

# The Node-weighted Steiner Problem in Graphs of Restricted Node Weights

Spyros Angelopoulos

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

**Abstract.** In this paper we study a variant of the Node-Weighted Steiner Tree problem in which the weights (costs) of vertices are restricted, in the sense that the ratio of the maximum node weight to the minimum node weight is bounded by a quantity  $\alpha$ . This problem has applications in multicast routing where the cost of participating routers must be taken into consideration and the network is relatively homogenous in terms of the cost of the routers.

We consider both online and offline versions of the problem. For the offline version we show an upper bound of  $O(\min\{\log \alpha, \log k\})$  on the approximation ratio of deterministic algorithms (where  $k$  is the number of terminals). We also prove that the bound is tight unless  $P = NP$ . For the online version we show a tight bound of  $\Theta(\max\{\min\{\alpha, k\}, \log k\})$ , which applies to both deterministic and randomized algorithms. We also show how to apply (and extend to node-weighted graphs) recent work of Alon *et al.* so as to obtain a randomized online algorithm with competitive ratio  $O(\log m \log k)$ , where  $m$  is the number of the edges in the graph, independently of the value of  $\alpha$ . All our bounds also hold for the Generalized Node-Weighted Steiner Problem, in which only connectivity between pairs of vertices must be guaranteed.

## 1 Introduction

### 1.1 Problem Definition

The *Node-Weighted Steiner Tree* problem (NWS) is defined as follows. Given an undirected graph  $G = (V, E)$  ( $|V| = n, |E| = m$ ) with a cost function  $c$  on the edges and vertices, 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 and vertices. The *Generalized Node-Weighted Steiner* problem (GNWS) is defined along the same lines, with the exception that instead of a set of terminals, we are given a requirements function  $r : V \times V \rightarrow \{0, 1\}$ . The objective is to find a minimum-cost subgraph of  $G$  which provides connectivity for all pairs of vertices in the set  $K = \{(v, u) \text{ with } r(v, u) = 1\}$ . For uniformity, we call  $K$  the set of *terminal pairs*. Clearly, GNWS generalizes NWS.

Depending on whether  $K$  is known in advance, we distinguish between the offline and online versions of the problem. In the latter version, every time a new *request* appears (i.e., a terminal, or a pair of terminals) the algorithm must guarantee connectivity for the new request by buying irrevocably, if necessary, certain edges and vertices.

Both the offline and online version of this problem are variants of the well-known Steiner Tree Problem and Generalized Steiner Problem (GSP) which have been studied extensively in the literature (c.f. section 1.2 for some representative results on this problem).

In this paper we are interested in the variant of node-weighted Steiner problems in which some restriction is placed on the node weights. Specifically, let  $\alpha$  denote the quantity  $\max_{v,u \in V} c(v)/c(u)$  (we assume non-negative edge and vertex costs). We call  $\alpha$  the *asymmetry* of the graph, since it provides an indication of the variance of the vertex costs. Our aim is to provide upper and lower bounds for both the approximation ratio and the competitive ratio as functions of  $\alpha$ .

The (classic) Steiner problem has wide applications in multicast routing and network design (see e.g., [7]). On the other hand, NWS captures situations where the cost considerations include not only network links but also network nodes (e.g., routers), and the cost of the node simply reflects how much we must pay to have it included in the connectivity network. A network of relatively homogenous routers, in terms of their cost, is then modelled by a graph of small asymmetry. Our setting is largely motivated by the work of Faloutsos *et al.* [8]; in their work, the asymmetry is defined in the context of a directed graph, with only an edge-cost function, as the maximum ratio of the costs of the two directed edges between any two vertices in the graph, and reflects network homogeneity in terms of the edge costs of antiparallel links.

**Summary of our results.** We first show that unless  $P=NP$ , offline NWS cannot be approximated within a factor better than  $O(\min\{\log \alpha, \log k\})$  (Theorem 1). The proof uses ideas from Berman’s reduction (see section 1.2), however we reduce from Set Cover of Bounded Set Size; the result then follows from Trevisan’s inapproximability result [19]. Theorem 2 shows that the hardness bound is asymptotically tight, by presenting an algorithm which is a combination of Ravi and Klein’s algorithm for GNWS and the constant-factor approximation algorithm of Goemans and Williamson for GSP. In Theorem 3 we show an asymptotically tight bound of  $\Theta(\max\{\min\{\alpha, k\}, \log k\})$  on the competitive ratio of (deterministic or randomized) online algorithms. Last, section 3.1 builds upon ideas found in the work of Alon *et al.* [3] which provides a general framework applicable to several online network optimization problems with edge cost functions. We show how a key lemma in their work can be extended to node-weighted problems which translates to a  $O(\log m \log k)$ -competitive randomized algorithm regardless of the asymmetry of the input graph. The bound is almost tight since the lower bound of  $\Omega(\frac{\log m \log k}{\log \log m + \log \log k})$  shown for online Set Cover in [2] applies to online NWS as well.

## 1.2 Related Work

Berman (see ref. in [14]) showed an approximation-preserving reduction from Set Cover to NWS, which implies, using results of [9] and [15], that NWS cannot be approximated within a factor of  $o(\log k)$  unless  $P=NP$ . We emphasize that the hardness result holds only when  $\alpha$  is unbounded, in particular, it is required that  $\alpha$  is as big as  $\Omega(n)$ . Klein and Ravi [14] presented an asymptotically optimal algorithm which guarantees an approximation ratio of  $2 \ln k$ . Guha and Khuller [11] improved the upper bound to  $1.35 \ln k$  (approximately).

Concerning the classic Steiner Tree problem (where  $w(v) = 0$ , for all  $v \in V$ ), Karp [13] showed very early that it is NP-hard, and is in fact APX-hard [5] [18]. Currently, the best upper bound for general graphs is 1.55 and is due to Robins and Zelikovsky [16]. For the Generalized Steiner Problem, the corresponding bound is 2 [10],[1]. In terms of online algorithms, Imase and Waxman [12] showed a tight bound of  $\Theta(\log k)$  on the competitive ratio of online Steiner Tree, a result which Berman and Coulston [4] showed extends to the online GSP.

Faloutsos *et al.* [8] considered a somewhat related problem to ours, namely the online Steiner tree problem on directed graphs of bounded edge asymmetry. They showed that a simple greedy algorithm is  $O(\min\{k, \beta \log k\})$  competitive, and they also proved a lower bound on the competitive ratio of every deterministic algorithm (which one can show extends to randomized algorithms) of  $\Omega(\min\{k, \beta \log k / \log \beta\})$ . Here  $\beta$  denotes the edge asymmetry, as defined earlier in this section. Their results are related to this work not only because they provide some motivation for our definition of node asymmetry, but also because NWS can be reduced to Steiner tree in directed graphs, as shown by Segev [17]. Note however, that applying this reduction by itself only cannot yield good bounds: in particular for offline NWS with asymmetry  $\alpha$ , the reduction creates a directed graph of edge asymmetry  $\beta$  which can be as high as  $\alpha$  (depending on the edge costs) and for such graphs it is not known how to approximate the Steiner Tree problem to a  $o(\beta)$  factor (note that an obvious upper bound on the approximation ratio using this technique is  $O(\beta)$ ).

## 1.3 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 a cost, such that for every edge  $e = (v, u) \in E$ ,  $e' = (v, u) \in E'$ , and  $c(e') = c(e) + c(v)$ . 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 NWS in  $G_p$  can be transformed in polynomial time to a solution to NWS in  $G$  without any increase to the solution's cost, and without affecting the approximation ratio (see, e.g., Theorem 3.2 in [20] 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). We can also use the following property, which is a folklore result for the classic Steiner Tree problem but applies to the node-weighted version as well: Let  $(G_p, K)$  be the input to NWS, then we can transform any solution (Steiner tree)  $T$  to a tree  $T'$  which has total cost at most that of  $T$ , and for which the number of Steiner nodes (i.e., vertices which are not terminals) is at most  $|K| - 2$ . This follows from the observation that we can assume that all Steiner nodes have degree at least three, since Steiner nodes of degree two can be replaced by a single edge in the path completion of  $G$ .

Throughout this paper we assume that  $\alpha$  is an integer, since we can always scale it to the nearest integer without affecting, asymptotically, the bounds. We assume that  $G$  is connected, since otherwise we can restrict the problem to the connected components of  $G$ . We will denote by  $c_{min}, c_{max}$  the minimum and maximum costs of vertices in  $V$ , respectively. Because  $G$  is part of the input in both the online and offline versions of the problem, we assume that  $\alpha$  is known to the algorithm. Last, we note that all our lower bounds will be presented in terms of NWS, while all our algorithms will be described in the context of the more general GNWS problem.

## 2 Offline Algorithms

**Theorem 1.** *Any polynomial-time algorithm for NWS in graphs of asymmetry  $\alpha$  has approximation ratio  $\Omega(\min\{\log k, \log \alpha\})$ , unless  $P = NP$ .*

*Proof.* We present a polynomial-time reduction of a variation of the set cover problem in which all sets have bounded size, to NWS of bounded asymmetry. Consider an instance  $I$  of set cover which consists of a universe of elements, denoted by  $U$  and a collection  $\mathcal{C}$  of sets, each containing *at most*  $B$  elements in  $U$ . The objective 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  $e \in S$ . 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 and each vertex  $v_s$  has weight  $B$ . 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  $k = |K| = |U|$ ). Note that this transformation gives rise to an instance  $I'$  of NWS in a graph of asymmetry  $B$ .

Denote by  $OPT(I')$ ,  $A'(I')$  the cost of the optimal algorithm and the cost of an approximation algorithm  $A'$  for NWS 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,

$$A'(I') = k + B \cdot |C'| \quad \text{and} \quad A(I) = |C| = |C'|. \quad (1)$$

Likewise, we have

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

where  $OPT(I)$  is the optimal cost for the instance  $I$  of set cover. Since both  $C$  and  $C_{opt}$  are solutions for  $I$ , we have

$$B|C'_{opt}| = B|C_{opt}| \geq k \quad \text{and} \quad B|C'| = B|C| \geq k \quad (3)$$

Suppose that NWS is approximable within a factor of  $o(\log \alpha) = o(\log B)$ , then using (2) and (3) we get

$$\begin{aligned} A'(I') &= o(\log B)OPT(I') = o(\log B)(k + B \cdot |C'_{opt}|) \\ &= o(\log B)B \cdot |C'_{opt}|, \end{aligned}$$

hence from (1) and (2) we get that  $|C| = o(\log B)|C_{opt}|$ , which means that bounded-size set cover is approximable within a factor of  $o(\log B)$ , which implies that  $P=NP$ , by the inapproximability result of Trevisan [19], namely that set cover on instances with sets of size at most  $B$  is hard to approximate within a factor of  $\ln B - O(\ln \ln B)$  unless  $P = NP$ . Since  $B \leq k$  the result follows.  $\square$

We now present an algorithm which has approximation ratio  $O(\min(\log k, \log \alpha))$ , thereby matching the lower bound of Theorem 1. We will assume that  $k > \alpha$ , since otherwise the lower bound of Theorem 1 is already known to be tight. The algorithm is a combination of the Klein-Ravi (KR) and Goemans-Williamson (GW) algorithms. The former works in iterations; in each iteration, a subset of currently *active* trees is merged into a single tree (i.e., a single connected component) by buying a vertex of smallest cost-efficiency, and minimum-cost paths from the vertex to the active trees in question. Here the term “active” reflects the fact that the tree contains terminals for which the requirement function is not satisfied by the current partial solution. More formally, a tree  $T$  is active iff there exist vertices  $u, v \in V$ , with  $u \in T$ ,  $v \notin T$  such that  $r(v, u) = 1$ . In addition, the cost-efficiency of a vertex  $v$  is defined as the minimum ratio, over all subsets of active trees, of the cost of  $v$  as well as the cost of the minimum-cost paths from  $v$  to each active tree in the subset, over the cardinality of the subset. In other words, the cost-efficiency is the minimum average cost paid so as to connect active subtrees by means of paths originating at  $v$ .

We emphasize that for the GSP instance  $(G'_p, r')$  we treat the graph as if the weights of the vertices do not exist, and that  $G'_p$  is the path completion of  $G'$ .

**Algorithm Offline GNWS** on input  $(G, r)$

- 1 Execute the KR algorithm until at most  $k/\alpha$  active trees remain
- 2 Create a new graph  $G'$  by contracting each tree in the partial solution to a single “supernode”. Each supernode gets vertex weight zero.
- 3 Define a new requirements function  $r'$  on  $G'$  and solve GSP on instance  $(G'_p, r')$  using the GW algorithm

The requirement function  $r'$  is defined in the “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  $r'(u, v) = 1$  if and only if there exist vertices  $\tilde{u}, \tilde{v} \in G$  which belong in trees  $T_1, T_2$  in  $F$ , respectively, with  $T_1 \neq T_2$ , such that  $r(\tilde{u}, \tilde{v}) = 1$ . 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 instance  $(G, r)$  of GNWS.

**Theorem 2.** *Algorithm Offline GNWS has approximation ratio  $O(\min(\log k, \log \alpha))$ .*

*Proof.* Consider the penultimate iteration of the KR algorithm in step 1. Since at least  $k/\alpha$  active trees remain, the cost of the partial solution maintained up to that iteration, i.e edges and vertices bought by KR excluding terminals<sup>1</sup> 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 [14], in particular the inequality following (4)).

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  $OPT$ .

It remains to bound the cost due to step 3. Denote by  $F = \{T_1, \dots, T_i\}$  the forest of trees returned by Offline GNWS. First, note that the cost of edges in the optimal solution to the GSP problem on instance  $(G'_p, r')$  is bounded by  $OPT$ . Since GW is a 2-approximation algorithm for GSP, the cost of the edges in the forest  $F$  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). By the argument in section 1.3 we can

---

<sup>1</sup> The analysis of the KR algorithm in [14] makes wlog the assumption that all terminal costs are zero. For this purpose our analysis for step 1 of Offline GNWS does not take into account the cost of terminals; instead we simply add their contribution later in the proof.

assume that 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

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

Therefore the total node-weight of the forest  $F$  is at most  $2OPT$ , and its total weight at most  $4OPT$ .

Putting everything together, the cost of the solution returned by the algorithm is  $O(\log \alpha)OPT$ .  $\square$

We note that the above algorithm is applicable, with the same performance guarantees, to a wider class of network design problems which can be formulated as cut-covering problems in which the family of cuts is defined by *proper* functions (see e.g., [10]). This follows from the following two facts: i) for such problems the KR algorithm upholds the properties we used in the proof of Theorem 2 ; and ii) the GW algorithm is still 2-approximation for the edge-weighted version of this problem.

### 3 Online Algorithms

**Theorem 3.** *The competitive ratio of deterministic online GNWS is  $\Theta(\max\{\min\{\alpha, k\}, \log k\})$ .*

*Proof.* For the lower bound, we consider the online NWS 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 [12]. Otherwise the adversary presents a graph  $G$  which is defined as follows: The vertex set of  $G$  consists of (disjoint) sets  $V_1$  and  $V_2$ , with  $|V_2| = k(k+1)/2$ . Partition  $V_2$  into  $\binom{k(k+1)}{2}$  sets of size  $k$  each, and for each set define a vertex which is adjacent to those  $k$  vertices: this gives rise to the set  $V_1$ . No more vertices or edges exist in  $G$ . The weight of each vertex in  $V_1$  is  $\alpha$ , 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  $V_1$  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 game. Clearly, the game can go on for  $k - 1$  rounds. At the end of the game,  $k$  terminals have been presented all of which are adjacent to a vertex in  $V_1$ , hence  $OPT = \alpha + k$ . On the other hand, the algorithm has bought  $k - 1$  vertices in  $V_1$ , hence its cost is at least  $\alpha(k - 1)$ , and in this case the competitive ratio is  $\Omega(\min\{\alpha, k\})$ .

For the upper bound, we propose an algorithm, denoted by  $A$ , which works in two phases. Let  $\bar{k}$  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  $\alpha > \bar{k}$ : namely, the algorithm connects the two terminals in the pair by means of a minimum-cost path. The second phase begins at the point where  $\bar{k}$  exceeds  $\alpha$ , at which point the algorithm switches to the Berman and Coulston [4] (BC) algorithm for (edge-weighted) GSP. 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). We remind the reader that once again, we work on the path completion graph.

The total cost due to the first phase is clearly  $\min\{\alpha, k\}OPT$ . For the second phase, since the BC algorithm is  $O(\log k)$ -competitive for GSP, it follows that the cost of the edges it buys is  $O(\log k)OPT$ . Observe also that since BC returns a forest of at most  $k$  trees, from the discussion in section 1.3 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  $C = 3k \cdot c_{max} \leq 3\alpha k \cdot c_{min}$ . However, the latter contribution to the cost is in effect only when  $k \geq \alpha$ . Given that in such case  $OPT \geq kc_{min} \geq \alpha c_{min}$ , we have that  $C \leq 3\alpha \cdot OPT = 3 \min\{\alpha, k\} \cdot OPT$ . Therefore, the total cost of A is

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

*Note:* The lower bound of Theorem 3 can be extended to randomized algorithms, using a somewhat larger graph as part of the adversarial input. We sketch the proof (and omit certain details). Again, the adversarial input consists of a graph on vertices  $V_1 \cup V_2$ ,  $|V_1| = \binom{|V_2|}{k}$  and for every set  $S$  of  $k$  vertices in  $V_2$  there is exactly one vertex in  $V_1$  adjacent to  $S$  (and only  $S$ ). We require that  $|V_2| \geq 2k^2$ . The edge and vertex weights remain the same. We define a probability distribution  $D$  on the sequence of terminals presented to any fixed deterministic online algorithm; from Yao's principle [21], 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 subset of  $V_2$  of cardinality  $k$ , in any order. It follows that every time the deterministic algorithm considers a terminal (except for the very first one), the probability it has already bought a vertex in  $V_1$  which can guarantee connectivity 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 bounded by  $(1 - 1/k)^{k-1}$ , thus the average cost of the algorithm is  $\Omega(\alpha k)$ , while the average optimal cost is still  $\alpha + k$ .

### 3.1 Randomized online algorithms for the general case (unbounded asymmetry)

Theorem 3 suggests that in the case of large asymmetry, namely when  $\alpha \in \Omega(k)$ , the competitive ratio of any deterministic or randomized algorithm is disappointingly bad. However, the lower bound requires a construction of a graph with a

large number of vertices. In fact, in this section we will show how to achieve a better upper bound when the number of vertices in the graph is subexponential on the total number of terminals. In particular, we present and analyze a randomized algorithm based on the framework of Alon *et al.* [3], which addresses broad classes of (edge-weighted) connectivity and cut problems<sup>2</sup>. We show how their approach can be extended to a variant of such problems in which nodes as well as edges are associated with a cost function. In this section we omit proofs due to space limitations, and only focus on how to adapt/modify the ideas of [3].

We will create an online algorithm for GNWS which consists of two distinct components. The first component maintains a fractional solution to the problem, i.e., a feasible weight assignment  $w$  for nodes and edges in the graph. Here, a feasible assignment is such that for any request, i.e., a pair of terminals  $(t_i, t_j)$  there is a flow from  $t_i$  to  $t_j$  of value at least 1, assuming that we treat the weights of both edges and vertices as capacities<sup>3</sup>. Note that nodes, and not only edges, are assigned capacities, in the sense that in any feasible flow the in-flow for any node cannot exceed the capacity (weight) of the node. When the algorithm receives a new request, the algorithm will update the weight assignments, by performing an appropriate *weight augmentation*. In particular, the algorithm may increase the weights of certain vertices and/or edges, but it will certainly not decrease any of the currently assigned weights. This guarantees that after each request is processed, a feasible fractional solution can be maintained; moreover this computation is accomplished in an online fashion. Naturally, one seeks a fractional solution of small cost (c.f., Lemma 2).

The second component of the algorithm is responsible for rounding the fractional solution to an integral one, and is performed during each step (i.e., after each request appears). This component is largely orthogonal to the first one, and as demonstrated in [3], randomized rounding can be applied successfully in a variety of connectivity/cut problems. We show how randomized rounding can yield an integral solution of cost within a factor of  $O(\log k)$  from the cost of the fractional solution maintained by the first component of the algorithm (c.f., Lemma 3).

We start by describing the first component of the algorithm. Following [3], we can assume that all edges and vertices have costs in the interval  $[1, m^2]$  (here we are also using the assumption that the graph is connected and hence  $m \geq n - 1$ ). Initially the algorithm gives each edge and vertex a fractional weight of  $1/m^3$ . Suppose that a new request  $(t_1, t_2)$  arrives; the algorithm will then update the weight assignment (potentially increasing certain weights, but not decreasing any of them). More specifically, if the maximum flow from  $t_1$  to  $t_2$  is at least 1, we do nothing; otherwise we perform a weight augmentation step as follows. Find a minimum-weight  $t_1 - t_2$  cut  $C$ , defined as a partition of  $V$  in sets  $S$  and

---

<sup>2</sup> [3] is focused on edge-weighted graphs, and the authors claim that the techniques can be extended to the vertex counterparts of the problems in which only vertices have costs. Here, we extend their major result to the case where both vertices and edges are associated with costs.

<sup>3</sup> The weights  $w$  should not be confused with the costs  $c$  of edges and vertices.

$V \setminus S$  which separates  $t_1$  and  $t_2$ . For clarity, we emphasize that the weight of  $C$  is the sum of weights of edges in the cut. Denote by  $S_1$  and  $S_2$  the subsets of  $V$  and  $V \setminus S$ , respectively, for which each vertex in  $S_1$  (resp.  $S_2$ ) is incident with at least one edge in  $C$ . For every edge  $e = (v, u) \in C$  let  $\delta_e = w(e) / \max\{c(e), c(v), c(u)\}$ . We increase  $w(e)$  by  $\delta(e)$ . In addition, for every vertex  $v \in S_1$  (resp.  $u \in S_2$ ) we increase  $w(v)$  (resp.  $w(u)$ ) by  $\sum_{e=(v,u):e \in C} \delta_e$  (resp.  $\sum_{e=(v,u):e \in C} \delta_e$ ). We repeat the process, with a new weight augmentation step, until the maximum flow from  $t_1$  to  $t_2$  is at least 1. At an intuitive level, the augmentation is “balanced” in the sense that we increase the weights of certain vertices only as much as it is needed, and a vertex weight is increased proportionally to the weight increase of incident edges in the cut.

Note that at the end of the weight augmentation step associated with  $C$  the total edge-weight of  $C$  does not exceed the total vertex-weight of either  $S_1$  or  $S_2$ . This guarantees that when the algorithm terminates, all connectivity demands are satisfied.

**Lemma 1.** *The number of weight augmentation steps performed by the algorithm is  $O(\log m) \cdot OPT_f$ , where  $OPT_f$  denotes the cost of the optimal fractional solution.*

**Lemma 2.** *The algorithm is  $O(\log m)$  competitive for fractional node-weighted connectivity problems.*

The second part of the algorithm involves rounding the feasible fractional solution (which is maintained at each step) in an online fashion. Recall that when a request  $(t_1, t_2)$  appears we first compute a fractional,  $O(\log m)$ -competitive solution using the algorithm described earlier. The rounding method is simple and is based on the rounding employed in the context of the multicast problem in [3]. More specifically, the rounding method involves keeping  $2 \lceil \log(k' + 1) \rceil$  independent random variables  $X_i$  distributed uniformly in the interval  $[0, 1]$ . 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  $\theta$  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  $e$  and vertices  $v$  of the graph with the property that  $w(e) \geq \theta$  and  $w(v) \geq \theta$ , i.e., all vertices and edges whose weight exceeds the current value of the threshold (initially  $I = \emptyset$ ).

The following Lemma follows easily using the ideas of Lemma 4.1 in [3]. The only change in the proof is that we have to account for the cost of the vertices added in the solution (and not only the edges), but essentially the proof remains the same.

**Lemma 3.** *Throughout the execution of the algorithm, the expected cost of the solution which the algorithm maintains is  $O(\log k' \log m OPT)$ , where  $OPT$  is the cost of the optimal integral solution so far. Furthermore, for any terminal pair  $T = (t_1, t_2)$  the probability that the algorithm does not allocate a path that connects  $t_1$  and  $t_2$  is small, namely  $1/k'^2$ .*

Lemma 2 and Lemma 3 imply that the randomized algorithm is  $O(\log k \log m)$ -competitive (for integral solutions), with the condition that if the algorithm fails to allocate a path, a path has to be bought explicitly (since this happens infrequently, the overhead to the total cost is negligible).

## 4 Concluding Remarks

In this paper we studied the Steiner Tree Problem and Generalized Steiner Problem at the presence of node and edge weights, and presented upper and lower bounds on the performance of online and offline algorithms as a function of the node asymmetry  $\alpha$ . Of course, as one may suggest, there are several possible ways to capture the variation of node weights in the graph; for instance, one may instead define  $\alpha$  as  $\max_{v,u \in V} \{|c(v) - c(u)|\}$ . Or rather we could insist that the cost of a vertex  $v$  is within a factor of at most  $\alpha$  from the cost of vertices adjacent to it, as well as edges incident with it. The latter definition takes into account, at least in a certain limited way, the cost of edges and is not biased towards the cost of nodes. It is easy to show that the upper and lower bounds shown in sections 2 and 3 carry over to the above definitions of asymmetry<sup>4</sup>.

As argued in the introduction, GNWS reduces to a related variant of the directed Steiner problem, in which the asymmetry  $\beta$  of edge costs, as defined in [8], does not exceed  $\alpha$ . Thus it would be very interesting to derive better upper bounds for the edge-asymmetric Steiner tree problem. In particular, can we show that the sublinear (in terms of  $k$ ) upper bound of [6] for offline Steiner trees and forests in directed graphs can be improved assuming a known bound on the edge asymmetry? Can we get  $o(\beta)$  approximation algorithms? The online version of this problem has been the topic of [8], but there still exists a gap between the known upper and lower bounds on the competitive ratio.

Last, since the asymmetric node-weighted and directed Steiner problems are related, we would like to study the combination of the two, namely the problem in which bounds on both vertex and edge asymmetry are known. Such a model would probably capture in a more realistic way practical network design problems.

## 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. The online set cover problem. In *Proceedings of the 35th annual ACM Symposium on the Theory of Computation*, pages 100–105, 2003.

---

<sup>4</sup> If we define  $\alpha$  as  $\max_{v,u \in V} \{|c(v) - c(u)|\}$ , it is convenient to assume that all costs are at least 1, otherwise a trivial bound of  $\Theta(k)$  can be shown for the online version.

3. N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. A general approach to online network optimization problems. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 570–579, 2005.
4. 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.
5. M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.
6. 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.
7. M. Faloutsos. *The Greedy the Naive and the Optimal Multicast Routing—From Theory to Internet Protocols*. PhD thesis, University of Toronto, 1998.
8. 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.
9. U. Feige. A threshold of  $\ln n$  for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.
10. M.X. Goemans and D.P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 6(24), 1995.
11. S. Guha and S. Khuller. Improved methods for approximating node weighted steiner trees and connected dominating sets. *Information and Computation*, (150):228–248, 1999.
12. M. Imase and B. Waxman. The dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991.
13. R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.
14. P. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *Journal of Algorithms*, (19):104–115, 1995.
15. 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.
16. 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.
17. A. Segev. The node-weighted steiner tree problem. *Networks*, (17):1–17, 1987.
18. M. Thimm. On the approximability of the Steiner tree problem. *Theoretical Computer Science*, 295(1):387–402, 2003.
19. 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.
20. V. Vazirani. *Approximation Algorithms*. Springer, 2001.
21. 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.