Improved Bounds for the Online Steiner Tree Problem in Graphs of Bounded Edge-Asymmetry

Spyros Angelopoulos*

Abstract

In this paper we consider the Online Steiner Tree problem in weighted directed graphs of bounded edge-asymmetry α. The edge-asymmetry of a directed graph is defined as the maximum ratio of the cost (weight) of antiparallel edges in the graph. The problem has applications in multicast routing over a network with non-symmetric links. We improve the previously known upper and lower bounds on the competitive ratio of any deterministic algorithm due to Faloutsos *et al.* [11]. In particular, we show that a better analysis of a simple greedy algorithm yields a competitive ratio of $O(\min\{k, \frac{\alpha \log k}{\log \log \alpha}\})$, where k denotes the number of terminals requested. On the negative side, we show a lower bound of $\Omega(\min\{k^{1-\epsilon}, \frac{\alpha \log k}{\log \log k}\})$ on the competitive ratio of every deterministic algorithm for the problem, for any arbitrarily small constant ϵ.

1 Introduction

The *Steiner Tree* problem in undirected graphs is defined as follows. Given an undirected graph $G = (V, E)$ with a cost function $c : E \rightarrow \mathbb{R}^+$ 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. The cost of the tree is defined as the sum of the costs of its edges. When the input graph is *directed*, the input to the problem must specify, in addition to G and K a specific vertex $r \in V$ called the *root*. The problem then translates to finding a minimum cost *arborescence* rooted at r which spans all vertices in K.

In the *online* version of the problem, terminals in K are requested in an online, sequential fashion. Every time a request (terminal) $t \in V$ arrives, the algorithm must ensure that there is a path from an earlier requested terminal to t, for the undirected version, or a directed path from r to t in the directed version of the problem, respectively. We assume that the graph G is known to the algorithm. In the standard framework of competitive analysis (see, e.g., [7]), the goal is to design online algorithms of small competitive ratio. In the context of Steiner tree problems the competitive ratio is defined as the supremum of the ratio of the cost of the tree (or arborescence) 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 versions of the problem have been studied extensively in the literature (c.f. section 1.1 for some representative results concerning the online version) and are often encountered in the context of several combinatorial optimization problems. In addition to interest from the point of view of theoretical analysis, the problem has important applications in 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 (edges of the underlying graph) so as to minimize the cost for supporting multicast routing through a tree, which is essentially identical to the Steiner tree problem formulation. For the interplay between the Steiner Tree problem and multicast applications, the interested reader is referred to the work of Faloutsos [10].

Most of the theoretical work on the Steiner tree problem and its generalizations is focused on undirected graphs (see eg [15] [1] [12] [6] [17] for some representative results concerning the offline version of the problem). On the other hand, research considering directed underlying graphs has been relatively limited (see e.g. [8] [16], once again for the offline case) . However, a directed graph is a more appropriate and realistic representation of a real-life network. As argued in [14], [9] studies on network traffic on backbones have revealed marked asymmetry in link utilization. For instance, one should expect that a typical subscriber to a home internet-cable service will incur more traffic on the incoming link ("download"), than the outgoing link ("upload"). Moreover [14] if the link is wireless, its quality/bandwidth is inherently asymmetric, due to differences in noise levels, power of transmission and mobility levels of its endpoints.

---

*David R. Cheriton School of Computer Science, University of Waterloo, Waterloo ON, Canada N2L 3G1. email: `sangelop@cs.uwaterloo.ca`

Motivated by such observations, Ramanathan [14] proposed several metrics which are meant to capture deviation from the symmetry observed in undirected graphs. Perhaps the most intuitive metric is the so-called *maximum edge-asymmetry* α of a directed graph $G = (V, E)$ (or simply *asymmetry*, for the rest of this paper) which is defined as the maximum ratio of the costs of antiparallel links in the graph. To define this measure 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

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

Note that undirected graphs can be seen as graphs of asymmetry $\alpha = 1$, while directed graphs in which there is at lest one pair of vertices v, u such that $(v, u) \in E$, but $(u, v) \notin E$ are graphs with unbounded asymmetry (meaning that $\alpha = \infty$). Between these extreme cases, graphs with relatively small asymmetry model networks which are relatively homogenous in terms of the quality/characteristics of antiparallel links.

Ramanathan presented a 2α-approximation algorithm for the offline Steiner problem in graphs of asymmetry α. In addition, Ramanathan showed that the same approximation ratio is guaranteed when a different metric for capturing the graph asymmetry is applied, namely the ratio, over all adjacent pairs of vertices in the graph, of the sum of the larger edge-pair costs over the sum of the smaller edge-pair costs. In this paper we focus exclusively on the maximum-edge asymmetry only, since it represents a more clean-cut and easy to estimate measure of the graph edge asymmetry.

In subsequent work, Faloutsos Pankaj and Sevcik [11] studied the online Steiner tree problem in graphs of asymmetry α. They showed that a simple greedy algorithm (which we denote by Greedy) has a competitive ratio of $O(\min\{k, \alpha \log k\})$. In particular, when a new terminal t is requested, Greedy will find the directed path of minimum cost from some vertex in the current arborescence to t, and buy such edges. Once an edge is bought it is assigned zero cost in subsequent iterations, to reflect that the edge has irrevocably become part of the solution. Intuitively, the upper bound is not too difficult to derive: first,note that the cost of Greedy is at most k times the optimal cost. Second, since Greedy is $O(\log k)$-competitive for undirected graphs (see Section 1.1) when we move to directed graphs, the competitive factor is multiplied by at most α (to account for the fact that the connection paths may choose the "wrong", expensive direction). On the negative side, Faloutsos *et al.* showed a lower bound on the competitive ratio of $\Omega(\min\{k, \frac{\alpha \log k}{\log \alpha}\})$ for every deterministic algorithm[1]. The construction for the lower bound is interesting, since not only indicates that the problem is not trivial, but also provides some intuition about a better analysis of the greedy algorithm.

In this paper we narrow the gap between the upper and lower bounds for the problem. In particular, we first provide a more elaborate analysis of Greedy and prove the following:

Theorem 1.1. Greedy *is* $O(\min\{k, \frac{\alpha \log k}{\log\log \alpha}\})$-*competitive.*

On the negative side, we show the following lower bound on the competitive ratio of deterministic algorithms:

Theorem 1.2. *The competitive ratio of every deterministic online algorithm is* $\Omega(\min\{k^{1-\epsilon}, \alpha \frac{\log k}{\log\log k}\})$, *for every constant* $0 < \epsilon < 1$.

Theorem 1.2 in conjunction with the lower bound of [11] yields

Corollary 1.1. *The competitive ratio of every deterministic online algorithm is* $\Omega(\max\{\min\{k, \frac{\alpha \log k}{\log \alpha}\}, \min\{k^{1-\epsilon}, \alpha \frac{\log k}{\log\log k}\}\})$ *for every constant* $0 < \epsilon < 1$.

Our results improve the known bounds in a variety of situations, depending on the parameters α and k. In particular, consider graphs which are highly asymmetric, in the sense that there is a constant $c > 1$ such that $k = \alpha^c$, for some constant c. In this case Theorem 1.1 and Theorem 1.2 yield a tight bound of $\Theta(\frac{\alpha \log \alpha}{\log\log \alpha})$ whereas the analysis of [11] only shows the trivial bounds of $\Omega(\alpha)$ and $O(\alpha \log \alpha)$.

An outline of the intuition behind the proof of Theorem 1.1 is given is Section 2. Section 3 is dedicated to the formal proof of Theorem 1.1. Section 4 outlines the proof of Theorem 1.2.

Due to space limitations, certain technical proofs are omitted in this extended abstract. Details on all proofs can be found in the full version of the paper [3].

1.1 Some related results

The Steiner tree problem has been extensively studied in several settings and variations. We overview only some of the results which are

---

[1] Note that when $\alpha \in \Omega(k)$ the lower bound on the competitive ratio due to [11] is $\Omega(k)$, which is obviously tight (using the trivial upper bound for the greedy algorithm). Thus the problem is interesting only when $\alpha \in o(k)$.

of particular relevance to this work, and pertain to online versions of the problem. For the online Steiner tree problem in *undirected* graphs, Imase and Waxman [13] showed a tight bound of $\Theta(\log k)$ on the competitive ratio of online Steiner Tree. In directed graphs of unbounded asymmetry, it is very easy to show that the competitive ratio of every algorithm, deterministic or randomized, is as large as the trivial bound, namely $\Theta(k)$. When the terminals are given as a sequence of points in the Euclidean plane, Alon and Azar [2] showed a lower bound of $\Omega(\log k/\log\log k)$ on the competitive ratio. For the so-called on-line *generalized* Steiner problem, in which pairs of terminals are requested sequentially and the algorithm must guarantee connectivity for every such requested pair, Berman and Coulston [5] proved a tight upper bound of $O(\log k)$, a result which improved the upper bound of $O(\log^2 k)$ due to Awerbuch *et al.* [4]. Both results apply to undirected graphs.

1.2 Preliminaries Given a directed graph of bounded asymmetry and an edge $e = (v, u) \in E$, it is always the case that its *antiparallel* edge $\overline{e} = (v, u)$ is always in E as well. Let $T = (r', V', E')$ be an arborescence rooted at r', we denote by $\hat{T}$ the graph (V', E''), with $E'' = E' \cup \{\overline{e} : e \in E'\}$. In words, $\hat{T}$ is the subgraph of G induced by the vertices of T, and induces all edges in T as well as all their antiparallel edges. For arborescence T and vertices v, u in T, we denote by $p_T(u, v)$ (resp. $p_{\hat{T}(u,v)}$) the simple directed path from u to v using exclusively edges in T (resp. $\hat{T}$). Note that such paths are uniquely defined (provided that $p_T(u, v)$ exists).

The cost of a directed path p is the total cost of all directed edges in p, and 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 $|K| = k$, with $OPT = c(T^*)$. 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' to the total cost of Greedy).

For convenience, we will be using the term "tree" to refer to a (rooted) arborescence.

2 Outline of the proof of Theorem 1.1

First, note that when $\alpha \in \Omega(k)$ the lower bound on the competitive ratio due to [11] is $\Omega(k)$, which is obviously tight (using a trivial upper bound for the greedy algorithm). Thus the problem is interesting only when $\alpha \in o(k)$. We will be assuming that α is integral since we can round α to the closest integer without affecting, asymptotically, the bounds. Let l be such that $\alpha^l = k$, which means that $l = \frac{\log k}{\log \alpha}$.

There are two main components in the proof. The first addresses the following question. Suppose that a subset $K' \subseteq K$ of $O(\alpha)$ terminals belongs in a (rooted) subtree T' of T^*. Suppose also that Greedy has been charged already the cost for serving a single terminal in K', but no other terminals in K' have arrived yet. Can we bound $c_{GR}(K')$ as a function of $c(T')$? Of course we can give trivial upper bounds: for every terminal $t \in K'$ (excluding the first terminal in K') we have $c_{GR}(t) = O(\alpha)c(T')$, hence $c_{GR}(K') = O(\alpha^2)c(T')$; even better we can use the fact that Greedy is log-competitive in undirected graphs, which implies that $c_{GR}(K') = O(\alpha \log |K'| c(T')) = O(\alpha \log \alpha \cdot c(T'))$. Note that it is not true that one can claim that $c_{GR}(K') = \alpha \cdot c(T')$; this would be true if the root of T' had already become part of the current tree Greedy builds (borrowing terminology from the literature on the Steiner problem in undirected graphs, Greedy should include appropriate *Steiner vertices* in the tree it builds). There is no easy way to guarantee this; in fact Theorem 1.2 shows this does not hold. However, we can still improve the bound to $O(\alpha \frac{\log \alpha}{\log\log \alpha} c(T'))$, as shown in Lemma 3.1.

The second component of the proof provides a framework for a recursive application of the previous observation. In particular, suppose that T^* can be partitioned into (roughly) α edge-disjoint trees $T_1, \ldots, T_\alpha$ of (roughly) the same number of terminals, namely α^{l-1}. Let v_i ($i \in [1, \alpha]$) denote the first terminal, among all terminals in T_i, to be requested; V_1 denote the set $\{v_i : i \in [1, \alpha]\}$ and $K_1, \ldots, K_\alpha$ denote the set of all remaining terminals in $T_1, \ldots, T_\alpha$ respectively. Using Lemma 3.1 we have that $c_{GR}(K) = c_{GR}(V_1) + \sum_{i=1}^{\alpha} c_{GR}(K_i) = O(\alpha \frac{\log \alpha}{\log\log \alpha} OPT) + \sum_{i=1}^{\alpha} c_{GR}(K_i)$. We then proceed recursively[2] at each subtree T_i. Note that trees T_i are edge-disjoint, hence at each level of the recursion the cost of Greedy increases by $O(\alpha \frac{\log \alpha}{\log\log \alpha})OPT$. Since there are roughly l levels of recursion, we derive the required upper bound on the cost of Greedy.

Naturally, the previous argument relies upon the ability to provide a decomposition of T^* into (roughly) α trees of (roughly) the same size. Moreover, the decomposition should be hierarchical, in the sense that we should be able to further decompose the resulting trees while upholding the above property. Lemma 3.6

---

[2] In the first level of the recursion, we could instead claim that $c_{GR}(V_1) = O(\alpha \cdot OPT)$, since we know that r is the root of T^*. However, this is true only for the first level, and for all subsequent levels we have to rely to Lemma 3.1.

proves the existence of such a "balanced", hierarchical decomposition of T^*.

3 Proof of Theorem 1.1

The following is a key Lemma in the analysis of GREEDY.

LEMMA 3.1. *Let* T' *be a subtree of* T^* *rooted at vertex* r' *and let* $K' \subseteq K$, *with* $|K'| = O(\alpha)$ *be a subset of* K *such that every terminal in* K' *is a vertex in* T'. *Let* w *denote the terminal which was requested the earliest among all terminals in* K'. *Then*[3] $c_{GR}(K') = c_{GR}(w) + O(\alpha \frac{\log \alpha}{\log\log \alpha})c(T')$.

In order to show Lemma 3.1, we will prove the lemma for the case in which T' and K' have a relatively simple structure (c.f. Lemma 3.2). The proof of Lemma 3.1 will then become substantially easier.

DEFINITION 3.1. *Let* T',K' *and* r' *be as defined in the statement of Lemma 3.1. We call the triplet* $\mathcal{C} = (T', K', r')$ *a* comb instance, *or* comb *for simplicity 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 directed paths* p_i *from* v_i *to* u_i. *No other edges are in* T'. *Finally the set* K' *is defined as the set of vertices* $\{u_1, \ldots, u_{k'}\}$. *We call* P *the* backbone *of* $\mathcal{C}$, *and the paths* p_i *the* terminal paths *of the comb. The vertex set of* $\mathcal{C}$ *is the set of vertices in* T'.

Figure 1 illustrates the structure of a comb. Note that the definition allows the paths p_i to be empty, in which case $v_i \equiv u_i$; in addition, the directed paths from v_i to v_{i-1} (as well as the path from r' to $v_{k'}$) may also be empty, in which case $v_i \equiv v_{i-1}$. For the proof of Lemma 3.2 we will assume, for simplicity and wlog, that such degeneracies do not arise.

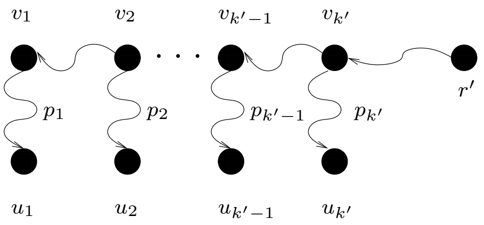

Figure 1: The structure of a comb instance. Wavy lines indicate directed paths between vertices.

We will make use of some auxiliary definitions. For terminal u_i in comb $\mathcal{C}$ we say that its *index* is i. We say that u_j *precedes* u_i in $\mathcal{C}$ (denoted by $u_j \prec u_i$) iff $j < i$. We say that u_j is *between* u_i and $u_{i'}$ iff $u_i \prec u_j \prec u_{i'}$. Given terminals u_i and u_j such that $u_i \prec u_j$ we call the path $p_{\hat{T}}(v_i, v_j)$ the *segment* between u_i and u_j.

The following lemma is a version of Lemma 3.1 in which T' and K' are restricted to form a comb.

LEMMA 3.2. *Given the comb* $\mathcal{C} = (T', K', r')$, *with* $|K'| = k' = O(\alpha)$, *let* $w \in K'$ *denote the terminal requested the earliest among all terminals in* K'. *Then* $c_{GR}(K') = c_{GR}(w) + O(\alpha \frac{\log \alpha}{\log\log \alpha})c(T')$.

Proof. Let π denote a permutation of $\{1, \ldots, k'\}$ such that $u_{\pi_1}, \ldots, u_{\pi_{k'}}$ is the sequence of the requests in K' in the order in which they are requested ($w = u_{\pi_1}$). Also let x be such that $x^x = |K'|$ which implies that $x = O(\frac{\log \alpha}{\log\log \alpha})$. For convenience we will assume that x is an integer.

In order to bound $c_{GR}(K' \setminus u_{\pi_1})$ we will 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}. We will show that

$$(3.1) \qquad C \stackrel{\text{def}}{=} \sum_{i=2}^{k'} c(q_i) = O(\alpha \frac{\log \alpha}{\log\log \alpha})c(T').$$

Since $c_{GR}(K' \setminus u_{\pi_1}) \leq \sum_{i=2}^{k'} c(q_i)$ the lemma will then follow.

In order to simplify the proof we will ignore the contribution to the cost C of all (directed) edges in the connection paths q_i's which belong in the tree T, and will only consider the contribution of edges in $\hat{T}$ but not in T. This is because the total contribution to C of edges in q_i which belong to T is at most $k'c(T) = O(\alpha c(T))$, which does not exceed asymptotically the bound we want to prove.

We first aim towards grouping together terminals as they are being requested; this will also facilitate our assignment of terminals to their mates. To this end we introduce the concept of a *run* and the concept of the *representative* of a run. The first terminal to be requested, namely u_{π_1} begins run 1. At the time u_{π_i} ($i \geq 2$) is requested, for every run j generated thus far, let $u_{h(j)}$ denote the terminal with the highest index in the comb among all terminals in run j requested so far. If $u_{h(1)} \prec u_{\pi_i}$ in the comb, then u_{π_i} becomes a member of run 1. Otherwise let π_l, π_r be the highest and lowest indices, respectively, with $l, r \leq i-1$, such that u_{π_i} is between u_{π_l} and u_{π_r} in the comb (if such a π_l does

---

[3] An alternative statement of the lemma is that $c_{GR}(K' \setminus \{w\}) = O(\alpha \frac{\log \alpha}{\log\log \alpha})c(T')$. In fact, in the proof of Theorem 1.1 we will use the latter statement. The same applies to the statement of Lemma 3.2.

not exist, then we set it to 0). If either i) $\pi_l = 0$; or ii) there is no run j with the property that $u_{\pi_l} \equiv u_{h(j)}$ **and** u_{π_r} is the representative of run j, then u_{π_i} starts a new run with the representative of the new run to be u_{π_r}. Otherwise, u_{π_i} becomes a member of the same run[4] as u_{π_l}, namely run j, and its *predecessor* in the run is u_{π_l}.

The above process produces a partition of K' into runs, and determines a representative of all runs other than run 1. We will be denoting by $rep(r)$ the representative of a run r; we will also be using the notation $rep(u_{\pi_i})$ to denote the representative of the run to which u_{π_i} is assigned. An example of the partition of K' into runs is given in the Appendix.

We introduce some additional definitions. Let $\sigma(i,j) \in \{1, \ldots k'\}$ be such that $u_{\sigma(i,j)}$ is the terminal of the j-th *lowest* index which belongs in run i. The *size of a run* is the number of terminals in the run. For a certain terminal u_{π_i}, denote by $s(u_{\pi_i})$ the segment between u_{π_i} and $rep(u_{\pi_i})$; we call $s(u_{\pi_i})$ the *segment of* u_{π_i}. Denote by $d(r)$ the cost of the segment of the first terminal in run r, namely $d(r) = c(s(u_{\sigma(r,1)}))$.

The following claim describes some properties related to a run.

Claim 3.1. *(i) Every terminal in a run other than run 1 precedes its representative in the comb. Furthermore, each terminal is the representative of at most one run. (ii) Suppose r, r' ($r \neq r'$) denote two runs such that there exists a terminal u in run r' such that $u_{\sigma(r,j)} \prec u \prec u_{\sigma(r,j+1)}$, i.e., u is between two consecutive terminals of run r. Then the whole run r' is contained between $u_{\sigma(r,j)}$ and $u_{\sigma(r,j+1)}$.*

Once the runs and the representatives have been determined, we can proceed with assigning mates to the terminals, using the following rules.

1. Terminal $u_{\sigma(1,j)}$ is always assigned $u_{\sigma(1,j-1)}$ as its mate, for $j \geq 2$.
2. The terminal of smallest index in run $i > 1$, i.e., $u_{\sigma(i,1)}$, is always assigned the representative of the run $rep(i)$ as its mate.
3. If the size of run $i > 1$ is at most x, then each terminal in the run i is assigned the representative of the run as its mate.
4. Otherwise (i.e., if the size of run $i > 1$ is larger than x), then
    (a) If $c(p_{\hat{T}}(v_{\sigma(i,j-1)}, v_{\sigma(i,j)})) \geq d(i)/x$ $(j \geq 2)$ then $u_{\sigma(i,j)}$ is assigned the representative of the run as its mate;
    (b) Otherwise, the mate of $u_{\sigma(i,j)}$ is set to be $u_{\sigma(i,j-1)}$.

Note that for every i, j $u_{\sigma(i,j)}$ is requested *after* $u_{\sigma(i,j-1)}$. Likewise, every terminal in any run (other than run 1) is requested *after* the representative of the run. This means that our assignment of terminals to mates is feasible, in the sense that the mate of a terminal is requested prior to the terminal itself. Recall that once terminals are assigned mates, the connection paths q_i's are uniquely determined.

Recall that C denotes the total cost due to the assignment of terminals to mates (see the definition in Eq (3.1)) via the connection paths q_i's. In order to bound C we observe that we can express C as the sum of two partial costs, which we call C_1 and C_2. Here, C_1 denotes the cost due to edges $\overline{e}$ such that e belongs in some terminal path p_i in T, and C_2 denotes the cost due to edges $\overline{e}$ such that e belongs in the backbone P of the comb. Indeed, our particular assignment of terminals to mates is motivated by the main objective to balance the contributions of C_1 and C_2 to the overall cost C. At an intuitive level, if a run other than run 1 has small size (at most x), then its representative can "afford" to act as the mate of all terminals in the run (see assignment rule 3), since this does not contribute much to the C_1 cost, and it definitely does not affect the C_2 cost. However, if the size of a run is large we must be more careful as assignment rule 4 suggests. For instance, we can no longer afford to assign all terminals in the run to the representative of the run as their mate: this would implode the C_1 cost. Instead, as long as a terminal in a run is "not too far away" with respect to the backbone cost (distance) from its predecessor in the run, we let the predecessor be its mate: the C_2 cost will not increase by much, in the sense that the average contribution of such terminals to C_2 will be kept low (rule 4b). Otherwise, i.e., when the terminal is indeed far away from its predecessor in the run, then we will choose the representative as the mate (rule 4a); since the later case will not arise too often for any given run (namely at most x times) the C_1 cost will be kept low. In the remainder of the proof we elaborate on this intuitive explanation.

First we show how to bound C_1. Denote by $C_{1,i}$ the contribution of the connection path q_i for terminal u_{π_i} to this cost which means that $C_1 = \sum_{i=2}^{k'} C_{1,i}$. Note that u_{π_i} contributes to C_1 only in two cases: i) If it is the mate of its successor in the run to which it belongs (rule 1, 4b). In this case, it contributes at most $\alpha c(p_i)$ to $C_{1,i}$; ii) If it is the mate of certain terminals in the run for which it is the representative (the remaining rules). Recall that from Claim 3.1 (i), u_{π_i} can be the

---

[4] Note that the representative of a run is not a member of the run.

representative of at most one run; let r denote this specific run. Then u_{π_i} can be the mate of $u_{\sigma(r,1)}$ (rule 2), as well as either at most x terminals in r (as follows when either rule 3 or 4a applies). The total contribution to $C_{1,i}$, in this case, is then bounded by $(x+1)\alpha c(p_i)$. Summarizing,

$$C_{1,i} \leq (x+2)\alpha \cdot c(p_i) = O(x\alpha \cdot c(p_i)). \tag{3.2}$$

Next define $C_{2,i}$ as the contribution of u_{π_i} to C_2. We say that u_{π_i} *contributes* the directed edge e, with $\overline{e} \in P$ when the path q_i includes e. For the remainder of this proof, we will call such edges *expensive*. Also, let $\overline{P}$ denote the directed path from v_1 to r in $\hat{T}$ and q_i' denote the intersection of $\overline{P}$ with q_i, namely the subpath of q_i which consists of expensive edges only, which means that $C_{2,i} = c(q_i')$. Let X denote the subset of K' which consists of terminals with non-zero contribution to C_2. We can think of the assignment of terminals in X to their mates as being done as the terminals in X are requested over time; more precisely, we can think of all edges in q_i' being "bought", as the connection path between the terminal and its mate is *established*, at the precise moment $u_{\pi_i} \in X$ is requested. In this view, every time an expensive edge in $\overline{P}$ is contributed due to such an assignment, we say that the *depth* of the edge increases by 1 (initially, i.e., before any terminals have been requested, all expensive edges have depth zero).

In addition, observe that u_{π_i} contributes to C_2 when it is assigned to its mate as a result of either rule 1 or rule 4b only. In other words, we are only considering cases in which $\overline{u}_{\pi_i}$ is the predecessor of u_{π_i} in its run.

Claim 3.2. *For a terminal $u_{\pi_i} \in X$, all expensive edges in q_i' have the same depth, right after q_i is established.*

Claim 3.2 asserts that it is meaningful to say that terminal u_{π_i} is of depth δ if right after it is assigned to its mate, and the connection path q_i is established, the depth of all expensive edges at the connection path becomes equal to δ. This implies that we can further partition X into sets $X_1, X_2 \ldots$ such that X_i consists of all terminals of depth i. Note that for all i with $u_{\pi_i} \in X_j$, the paths q_i' are edge-disjoint.

The following lemma shows that the contribution of a terminal to C_2 decreases exponentially with its depth.

Lemma 3.3. *For a terminal $u_{\pi_i} \in X_j$, $C_{2,i} \leq \frac{\alpha c(P)}{x^{j-1}}$.*

Proof. By induction on j. The claim is trivially true for $j = 1$ from the disjointness of all q_i''s for terminals in X_1. Suppose the claim holds for j, we will show it holds for $j+1$. Consider the set of terminals X_{j+1}. Recall that every terminal in X_{j+1} will buy expensive edges of current depth exactly j prior to the assignment of the said terminal to its mate, then right after the assignment the depth of such edges increases by one. Let u_{π_i} be a terminal in X_{j+1}, q_i its connection path, and $r > 1$ the run to which it belongs. We want to show that the whole run r is contained between two terminals $\overline{u}_{\pi_l}$ and u_{π_l} with the property that $u_{\pi_l} \in X_j$, and that in addition $\overline{u}_{\pi_l} \prec rep(r) \preceq u_{\pi_l}$ (here $\preceq$ denotes either precedence or equivalence).

We begin with a simple observation. Consider the set Q of paths of the form q_l' such that $u_{\pi_l} \in X_j$. As noted earlier, any two paths in Q are edge disjoint. We claim that q_i' is a subpath of one of the paths in Q. Note first that every edge $e \in q_i'$ must belong in some path $q \in Q$, since the depth of all edges in q_i' become $j+1$ once q_i' is established. Then we can argue that if q_i' was not a subpath of a path in Q, then there would exist a terminal $u' \in X_j$ other than $\overline{u}_{\pi_i}$ which is requested earlier than u_{π_i} and such that $\overline{u}_{\pi_i} \prec u' \prec u_{\pi_i}$ in the comb, a contradiction, since that would mean that $\overline{u}_{\pi_i}$ cannot be the predecessor of u_{π_i} in r.

We know, therefore, that there exists a terminal $u_{\pi_l} \in X_j$ for which q_i' is a subpath of q_l'. Since u_{π_l} is requested earlier than u_{π_i}, and $u_{\pi_i} \prec u_{\pi_l}$ in the comb, we have that $u_{\pi_i} \prec rep(r) \preceq u_{\pi_l}$, hence every terminal in r must precede u_{π_l}. In addition, if $r' \neq r$ denotes the run in which the (consecutive) members $\overline{u}_{\pi_l}$ and u_{π_l} belong, we know that there exists a member of run r, namely u_{π_i} which is between $\overline{u}_{\pi_l}$ and u_{π_l}. Claim 3.1, in particular statement (ii), will then guarantee that all elements in r are between $\overline{u}_{\pi_l}$ and u_{π_l}.

We thus derive that $C_{2,i} \leq d(r)/x \leq C_{2,l}/x$. The first inequality follows from the assignment of terminals to mates, when applying rule 4b (Note that rule 1 cannot apply since $u_{\pi_i} \in X_{j+1}$ has to belong to a run $r > 1$). The second inequality follows from the previously shown property concerning the containment of run r, including $rep(r)$, between $\overline{u}_{\pi_l}$ and u_{π_l}. By the induction hypothesis, $C_{2,l} \leq \frac{\alpha c(P)}{x^{j-1}}$ and the lemma follows. □

We now proceed to bound C_2. Denote by $c_2(X_j)$ the contribution of X_j to C_2. Recall that paths of the form q_l', for all $u_{\pi_l} \in X_j$ are edge-disjoint, for fixed j. Hence $c_2(X_j) \leq \alpha \cdot c(P)$. Combining this fact with Lemma 3.3 we have $c_2(X_j) = \min\{\alpha c(P), \frac{\alpha c(P)}{x^{j-1}}|X_j|\}$. Therefore,

$$C_2 = \sum_j c_2(X_j) = \sum_j \min\{\alpha c(P), \frac{\alpha c(P)}{x^{j-1}}|X_j|\} \tag{3.3}$$

Note that (3.3) is maximized when $|X_j| = x^{j-1}$, for all $j \geq 2$, which implies that

$$C_2 \in O(x\alpha c(P)) \tag{3.4}$$

We are now ready to conclude the proof of Lemma 3.2. Observe that $C = C_1 + C_2 = \sum_{i=1}^{k'} C_{1,i} + C_2 = O(x\alpha \sum_{i=1}^{k'} c(p_i)) + O(x\alpha c(P)) = O(x\alpha(\sum_{i=1}^{k'} c(p_i) + c(P))) = O(x\alpha c(T'))$, where the third equality follows from (3.2) and (3.4). Hence $C = O(\alpha \frac{\log \alpha}{\log\log \alpha} C(T'))$. □

We will now highlight how Lemma 3.2 can be used to prove Lemma 3.1.

Proof of Lemma 3.1. We will show how to partition K' into two collections of sets: The first collection, denoted by I, will consist of terminals which asymptotically do not affect much the overall cost of GREEDY; we will call such terminals *ignored*. The second collection will consist of a partition of $K' \setminus I$ into near-disjoint comb-instances. In particular, by near-disjoint we mean that any edge in T' will be shared by at most two comb instances.

The decomposition is determined by visiting terminals in K' in the order they are requested. Let σ' denote the sequence of terminals in K', and $\sigma'[i]$ the i-th requested terminal in K'. We initialize the set I as well as the collection of comb instances, denoted by $\mathcal{C}$, to empty sets. For any terminal $t \in \sigma'$ let q_t denote the path $p_{T'}(r', t)$, namely the path from r' to t in T' and $\overline{q}^t$ denote the path $p_{\hat{T}}(t, r')$, namely the path from t to r' which follows all edges antiparallel to edges in q_t.

The first terminal in σ', namely $\sigma'[1] = w$, induces a (trivial) comb instance of the form $\mathcal{C}_1 = (q_w, w, r')$, with backbone q_w, and a single, empty terminal path[5]. We say that we *assign* w to $\mathcal{C}_1$, and we add $\mathcal{C}_1$ to $\mathcal{C}$.

Consider now terminal $t = \sigma'[i]$ with $i > 1$. Focus on the sequence of vertices, in the order they are visited, in the directed path $\overline{q}_t$. Let v denote the first vertex in this sequence which belongs to the vertex set of some $\mathcal{C}_l \in \mathcal{C}$, with the convention that in the case where t itself is in the vertex set of $\mathcal{C}_l$, we consider v to be vertex t (if v belongs to more than one combs, then we choose any of such combs to be $\mathcal{C}_l$.) Note that the sequence of vertices must include r' which is in $\mathcal{C}_1$, so such a v always exists. We consider the following cases, and make appropriate decisions *in this order:*

- *Case 1.* v is a terminal $\sigma'[j]$ with $j < i$. In this case we add t to the set of ignored terminals I.

- *Case 2.* v is a vertex in the backbone of $\mathcal{C}_l$. In this case we assign t to $\mathcal{C}_l$ and we update $\mathcal{C}_l$ by adding the corresponding terminal path $p_{T'}(v, t)$ (possibly empty) to $\mathcal{C}_l$.

- *Case 3.* If none of the above happen, then it must be the case that v belongs in one of the terminal paths for $\mathcal{C}_l$. In particular, there must exist some $j < i$, such that terminal $s = \sigma'[j]$ is a terminal already assigned to $\mathcal{C}_l$, and v is a vertex in the terminal path corresponding to s in $\mathcal{C}_l$ which does not belong in the backbone of $\mathcal{C}_l$ and is not s either. Let s' denote the vertex of the terminal path for s in $\mathcal{C}_l$ which also belongs in the backbone of $\mathcal{C}_l$. In this particular case, we say that t *initiates* a new comb $\mathcal{C}_{\rho+1}$, where ρ is the highest current index of combs in the collection $\mathcal{C}$. More precisely, we create a new comb $\mathcal{C}_{\rho+1}$, rooted at s', with $p_{T'}(s', s)$ as its backbone, and the path $p_{T'}(v, t)$ as the terminal path for t. We assign t to $\mathcal{C}_{\rho+1}$; we also say that terminal s *pays for initiating* comb $\mathcal{C}_{l+1}$.

Figure 2 illustrates the application of the decomposition.

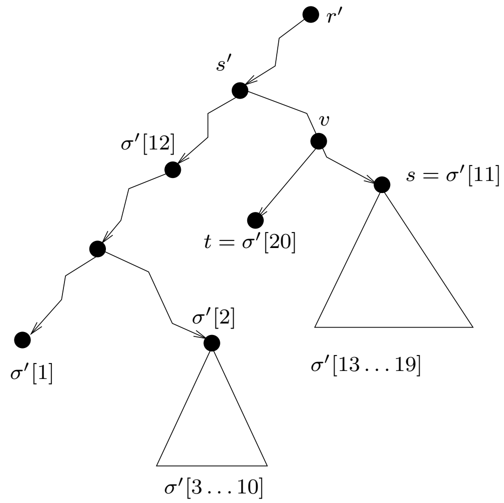

Figure 2: An example of the comb decomposition. Here, terminals $\sigma'[1]$, $\sigma'[2]$, $\sigma'[11]$ and $\sigma'[12]$ are assigned to $\mathcal{C}_1$. Terminals $\sigma'[3 \ldots 10]$ and $\sigma'[13, \ldots 19]$ which belong in the subtrees of $\sigma'[2]$ and $\sigma'[11]$, respectively, are all ignored terminals. Terminal $t = \sigma'[20]$ initiates $\mathcal{C}_2$, with terminal $s = \sigma'[11]$ paying for initiating $\mathcal{C}_2$.

Once the whole sequence σ' is processed as above, we are left with a set of ignored terminals I and a partition of terminals in $K' \setminus I$ into sets K'_i such that all terminals in K'_i are assigned to comb $\mathcal{C}_i$. Denote by t_i^1 the terminal which initiates comb $\mathcal{C}_i$, and s_i^1 the terminal which pays for initiating that comb. From the decomposition we have

(3.5)
$$c_{GR}(K') = c_{GR}(I) + \sum_i c_{GR}(t_i^1) + \sum_i c_{GR}(K'_i \setminus \{t_i^1\}).$$

---

[5] Recall that our definition of a comb instance allows combs in which the terminal paths are empty. This is one case where such an instance arises.

The following lemma concludes the proof (see [3] for details).

Lemma 3.4. *$c_{GR}(K')$, as expressed by (3.5) is bounded by $c_{GR}(w) + O(\alpha\frac{\log\alpha}{\log\log\alpha})c(T')$*

□

As outlined in Section 2, we aim towards applying Lemma 3.1 in a recursive manner. To this end,we now proceed to decompose the optimal arborescence T^* into a "balanced" family of arborescences. We are interested in decompositions which are edge-disjoint. Vertex-disjointness is not critical, however, in the event a terminal vertex belongs to more than one tree in the decomposition, we insist that the terminal itself is *assigned* to exactly one tree in the decomposition. Effectively, this will allow us to treat the trees as if they were "terminal-disjoint", even though the trees are not necessarily vertex-disjoint. This is a convention we make and its only purpose is to simplify the proofs.

Lemma 3.5. *Let T be a Steiner tree for a set of k terminals, and x a number such that $1 \le x \le k$. Then T can be decomposed in at most $\lceil\frac{k}{x}\rceil$ and at least $\frac{k}{2x}$ edge-disjoint arborescences, with each arborescence assigned between x and $2x$ terminals.*

Definition 3.2. *Let $D(T)$ denote the set of trees obtained by applying an edge-disjoint decomposition D to tree T with k terminals. We say that D is* balanced *for tree T if every tree in $D(T)$ is assigned between $\frac{k}{4\alpha}$ and $\frac{4k}{\alpha}$ terminals, and $\frac{\alpha}{4} \le |D(T)| \le 4\alpha$.*

In what follows, let $k(T')$ denote the number of terminals assigned to a tree T', as the result of a decomposition. Recall that K denotes the set of all terminal requests, with $|K| = k$. Let τ be such that $\alpha^\tau = k$, which means that $\tau = \frac{\log k}{\log\alpha}$.

Definition 3.3. *A* balanced *hierarchical decomposition H of* height *L of a Steiner tree T consists of L families of sets, denoted by $\mathcal{F}_1, \dots \mathcal{F}_L$, defined recursively as follows. Let D_1^T be a balanced decomposition for tree T. First, we define $\mathcal{F}_1$ to be the set $D_1^T(T)$, and every tree in $\mathcal{F}_1$ is said to be* at level 1. *We define $\mathcal{F}_{i+1}$ in terms of $\mathcal{F}_i$ as $\mathcal{F}_{i+1} = \{D_{i+1}^{T'}(T') | T' \in \mathcal{F}_i\}$, where $D_{i+1}^{T'}$ is a balanced tree decomposition for tree T' (with respect to the terminals assigned to T'). We also say that every tree in $\mathcal{F}_{i+1}$ is at level $i+1$. Finally, H must be such that every tree at level L is assigned at most $c\cdot\alpha$ terminals, for some constant c.*

Lemma 3.6. *For every tree Steiner tree T with k terminals there exists a balanced hierarchical decomposition H of T such that for every tree T' at level i, we have $\frac{1}{4}\alpha^{\tau-i} \le k(T') \le 4\alpha^{\tau-i}$, and the height of the decomposition is $L = O(\tau)$.*

Let H^* denote the balanced decomposition of T^* which satisfies the conditions of Lemma 3.6, and let $\mathcal{F}_1, \dots \mathcal{F}_L$ be its levels. In what follows we prove the main result of this section.

Proof of Theorem 1.1. Let σ denote the input sequence of terminals. Define the *depth* of a terminal $t = \sigma[i]$ in σ, as the smallest index j such that t is assigned to some tree $T \in \mathcal{F}_j$ **and** no terminal in $\sigma[1 \dots i-1]$ is assigned to T; we also say that t is *associated* with tree T. If such a j does not exist, then the depth of the terminal t is defined to be $L+1$, and t is associated with the (unique) tree of level L to which is is assigned in H^*. Note that every terminal is associated with a unique tree. We can then partition K into disjoint sets $S_1, \dots S_{L+1}$, where S_i denotes the set of terminals of depth i. Let $c_{GR}(S_i)$ denote the cost induced by Greedy on S_i.

Lemma 3.7. *$c_{GR}(S_i) = O(\alpha\frac{\log\alpha}{\log\log\alpha}OPT)$.*

Proof. Consider first S_1. There are at most 4α trees in $\mathcal{F}_1$, hence at most 4α terminals in S_1. The first terminal ($\sigma[1]$) incurs a cost of at most OPT. We can now invoke Lemma 3.1 to get that $c_{GR}(S_1) = O(\alpha\frac{\log\alpha}{\log\log\alpha}OPT)$.

In general, consider set S_i, with $2 \le i \le L$. Recall that $\mathcal{F}_i$ is derived by applying a balanced decomposition to each tree in $\mathcal{F}_{i-1}$, namely $\mathcal{F}_i = \bigcup_{T\in\mathcal{F}_{i-1}} D_i^T(T)$. For each $T \in \mathcal{F}_{i-1}$ there are $|D_i^T(T)|$ trees in $\mathcal{F}_i$, and each such tree is assigned at most one terminal in S_i. Denote by S_i^T the subset of S_i which consists of terminals assigned to trees in $D_i^T(T)$. Since D_i^T is a balanced decomposition for T by Definition 3.2 we know that $|D_i^T(T)| \le 4\alpha$, hence $|S_i^T| \le 4\alpha$. In addition, by the definition of the depth of a terminal, we know that there is one terminal in S_j with $j \le i-1$, say terminal w, which is assigned to T and has been requested earlier than all terminals in S_i^T (if this was not the case, one terminal in S_i^T would have depth smaller than i, a contradiction). In addition, the contribution of w to the cost of Greedy has already been taken into account when bounding $c_{GR}(S_j)$. This means that we can apply Lemma 3.1 for the tree T, the set of terminals S_i^T and w as a terminal requested earlier than all terminals in S_i^T, and get that

$$c_{GR}(S_i^T) = O(\alpha\frac{\log\alpha}{\log\log\alpha}c(T)),$$

Finally, since every terminal in S_i is associated with a tree of level i, note that $S_i = \bigcup_{T\in\mathcal{F}_{i-1}} S_i^T$. In addition,

$\mathcal{F}_{i-1}$ consists of edge-disjoint trees, therefore $c_{GR}(S_i) = \sum_{T\in\mathcal{F}_{i-1}} c_{GR}(S_i^T) = O(\alpha\frac{\log\alpha}{\log\log\alpha}\sum_{T\in\mathcal{F}_{i-1}} c(T))$, which is $O(\alpha\frac{\log\alpha}{\log\log\alpha}c(T^*))$. For the case $i=L+1$ the bound on $c_{GR}(S_{L+1})$ follows along the same lines and the fact that from the definition of H^*, $O(\alpha)$ terminals are associated with each tree at level $L+1$. □

To conclude the proof of Theorem 1.1 , we have that the total cost of GREEDY is bounded as follows:

$$\begin{aligned} c_{GR}(K) &= \sum_{i=1}^{L+1} c_{GR}(S_i) = (L+1)\cdot O(\alpha\frac{\log\alpha}{\log\log\alpha}OPT) \\ &= O(\tau)O(\alpha\frac{\log\alpha}{\log\log\alpha}OPT) \\ &= O(\alpha\frac{\log k}{\log\log\alpha}OPT) \end{aligned}$$

where the second and third equalities follow from Lemma 3.7 and Lemma 3.6, respectively.

Hence $c_{GR}(K) = O(\min\{k, \alpha\frac{\log k}{\log\log\alpha}\})OPT$. □

4 Outline of the proof of Theorem 1.2

Consider the graph G illustrated in Figure 3: 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. In particular, each upwards edge has cost α times the cost of its antiparallel upwards edge. Let P denote the directed path from r to t, whose total cost is defined to be equal to 1.

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 wlog 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 in 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 $r \rightarrow v_1$ and $v_x \rightarrow t$). 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 $v_i \rightarrow v_{i+1}$. In addition, we insert x vertices of the form u_i^l, with $l \in [1, x]$ 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 $\Theta(k)$.[6]

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).

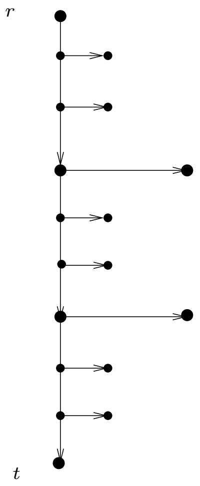

Figure 3: The structure of graph G, for the case $x = 2$. Only "cheap" edges are shown while "expensive" antiparallel edges are omitted

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 (ie 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 (since there is a single path connecting r to t in G) and enforcing this requirement requires a much more complicated adversarial input graph. Indeed, much of the details in the formal proof of Theorem 1.2 is dedicated to addressing this issue (a formal definition of the actual adversarial graph, and a detailed analysis can be found in the full version of the paper [3]).

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

---

[6] Note the similarity between this construction and the concepts of the *comb* and the *run* which we used in the proof of the upper bound. In fact, their definition was motivated by this adversarial construction, and vice versa.

"lower" vertex which was requested before u. In the second and third cases, a cost (roughly) 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 $\Omega(k_j)$ for round 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 optimal algorithm will buy the path P and all edges of the form (v, u), for a total cost which can be shown that is bounded by a constant.

Acknowledgements. I would like to thank Michalis Faloutsos for early discussions on the problem, as well as Alejandro López-Ortiz and Reza Dorrigiv for their helpful comments.

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 and Y. Azar. On-line Steiner trees in the euclidean plane. *Discrete and Computational Geometry*, 10:113–121, 1993.

[3] S. Angelopoulos. Improved bounds for the online Steiner tree problem in graphs of bounded edge-asymmetry. Technical Report CS-2006-36, David R. Cheriton School of Computer Science, University of Waterloo, 2006.

[4] B. Awerbuch, Y. Azar, and Y. Bartal. On-line generalized Steiner problem. In *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 68–74, 1996.

[5] 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.

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

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

[8] 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.

[9] K. Claffy, G. Polyzos, and H.W. Braun. Traffic characteristics of the T1 NSFNET backbone. In *Proceedings of INFOCOM*, 1993.

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

[11] 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.

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

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

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

[15] G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 770–779, 2000.

[16] A. Segev. The node-weighted Steiner tree problem. *Networks*, (17):1–17, 1987.

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

Appendix

Consider the comb instance of Figure 4, with K' consisting of 12 terminals (for simplicity, we represent terminal u_i, with $1 \leq i \geq 12$, by its index i). Suppose the terminals are given in the order

3,7,1,11,4,6,5,2,8,12,10,9

The decomposition is then as follows:

run	representative	terminals
1	–	3,7,11,12
2	3	1,2
3	7	4,6
4	6	5
5	11	8,10
6	10	9

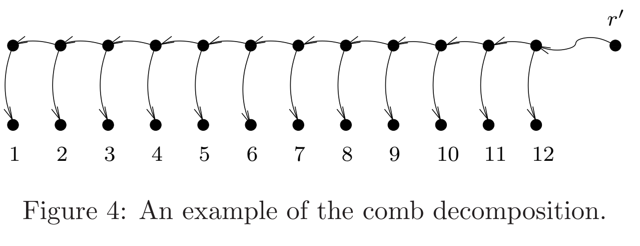

Figure 4: An example of the comb decomposition.