlive variant of the problems. Specifically, in Theorem 10 we show an asymptotically tight bound of $\Theta(\max\{\min\{\alpha, k\}, \log k\})$ on the competitive ratio of (deterministic or randomized) online algorithms for NWST and GNWS. Concerning online priority Steiner problems, the main technical result is a tight bound on the competitive ratio of deterministic algorithms for both PST and GPS equals $\Theta\left(b\log\frac{k}{b}\right)$ (if $k > b$), and $\Theta(k)$, otherwise (Theorem 11 and Theorem 12). The bounds extend to randomized algorithms (Theorem 17). Although the upper bound is relatively simple, the lower bound requires an inductively-defined construction of an adversarial graph that combines several components, and which enforces every asymptotically optimal algorithm to have essentially the same behavior as the greedy algorithm.

Next, we study the competitiveness of PST in directed graphs. This is motivated by the above-mentioned fact that PST is a useful formulation of multicasting with QoS guarantees. Moreover, one should take into account that directed graphs provide a more accurate and realistic representation of a network. Indeed, studies on the traffic of network backbones reveal marked asymmetry in parameters such as speed, link utilization and reliability [16]. We first observe that if antiparallel links have the same costs, but their priorities can differ by as little as one, then a tight bound of $\Theta(k)$ on the competitive ratio follows easily. Hence we focus on the case in which antiparallel links have the same priority, but their costs may vary. In particular, we consider directed graphs of bounded edge-cost asymmetry γ, as defined in Section 1.2. More precisely, we derive an upper bound of $O\left(\min\left\{\max\left\{\gamma b\frac{\log(k/b)}{\log\gamma}, \gamma b\frac{\log(k/b)}{\log\log(k/b)}\right\}, k\right\}\right)$ on the performance of the greedy algorithm (Theorem 15), and a corresponding lower bound of $\Omega\left(\min\left\{\gamma b\frac{\log(k/b)}{\log\gamma}, k^{1-\epsilon}\right\}\right)$, where ϵ is any arbitrarily small constant, for any online algorithm (Theorem 16). In other words, we demonstrate how to apply parameterized analysis of an online problem with three instance-related parameters: the number of terminals, the edge asymmetry and the number of priority levels. This analysis succeeds in separating the greedy from the naive algorithm, for a large range of the parameters γ and b, as will be argued in Section 4.3.

Last, we show how to use techniques introduced by Alon *et al.* [2] in order to obtain a $O(\log k \log m)$-

competitive randomized algorithm for online PST and GPS, where m is the number of edges in the graph (Theorem 20). The result demonstrates that when k is comparable to m (and thus n as well), there exist efficient online algorithms regardless of the number of priority levels.

Table 1 summarizes the main results of this paper, in the context of previous work.

Table 1: Summary of previous work and main results in this paper. UB and LB abbreviate "upper bound" and "lower bound", respectively. Results without citation appear in this work.

		Node-weighted Steiner	Priority Steiner	Directed Steiner
Online non-parameterized	UB	$O(k)$	$O(k)$	$O(k)$ [38]
	LB	$\Omega(k)$	$\Omega(k)$	$\Omega(k)$ [38]
Online parameterized	UB	$O(\max\{\min\{\alpha, k\}, \log k\})$	$O(\min\{b \log \frac{k}{b}, k\})$	$O\left(\min\left\{\max\left\{\gamma \frac{\log k}{\log \gamma}, \gamma \frac{\log k}{\log\log k}\right\}, k\right\}\right)$ [5]
	LB	$\Omega(\max\{\min\{\alpha, k\}, \log k\})$	$\Omega(\min\{b \log \frac{k}{b}, k\})$	$\Omega\left(\min\left\{\max\left\{\gamma \frac{\log k}{\log \gamma}, \gamma \frac{\log k}{\log\log k}\right\}, k^{1-\epsilon}\right\}\right)$ [4, 6]
Offline non-parameterized	UB	$O(\log k)$ [29]	$O(\log k)$ [14]	$O(k^{\epsilon})$ [13]
	LB	$\Omega(\log k)$ [29]	$\Omega(\log\log n)$ [15]	$\Omega(\log k)$ [13]
Offline parameterized	UB	$O(\min\{\log \alpha, \log k\})$	$O(\min\{b, \log k\})$ [14]	2γ [33]
	LB	$\Omega(\min\{\log \alpha, \log k\})$		

2 Preliminaries

Given a (simple) path P between two vertices v, u in G, the cost of P is the sum of the costs of all vertices and edges in P, *excluding* the end vertices v and u. Note that a path of minimum cost can be computed in polynomial time, by constructing a directed graph $G' = (V, E')$ in which only edges have cost, such that for every edge $e = (v, u) \in E$, $e' = (v, u) \in E'$, and $c(e') = c(e) + c(u)$. It is easy to see that a shortest path from v to u in G' translates to a minimum-cost path from v to u in G.

Given graph G with edge and vertex weights, define the *shortest path completion of* G (or simply path completion), as the complete graph $G_p = (V, E')$ in which every vertex has the same cost as in V, and the cost of an edge $e = (v, u) \in E'$ is the cost of a minimum-cost path between v and u in G. Note that the path completion can be computed in polynomial time. Following a standard practice in the study of Steiner trees we observe that, when we need so, we can restrict our attention to the path completion of G, in the sense that a solution to NWST in G_p can be transformed in polynomial time to a solution to NWST in G without any increase to the solution's cost, and without affecting the approximation ratio (see, e.g., Theorem 3.2 in [37] which is cast in the context of the metric completion of the graph, but also can be applied in the case of the path completion, even though the latter does not necessarily give rise to metric distances).

Using standard terminology, given a Steiner tree $T = (V_T, E_T)$ for a set of terminals K, we call vertices in $V_T \setminus K$ *Steiner* vertices. We will use the following property.

Observation 1. *Let (G_p, K) be the input to* NWST*, then we can transform any solution (Steiner tree) T to a a tree T' which has total cost at most the cost of T, and for which the number of Steiner nodes (i.e., vertices which are not terminals) is at most $|K| - 2$.*

Proof. We will show that there exists a Steiner tree T' for K of cost at most the cost of T such that every Steiner vertex in T' has degree at least 3. Let $S(T')$ denote the set of Steiner vertices of tree T', and $deg_{T'}(v)$ denote the degree of v in T'. Then, we have that

$$3|S(T')| + |K| \leq \sum_{v \in S(T') \cup K} deg_{T'}(v) = 2(|K| + |S(T')| - 1),$$

which yields the desired result. Suppose that in T there exists a Steiner vertex v of degree 2 (note that Steiner vertices of degree 1 can be removed from the tree without increasing its cost). Let u_1, u_2 denote the neighbors of v in T. Then, using the definition of the path completion G_p, we can replace the path u_1, v, u_2 by the single edge (u_1, u_2) without increasing the cost of the tree. Repeating this process, we end up with a tree with the desired property. □ □

The following straightforward observation will simplify some cost manipulations in the proofs.

Observation 2. *For* NWST *algorithms, when computing the cost of solutions, we can assume, without loss of generality, that terminals have zero weight.*

This is because the contribution of terminals to the total node weight is the same in any online algorithm as well as in the offline optimal solution. Note that this is not to say that their weights are actually zero (which would imply infinite asymmetry), but rather that we can treat them as zero, without affecting the bounds.

Throughout this paper we assume that parameters such as α, β, γ are integers, since we can always scale them to the nearest integer without affecting, asymptotically, the bounds. Furthermore, we can assume that each parameter is at least a sufficiently large constant (e.g, 8). This is due to the fact that if any of these parameters is bounded by a constant, then the corresponding competitive ratios are asymptotically the same as for a value of the said parameter equal to 1.

We assume that G is connected, since otherwise we can restrict the problem to the connected components of G. We denote by c_{min}, c_{max} the minimum and maximum costs of vertices in V, respectively. Since G is part of the input in both the online and offline versions of the problem, we assume that the parameters α and b are known to the algorithm. A greedy online algorithm for either problem connects the terminal to the current solution via a feasible, shortest-cost path. Edges already bought have their cost set to zero, to reflect that they can be used for free in the future.

Theorem 3 ([9])**.** *For the online GS problem, algorithm BC due to Berman and Coulston is $O(\log k)$-competitive. The bound is asymptotically optimal.*

Theorem 4 ([29])**.** *For online* GNWS*, algorithm KR due to Klein and Ravi is $O(\log k)$ competitive. The bound is asymptotically optimal.*

Theorem 5 ([21])**.** *For the offline GS problem, algorithm GW due to Goemans and Williamson is a 2-approximation.*

We also note that Alon *et al.* [2] have studied online algorithms for broad classes of (edge-weighted) connectivity and cut problems. Their approach applies to online problems that have integer programming formulations whose LP-relaxation models either a cut or a connectivity problem. In order to obtain efficient online randomized algorithms [2] apply a two-step approach: in the first step, a solution to the *fractional* problem (i.e., the one modeled by the LP relaxation) is obtained using the *multiplicative updates* technique. In the second step, this fractional solution is rounded, in an online fashion, to an integral one. This gives rise to a randomized online algorithm, whose competitive ratio is the product of the performance ratios of the two steps. [2] applied this technique to a wide variety of problems such as online facility location, online multicast, and online group Steiner.

Last, in order to show lower bounds on randomized algorithms, we resort to a standard tool, namely *Yao's principle* [40], according to which the ratio of the average cost of the algorithm over the average optimal cost is a lower bound on the competitive ratio of every randomized algorithm against an oblivious adversary (assuming cost minimization problems).

3 Parameterized analysis of NWST and GNWS

Recall that in the context of node-weighted problems, the relevant parameters are the number of terminals k and the parameter α which is equal to the ratio of the maximum node weight to the minimum node weight in the graph.

3.1 Approximation algorithms

We first address the inapproximability of NWST. To this end, we use a reduction from the *Bounded-size Set Cover* problem, or BSSC, for brevity. The input to this problem consists of a universe U of elements and a collection $\mathcal{C}$ of sets, with each such set containing at most B elements in U, for some parameter B. The objective, as in the general set cover problem, is to find a collection $C \subseteq \mathcal{C}$ of minimum cardinality such that for every $e \in U$ there exists $S \in C$ such that $e \in U$. As shown by Trevisan [36], BSSC is not approximable within a factor of $\ln B - O(\ln \ln B)$ unless P=NP.

Theorem 6. *Any polynomial-time algorithm for* NWST *in graphs of asymmetry α has approximation ratio $\Omega(\min\{\log k, \log \alpha\})$, unless $P = NP$.*

Proof. The proof is based on a reduction from BSSC. Consider an instance I of BSSC which consists of a universe of elements, denoted by U and a collection $\mathcal{C}$ of sets, each containing at most α elements in U. The reduction is as follows: G is defined as a graph with a vertex v_S for each set $S \in \mathcal{C}$, and a vertex v_e for every element $e \in U$. Each v_e vertex has weight equal to $1/\alpha$ and each vertex v_S has weight 1. All pairs of vertices of the form v_S, $v_{S'}$ are pairwise adjacent; furthermore, v_S and v_e are adjacent if and only if $e \in S$. All edge weights are zero. Last, we define the set of terminals K as the set of all v_e vertices in G (hence $k = |U|$). Note that this transformation gives rise to an instance I' of NWST, and that G has asymmetry α.

Denote by $OPT(I')$, $A'(I')$ the cost of the optimal algorithm and the cost of an approximation algorithm A' for NWST on the above instance I', respectively. Also denote by C'_{opt}, C' the set of vertices of the form v_S in (any fixed) optimal solution and the solution of A', on input I', respectively. It is easy to see that a Steiner tree for K in G corresponds to a set cover for the instance $(U, \mathcal{C})$, in that the set of vertices C' corresponds to a collection of sets (which we denote by C) which cover U. Define A as the set cover algorithm which, on instance I, selects all sets in C. Since all edges in G have zero weight,

$$A'(I') = k/\alpha + |C'| \quad \text{and} \quad A(I) = |C| = |C'|. \tag{1}$$

Likewise, we have

$$OPT(I') = k/\alpha + |C'_{opt}| \quad \text{and} \quad OPT(I) = |C_{opt}| = |C'_{opt}|. \tag{2}$$

where $OPT(I)$ is the optimal cost for the instance I of set cover, and C_{opt} is the collection of sets in I corresponding to C'_{opt}. Suppose that NWST is approximable within a factor ρ, then

$$A'(I') \leq \rho \cdot OPT(I') = \rho(k/\alpha + |C'_{opt}|) \qquad \text{(from (2))},$$

The solution obtain via the reduction for the instance I of BSSC has cost at most $A'(I)$, and from (2) we obtain that we can approximate BSSC to a factor ρ_{BSSC} such that

$$\rho_{BSSC} \leq \rho \cdot \frac{k/\alpha + |C'_{opt}|}{|C'_{opt}|}. \tag{3}$$

We now distinguish two cases, depending on whether $\alpha \leq k$ or not. Suppose first that $\alpha \leq k$. In this case, since C_{opt} is a feasible solution for I, we have that

$$\alpha |C'_{opt}| = \alpha |C_{opt}| \geq k. \tag{4}$$

Combining (3) and (4) we obtain that $\rho \geq \rho_{BSSC}/2 \geq \frac{\ln \alpha}{2} - O(\ln \ln \alpha)$, where the last inequality follows from the inapproximability of BSSC [36].

Suppose next that $\alpha > k$. In this case, we obtain again directly from (3) that $\rho \geq \rho_{BSSC}/2$, which in combination with the inapproximability of BSSC implies that $\rho \geq \frac{\ln k}{2} - O(\ln \ln k)$, which completes the proof. □ □

We now present an algorithm that has approximation ratio $O(\min(\log k, \log \alpha))$, thereby matching the lower bound of Theorem 6. We first note that when $\alpha \geq k$, the Klein-Ravi (KR) algorithm is $O(\log k)$-approximation, hence optimal. We may thus assume that $\alpha < k$. In this case, our algorithm is the a combination of the Klein-Ravi and Goemans-Williamson (GW) algorithms. For convenience, we borrow some of the notation in [29, 21]: in particular, we use the concept of a *requirement function* $r : V \times V \to \{0, 1\}$, defined as $r(u, v) = 1$ if and only if the pair of vertices u, v is not yet connected at the end of the current iteration of the algorithm (and thus initially $r(s_i, t_i) = 1$ for all pairs $(s_i, t_i) \in K$, and $r(u, v) = 0$ for all other pairs not in K). We remind the reader, in particular, that in each iteration of the KR algorithm, a subset of currently *active* trees is merged into a single tree (i.e., into a single connected component) by buying a vertex of highest cost-efficiency, and paths from the vertex to the active trees in question (the definition of cost-efficiency is as in [29] and is not critical in our analysis). Here, the term "active" reflects the fact that the tree contains vertices for which the requirement function is not satisfied by the current partial solution. More formally, a tree T is active if and only if there exist vertices $u, v \in V$, with $u \in T$, $v \notin T$ such that $r(u, v) = 1$. Prior to the execution of the algorithm $r(u, v) = 1$ if and only if $(u, v) \in K$.

Algorithm 1 Algorithm OFFLINEGNWS on input (G, r)

1: Execute the KR algorithm until at most k/α active trees remain.
2: Create a new graph G' by contracting each tree in the partial solution to a single "supernode". Each supernode is assigned vertex weight zero.
3: Define a new requirements function r' on G' and solve GS on instance (G'_p, r') using the GW algorithm.

We emphasize that for the GS instance (G'_p, r'), all nodes have zero weight, and that G'_p is the path completion of G'. We also remind the reader that GS is the (plain) Generalized Steiner problem, with no node weights.

The requirements function r' in step 3 of the algorithm is defined in a very natural way. In particular, at the end of step 1, the set of edges and vertices which have been bought induces a forest F, with each tree T in the forest corresponding to a supernode v_T in the vertex set of G'. We define r' for supernodes $u, v \in G'$ to be such that $r'(u, v) = 1$ if and only if there exist vertices $\tilde{u} \in T_1$ and $\tilde{v} \in T_2$, for trees T_1, T_2 in F such that $r(\tilde{u}, \tilde{v}) = 1$ at the end of step 1. Informally, r' is determined by all vertices whose connectivity requirement has not been satisfied by the end of step 1. Let OPT denote the optimal solution cost for the instance (G, r) of GNWS.

Theorem 7. *Algorithm* OFFLINEGNWS *has approximation ratio* $O(\min(\log k, \log \alpha))$.

Proof. We upper bound the cost of each one of the steps 1 and 3 of the algorithm with respect to the optimal cost OPT. Recall that from Observation 2, we can assume that all terminals have zero weight.

Consider first step 1, and more precisely the penultimate iteration of the KR algorithm in it. Since at least k/α active trees remain, the cost of the partial solution maintained up to that iteration, i.e., edges and vertices bought by KR is upper bounded by

$$2\ln\frac{k}{k/a}OPT = O(\ln\alpha)OPT.$$

The above follows by the analysis of the KR algorithm (see Section 4 in [29], in particular the two inequalities following (4) in [29]). More precisely, in any iteration of the KR algorithm, [29] shows that the cost of the algorithm up to that iteration is bounded by the quantity $2\ln\frac{k}{\phi}OPT$, where ϕ is the number of active trees that remain up to that point.

The KR algorithm has the property that in every iteration it connects $q \geq 2$ active trees in a new component at an average cost of at most OPT/q, which implies that the cost of the last iteration in step 1 is bounded by OPT. Overall, step 1 contributes cost $O(\ln\alpha)OPT$.

It remains to bound the cost due to step 3; here, we bound separately the vertex and edge costs. Denote by $F = \{T_1, \ldots, T_l\}$ the forest of trees returned by OFFLINEGNWS. First, note that the cost of edges in the optimal solution to the GS instance (G'_p, r') is bounded by OPT. Since GW is a 2-approximation algorithm for GS (Theorem 5), the cost of the edges in the forest is at most $2OPT$. Next, let k_i denote the number of terminal pairs in tree T_i (here, we stress that the term terminal refers to the instance (G'_p, r'), namely a supernode which corresponds to an active tree at the end of step 1 is considered a single terminal). From Observation 1, the number of Steiner nodes in T_i is bounded by $2k_i - 2 \leq 2k_i$. Therefore the total weight of Steiner nodes in F is at most

$$\sum_{i=1}^{l} 2k_i \cdot c_{max} \leq 2\frac{k}{\alpha}c_{max} = 2k \cdot c_{min} \leq 2OPT,$$

where the first inequality follows from the fact that at most k/α active trees remain at the beginning of step 2, the equality follows the definition of the asymmetry, and the second inequality follows from the fact that the weight of each terminal in K also appears in OPT. We conclude that the total node-weight of the forest F is at most $2 \cdot OPT$, and its total weight at most $3 \cdot OPT$, which in turn is a bound for step 3 of the algorithm.

Adding the contribution of both steps, we obtain that the algorithm has overall cost $O(\log\alpha)OPT$. □

□

3.2 Online Algorithms

Lemma 8. *The competitive ratio of deterministic online* GNWS *is* $\Omega(\max\{\min\{\alpha, k\}, \log k\})$.

Proof. We consider the online NWST problem (the lower bound carries over to online GNWS). If $\alpha < \log k$, a lower bound of $\Omega(\log k)$ follows by the construction of Imase and Waxman [27] for the online edge-weighted Steiner tree problem. Otherwise the adversary presents a graph G which is defined as follows: The vertex set of G consists of (disjoint) sets V_1 and V_2 (we require, in particular, that $|V_2| \geq k(k+1)/2$). The set V_1 is defined in terms of V_2 as follows: Each subset of k vertices in V_2 defines a vertex $u \in V_1$ which is adjacent to these k vertices in V_2. No more vertices or edges exist in G. The weight of each vertex in V_1 is α, and each vertex in V_2 has unit weight, whereas the weight of all edges is zero.

The adversary will present a nemesis sequence consisting of vertices in V_2, determined by a game against the algorithm. In the first round of the game, the adversary picks any two vertices in V_2 as the first

two terminals in the sequence; the algorithm buys a vertex in V_1 so as to guarantee connectivity. In each subsequent round, the adversary presents a new terminal, namely a vertex in V_2, which is not adjacent to vertices bought in earlier rounds. The algorithm then must buy a new vertex in V_1 and the adversary repeats the above strategy. Clearly, the game lasts $k-1$ rounds. At the end of the game, k terminals have been presented all of which are adjacent to some vertex in V_1, hence $OPT \leq \alpha + k$. On the other hand, the algorithm has bought $k-1$ vertices in V_1, hence its cost is at least $\alpha(k-1)$ which in turn yields a bound on the competitive ratio of $\Omega(\min\{\alpha, k\})$. □ □

We observe that the lower bound of Lemma 8 can be extended to randomized algorithms, using the same graph as part of the adversarial input; we require, however, a larger set of vertices V_2, namely we require that $|V_2| \geq 2k^2$. We define a probability distribution D on the sequence of terminals presented to any fixed deterministic online algorithm; from Yao's principle [40], the ratio of the average cost of the algorithm over the average optimal cost is a lower bound on the competitive ratio of every randomized algorithm against an oblivious adversary. In particular D chooses uniformly at random a set $K \subseteq V_2$, with $|K| = k$; then the input to the deterministic algorithm is the set K, in any order. It follows that every time the deterministic algorithm considers a terminal $t \in K$ (except for the very first terminal), the probability it has already bought a vertex in V_1 which is adjacent to t is at most $\frac{k}{2k^2-k^2} = \frac{1}{k}$, and hence the probability that for each of the $k-1$ terminals the algorithm must buy a new vertex in V_1 is at least $(1-1/k)^{k-1} \geq \mathrm{e}^{-1}$, thus the average cost of the algorithm is $\Omega(\alpha k)$, while the average optimal cost is still $\alpha + k$.

Next, we show a matching upper bound. More precisely, we propose an algorithm, denoted by A, which works in two phases. Let $\overline{k}$ denote the number of pairs presented by the adversary thus far, and k the total number of pairs that will be presented eventually (A does not know k). The first phase is a greedy phase and lasts for as long as $\alpha > \overline{k}$: namely, the algorithm connects the two terminals in the pair by means of a minimum-cost path. The second phase starts when $\overline{k}$ exceeds α, at which point the algorithm switches to the Berman and Coulston (BC) algorithm for (edge-weighted) GS [9]. More precisely, the algorithm treats all edges bought during the greedy phase as having weight zero (meaning that it already paid for them). In addition, it ignores vertex weights (or alternatively, treats vertices as if they have zero weight).

Lemma 9. *Algorithm A described above has competitive ratio $O(\max\{\min\{\alpha, k\}, \log k\})$.*

Proof. The total cost due to the first phase is clearly at most $\min\{\alpha, k\}OPT$. For the second phase, since the BC algorithm is $O(\log k)$-competitive for GS (Theorem 3) it follows that the cost of the edges it buys is $O(\log k)OPT$. Moreover, since BC returns a forest of at most k trees, using Observation 1, it follows that at most $2k$ vertices of the forest are Steiner vertices. Therefore the total node-cost of the forest is bounded by the quantity $C = 4k \cdot c_{max} \leq 4\alpha k \cdot c_{min}$. However, the latter contribution to the cost is in effect only when $k \geq \alpha$, otherwise we do not run BC. Given that in such case $OPT \geq kc_{min} \geq \alpha c_{min}$, we have that $C \leq 4\alpha \cdot OPT = 4\min\{\alpha, k\} \cdot OPT$ Therefore, the total cost of A is

$$O(\log k \cdot OPT + \min\{\alpha, k\} \cdot OPT) = O(\max\{\min\{\alpha, k\}, \log k\} \cdot OPT).$$

□ □

Combining Lemmas 8 and 9 we obtain the following theorem.

Theorem 10. *The competitive ratio of deterministic online* GNWS *is* $\Theta(\max\{\min\{\alpha, k\}, \log k\})$.

4 Parameterized analysis of online PST and GPS

In this section we study the online priority Steiner tree problems. Recall that for this problem, the relevant parameters are the number of terminals k as well as the number of priority levels b.

4.1 Upper bound

We analyze a relatively simple algorithm, called ONLINEGPS which applies not only to PST but also to its generalization, namely GPS (as defined in Section 1). This algorithm uses, as subroutine, the online algorithm for the (plain) Generalized Steiner problem (GS) in undirected graphs due to Berman and Coulston [9], which we denote by BC. In Section 4.2 we prove that this algorithm is asymptotically optimal.

Denote by $\overline{k}$ the number of terminals revealed so far. ONLINEGPS works in two phases. In the first phase, as long as $\overline{k} \leq b$ ONLINEGPS connects each terminal pair by a minimum-cost path. The second phase begins when $\overline{k} > b$: during this phase, ONLINEGPS serves each request by running the BC algorithm. More precisely, let G_j denote the subgraph of G induced by all edges e of priority at least j, for all integers $1 \leq j \leq b$. During the second phase, ONLINEGPS partitions the sequence of requests into subsequences $R_1, \ldots, R_b$ such that R_j consists of all requests in R of priority equal to j. Each sequence in R_j is served by running BC on graph G_j.

Theorem 11. *The competitive ratio of* ONLINEGPS *is* $O\left(b \log \frac{k}{b}\right)$*, if* $k > b$ *and* $O(k)$*, otherwise.*

Proof. If $k \leq b$, then ONLINEGPS executes only the first phase, hence its cost is at most $k \cdot OPT$. Next, suppose that $k > b$. Let OPT_j denote the cost of the optimal solution to the Generalized Steiner Problem on graph G_j, and for the set of terminals R_j, (here, G_j and R_j are as defined above). Let $c(R_j)$ denote the cost of ONLINEGPS for the sequence of requests in R_j; we can assume, without loss of generality, that $|R_j| > 0$, since otherwise R_j does not contribute any cost. Since the BC algorithm is $O(\log |R_j|)$-competitive, it follows that $c(R_j) = O(\log |R_j|)OPT_j = O(\log |R_j|)OPT$ (since $OPT_j \leq OPT$). Therefore, the cost incurred by ONLINEGPS in the second phase is bounded by

$$c(R) = O\left(\sum_{j=1}^{b} \log |R_j| OPT\right) = O\left(\log \prod_{j=1}^{b} |R_j|\right) OPT,$$

which is maximized when $|R_j| = \frac{k}{b}$ (for $k > b$.) Hence $c(R) = O\left(b \log \frac{k}{b} \cdot OPT\right)$. Since the contribution of the first phase is at most $b \cdot OPT$, we obtain that the total cost of the algorithm is $O\left(b \log \frac{k}{b} \cdot OPT\right)$, which concludes the proof. □ □

4.2 Lower bound

In this section, we prove that the algorithm of Section 4.1 is asymptotically optimal. More precisely, we prove the following theorem:

Theorem 12. *The competitive ratio of every deterministic online algorithm for online* PST *and* GPS *is* $\Omega\left(b \log \frac{k}{b}\right)$*, if* $k > b$ *and* $\Omega(k)$*, otherwise.*

We prove the bound for online PST, then the result carries over to online GPS. We distinguish two cases, based on whether $k > b$. Depending on the case, the adversary will present a suitable input graph G and a request sequence σ of k terminals (the latter is derived by means of a game between the algorithm and the adversary). The case $k > b$ is substantially harder than the case $k \leq b$ (which is also reflected, in some sense, in the upper bound of Theorem 11).

4.2.1 Case $k > b$

We present a game between a deterministic online algorithm and the adversary which results in a competitive ratio at least $\Omega\left(b \log \frac{k}{b}\right)$. The adversarial graph G will be constructed in b consecutive *phases*. Each phase, in turn, will be defined in terms of additions of appropriate vertices and edges, which occurs in consecutive *rounds*.

High-level description of the proof. We begin with some intuition and an informal outline of the proof. Recall that for the standard online Steiner tree problem in an undirected graph, there is a $\Omega(\log k)$ lower bound on the competitive ratio of any online algorithm [27]. This bound is obtained by a game between an adversary and the online algorithm on an inductively defined graph. Figure 1 illustrates the main idea behind this construction. Consider first graph G_1, and suppose that terminals r, u_1 are first requested. The online algorithm (say A), will connect u_1 to r via a path that passes either through either $u_{2,1}$ or $u_{2,2}$. Then the adversary will request next vertex $u_{2,2}$, or $u_{2,1}$, respectively. This ensures that A pays a cost of 3/2, whereas OPT has a cost equal to 1. One can "boost" this bound by forcing the algorithm to make further "bad" choices; this can be done, informally, by replacing, in G_1, the edges of cost $1/2$ with copies of G_1, where now edges have weight $1/4$. This gives rise to graph G_2. The adversary now will request vertices r, u_1 first, one vertex of type $u_{2,*}$, and two vertices of type $u_{3,*}$, in such a way that all requested vertices lie on a path from r to u_1. This means that OPT still pays 1, whereas A now pays cost $1 + 1/2 + 2 \cdot 1/4 = 1 + 2 \cdot 1/2$. We can then continue by defining recursively $G_3, \ldots, G_{\log k}$; then a similar game on the final graph $G_{\log k}$ yields a cost for A that is $\Omega(\log k)$, whereas OPT still pays cost 1.

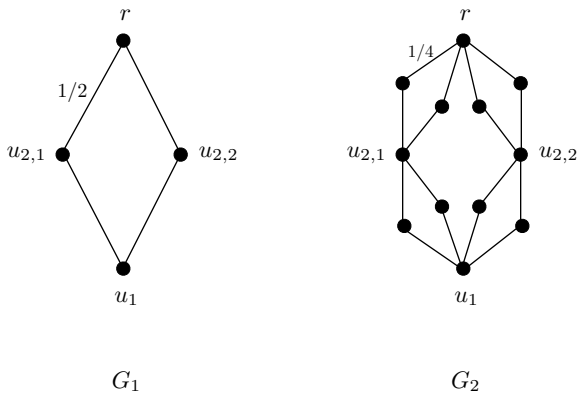

Figure 1: An illustration of the adversarial graph of Imase and Waxman [27]. In G_1 all edges have weights equal to $1/2$, whereas in G_2 all edges have weights equal to $1/4$.

Turning to online PST, we use some elements of the above construction so as to define a game that consists of b phases. In phase i, only terminals of priority i will be requested. By assigning appropriate values to the edge priorities of the graph, we enforce a situation in which the parts of the graph that were bought in phases $1, \ldots i-1$ will not be useful, for the online algorithm, in phase i. Thus, each phase will incur a lower bound of $\Omega(\log \frac{k}{b})$, we achieve overall a lower bound of $\Omega(b \log \frac{k}{b})$, as wanted.

There are certain technical challenges that one must overcome in this approach. Namely, one must

guarantee that for the online algorithm, phases are sufficiently "disjoint", as explained above, whereas for OPT, all requested terminals are still on a simple path (as in the standard online Steiner tree construction). To do so, instead of using the "diamond"-shaped construction illustrated in Figure 1, we use a construction in which there are many vertical-shaped edges (which we call "columns"), as depicted in Figure 2. The terminals will connect to the columns by means of edges to certain subset of vertices (which we will call s-sets). Finally, in order to simplify the proof (namely, so as to argue more easily about the disjointness of phases), we use "tree-like" constructions of s-sets. In this way, the adversary will be able to request successive terminals whose s-sets are in a parent-child relation, thus giving the adversary a lot of power in moving the game towards columns that have not yet been used by the algorithm.

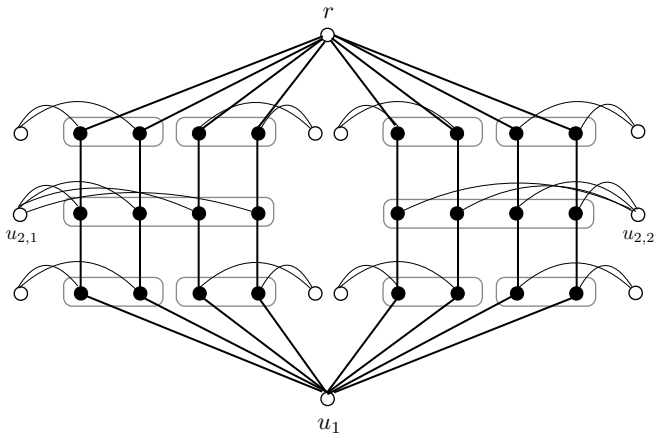

Figure 2: A different construction of the adversarial graph G_2 of Figure 1. Here, only non-solid vertices can be requested as terminals. Solid edges have weight $1/4$, all other edges have 0 weight. Rectangles depict the s-sets to which potentially requested terminals connect. Note that there are 8 columns in total.

We now proceed with the technical details of the proof. We first show how to construct the adversarial graph, using the intuition outlined above. Then we define the game between the algorithm and the adversary, and finish by bounding the costs of the online algorithm and the optimal offline algorithm.

Construction of the adversarial graph. We begin with defining some auxiliary constructions. Let $T_1 = \{v_1, \ldots, v_l\}$ and $T_2 = \{v'_1, \ldots, v'_l\}$ be two disjoint sets of l vertices each (we should think of T_1 as lying higher than T_2, as illustrated in Figure 3). For every $i \leq l$, we define the *distance* between v_i and v'_i to be equal to 1. We call index i the i-th *column*, and require that l is a power of 2, and that it is large compared to k (e.g., at least 2^k). For a column $i \leq l$, we say that we *insert a vertex w at depth $d < 1$ in column i* if we introduce a new vertex w at distance d from vertex v_i and distance $1 - d$ from vertex v'_i. We also say that a collection of vertices which are at the same depth form a *layer* of vertices. Intuitively, the depth of a vertex w describes how far away w is from the top layer T_1, or equivalently, how close it is to the bottom layer T_2. The *distance* between two inserted vertices is the absolute value of the difference of their depths. We denote by E the set of all l columns (see Figure 3 for an illustration).

On the set E of columns, we define a construction $B(E, \beta)$ called *block* which inserts vertices and edges

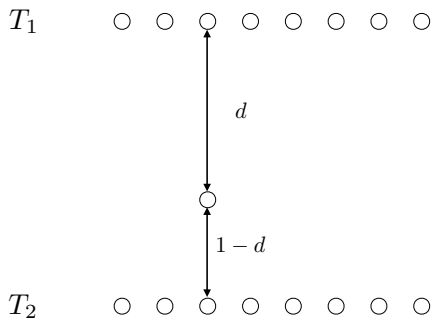

Figure 3: An illustration of the insertion of a vertex at depth d. Here, T_1 and T_2 consist of eight vertices each, and the vertex is inserted at the third column.

in $\log(k/b)-1$ consecutive *rounds*[1]; here β is an integral parameter, with $\beta \in [1,b]$. The rounds are defined inductively as follows: In round 1, l vertices $w^{1,1}\ldots,w^{1,l}$ are inserted, one in each column of E, at depth equal to 1/2. We partition the l vertices in two groups, $S^{1,1}$ and $S^{1,2}$, consisting of vertices $\{w^{1,1},\ldots w^{1,l/2}\}$, and $\{w^{1,l/2+1},\ldots w^{1,l}\}$, respectively. We refer to such groups of w vertices as *s-sets*. For each one of the two s-sets we add a corresponding vertex, namely we add vertices $u^{1,1}$ and $u^{1,2}$. For every vertex $w \in S^{1,1}$ (resp. $w \in S^{1,2}$) we add the edge $(w,u^{1,1})$ (resp. $(w,u^{1,2})$, which has zero cost and priority equal to β.

We now describe the j-th round in the construction of $B(E,\beta)$, assuming that rounds $1,\ldots,j-1$ have been defined. Let d denote the size of the s-set of smallest cardinality that was inserted in round $j-1$. For every $i \in [1,2^j-1]$, we insert l vertices at depth $i/2^j$, one for each column in E, *unless* some other w vertex has been inserted at this same depth in a previous round, in which case we do not perform any additional insertion. This has the effect that 2^{j-1} additional *layers* of vertices are inserted in round j. We call the i-th deepest layer of vertices that is inserted in round j the *i-th layer of round j*, and denote by $w^{j,i,1}\ldots w^{j,i,l}$ its vertices. We partition the l w-vertices of the i-th layer of round j in $(2^i l)/d$ s-sets (from left to right), all of the same size $d/2^i$: In particular, the s-set $S^{j,i,q}$ (in words, the q-th s-set, from left to right, of the i-th layer of round j) consists of $d/2^i$ vertices $\{w^{j,i,\frac{(q-1)d}{2^i}+1},\ldots w^{j,i,\frac{qd}{2^i}}\}$. For each such s-set of the form $S^{j,i,q}$, we add a vertex $u^{j,i,q}$, and for every vertex $w \in S^{j,i,q}$ we add the edge $(w,u^{j,i,q})$ of zero cost and priority β. The *depth* of a u-vertex is defined as the depth of any of the w-vertices in its corresponding s-set, and its *height* is defined to be equal to one minus its depth. This completes the definition of round j, and thus the definition of the block construction as well. Figure 4 illustrates the block construction.

Finally, we turn $B(E,\beta)$ into a well-defined graph by inserting edges between any two pairs of consecutive vertices in all columns. More precisely, for each pair of consecutive vertices w,w' of every column i, we add an edge (w,w') (which we call *vertical* edge) of cost equal to the distance between w and w' in the column and priority equal to b. We use the same notation $B(E,\beta)$ to refer to this graph, when clear from context.

We say that an s-set S *crosses* a certain set of columns C if and only if the set of columns in which the vertices of S lie intersects C. Two s-sets cross each other if and only if the intersection of the sets of columns crossed by each one is non-empty. Note that there is a 1-1 correspondence between s-sets and u-vertices, which means that several properties and definitions pertaining to s-sets carry over to the corresponding u vertices (e.g., we say that two u vertices cross if their s-sets cross).

[1]Throughout the paper we omit floors and ceilings since they do not affect the asymptotic behavior of the algorithms.

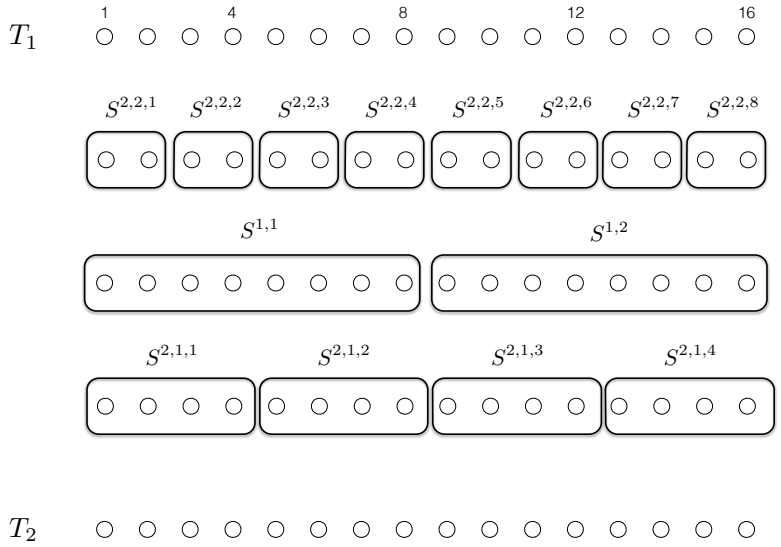

Figure 4: An illustration of the block construction $B(E, \beta)$, assuming a set E of 16 columns, and two rounds. For simplicity, the figure illustrates only the partition of w-vertices into s-sets, and omits the u vertices. $S^{1,1}$ and $S^{1,2}$ are the s-sets of round 1. The sets $S^{2,1,1} \dots S^{2,1,4}$ are the s-sets at the first layer of round 2, and $S^{2,2,1} \dots S^{2,2,8}$ are the s-sets at the second layer of round 2.

The adversarial graph G is defined by performing a series of block constructions, in b consecutive *phases*. Let T_1, T_2 and E be as defined earlier. Phase 1 consists only of the insertion of block $B(E, 1)$. Suppose that phase $p-1 < b$ has been defined, we next define phase p. Let d denote the smallest cardinality among s-sets inserted in phase $p-1$. In other words, d is the cardinality of s-sets in the last (i.e., highest) layer inserted in the last round of phase $p-1$. This induces a partition of the columns in E in d sets, where the i-th set in this partition, say set E_i, consists of columns $(i-1)d+1, \dots id$. Phase p consists of inserting l/d blocks, namely blocks $B(E_i, p)$, for all $i \in [1, l/d]$ (see Figure 5 for an illustration).

One subtle point has to do with w-vertices of the same column, which are at the same depth but belong to different phases. Say w_1 and w_2 are such vertices, added in phases j_1 and j_2, respectively. For every such pair, there is an edge (w_1, w_2) in G of zero weight and priority equal to $\min\{j_1, j_2\}$. In words, this means that using this edge is free, however it can be of use only if we route traffic at a level at most the minimum of the two corresponding phases.

This essentially completes the construction of G. We also add a root vertex r and edges of the form (r, v), for all $v \in T_1$, as well as edges of the form (t, v'), for all $v' \in T_2$ (where t is a new vertex). These edges are assigned zero cost and priority b.

Before proceeding to the description of the game between the algorithm and the adversary, we provide some alternative classification of vertices added in round j of phase p, which will make the description of the game easier. Consider the collection of deepest s-sets inserted during round j of phase p. This collection induces a partition of the set of all columns E into disjoint sets $E_p^{j,1}, E_p^{j,2}, \dots$, from left to right: every s-set inserted during round j of phase p will cross only one of the sets of columns $E_p^{j,1}, E_p^{j,2}, \dots$. We then say that an s-set (or a corresponding u-vertex) inserted during round j of phase p that crosses edges of the set $E_p^{j,i}$ belongs in *component* $C_p^{j,i}$. This provides a convenient way to identify s-sets and u-vertices in this

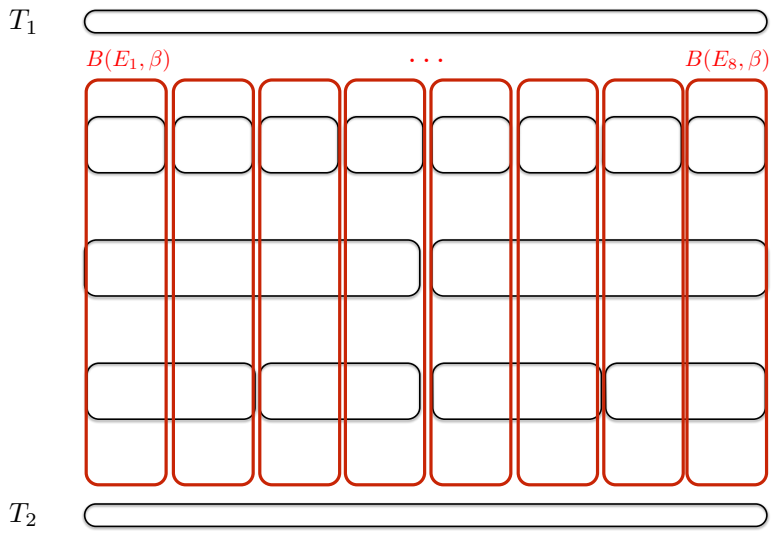

Figure 5: An illustration of a phase in the construction of G. Here, $B(E, 1)$ consists of 2 rounds, similar to Figure 3, i.e., three layers are inserted in total. There are eight blocks $B(E_1, 2), \ldots B(E_8, 2)$ inserted in phase 2, each crossing the columns of the highest S-set of $B(E, 1)$; each of the blocks $B(E, 2)$ is represented by a rectangle.

construction. Namely, we use the notation $C_p^{j,i}(level, pos)$ to describe the s-set which belongs in component $C_p^{j,i}$, is in the $level$-th deepest layer among layers of the component $C_p^{j,i}$ and is also the pos-th s-set (from left to right) in this component. We call the s-set $C_p^{j,i}(1, 1)$ the *s-root* of component $C_p^{j,i}$ (by definition, the unique deepest s-set of the component).

As an example of the above definitions, in Figure 4, the component $C_1^{2,3}$ consists of the sets $S^{2,1,3}$, $S^{2,2,5}$ and $S^{2,2,6}$. The root of this component is $S^{2,1,3}$, and $S^{2,2,5}$ and $S^{2,2,6}$ are its left and right children, respectively.

The purpose of this notation is to define a *parent/child* relation among s-sets in the same component, which is at the heart of the game between the algorithm and the adversary, as it will become clear later. From construction, every s-set in layer L of $C_p^{j,i}$, say the set $C_p^{j,i}(L, pos)$ crosses exactly two other s-sets in the same component which are inserted at layer $L + 1$, namely sets $C_p^{j,i}(L, 2pos)$ and $C_p^{j,i}(L, 2pos + 1)$, which we call the *left* and *right* child of $C_p^{j,i}(L, pos)$, respectively. The only exception is the highest layer of the component, which does not have any children.

We emphasize that definitions that relate s-sets apply also to u-vertices, due to their 1-1 correspondence. For instance, we say that a u-vertex is the *u-root*, or simply *root* of a component if its corresponding s-set is the s-root of the component, or we say that two u-vertices are in a child/parent relation if the corresponding s-sets are in such relation (in the same component).

The game between the algorithm and the adversary. We show how to construct an adversarial sequence σ of at most k requests (the actual number of requests will be $k-b+1$, but this does not affect the asymptotic analysis). The request sequence σ is constructed in b *phases*, and each phase will request a u-vertex which was added in the corresponding phase in the construction of G (with the exception of the very first request,

which is for vertex t). Every requested vertex in phase $p \leq b$ will be assigned a priority equal to the phase index, namely p. For every such vertex u, the online algorithm must establish a *connection path* from r to u denoted by $path(u)$ (of priority at least $p(u)$), possibly buying new edges. We can assume, without loss of generality, that the algorithm provides a single connection path for each requested terminal (if more than one such paths exist, the adversary can declare one arbitrarily as $path(u)$, and ignore all others; this only strengthens the bound. Similar arguments have been used in [18] and [4]).

All phases consist of $\log(k/b) - 1$ rounds, with the exception of phase 1, which consists of $\log(k/b)$ rounds (with round 0 an "initialization" round). To give some intuition about the game, we first describe the actions of the adversary on the first three rounds of phase 1, then provide a more formal iterative description of the game. Every request in phase 1 is assigned priority equal to 1. In round 0, vertex t is requested, and the connection path $path(t)$ chosen by the algorithm will cross exactly one of the sets $S^{1,1}$ and $S^{1,2}$ (which are also the s-roots of the trivial components $C_1^{1,1}$ and $C_1^{1,2}$). Round 1 consists of a single request to a u-vertex determined as follows: if $path(t)$ crosses the s-root of $C_1^{1,1}$ (resp. $C_1^{1,2}$), then the adversary requests the root of $C_1^{1,2}$ (resp. $C_1^{1,1}$). At this point, no matter what the connection paths chosen so far, there is a component $C_1^{2,x}$, for some $x \in [1, 4]$, such that no columns of $C_1^{2,x}$ are crossed by existing connection paths[2]. Round 2 then begins, during which the adversary will request one u vertex per layer of $C_1^{2,x}$. In the first request of the round, the adversary requests the root of $C_1^{2,x}$. The connection path for the latter will cross exactly one of its children: if it crosses the left child, then the next request will be to the right child and vice versa. In total 2 requests to u-vertices are made in this round.

For a formal description of the algorithm/adversary game, we first need a straightforward proposition concerning u-vertices, which follows from the construction of G. The proposition will guarantee that the actions of the adversary are well-defined.

Proposition 13. *Let u be a u-vertex requested by the adversary which belongs to component $C_p^{j,x}$, for some round j and index x. Suppose that u satisfies the precondition that prior to its request in σ, no connection path for previously requested vertices crosses u. Then the following hold:*

(a) If u is the highest u-vertex in $C_p^{j,x}$, (i.e., u has no children in $C_p^{j,x}$) and $j < \log(k/b) - 1$, then after $path(u)$ is established, there is a component $C_p^{j+1,y}$ whose root does not cross any connection paths for all previous requests.

(b) If u is not the highest u-vertex in $C_p^{j,x}$ (i.e., u has children in $C_p^{j,x}$) then after $path(u)$ is established, there is a child of u in $C_p^{j,x}$ that is not crossed by any connection paths for all previous requests.

(c) If u is the highest u-vertex in $C_p^{j,x}$, and $j = \log(k/b) - 1$, then after $path(u)$ is established, there is a component $C_{p+1}^{1,y}$ whose root does not cross any connection paths of previous requests.

Proof. **(a)** Let S denote the s-set to which u corresponds. Since S is the highest s-set in round j, from construction of G, there will be two s-sets which are the roots of components $C_p^{j+1,y}$, $C_p^{j+1,y'}$, for two indices y and y' with the property that they each cross half the columns crossed by S. Hence, the connection path for u will cross only one of these two sets (without loss of generality, $C_p^{j+1,y'}$). Then $C_p^{j+1,y}$ is the component with the desired property.

[2]We say that a connection path for a terminal crosses an s-set (or a corresponding u-vertex) if it contains (vertical) edges in a column that is crossed by the s-set).

(b) Let S denote the s-set to which u corresponds. Since S is not the highest S-set in round j, it has two children in component $C_p^{j,x}$, with the property that they each cross half the columns crossed by S. The argument proceeds as in (a).

(c) Let S denote the s-set to which u corresponds. Since S is the highest s-set in the last round of phase p, phase $p+1$ will involve (in its first round), the introduction of two components, say $C_{p+1}^{1,y}$, $C_{p+1}^{1,y'}$ such that their corresponding roots cross half of the columns crossed by S. Once again, the argument proceeds as in (a). □ □

The game proceeds as follows: First, suppose that rounds $1 \dots j$ of phase p have been defined (with $j < \log(k/b) - 1$), we now describe the requests of round $j+1$ of phase p. Let s denote the s-set of the highest u-vertex requested in round j. Then, from Proposition 13(a), there is a component, namely $C_p^{j+1,y}$ for some index y such that no connection path established so far crosses the s-root of $C_p^{j+1,y}$. Round $j+1$ begins with the adversary requesting the u-root of this component. The connection path for this request will cross only one of its children in the component, and the child that is not crossed will be the 2nd terminal to be requested in this round. Round j of this phase proceeds with requesting u-vertices in an upwards fashion, one u-vertex per layer of $C_p^{j+1,y}$: when a vertex is requested, the next request is the unique child of the vertex that is not crossed by any connection paths up to that point (which can be done as suggested by Proposition 13(b)). The round continues until the highest vertex in component $C_p^{j+1,y}$ is requested, i.e., continues with 2^j requests in total.

It remains to argue how to generate the first request for any phase $p > 1$. Note that the last request of phase $p-1$ is to a u-vertex which is highest in some component $C_{p-1}^{\log(k/b)-1,x}$ for some index value x. Proposition 13(c) guarantees that there is a component $C_p^{1,y}$, for some index y, whose root does not cross any connection paths of previous requests. Then the first request in phase p (indeed the only request of the first round in phase p) is to the u-root of $C_p^{1,y}$.

Analysis. We first observe that the adversarial sequence of requests σ is such that for any request u, there exists a column in u that crosses all previously requested u-vertices in σ (this is easy to show by means of a straightforward induction). This implies that all requested u-vertices in σ share a common column, say i. Then, an offline solution suffices to buy: i) all vertical edges of column i (for a total cost equal to 1); ii) edges from a w vertex in column i to a requested u-vertex (zero cost); iii) edges of the form (r, v_i), and (t, v_i') (at zero cost). Hence the optimal offline cost is at most 1.

On the other hand, consider the requests which belong in round j of phase p. These requests are part of a component $C_p^{j,x}$ for some index x. Due to the actions of the adversary, as well as Proposition 13(b), every time a request is issued for a u-vertex in $C_p^{j,x}$ other than the root of this component the algorithm must pay cost at least $1/2^j$. Since there are 2^{j-1} requests in σ that belong to $C_p^{j,x}$, the algorithm pays a cost which is at least $1/2 - 1/2^j$ during the round. Since there are b phases and $\log(k/b) - 1$ rounds in each phase, the overall cost is $\Omega(b \log \frac{k}{b})$.

4.2.2 Case 2: $k \leq b$.

In this case, the adversarial graph is based on a tree-like construction which is defined as follows. In addition to the root r, there are k levels of vertices: level 1 consists only of vertex $u_{1,1}$, whereas level i ($1 < i \leq k$) consists of vertices $u_{i,1} \dots u_{i,2^{i-1}}$. Every vertex $u_{i,j}$, with $i < k$ is connected to vertices $u_{i+1,2j}$, $u_{i+1,2j+1}$, by means of edges with priority i and unit cost. Edges incident with the root vertex r have priority b (and zero cost). Every vertex that is requested will be assigned a priority equal to its level, i.e., if vertex $u_{i,j}$

is requested, it will be assigned priority equal to i. The game proceeds in rounds: In the first round, the

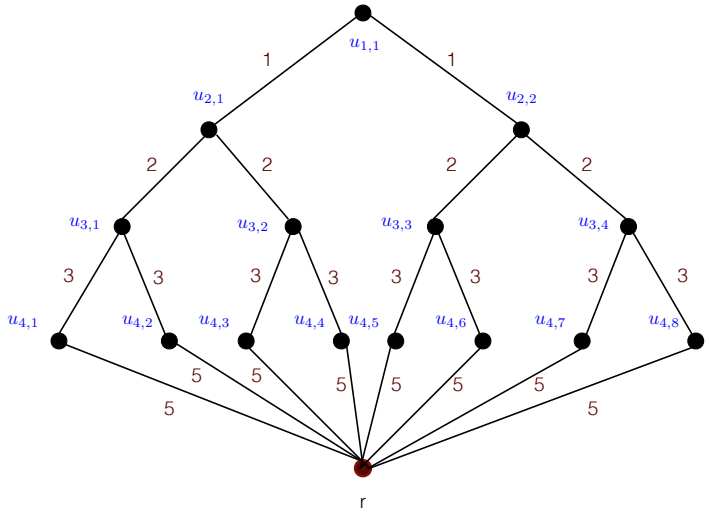

Figure 6: An illustration of the adversarial graph for $k = 4$ and $b = 5$. The edge weights denote their priority levels.

adversary requests $u_{1,1}$. The algorithm will establish a connection path from r to $u_{1,1}$, say $path(u_{1,1})$ of cost k; based on the choice of this path, in round 2 the adversary will request the unique vertex $u_{2,j}$ such that $path(u_{1,1})$ does not visit $u_{2,j}$. The game proceeds in a similar fashion: let u_i denote the last vertex requested, during round i. In round $i+1$ the adversary requests the unique vertex $u_{i+1,j}$ such that $path(u_{i,1})$ does not visit $u_{i+1,j}$. Note that from the choice of priorities of edges and requested vertices, the connection path for $u_{i+1,j}$ must be a path from the root r which does not use edges of connection paths established in earlier rounds. Hence the cost of this connection path is then $k - i$, which implies that the overall cost of the algorithm is $\sum_{i=1}^{k}(k-i) = \Theta(k^2)$. Finally, note that all requests can be served, offline, by a single path from r to $u_{1,1}$ which visits all requests (one per level), and has total cost k, which establishes that the competitive ratio of any algorithm is, in this case, $\Omega(k)$.

4.3 The online directed PST problem

In this section we study the online PST problem in the setting of directed graphs. Recall that the relevant parameters here are the number of terminals k, the number of priority levels b, as well as the edge-asymmetry of the graph γ, defined as the maximum ratio of weights of antiparallel edges in the graph.

To the best of our knowledge, priority Steiner Tree problems have been studied only in the context of undirected graphs. However, as noted in Section 1.3, these problems are useful in modeling multicasting with QoS guarantees; hence, a directed graph may provide a more realistic representation of the underlying network. Suppose then that G is directed, and that we seek a minimum-cost *arborescence*, namely a directed subgraph of G such that there is a unique directed path from the root to every terminal in K, with the property that this path satisfies the QoS requirements of the said terminal. What can we say about the competitiveness of this problem?

The first observation to make is that the competitiveness of PST changes dramatically when G is directed. Specifically, since the online Steiner tree problem is $\Omega(k)$-competitive for directed graphs [38], the same lower bound carries over to online PST. More importantly, the same bound applies even in graphs in

which antiparallel edges have the same cost (i.e., $\gamma = 1$), as long as we allow the priorities of antiparallel edges to differ by as little as one.

Proposition 14. *In directed graphs with $\gamma = 1$ and at least two priority levels, any algorithm for* PST *is $\Omega(k)$-competitive.*

Proof. The adversarial graph is based on a directed variant of the graph of Figure 6. Here, there are directed edges from any vertex at level i, say $u_{i,j}$ to two of its children at level $i+1$, namely $u_{i+1,2j}$, $u_{i+1,2j+1}$. The directed edge $(u_{i,j}, u_{i+1,2j})$ has priority 1 and cost 1, whereas the antiparallel directed edge $(u_{i+1,2j}, u_{i,j})$ has priority 2, and cost again 1. Similar weight assignments hold for $(u_{i,j}, u_{i+1,2j+1})$ and $(u_{i+1,2j}, u_{i,j})$. All edges from the root r to the u-vertices have priority 2. Last, every request at level i has priority equal to 2. We can then repeat the argument of the proof of Theorem 12, Case $k \leq b$, so as to show a lower bound of $\Omega(k)$ on the competitive ratio. □ □

In light of Proposition 14 we thus focus on graphs of bounded edge-cost asymmetry, denoted by γ, as defined in Section 1.2, assuming that antiparallel links have the same priority. In other words, the "directedness" of a graph manifests itself in terms of edge costs, and not in terms of edge priorities. We first analyze the competitiveness of the greedy algorithm in this model. Here, the greedy algorithm connects a request to terminal t to the current arborescence (i.e., the current tree rooted at r), by means of a (directed) least-cost path of sufficient priority, namely $p(t)$ (finding such a path is easy). For the purpose of the analysis, we determine a classification of terminals according to their priority (similar to the proof of Theorem 11), in combination with the analysis of the greedy algorithm for the edge-asymmetric Steiner tree problem of [5].

Theorem 15. *The competitive ratio of the greedy algorithm for* PST *is*

$$O\left(\min\left\{\max\left\{\gamma b\frac{\log(k/b)}{\log\gamma}, \gamma b\frac{\log(k/b)}{\log\log(k/b)}\right\}, k\right\}\right).$$

Proof. Using the terminology of the proof of Theorem 11, let G_j denote the subgraph of G induced by all edges e of priority at least j, for all $j \leq b$. For any sequence of requests $R = r_1, \ldots r_k$, partition R into subsequences $R_1, \ldots, R_b$ such that R_j consists of all requests in R of priority equal to j. Let (G_j, R_j) denote an instance of the (plain) online asymmetric Steiner tree problem on graph G_j and request sequence R_j: here we ignore the priority of edges and terminals, and our objective is to minimize the cost of the Steiner forest for the sequence R_j without concerns about priorities. Let OPT_j denote the cost of the Steiner forest for the above instance.

From the bound on the competitive ratio of the greedy algorithm for asymmetric Steiner tree due to [5], we deduce that

$$c(R_j) = O\left(\min\left\{\max\left\{\gamma\frac{\log|R_j|}{\log\gamma}, \gamma\frac{\log|R_j|}{\log\log|R_j|}\right\}, |R_j|\right\}\right) OPT \tag{5}$$

where $c(R_j)$ denotes the cost paid by the algorithm to serve requests in R_j.

Let $f_1(x) = \log(x)$ and $f_2(x) = \log(x)/\log\log(x)$. We observe that the functions $f_1(x)$ and $f_2(x)$ are concave. It then follows that

$$\sum_{j=1}^{b}\frac{\log|R_j|}{\log\log|R_j|} \leq b\cdot\frac{\log(k/b)}{\log\log(k/b)} \quad \text{and} \quad \sum_{j=1}^{b}\log|R_j| \leq b\cdot\log(k/b) \tag{6}$$

Let $c(R)$ denote the total cost of the greedy algorithm for the request sequence R. Note that

$$c(R) = \sum_{j=1}^{b} c(R_j). \tag{7}$$

Furthermore, it can be readily seen that any two functions f, g over the positive reals satisfy the following two properties.

$$\sum_{j=1}^{b} \min\{f(x_j), g(x_j)\} \leq \min\left\{\sum_{j=1}^{b} f(x_j), \sum_{j=1}^{b} g(x_j)\right\}, \text{ and} \tag{8}$$

$$\sum_{j=1}^{b} \max\{f(x_j), g(x_j)\} \leq \sum_{j=1}^{b} (f(x_j) + g(x_j)) \leq 2\max\left\{\sum_{j=1}^{b} f(x_j), \sum_{j=1}^{b} g(x_j)\right\}. \tag{9}$$

Combining (5), (6), (7) and using properties (8), (9), we obtain that

$$c(R) = O\left(\min\left\{\max\left\{\gamma b \frac{\log(k/b)}{\log\gamma}, \gamma b \frac{\log(k/b)}{\log\log(k/b)}\right\}, k\right\}\right) \cdot OPT.$$

□ □

The following theorem shows that the greedy algorithm is near-tight for PST in graphs of edge-asymmetry γ.

Theorem 16. *Every online algorithm for* PST *in graphs of asymmetry γ has competitive ratio $\Omega\left(\min\left\{\gamma b \frac{\log(k/b)}{\log\gamma}, k^{1-\epsilon}\right\}\right)$, for arbitrarily small constant ϵ.*

Proof. For a high-level description of the proof, we combine ideas from Theorem 12 and the lower-bound constructions of [18] and [4]. We then argue that the adversarial sequence will force any algorithm to pay a cost of $\Omega\left(\min\left\{\gamma \frac{\log(k/b)}{\log\gamma}, k/b\right\}\right)$, in each of a total number of b phases.

We first describe the adversarial graph, and we begin with the definition of some auxiliary constructions. Consider first a set E of columns, as defined in Section 4.2 (Case $k > b$), and as depicted in Figure 3. For the convenience of the reader, we replicate the definition in the remainder of this paragraph. Let $T_1 = \{v_1, \ldots, v_l\}$ and $T_2 = \{v_1, \ldots, v_l'\}$ be two disjoint sets of l vertices each (we should think of T_1 as lying higher than T_2). For every $i \leq l$, we say that the distance between v_i and v_i' is equal to 1. We call index i the i-th *column*, and require that l is a power of 2, and that it is large compared to k (e.g., at least 2^k). For a column $i \leq l$, we say that we *insert a vertex w at depth $d < 1$ in column i* if we introduce a new vertex w at distance d from vertex v_i and distance $1 - d$ from vertex v_i'. We denote by E the set of all l columns.

On the collection of columns E we define a construction called *block* which we denote by $B(E, \beta)$, which is similar, although not identical to the block construction of Theorem 12, and inserts vertices and edges in $\Theta\left(\frac{\log(k/b)}{\log\gamma}\right)$ *rounds*[3]. Here β is an integral parameter, with $\beta \in [1, b]$. The rounds are defined as follows: In the first round, for every $i \in [1, \gamma - 1]$, we insert l vertices (which for consistency with the earlier proof we call w-vertices) at depth i/γ, one for each column of E. Moreover, we group the w-vertices in s-sets as follows: the vertices on the i-th deepest layer are grouped into s-sets of size $l/2^i$. For every s-set we introduce an associated u-vertex, which is connected to the corresponding s-set by means of

[3]Without loss of generality we once again omit flours and ceilings, and assume all fraction quantities (as well as γ) are integral.

antiparallel edges of zero cost (or, more formally, of the same infinitesimally small cost) and priority equal to the parameter β.

Suppose that rounds $1, \dots j-1$ have been defined in the construction of $B(E, \beta)$, we show how to define round j. Let d denote the size of the s-set of smallest cardinality that was inserted in round $j-1$. For every $i \in [1, \gamma^j - 1]$, we insert l vertices at depth i/γ^j, one for each column in E, *unless* some other w vertex has been inserted at this same depth in a previous round, in which case we do not perform any additional insertion. This has the effect that $\Theta(\gamma^j)$ additional *layers* of vertices are inserted in round j. We call the i-th deepest layer of vertices that is inserted in round j the *i-th layer of round j*, and denote by $w^{j,i,1} \dots w^{j,i,l}$ its vertices. We partition the l w-vertices of the i-th layer of round j in $(2^i l)/d$ s-sets (from left to right), all of the same size $d/2^i$: In particular, the s-set $S^{j,i,q}$ (in words, the q-th s-set, from left to right, of the i-th layer in round j) consists of $d/2^i$ vertices $\{w^{j,i,\frac{(q-1)d}{2^i}+1}, \dots w^{j,i,\frac{qd}{2^i}}\}$. For each such s-set of the form $S^{j,i,q}$, we add a vertex $u^{j,i,q}$, and for every vertex $w \in S^{j,i,q}$ we add the edges $(w, u^{j,i,q})$ and $(u^{j,i,q}, w)$ of the same infinitesimally small cost and priority β. This completes the definition of round j.

Finally, we turn $B(E, \beta)$ into a well-defined graph by inserting *directed* edges connecting any two pairs of consecutive vertices in all columns. More precisely, for each consecutive pair of vertices w, w' of every column i, such that w is higher that w', we insert the edge (w, w') of cost equal to the distance of w and w' in the column i, and priority equal to b. We also insert the antiparallel edge (w', w), of cost $\gamma c(w, w')$, and priority again b. Informally, the "downwards" direction is cheap, whereas the "upwards" direction is expensive.

The adversarial graph G is defined by performing a series of block constructions, in b consecutive *phases*. Let T_1, T_2 and E be as defined earlier. Phase 1 consists only of the insertion of block $B(E, 1)$. Suppose that phase $p-1 < b$ has been defined, we next define phase p. Let d denote the smallest cardinality among s-sets inserted in phase $p-1$ (in other words, d is the cardinality of s-sets in the last (i.e., highest) layer inserted in the last round of phase $p-1$). This induces a partition of the columns in E in l/d sets, where the i-th set in this partition, say set E_i, consists of columns $(i-1)d+1, \dots id$. Phase p consists of inserting d blocks, namely blocks $B(E_i, p)$, for all $i \in [1, d]$.

As in the proof of Theorem 12, a subtle point has to do with vertices which are inserted in different phases yet they are in the same layer. Say w_1 and w_2 are vertices with this property, with w_1 added in phase j and w_2 added in phase j'. For every such pair, there are edges (w_1, w_2) and (w_2, w_1) of the same infinitesimally small cost and priority equal to $\min\{j, j'\}$. In words, this means that using these edges is free, however they can be of use only if we route traffic at a level at most the minimum of the two corresponding phases.

This essentially completes the construction of G. We also add a root vertex r and edges of the form (r, v), (v, r) for all $v \in T_1$, as well as edges of the form (t, v'), (v', t) for all $v' \in T_2$. These edges are assigned zero cost (or the same infinitesimally small cost) and priority b. Note that the asymmetry of the resulting graph is γ.

We conclude the description of the adversarial graph by making two important observations: First, the construction $B(E, \beta)$, for any value β is defined in such a way so as to be conceptually similar to the (recursively defined) adversarial graph of [18]. Thus, at an intuitive level, we aim at a lower bound which will be equal to the lower bound of [18], but replacing k with k/b, and multiplying by the overall number of phases b.

The second observation is that G is very similar to the adversarial graph of Theorem 12. The significant difference is that round j adds $\Theta(\gamma^j)$ layers of vertices instead of only 2^j. This also affects the overall number of rounds, which is now $\Theta\left(\frac{\log(k/b)}{\log \gamma}\right)$ instead of $\log \frac{k}{b} - 1$.

Given these similarities, we show how to produce the adversarial sequence and how the cost-analysis is

performed (we omit technical details that follow the analysis of [18] and [4]). For the adversarial sequence, we request u-vertices in the same manner as in the proof of Theorem 12 (Case 1). The only difference is that within round j, $\Theta(\gamma^j)$ terminals are requested instead of $\Theta(2^j)$. Again, within a round, the next vertex to be requested is the child of the last requested vertex.

Concerning the cost analysis, the cost of the optimal algorithm is at most one (as argued in the main result). On the other hand, consider a request at round j of phase p. There are two possibilities concerning the connection path that the algorithm can choose so as to serve this request. The first possibility it that the connection path originates from the root r (i.e., uses the "cheap", downwards edges of some column). In this case, the algorithm will pay a cost equal to the depth of the requested u-vertex. This holds because at the time of u's request, no connection path of any previous terminal crosses u.

The second possibility is that the connection path for u originates from a previously requested terminal u', of the same phase (thus the connection path does not visit r). In this case, we can argue that the algorithm must pay at least $\Omega(\gamma \cdot (1/\gamma^j))$ This follows from the fact that $\Theta(\gamma^j)$ terminals are requested in rounds $\leq j$ (of any given phase), and that the smallest distance between any two such vertices is $\Omega(\gamma \cdot (1/\gamma^j))$ (because the connection path should follow the "upwards", "expensive" direction). Once again, we use the fact that the algorithm/adversary game guarantees that when u is requested, no connection path of any previous terminal crosses u.

Summarizing, a terminal requested at round j of phase p of the game will contribute a cost equal to

$$\Omega\left(\min\left\{1/\gamma^{j-1}, d\right\}\right),$$

where d is the depth of the terminal.

Recall that there are $\Theta(\gamma^j)$ terminals in round j, from Theorem 1.2 in [4], their contribution to the cost is

$$\Omega\left(\min\left\{1/\gamma, \text{number of terminals in round } j\right\}\right),$$

Finally, we conclude that the total contribution of all $\Theta(\log(k/b)/\log\gamma)$ rounds in all b phases is

$$\Omega\left(\min\left\{\gamma b\frac{\log(k/b)}{\log\gamma}, k^{1-\epsilon}\right\}\right),$$

for arbitrarily small ϵ. □ □

We conclude this section with a discussion concerning the significance of theorems 15 and 16. The results show that when γb is small in comparison to k (e.g, when $\gamma b \in O(k^{1-\epsilon})$, for constant ϵ), then the greedy algorithm is a factor at most $O(\log\log(k/b))$ from the best online algorithm. Otherwise, e.g., when $\gamma b = \Omega(k)$, the corresponding gap is at most k^ϵ. In contrast, recall that the naive algorithm has competitive ratio $\Theta(k)$ independent of the parameters γ and b. Thus, according to Theorem 16, its performance is a factor as big as $\Theta(\frac{k}{\log k})$ from optimal, even when γ and b are small compared to k (e.g., constant). This demonstrates that the parameterized analysis of Theorem 15 achieves a relative separation of the two algorithms even in the setting of directed graphs.

4.4 Randomized algorithms

In this section we study the effect of randomization in online priority Steiner tree problems. We begin by showing that the lower bound of Theorem 12 holds even when considering randomized algorithms against an oblivious adversary. For the remainder of this section, we assume undirected graphs (i.e., $\gamma = 1$).

Theorem 17. *The competitive ratio of every randomized online algorithm against the oblivious adversary for online* PST *and* GPS *is* $\Omega\left(b \log \frac{k}{b}\right)$*, if* $k > b$ *and* $\Omega(k)$*, otherwise.*

Proof. We resort to Yao's principle [40]. It suffices to show that there exists an input graph, and a distribution over a sequence of k requests such that the average cost incurred by any deterministic algorithm (over this distribution) is at least $\Omega\left(b \log \frac{k}{b}\right)$, if $k > b$ and $\Omega(k)$, otherwise. times the average optimal cost.

The input graph is the graph G constructed as in the proof of Theorem 12, depending on whether $k > b$ or not. The distribution for the case $k > b$ is motivated by the request sequence in the deterministic case (see Section 4.2.1): more specifically, let u be the i-th u-vertex requested by the adversary during round j of phase p, and suppose that u belongs in component $C_p^{j,x}$. Then the $(i+1)$-th request (of round j and phase p) is chosen uniformly at random: with probability 1/2, the left child of u in component $C_p^{j,x}$ is requested, otherwise, the right child of u is requested. The first u-vertex of a round (of any phase), is chosen deterministically, so as to enforce Proposition 13(a) and (c). This ensures that any deterministic algorithm will pay, on average, a cost at least half the cost determined in the analysis of Theorem 12, whereas the optimal average cost remains as in the same analysis.

A similar argument can be made for the case $k \leq b$. In this case we use the adversarial tree-like graph of Section 4.2.2. Here, assuming that u is the last-requested vertex, we choose with equally probability which of its children will be subsequently requested. This guarantees that the deterministic algorithm still pays $\Omega(k^2)$, in expectation, whereas the optimal cost is at most k. □ □

Even though Theorem 12 and Theorem 17 are tight, they suggest that in the case in which b is large, namely when $b \in \Omega(k)$, the competitive ratio of any deterministic or randomized algorithm is disapointingly bad (i.e., $\Omega(k)$). However, observe that the adversarial graph G requires at least an exponential number of vertices (and edges) in the number of requested terminals k. Thus the following question arises: if the number of edges is comparable to the number of terminals, can we achieve better competitive ratios? Note that this type of question has already been addressed in the context of the (plain) online Steiner tree problem, as well as the online node-weighted Steiner tree problem (see Section 1 for some relevant results, in particular, the results of [7, 31, 26]).

Concerning the online priority Steiner tree problem, we show how to use the framework of Alon *et al.* [2], which studied online algorithms for broad classes of (edge-weighted) connectivity and cut problems using the multiplicative-updates approach. In particular, we note that GPS has a a formulation that matches the framework of [2] as follows. We are given graph $G = (V, E)$, with cost and priority functions defined over E as $c : E \to \mathbb{R}^+$ and $p : E \to \mathbb{R}^+$, respectively. A feasible *integral* solution to GPS is an assignment of *weights* (capacities) w from $\{0, 1\}$ such that for each request of the form (s_i, t_i, b_i) there is a flow from s_i to t_i in G that is routed only through edges of priority at least b_i and which has value at least 1. A feasible *fractional* solution is an assignment of *weights*[4] (capacities) w from $[0, 1]$ such that for each request of the form (s_i, t_i, b_i) there is a flow from s_i to t_i in G that is routed only through edges of priority at least b_i and which has value at least 1. The cost of the solution w, in both cases, is defined as $\sum_{e \in E} w_e c_e$.

We can then use the two-step approach of [2] in order to obtain an online randomized algorithm. The first step consists of computing an online fractional solution. This can be done, in black-box fashion, using the multiplicative-updates algorithm described in Section 3.1. of [2]. Then Theorem 3 of [2] implies the following:

Lemma 18. *There is an online fractional algorithm for* GPS *that is* $O(\log m)$ *competitive, where* m *is the number of edges in the graph.*

[4]The weight $w(e)$ of an edge should not be confused with its cost $c(e)$.

The second step in the approach s responsible for rounding the fractional solution to an integral one, and is performed on-line, during each step, i.e. after each pair request is issued. The rounding method is simple and is based on the rounding employed in the context of the multicast problem in [2]. More specifically, the method involves keeping $2\lceil \log(k'+1) \rceil$ independent random variables X_i distributed uniformly in the interval $[0,1]$, for all edges e such that $p(e) \geq p_i$. Here, k' denotes the number of terminal pairs served by the online algorithm up to the current point, which implies that the number of the random variables increases as the algorithm serves more and more requests. Define the threshold θ as $\min_{j=1}^{2\lceil \log(k'+1) \rceil} \{X_j\}$. The algorithm will then update its current integral solution I by adding in I all edges e of the graph with the property that $w(e) \geq \theta$ i.e., all edges whose weight exceeds the current value of the threshold (initially $I = \emptyset$). Using the same argument as in the proof of Lemma 7 in [2], we conclude that this rounding method introduces a multiplicative overhead of $O(\log k)$ in comparison to the cost of the fractional solution.

Lemma 19. *There is an online randomized algorithm that given a fraction solution for* GPS *produces an online integral solution which is $O(\log k)$ competitive.*

Combining Lemmas 18 and 19 we obtain the following:

Theorem 20. *There exists a randomized algorithm for* GPS *and* PST *with competitive ratio $O(\log k \log m)$, where m is the number of edges in the graph.*

5 Conclusion

In this paper we presented a parameterized analysis of the online node-weighted and the priority Steiner tree problems. More precisely, we gave tight upper and lower bounds on the competitive ratio of online algorithms by considering, as relevant parameters, the ratio of the maximum to the minimum node weight and the number of priority levels, respectively (in addition to the number of terminals in the request sequence). Our analysis confirms the intuitive expectation that naive algorithms (which are deemed optimal assuming competitive analysis solely over the number of terminals) are inferior to more sophisticated algorithms once the additional relevant parameters are taken into consideration. We have thus proven a strict separation between these two classes of algorithms which was yet to be reflected by "standard" competitive analysis.

An interesting direction for future work is to pursue a more refined classification of the performance of algorithms in the context of "hard" Steiner tree problems (i.e., problems with worst-case competitive ratio $\Omega(k)$) under different settings. For example, it is known that when the requests are generated randomly, (in particular, assuming requests are i.i.d. over the set of vertices), one can obtain constant competitive ratios for online Steiner tree in undirected graphs [20]. A natural question is what are the corresponding competitive ratios of the node-weighted and priority Steiner tree problem in this stochastic setting. What about the online directed Steiner tree problem? Another setting that is worth studying is the setting in which we allow the online algorithm some limited power in deferring or revoking its decisions. Suppose, for instance, that we allow the algorithm to change a single edge of the current Steiner tree, upon each request (instead of only adding edges). It turns out that this seemingly marginal increase in power yields a constant competitive ratio for the (plain) online Steiner tree problem [23]. It would be interesting to study whether significant performance improvement is possible for hard online Steiner tree problems under this setting.

References

[1] A. Agrawal, P. N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner tree problem on networks. *SIAM Journal on Computing*, 24:440–456, 1995.

[2] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. A general approach to online network optimization problems. *Transactions on Algorithms*, 2(4):640–660, 2006.

[3] N. Alon and Y. Azar. On-line Steiner trees in the Euclidean plane. *Discrete and Computational Geometry*, 10:113–121, 1993.

[4] S. Angelopoulos. Improved bounds for the online steiner tree problem in graphs of bounded edge-asymmetry. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 248–257, 2007.

[5] S. Angelopoulos. A near-tight bound for the online steiner tree problem in graphs of bounded asymmetry. In *Proceedings of the 16th Annual European Symposium on Algorithms (ESA)*, pages 76–87, 2008.

[6] S. Angelopoulos. Ion the competitiveness of the online asymmetric and euclidean steiner tree problems. In *Proceedings of the 7th International Workshop on Approximation and Online Algorithms (WAOA)*, pages 1–12, 2009.

[7] B. Awerbuch, Y. Azar, and Y. Bartal. On-line generalized Steiner problem. *Theoretical Computer Science*, 324(2–3):313–324, 2004.

[8] P. Basu, C. Chau, R. J. Gibbens, Saikat Guha, and R. E. Irwin. Multicasting under multi-domain and hierarchical constraints. In *11th International Symposium and Workshops on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 524–531, 2013.

[9] P. Berman and C. Coulston. Online algorithms for Steiner tree problems. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 344–353, 1997.

[10] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.

[11] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

[12] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM*, 60(1):6, 2013.

[13] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed steiner problems. *Journal of Algorithms*, 1(33):73–91, 1999.

[14] M. Charikar, J. Naor, and B. Schieber. Resource optimization in QoS multicast routing of real-time multimedia. *IEEE/ACM Transactions on Networking*, 12(2):340–348, 2004.

[15] J. Chuzhoy, A. Gupta, J. Naor, and A. Sinha. On the approximability of some network design problems. *Transactions on Algorithms*, 4(2), 2008.

[16] K. Claffy, G. Polyzos, and H.W. Braun. Traffic characteristics of the t1 nsfnet backbone. In *Proceedings of INFOCOM*, 1993.

[17] M. Faloutsos. *The Greedy the Naive and the Optimal Multicast Routing–From Theory to Internet Protocols*. PhD thesis, University of Toronto, 1998.

[18] M. Faloutsos, R.Pankaj, and K. C. Sevcik. The effect of asymmetry on the on-line multicast routing problem. *Int. J. Found. Comput. Sci.*, 13(6):889–910, 2002.

[19] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[20] N. Garg, A. Gupta, S. Leonardi, and P. Sankowski. Stochastic analyses for online combinatorial optimization problems. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, pages 942–951, 2008.

[21] M.X. Goemans and D.P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 6(24), 1995.

[22] A. Gu, A. Gupta, and A. Kumar. The power of deferral: maintaining a constant-competitive steiner tree online. In *Proceedings of the 45th Symposium on Theory of Computing Conference (STOC)*, pages 525–534, 2013.

[23] Albert Gu, Anupam Gupta, and Amit Kumar. The power of deferral: maintaining a constant-competitive steiner tree online. *SIAM Journal on Computing*, 45(1):1–28, 2016.

[24] S. Guha and S. Khuller. Improved methods for approximating node weighted steiner trees and connected dominating sets. *Information and Computation*, (150):228–248, 1999.

[25] A. Gupta and A. Kumar. Online steiner tree with deletions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 455–467, 2014.

[26] M. T. Hajiaghayi, V. Liaghat, and D. Panigrahi. Online node-weighted steiner forest and extensions via disk paintings, 2013.

[27] M. Imase and B. Waxman. The dynamic Steiner tree problem. *SIAM Journal on Discrte Mathematics*, 4(3):369–384, 1991.

[28] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[29] P. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *Journal of Algorithms*, (19):104–115, 1995.

[30] K. J. Lai, C. P. Gomes, M. K. Schwartz, Kevin S. McKelvey, D. E. Calkin, and C. A. Montgomery. The steiner multigraph problem: Wildlife corridor design for multiple species. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, (AAAI)*, 2011.

[31] J. Naor, D. Panigrahi, and M. Singh. Online node-weighted steiner tree and related problems. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 210–219, 2011.

[32] C. A. S. Oliveira and P. M. Pardalos. A survey of combinatorial optimization problems in multicast routing. *Comput. Oper. Res.*, 32(8):1953–1981, 2005.

[33] S. Ramanathan. Multicast tree generation in networks with asymmetric links. *IEEE/ACM Trans. Netw.*, 4(4):558–568, 1996.

[34] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the twenty-ninth annual ACM Symposium on Theory of Computing*, pages 475–484, 1997.

[35] M. Thimm. On the approximability of the Steiner tree problem. *Theoretical Computer Science*, 295(1):387–402, 2003.

[36] L. Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of the thirty-third annual ACM Symposium on Theory of Computing*, pages 453–461, 2001.

[37] V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[38] J. Westbrook and D. C. K. Yan. Linear bounds for on-line Steiner problems. *Information Processing Letters*, 55(2):59–63, 1995.

[39] J. Westbrook and D. C. K. Yan. The performance of greedy algorithms for the on-line Steiner tree and related problems. *Mathematical Systems Theory*, 28(5):451–468, 1995.

[40] A. C.-C. Yao. Probabilistic computations:towards a unified measure of complexity. In *In Proceedings of the 17th Annual IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1997.