Parameterized Analysis of Online Steiner Tree Problems

Spyros Angelopoulos

Max Planck Institut für Informatik
66123, Saarbrücken, Campus E1 4, Germany
sangelop@mpi-inf.mpg.de

Abstract. Steiner tree problems occupy a central place in both areas of approximation and on-line algorithms. Many variants have been studied from the point of view of competitive analysis, and for several of these variants tight bounds are known. However, in several cases, worst-case analysis is overly pessimistic, and fails to explain the relative performance of algorithms. We show how parameterized analysis can help resolve this problem. As case studies, we consider the Steiner tree problem in directed graphs and the priority Steiner tree problem.

Keywords. Online computation, competitive analysis, Steiner tree problems

1 Introduction

The *Steiner tree* problem occupies a central place in the area of approximation and online algorithms. In its standard version, the problem is defined as follows. Given an undirected graph $G = (V, E)$ with a weight (cost) function $c : E \to \mathbb{R}^+$ on the edges, and a subset of vertices $K \subseteq V$ with $|K| = k$ (also called *terminals*), the goal is to find a minimum-cost tree which spans all vertices in K. When the input graph is *directed*, the input to the problem must specify, in addition to G and K, a vertex $r \in V$ called the *root*. The problem is then to find a minimum cost *arborescence* rooted at r which spans all vertices in K.

In the *online* version of the problem, the terminals in K are revealed to the algorithm as a sequence of requests. When a request for terminal $u \in V$ is issued, and assuming a directed graph, the algorithm must guarantee a directed path from r to u. The input graph G is assumed to be known to the algorithm. Using the standard framework of competitive analysis (see, e.g., [9]), the objective is to design online algorithms of small *competitive ratio*. More precisely, the competitive ratio is defined as the supremum (over all request sequences and input graphs) of the ratio of the cost of the arborescence produced by the algorithm over the optimal off-line cost assuming complete knowledge of the request set K.

Apart from its theoretical importance and its application in several combinatorial optimization problems, the Steiner tree problem is useful in modeling

Dagstuhl Seminar Proceedings 09171
Adaptive, Output Sensitive, Online and Parameterized Algorithms
http://drops.dagstuhl.de/opus/volltexte/2009/2121

efficient multicast communication over a network. Indeed, multicasting involves dissemination of information from a designated source to members of a subscribing group. The online variant describes then the situation in which subscribers to the service are not known beforehand, but rather issue dynamic requests to join the group. The reader is referred to [16] for a study of the relation between Steiner tree problems and network multicasting.

In this paper we focus on the following on-line Steiner tree problems, and review some recent results obtained in [2,3,4].

• In the *asymmetric* Steiner tree problem, the underlying graph is directed, and a specific vertex $r \in V$ is designated as the *root*. We define the *asymmetry* α of graph G as the maximum ratio of the cost of antiparallel links in G. More formally, 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 is defined as

$$\alpha = \max_{\{v,u\} \in A} \frac{c(v, u)}{c(u, v)}$$

Given a set $K \subseteq V$ of terminals, we seek the minimum cost *arborescence* rooted at r that spans all vertices in K. In addition, our aim is to express the performance of the algorithm in terms of the parameters k and α.

• The *priority Steiner tree problem*. Here, the underlying network is represented by a graph $G = (V, E)$ (which unless specified, is assumed to be undirected). Let $r \in V$ denote the source of the multicast group (which we also call *root*). We let $K \subseteq V$ denote the set of receivers of the multicast group, also called *terminals*.

In addition, the network can support a set of different *Quality of Service* (QoS) levels that are modeled by b integral *priorities* $1 \dots b$, where b is the highest priority and 1 the lowest. Every edge $e \in E$ is associated with its own *priority value* $p(e)$, which reflects the level of QoS capabilities of the corresponding link (e.g., the bandwidth of the link) as well as with a cost value $c(e)$ which reflects the cost incurred when including the link in the multicast tree. Last, every terminal $t \in K$ is associated with a priority $p(t) \in [1, b]$, which describes the QoS level it requires. A feasible Steiner tree T in this model is a tree rooted in r that spans K, and is such that for every terminal $t \in K$, the priority of *each* edge in the path from r to t in T is at least $p(t)$. The interpretation of this requirement is that terminal t should be able to receive traffic at a level of QoS at least as good as $p(t)$. The objective of the problem is to identify a feasible tree of smallest cost, where the cost of the tree is defined as the total edge-cost of the tree.

The asymmetric Steiner tree problem is motivated by the observation that a directed graph is a more appropriate and realistic representation of a real network. A typical communication network consists of links asymmetric in the quality of service they provide. In [13], studies on the traffic of network backbones reveal marked asymmetry in parameters such as speed, link utilization and reliability. For instance, a subscription to a home internet-cable service will normally incur more traffic on the incoming ("download") link than the outgoing ("upload") link. Nevertheless both directed links are expected to be present,

albeit they may have different characteristics. This situation becomes even more prevalent in wireless networks, due to differences in noise levels, power of transmission, and mobility levels of its endpoints.

Note that according to this measure, undirected graphs are graphs of asymmetry $\alpha = 1$, 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 ($\alpha = \infty$). Between these extreme cases, graphs of small asymmetry model networks that are relatively homogeneous in terms of the cost of antiparallel links.

On the other hand, the priority Steiner tree problem can formulate multicast communication in an environment comprised by heterogeneous users. A multicast group consists of a source which disseminates data to a number of receivers, i.e., the members of the group. These members may vary significantly in their characteristics (such as the bandwidth of the end-connection or their computational power), which implies that they may require vastly different QoS guarantees in terms of the delivered traffic. The objective is to deliver information to all members of the group, while meeting the requirements of each individual member. Furthermore, this dissemination must be efficient in terms of utilization of network links.

Parameterized analysis The asymmetric Steiner tree problem is a parameterized version of the Steiner tree problem in directed graphs. It has long been known, that in general directed graphs (i.e., graphs of unbounded asymmetry), 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 [17]. This motivates the study of the problem in graphs of bounded asymmetry. In particular, one should expect that in graphs of bounded α, the competitive ratio should be a function of both α and k; moreover, the performance of an efficient algorithm should degrade gently as the asymmetry increases.

A similar picture can be drawn for the priority Steiner tree problem. When b at least as large as k (the number of terminals), it is easy to show that the competitive ratio is $\Theta(k)$, which is again matched by the naive algorithm that buys least-cost paths from the root of sufficient priority. In other words, the competitive ratio is not a reliable measure unless the analysis is performed assuming a given value of b. Once again, we aim to express the competitive ratio of algorithms as a function of both the number of terminals k and the number of the priority levels b, and we expect that as b increases, the competitiveness of the problem degrades.

1.1 Related work

Steiner tree problems have been extensively studied from the point of view of online algorithms. For graphs of either constant or unbounded asymmetry, the competitive ratio is tight. For the former class, Imase and Waxman [15] showed that a simple greedy algorithm is optimal and achieves competitive ratio $\Theta(\log k)$.

Berman and Coulston [8] extended the result to the Generalized Steiner problem by providing a more sophisticated algorithm. The performance of the greedy algorithm for online Steiner Trees and its generalizations has also been studied by Awerbuch *et al.* [5] and Westbrook and Yan [18]. 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 [1].

The first study of the online asymmetric Steiner tree problem is due to Faloutsos *et al.* [14] who showed that a simple greedy algorithm has competitive ratio $O(\min\{\alpha \log k, k\})$. On the negative side, they showed a lower bound of $\Omega\left(\min\left\{\frac{\alpha \log k}{\log \alpha}, k\right\}\right)$ on the competitive ratio of every deterministic algorithm.

It is important to note that when $\alpha \in \Omega(k)$ the lower bound on the competitive ratio due to [14] is $\Omega(k)$, which is obviously tight (using the trivial upper bound of $O(k)$ for the greedy algorithm). Thus the problem is interesting only when $\alpha \in o(k)$.

The priority Steiner tree problem was introduced by Charikar, Naor and Schieber [11]. In their work, they provided a $O(\min\{\log k, b\})$ approximation algorithm. The question of whether the problem could be approximated within a constant factor was left open in [11], and sparked considerable interest in this problem, until Chuzoy *et al.* [12] showed a lower bound of $\Omega(\log\log n)$ (under the complexity assumption that NP has slightly superpolynomial time deterministic algorithms). Interestingly, this is one of few problems for which a $\log\log$ inapproximability result is currently the best known.

2 Results and techniques

In this section we summarize some recent improved bounds for the online asymmetric and priority Steiner tree problems. We provide only sketches of the proofs; the interested reader is referred to [2,3,4] for the technical details.

The upper bounds for both problems are achieved by simple greedy algorithms (to which we refer as Greedy). For the asymmetric Steiner tree problem Greedy works by connecting each requested terminal u to the current arborescence by buying the edges in a least-cost directed path from the current arborescence to u. For the priority Steiner tree problem, Greedy serves a request of the form (u, p) (namely, a new terminal u issues a request for a path of priority at least p), by buying the edges in a least-cost path from the current arborescence to u, such that there exists a path from the root to u of priority at least p (among edges already bought).

2.1 Summary of results

For the online asymmetric Steiner we derive following upper bound:

Theorem 1 ([2,3]). *The competitive ratio of* Greedy *for an input graph of asymmetry α and a request sequence of k terminals is*
$O\left(\min\left\{\max\left\{\alpha\frac{\log k}{\log \alpha}, \alpha\frac{\log k}{\log\log k}\right\}, k\right\}\right)$.

On the negative side, one can show the following near-tight lower bound:

Theorem 2 ([3]). *Any deterministic algorithm for the Steiner tree problem in graphs of asymmetry α has competitive ratio $\Omega\left(\min\left\{\max\left\{\alpha\frac{\log k}{\log\alpha}, \alpha\frac{\log k}{\log\log k}\right\}, k^{1-\epsilon}\right\}\right)$ (where ϵ is any arbitrarily small constant).*

For the online priority Steiner tree problem, we obtain a tight bound for both deterministic and randomized algorithms:

Theorem 3 ([4]). *The competitive ratio of every deterministic or randomized online algorithm for priority Steiner tree is $\Theta\left(\min\{b\log\frac{k}{b}, k\}\right)$.*

Last, as a combination of the two problems, we study the competitiveness of priority Steiner tree assuming directed graphs. If antiparallel links have the same costs, but their priorities can differ by as little as one, it is easy to show a tight bound of $\Theta(k)$ on the competitive ratio. 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 α. For this case, we derive an upper bound of $O\left(\min\left\{\max\left\{\alpha b\frac{\log(k/b)}{\log\alpha}, \alpha b\frac{\log(k/b)}{\log\log(k/b)}\right\}, k\right\}\right)$, and a corresponding lower bound of $\Omega\left(\min\left\{\alpha b\frac{\log(k/b)}{\log\alpha}, k^{1-\epsilon}\right\}\right)$ (where ϵ is any arbitrarily small constant).

2.2 Outline of techniques

We begin with some preliminary notation and definitions. The cost of a directed path p will be denoted by $c(p)$. We denote by $c(T)$ the cost of arborescence T, namely the sum of the cost of the directed edges in T. We emphasize that only edges in T and none of their antiparallel edges contribute to $c(T)$. We will always use T^* to denote the optimal arborescence on input (G, K), with $|K| = k$, and $OPT = c(T^*)$. For any $K' \subseteq K$, we let $c_{GR}(K')$ denote the cost that Greedy pays on the subset K' of the input (in other words, the contribution of terminals in K' towards the total cost of Greedy). Similar definitions extend to the priority Steiner tree problem.

Asymmetric Steiner tree

Upper bound In order to prove Theorem 1, we first show that it applies to situations in which the spanning arborescence has a fairly simple structure: in particular, to instances called *comb instances* in [2] (see Figure 1 for an illustration).

Definition 1. *Let T' denote a tree rooted at vertex $r' \in V$ and let $K' \subseteq K$, with $|K'| = k'$. We call the triplet $\mathcal{C} = (T', K', r')$ a* comb instance*, or simply* comb *if the following hold: T' consists of a directed path P from r' to a certain vertex v_1, which visits vertices $v_{k'}, \ldots, v_1$ in this order (but possibly other vertices too); there are also disjoint directed paths t_i from v_i to u_i. No other edges are in T'.*

Finally the set K' is precisely the set $\{u_1, \ldots, u_{k'}\}$. We call P the backbone *of $\mathcal{C}$, and the paths t_i the* terminal paths *of the comb. The vertex set of $\mathcal{C}$ is the set of vertices in T'.*

The following is a key theorem in the analysis of GREEDY. Essentially the theorem states that we can achieve the desired competitive ratio if we know, for instance, that (T^*, K, r) is a comb.

Theorem 4. *Given the comb $\mathcal{C} = (T', K', r')$, let $z \in K'$ denote the terminal requested the earliest among all terminals in K'. Then $c_{GR}(K') = c_{GR}(z) + O\left(\max\left\{\alpha\frac{\log k'}{\log \alpha}, \alpha\frac{\log k'}{\log\log k'}\right\}\right) c(T')$.*

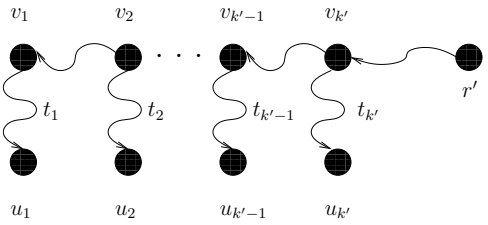

Fig. 1. The structure of a comb instance.

Given Theorem 4, the main result follows by partitioning the set of all requests K into a collection of near-disjoint comb-instances. Here, by near-disjoint we require that every edge in T^* appears in at most two comb instances.

In order to prove Theorem 4, let π denote a permutation of $\{1, \ldots, k'\}$ such that $\sigma = u_{\pi_1}, \ldots, u_{\pi_{k'}}$ is the sequence of the requests in K' in the order in which they are requested (hence $z = u_{\pi_1}$). Note that we aim towards bounding $c_{GR}(K' \backslash u_{\pi_1})$. To this end, we need to determine an assignment for every terminal u_{π_i} with $2 \leq i \leq k'$ to a *specific* terminal $\overline{u}_{\pi_i} \in \{u_{\pi_1}, \ldots u_{\pi_{i-1}}\}$. We call terminal $\overline{u}_{\pi_i}$ the *mate* of u_{π_i}. Let q_i denote the directed path in $\hat{T}$ from $\overline{u}_{\pi_i}$ to u_{π_i}, also called the *connection path* for u_{π_i}. Since $c_{GR}(K' \setminus u_{\pi_1}) \leq \sum_{i=2}^{k'} c(q_i)$ suffices to show that:

$$C \stackrel{\text{def}}{=} \sum_{i=2}^{k'} c(q_i) = O\left(\max\left\{\alpha\frac{\log k'}{\log \alpha}, \alpha\frac{\log k'}{\log\log k'}\right\}\right) c(T'). \tag{1}$$

Comb instances were identified in [2] as the hard instances for the problem, and for such instances, a weaker version of Theorem 4 was proved (c.f. Lemma 3.2 in [2]). More precisely, the definition of the comb in [2] requires a strict upper bound of $O(\alpha)$ on the number of terminals in the comb: this leads to an upper bound for $c_{GR}(K' \setminus u_{\pi_1})$ equal to $O\left(\alpha\frac{\log \alpha}{\log\log \alpha}\right) c(T')$. The proof of the main result in [2] proceeds then by first extending the result to all subtrees of T^* of

$O(\alpha)$ terminals (not necessarily combs), and then by applying it, in a recursive manner, in a hierarchical partition of T^* in trees of $O(\alpha)$ terminals each. This process yields an additional multiplicative overhead of $\log k/\log\alpha$ compared to the cost incurred by a comb instance of size $O(\alpha)$.

In [3] we follow a different approach. We allow the combs to contain an arbitrarily large number of terminals, which may very well be in $\Omega(\alpha)$. This allows us to bypass the need for recursion, and thus to save the factor $\log k/\log\alpha$. Instead, as already mentioned, suffices to decompose T^* (and K) into a collection of near-disjoint comb instances. In this more general setting, some of the high-level proof ideas remain as in [2]: we still partition the terminals in a comb in appropriately defined subsets called *runs* which dictate how to select the proper mate for each terminal. However, the definition of runs and the assignment of mates in [2] is not applicable anymore when $k' \in \Omega(\alpha)$: more specifically, a connection path can be as costly as the cost of the backbone (which becomes far too expensive if the number of terminals in the comb is in $\Omega(k)$). Instead, a substantially more involved assignment is required. We then perform an amortize analysis so as to bound the overall cost incurred by the assignment.

Lower bound We sketch how to derive a lower bound of $\Omega(\min\{\alpha\frac{\log k}{\log\log k}, k^{1-\epsilon}\})$, which in conjunction with the lower bound of [14]) yields Theorem 2. Consider the graph G illustrated in Figure 2: this graph will provide the motivation behind the definition of the actual adversarial input graph. Graph G is such that all "downwards" edges are cheap, whereas all "upwards" edges are expensive, and each has cost α times the cost of its antiparallel upwards edge. In particular, there is a path from the root r to a vertex t such that the cost of the directed path $r \to t$ is equal to 1 (hence the cost of the path $t \to r$ is α). Call P the path from s to t.

In addition, there are $\Theta(k)$ pairs of vertices, of the form (v_i, u_i), defined in a recursive manner as follows. Let x be such that $x^x = k$, hence $x = \Theta(\frac{\log k}{\log\log k})$, and assume without loss of generality that x is integral. The first group of vertices, namely group K_1, consists of x pairs such that the $v_i's$ are all evenly distributed over the path P (namely the cost of the path from v_i to v_{i+1}, over edges of P is the same for all i's such that $v_i \in K_1$, and is also the same as the cost of the path $r \to v_1$ and $v_x \to t$). In addition, the cost of the edge (v_i, u_i) is equal to $\frac{1}{x^2}$ whereas is antiparallel edge has cost $\alpha\frac{1}{x^2}$.

Suppose now that groups $K_1, \ldots K_j$ have been defined, we will show how to define K_{j+1}. For every pair of consecutive vertices (v_i, v_{i+1}) in P such that each of v_i, v_{i+1} belongs in some K_m with $m \leq j$, we insert x vertices of the form $v_i^1, \ldots, v_i^x$, all distributed evenly over the path $v_i \to v_{i+1}$. In addition, we insert x vertices of the form u_i^l, $l \in [1, x]$ such that the cost of the edge (v_i^l, u_i^l) is $\frac{1}{x^{j+2}}$, while the antiparallel edge (u_i^l, v_i^l) has cost $\alpha\frac{1}{x^{j+2}}$. We do the same with the pairs (r, v_1) and $v_{last}, t)$, where v_{last} is the bottom vertex among all groups K_m with $m \leq j$ We continue until x groups have been defined. Note that the total number of u-vertices is then $\Theta(k)$.

The adversary will request u-vertices as terminals in *rounds*. In particular, in round i, with $1 \leq i \leq x$, the adversary will request the u-vertices of group

K_i, in a bottom-up manner (i.e., starting from the u-vertex which is the farthest away from r and ending with the one that is the closest).

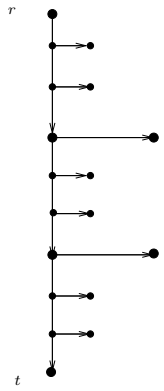

Fig. 2. The structure of the adversarial graph G, for the case $x = 2$. Only "cheap" edges are shown while "expensive" antiparallel edges are omitted.

To illustrate the intuition behind our argument, we will make the following assumption: suppose that every time the algorithm establishes a new connection path for a certain request (i.e., a path from a previously requested terminal), *several new edges must be bought*, in the sense that there is little overlapping between the new connection path and the ones which have already been established. Of course this is not the case in G itself, and enforcing this requirement is by no means a trivial task. In fact, much of the details in the formal proof of Theorem 2 is dedicated to this issue.

Consider then the l-th u-vertex (terminal) requested in round j. There are three options concerning the connection path for this u-vertex: either i) will originate in r; or ii) originate in a "higher" vertex which was requested in an earlier round; or iii) originate in a "lower" vertex which was requested before u. In the second and third cases, a cost (roughly) of at least $\alpha\frac{1}{(x+1)^j}$ will be incurred. It is easy to show that round j consists of $\Theta((x+1)^j)$ vertices. Thus, if the majority of the requests in round j fall in the last two cases, a cost of $\Omega(\alpha)$ is incurred for the round. Otherwise, the majority of the requests in the round are for terminals u which incur cost roughly the cost of the directed path from r to u, which translates to a total cost of $\Omega(k_j)$, where k_j is the number of requests in K_j. This argument shows that each round contributes a cost of $\Omega(\min\{k_j, \alpha\})$. Since there are x rounds, the result will then follow by combining the contribution of each round to the overall cost, and the observation that the the optimal algorithm will buy the path P and all edges of the form (v, u), for a total cost which can be shown that it is bounded by a constant.

Priority Steiner tree

Upper bound We show a simple algorithm for the problem that is asymptotically optimal. The result also implies that algorithm GREEDY has the same competitive ratio. We use the online greedy algorithm for the (plain) Steiner tree problem in undirected graphs, which we denote by GR. In particular, our algorithm maintains b Steiner trees, $T_1, \ldots T_b$, one for each priority value, in an online fashion (the trees may not be edge-disjoint, i.e., T_i may share edges with T_j). When a request $r_i = (v_i, b_i)$ is issued, the algorithm will assign node v_i to forest T_{b_i}, by running the GR algorithm for terminal v_i (ignoring the priority values). We denote the resulting algorithm by PRGR.

For any fixed integer $j \leq b$, let G_j denote the subgraph of G induced by all edges e of priority at least j. 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 Steiner tree problem on graph G_j and request sequence R_j: here we ignore the priority of edges and requests, and our objective is to minimize the cost of the Steiner tree for the sequence R_j without concerns about priorities. Let OPT_j denote the cost of the optimal Steiner tree for the above instance. Since the greedy algorithm is $O(\log |R_j|)$-competitive, it follows that $c(R_j) = O(\log |R_j|)OPT_j = O(\log |R_j|)OPT$ (this follows from the fact that $OPT_j \leq OPT$). Here, we denote by $c(R_j)$ the cost paid by PRGR on request set R_j (which is the same as the cost of the greedy algorithm for instance (G_j, R_j), and by OPT the cost of the optimal offline solution to the problem.

Therefore, the total cost of PRGR on sequence R 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 $c(R) = O\left(b \log \frac{k}{b} \cdot OPT\right)$.

In addition, $c(R) \leq k \cdot OPT$. Therefore, $c(R) = O\left(\min\left\{b \log \frac{k}{b}, k\right\}\right) OPT$, when $k > b$, and $c(R) = O(k)OPT$, otherwise.

Lower bound The crux in the proof of the lower bound in the statement of Theorem 3 is to use an input graph and an appropriate request sequence such that the following hold: there are b different groups of terminals, with each group consisting of $\Theta(k/b)$ terminals. Group $i \in [1, b]$ consists of terminals with priority equal to i. Furthermore, the construction enables us to derive a lower bound of $\Omega(\log(k/b)$ on the total cost paid by any deterministic or randomized algorithm to serve the requests in group i. This is achieved by creating a construction based on the idea of the *diamond graph* for the standard online Steiner tree problem (see [15]).

3 Conclusions and open problems

The objective of this report is to demonstrate that parameterized analysis can yield more realistic bounds for online Steiner tree problems. As case studies we

considered the directed and priority Steiner tree problems. For the former, the parameter of interest is the asymmetry of the graph, whereas for the latter, the total number of different priority levels. Since in real networks these parameters usually attain small values, our analysis establishes bounds which we believe reflect better the true complexity of the problems.

We expect that similar types of analysis can be applicable to other on-line optimization problems. A representative example is the problem of *file allocation* in general undirected networks, which is a well-studied problem in both theory and applications of distributed systems. The influential work of Bartal *et al.* [7] shows that if A is a c-competitive algorithm for the online Steiner tree problem (in undirected graphs), then it is possible to derive a (randomized) $(2+\sqrt{3})c$-competitive algorithm for file allocation. In a similar vein, Awerbuch *et al.* [6] show that a (stronger) upper bound of $O(\min\{\log k, \log Diam\})$ on the competitive ratio of the greedy Steiner tree algorithm implies a deterministic algorithm for file allocation with the same competitive ratio (here $Diam$ denotes the diameter of the graph). Notice that, as with multicasting, file allocation is a problem motivated by distributed networks, and once again, it would only be natural to study it under directed graphs. Therefore, we expect results for online Steiner tree problems will provide a useful tool in resolving other online network problems, in settings which are more realistic in practice.

Similar questions can be asked for approximation algorithms. For instance, the directed Steiner tree problem in graphs of asymmetry α is $O(\min\{\alpha, k^{\epsilon}\})$ approximable, as it follows from the result of Charikar *et al.* [10]. We conjecture that this bound can be improved to $O(\min(\alpha^{\epsilon}, k^{\epsilon}))$, for arbitrarily small ϵ.

References

1. N. Alon and Y. Azar. On-line Steiner trees in the Euclidean plane. *Discrete and Computational Geometry*, 10:113–121, 1993.
2. S. Angelopoulos. Improved bounds for the online Steiner tree problem in graphs of bounded edge-asymmetry. In *Proceedings of the 18th Annual Symposium on Discrete Algorithms (SODA)*, pages 248–257, 2007.
3. 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.
4. S. Angelopoulos. Online priority Steiner tree problems. In *Proceedings of the 20th Symposium on Algorithms and Data Structures (WADS)*, 2009. To appear.
5. B. Awerbuch, Y. Azar, and Y. Bartal. On-line generalized Steiner problem. *Theor. Comp. Sci.*, 324(2–3):313–324, 2004.
6. B. Awerbuch, Y. Bartal, and A. Fiat. Competitive distributed file allocation. *Information and Computation*, 185(1):1–40, 2003.
7. Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management. *Journal of Computer and System Sciences*, 51(3):341–358, 1995.
8. P. Berman and C. Coulston. Online algorithms for Steiner tree problems. In *Proc. of the 39th Symp. on the Theory of Computing*, pages 344–353, 1997.
9. A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

10. 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.
11. 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.
12. J. Chuzhoy, A. Gupta, J. Naor, and A. Sinha. On the approximability of some network design problems. *Trans. on Algorithms*, 4(2), 2008.
13. K. Claffy, G. Polyzos, and H.W. Braun. Traffic characteristics of the T1 NSFNET backbone. In *Proceedings of INFOCOM*, 1993.
14. 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.
15. M. Imase and B. Waxman. The dynamic Steiner tree problem. *SIAM Journal on Discrte Mathematics*, 4(3):369–384, 1991.
16. 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.
17. J. Westbrook and D. C. K. Yan. Linear bounds for on-line Steiner problems. *Information Processing Letters*, 55(2):59–63, 1995.
18. J. Westbrook and D. C. K. Yan. The performance of greedy algorithms for the on-line Steiner tree and related problems. *Math. Syst. Theory*, 28(5):451–468, 1995.