A General Approach to Online Network Optimization Problems

Noga Alon[*] Baruch Awerbuch[†] Yossi Azar[‡] Niv Buchbinder [§]
Joseph (Seffi) Naor [¶]

Abstract

We study a wide range of online graph and network optimization problems, focusing on problems that arise in the study of connectivity and cuts in graphs. In a general online network design problem, we have a communication network known to the algorithm in advance. What is not known in advance are the bandwidth or cut demands between nodes in the network. Our results include an $O(\log m \log n)$ competitive randomized algorithm for the online non-metric facility location and for a generalization of the problem called the multicast problem. In the non-metric facility location m is the number of facilities and n is the number of clients. The competitive ratio is nearly tight. We also present an $O(\log^2 n \log k)$ competitive randomized algorithm for the online group Steiner problem in trees and an $O(\log^3 n \log k)$ competitive randomized algorithm for the problem in general graphs, where n is the number of vertices in the graph and k is the number of groups. Finally, we design a deterministic $O(\log^3 n \log \log n)$ competitive algorithm for the online multi-cut problem. Our algorithms are based on a unified framework for designing online algorithms for problems involving connectivity and cuts. We first present a general $O(\log m)$-deterministic algorithm for generating fractional solution that satisfies the online connectivity or cut demands, where m is the number of edges in the graph. This may be of independent interest for solving fractional online bandwidth allocation problems, and is applicable to the node version as well. The integral solutions are obtained by an online rounding of the fractional solution. This part of the framework is problem dependent, and applies various tools including results on the approximate max-flow min-cut for multicommodity flow, the HST method and its extensions, certain rounding techniques for dependent variables, and Räcke's new hierarchical decomposition of graphs.

1 Introduction

We study a wide range of graph and network optimization problems, focusing on problems that arise in the study of connectivity and cuts in graphs. Such problems are associated with an input graph $G = (V, E)$ (directed or undirected), a cost function $c : E \to \mathbb{R}^+$, and a requirement function f (to be defined for each problem separately). The goal is to find a minimum cost subgraph that satisfies f. Our model is online; that is, the requirement function is not known in advance and it is given "step by step" to the algorithm, while the input graph is known in advance.

Network design problems are typically defined by a requirement function f that specifies for each cut in the graph the minimum coverage required for it. Since we are considering an online version of network design problems we concentrate on the following subclass which we call *generalized connectivity*. The requirement function f is a set of demands of the form $D = (S, T)$, where S and T are subsets of vertices in the graph such that $S \cap T = \emptyset$. A feasible solution is a set of edges, such that for each demand $D = (S, T)$ there is a path from a vertex in S to a vertex in T. Examples of problems belonging to this class are Steiner trees, generalized Steiner trees, and

[*]Schools of Mathematics and Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel. Email: nogaa@post.tau.ac.il. Research supported in part by a US-Israel BSF grant, by the Israel Science Foundation and by the Hermann Minkowski Minerva Center for Geometry at Tel Aviv University.

[†]Johns Hopkins University, Baltimore, MD 21218. E-mail: baruch@blaze.cs.jhu.edu. Supported by Air Force Contract TNDGAFOSR-86-0078, ARPA/Army contract DABT63-93-C-0038, ARO contract DAAL03-86-K-0171, NSF contract 9114440-CCR, DARPA contract N00014-J-92-1799, and a special grant from IBM.

[‡]School of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel. Email: azar@post.tau.ac.il. Research supported in part by the Israel Science Foundation and by the IST Program of the EU.

[§]Computer Science Dept., Technion, Haifa 32000, Israel. E-mail: nivb@cs.technion.ac.il.

[¶]Computer Science Dept., Technion, Haifa 32000, Israel. E-mail: naor@cs.technion.ac.il. Research supported in part by a US-Israel BSF grant, and by EU contract IST-1999-14084 (APPOL II).

the group Steiner problem. Less obvious examples are the set cover problem and the non-metric facility location problem, described below.

Cut problems in graphs involve separating sets of vertices from each other. We concentrate on a family of cut problems which we call *generalized cuts*. The requirement function f is a set of demands of the form $D = (S, T)$, where S and T are subsets of vertices in the graph such that $S \cap T = \emptyset$. A feasible solution is a set of edges that separates for each demand $D = (S, T)$, any two vertices $s \in S$ and $t \in T$. Examples of problems belonging to this class are multiway cuts and multi-cut.

There is a natural linear relaxation for the problems that we are considering. For generalized connectivity problems, a feasible fractional solution associates a fractional weight (capacity) with each edge, such that for each demand $D = (S, T)$ a unit of flow can be sent from S to T, where the flow on each edge does not exceed its weight. For generalized cuts, a feasible fractional solution associates a fractional weight (length) with each edge, which we interpret as inducing a distance function. The constraint is that for each demand $D = (S, T)$, the distance between any two vertices $s \in S$ and $t \in T$ is at least 1. Since many of the problems that we are considering are NP-hard, this linear relaxation is very useful for computing an approximate solution. Please refer to [17] for more details. In addition, fractional solutions have a motivation of their own in certain network design problems and bandwidth allocation problems.

Previous work: Network optimization problems in an online setting have been studied extensively. The online Steiner problem was considered in [14] who gave an $O(\log n)$-competitive algorithm and showed that in a general metric space this is indeed best possible. The generalized Steiner problem was considered in [3], where an $O(\log^2 n)$-competitive algorithm is given. This was improved to an $O(\log n)$-competitive ratio algorithm by [5]. The online version of the **metric** facility location problem was also considered recently. Meyerson [15] gave a randomized $O(\log n)$-competitive algorithm which was improved to a deterministic $\Theta(\frac{\log n}{\log \log n})$-competitive algorithm by Fotakis [7]. Recently, a deterministic online $O(\log n \log m)$-competitive algorithm for the set cover problem was given by [1] where n is the number of elements and m is the number of sets. A lower bound of $\Omega(\frac{\log n \log m}{\log \log m + \log \log n})$ was also shown for any deterministic online algorithm for the online set cover problem.

There is a vast literature on efficient approximation algorithms for problems involving connectivity and cuts. The reader is referred to [13, 17] for more details.

1.1 Results. We study generalized connectivity and cuts problems in a unified framework. The idea is to first compute a fractional solution online and then round this solution to an integral one. We provide a general deterministic procedure that computes a near-optimal fractional solution to any problem belonging to our class of problems. Specifically, the competitive ratio that we achieve is $O(\log m)$, where m is the number of edges in the graph. This algorithm can easily be extended to the vertex counterparts of the problems. We also show a matching lower bound of $\Omega(\log m)$ on the competitive ratio of any deterministic or randomized algorithm for this problem. Our algorithm draws on ideas taken from the algorithm of [1] for the online set cover problem.

We next describe our results on converting online a fractional solution into an integral solution. This rounding is problem dependent and we describe the rounding for each of the special cases considered.

The first problem we consider is *non-metric facility location*. In this problem we are given a set of possible facilities, each with a setup cost, and a set of clients, each with a connection cost to the facilities. The goal is to find a solution that minimizes the sum of the setup costs and the service costs. In the online case of the non-metric facility location, clients arrive online. The set cover problem is a special case of this problem in which the facilities are sets and the connection cost is either zero or infinite, depending on whether or not an element belongs to a set.

Next, we consider the *multicast* problem that generalizes the non-metric facility location problem. In the multicast problem we are given a set of weighted rooted trees containing a set of terminals. Each terminal is associated with at most one node in each tree. The goal is to find a minimum weight set of subtrees that contain all the terminals, where a subtree must contain the root of the tree it belongs to. In the online case, the terminals arrive online, and upon arrival of a terminal it is necessary to connect it to the root of a tree containing it. The non-metric facility location is a special case of the multicast problem. Each facility corresponds to a tree of depth two. A tree has one edge emanating from the root with weight equal to the setup cost of the facility, and then there are edges to the leaves, where each leaf corresponds to a client, and the weight of an edge is equal to the connection cost of the client to the facility.

Finally, in the realm of generalized connectivity, we consider the group Steiner problem on trees and general graphs [9]. In the group Steiner tree problem on a rooted tree we are given a weighted rooted tree $T = (V, E, r)$, and groups $g_1, g_2, \ldots g_k \subset V$. The goal is to find a minimum rooted subtree $T' = (V', E', r)$ that contains at least one vertex from each group. In the online version of the problem, the groups arrive online. The multicast problem is a special case of the group Steiner problem on rooted trees. Given an instance of the multicast problem, the roots of the trees can be connected to a joint root using edges of zero weight. A group contains all the nodes associated with a particular terminal. Notice that this reduction creates a special instance of the group Steiner tree problem in which any two paths from the root to vertices belonging to the same group are disjoint.

In the multi-cut problem we are given an undirected graph with costs (capacities) and a set of source-sink pairs. The goal is to find a minimum cost set of edges that disconnects each source-sink pair. In the online version of the problem, the source-sink pairs arrive online, and upon arrival of a pair it is necessary to disconnect it. We show an online algorithm for the multi-cut problem using the constructive version of a remarkable result of Räcke [16] for the hierarchical decomposition of graphs ([6] and [12]) together with an approximate max-flow min-cut theorem on trees [11]. This decomposition is used along with an online primal-dual algorithm for the problem on trees.
Specifically, we obtain the following results.

- A randomized $O(\log m \log n)$ competitive algorithm for the online multicast problem on trees, where m is the number of edges, and n is the number of requested terminals.
- A randomized $O(\log m \log n)$ competitive algorithm for the online non-metric (and metric) facility location problem, where m is the number of possible facilities and n is the number of clients.
- A randomized $O(\log^2 n \log k)$-competitive algorithm for the online group Steiner problem on trees, where k is the number of groups, and n is the number of leaves in the tree. This implies a randomized $O(\log^3 n \log k)$-competitive algorithm for general graphs using hierarchically well-separated trees [4, 8]
- A deterministic $O(\log^3 n \log \log n)$ competitive algorithm for the online multi-cut problem in general graphs. Improved bounds are obtained for planar graphs and for trees.

2 Preliminaries

In this section we formally define our problems. Let $G = (V, E)$ be a graph (directed or undirected) with cost function $c : E \to \mathbb{R}^+$ associated with the edge set E. Suppose further that there is a weight function (or capacity function) $w : E \to \mathbb{R}^+$ associated with the edge set E. The *cost* of w is defined to be $\sum_{e \in E} w_e c_e$.

Let $A \subset V$ and $B \subset V$ be subsets of V such that $A \cap B = \emptyset$. Let G' be the graph obtained from G by adding a super-source s connected to all vertices in A and a super-sink t connected to all vertices in B. The edges from s to A are directed into A and have infinite weight, and the edges from B to t are directed into T and have infinite weight. There is flow from A to B of value α if there exists a exists a legal flow function f that sends α units of flow from s to t satisfying the capacity function w. The shortest path from A to B is defined to be the shortest path with respect to w from any vertex $u \in A$ to any vertex $v \in B$ (i.e. the minimal distance between any pair of vertices in A and B)

A requirement function is a set of demands of the form $D_i = (S_i, T_i)$, $1 \le i \le k$, where $S_i \subset V$ and $T_i \subset V$ and $S_i \cap T_i = \emptyset$.

We first define the *generalized connectivity* problem. The input is a requirement function f. A feasible *integral* solution is an assignment of weights (capacities) w from $\{0, 1\}$ to E, such that for each demand $D_i = (S_i, T_i)$, $1 \le i \le k$, there is a flow function from S_i to T_i of value at least 1. A feasible *fractional* solution is an assignment of weights (capacities) w from $[0, 1]$ to E, such that for each demand $D_i = (S_i, T_i)$, $1 \le i \le k$, there is a flow function from S_i to T_i of value at least 1. We note that the flow constraint has to be satisfied for each demand (S_i, T_i) separately. The cost of a solution is defined to be the cost of w.

We now define the *generalized cuts* problem. The input is a requirement function f. A feasible *integral* solution is a set of edges $E' \subseteq E$ that separates for each demand $D_i = (S_i, T_i)$ any two vertices $a \in S_i$ and $b \in T_i$. Alternatively, we can think of each edge $e \in E'$ as having weight $w(e) = 1$. Thus, the weight function w induces a distance function on the graph such that the distance between vertices separated by E' is at least 1. A feasible *fractional* solution is an assignment of weights w from $[0, 1]$ to E, such that for each demand $D_i = (S_i, T_i)$, $1 \le i \le k$, the distance induced by w between each $a \in S_i$ and $b \in T_i$ is at least 1. The cost of a solution is defined to be the cost of w.

In an online setting, the graph $G = (V, E)$ along with the cost function c is known to the algorithm (as well as to the adversary) in advance. The set of requests of the form $D_i = (S_i, T_i)$ is then given one-by-one to the algorithm in an online fashion. Upon arrival of a new demand, the algorithm must satisfy it by increasing the weights of edges in the graph. (The algorithm is not allowed to decrease the weight of an edge.) Thus, previous demands remain satisfied. The performance of the algorithm, the *competitive ratio*, is defined to be the ratio between the cost of the solution produced by the algorithm and the cost of an optimal solution that satisfies the given demands with minimum cost.

We now define the special cases that we consider in the context of generalized connectivity.

The *multicast* problem in trees is defined as follows: Let $X = \{1, 2, \ldots, n\}$ be a ground set of terminals, and let $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ be a family of rooted trees with a cost function $c : E \to \mathbb{R}^+$ associated with the edges. Each tree leaf is associated with a subgroup of the terminals, where any terminal i belongs to at most one leaf subgroup in each of the trees. A *cover* is a collection of rooted subtrees $\mathcal{T}' = \{T'_1, T'_2, \ldots, T'_m\}$, where $T'_i \subseteq T_i$ $(1 \le i \le m)$, such that the union of the subgroups of the leaves in $\mathcal{T}'$ is the set X. The cost of a cover is the sum of the costs of the edges in the subtrees in $\mathcal{T}'$. The goal is to find a cover of minimum cost.

This problem is a natural generalization of the set cover problem. Think of each set as being represented by a tree containing only one edge, where one vertex is a root and the other vertex a leaf. The cost of the edge is equal to the cost of the set and the leaf is associated with the elements belonging to the set. We note that the multicast problem in trees has an $O(\log n)$-approximation algorithm.

The *non-metric facility location* problem is a special case of the multicast problem. In the non-metric facility location problem there are m possible locations for opening facilities denoted by $F = \{f_1, f_2, \ldots, f_m\}$. There is a setup cost for opening each of the facilities. There are also n clients $R = \{r_1, r_2, \ldots, r_n\}$ and a connection cost function $c : F \times R \to \mathbb{R}^+$ denoting the cost of connecting each client to each facility. A solution is a subset $F' \subseteq F$ and a mapping of the clients to the facilities in F'. The goal is to minimize the cost of the solution which is defined to be the setup cost of the facilities in F' plus the connection cost of the clients as defined by the mapping of clients to facilities in F'. The problem is clearly an instance of the multicast problem, where there is a tree corresponding to each facility and all trees have depth two. The cost of the "root" edge is equal to the setup cost of the facility and the cost of each "leaf" edge (that corresponds to a client) is equal to the cost of connecting the client to the facility.

The *group Steiner tree* problem in a rooted tree is defined as follows. We are given a rooted tree $T = (V, E, r)$ with non-negative cost function $c : E \to \mathbb{R}^+$, and groups $g_1, g_2, \ldots g_k \subset V$. The objective is to find a minimum cost rooted subtree T' that contains at least one vertex from each of the groups g_i, $1 \le i \le k$. The multicast problem is a special case of this problem. Given an instance of the multicast problem, we can connect the roots of all the multicast trees using edges of cost 0 to a joint root r. Group g_i, $1 \le i \le n$, is defined to be the set of leaves containing terminal t_i. Notice that this reduction creates a special instance of the group Steiner tree problem in which any two paths from the root to vertices belonging to the same group are disjoint.

The group Steiner tree problem has an $O(\log n \log k)$ approximation algorithm, where k is the number of groups, and n is the number of leaves in the tree [9]. In general (undirected graphs, the best approximation factor known for the group Steiner problem is $O(\log^2 n \log k)$ by combining [9] with [8].

The *multi-cut* problem in undirected graphs is a special case of the previously defined generalized cuts problem, where the requirement function f is a set of source-sink pairs $\{s_i, t_i\}$, $1 \le i \le k$. The best approximation factor for this problem is $O(\log k)$ [10].

3 Computing a fractional solution online

In this section we describe our online algorithm for computing a near-optimal fractional solution for both the generalized connectivity and the generalized cuts problems. We first describe the algorithm for the generalized connectivity problem (Section 3.1) and then explain the changes needed for the generalized cuts problem (Section 3.2). Let $|V| = n$ and $|E| = m$. The competitive ratio of our algorithm is $O(\log m)$ and it is defined with respect to an optimal offline fractional solution. We note that our method is applicable to both vertex and edge versions of our problems, as well as for directed and undirected graphs.

Let us denote the cost of an optimal fractional solution, OPT, by α. We first claim that by using the doubling technique, we can assume that the value of α is known up to a factor of 2. Initially, we can start guessing $\alpha = \min_{e \in E} c_e$, and then run the algorithm with this bound on the optimal solution. If it turns out that the value of the optimal solution is larger than our current guess for it, (that is, the cost of the fractional solution exceeds $\Theta(\alpha \log m)$), then we can "forget" about all weights given so far to the edges, update the value of α by doubling it, and continue on. We note that the fractional cost of the edges that we have "forgotten" about can increase the cost of our solution by at most a factor of 2, since the value of α is doubled in each step.

We thus assume that α is known. We next claim that we can assume that for all edges having weight strictly less than 1 in the online solution, their cost is between 1 and $2m^2$. This is justified as follows. First, we double the weights of all the edges in OPT and round down to zero all edges that have weight less than $1/m$. This results in a feasible solution of cost 2α which we denote by OPT$'$. Since the weights of the edges in OPT$'$ are at least $1/m$, the cost of the edges in OPT$'$ cannot exceed $2m\alpha$. Hence, all edges in the graph having cost more than $2m\alpha$ can be ignored. We now set to 1 in our solution the weight of all edges having cost less than α/m. This can increase the competitive factor of our algorithm by at most a factor of 2. Thus, the costs of all the edges in the graph are between α/m and $2m\alpha$, and the costs can further be normalized so that the minimum cost is 1 and the maximum cost is at most $2m^2$.

We note that by ignoring edges in the graph we mean that they are not considered by our online algorithm. For the generalized connectivity problem it means that we do not use these edges to connect vertices, and hence they can be removed from the graph. For the generalized cut problem it means that such edges do not participate in a cut, and hence we can identify the two endpoints of such edges.

3.1 Generalized Connectivity. We describe an online algorithm with competitive factor $O(\log m)$. All logarithms are to the base 2. Initially, the algorithm gives each edge a fractional weight of $1/2m^3$. Assume now that the algorithm receives a new demand (S, T). The following is performed in this case.

1. If the maximum flow from S to T is at least 1, then do nothing.
2. Else: While the flow between S and T is less than 1, perform a *weight augmentation*:
   - Compute a minimum cut $\mathcal{C}$ between S and T.
   - For each edge $e \in \mathcal{C}$, $w_e \leftarrow w_e(1 + \frac{1}{c_e})$.

We now analyze the performance of the algorithm upon termination, i.e., when the algorithm gets the full requirement function.

LEMMA 3.1. *When the algorithm terminates, all connectivity demands are satisfied.*

Proof. Follows immediately from the algorithm. □

LEMMA 3.2. *The number of weight augmentation steps performed during the run of the algorithm is at most*

$$6\alpha \log_2 m + 4\alpha = O(\alpha \log m).$$

Proof. Obviously, for each edge $e \in E$, $w_e \leq 1 + \frac{1}{c_e}$ always holds, since an edge which has weight exceeding 1 cannot be part of a minimum cut with total weight less than 1. Consider the following potential function:

$$\Phi = \sum_{e \in E} c_e w_e^* \log_2(w_e)$$

where w_e^* is the weight of edge e in OPT$'$. We now show three properties of Φ:

- The initial value of the potential function is: $-6\alpha \log_2 m - 2\alpha$.
- The potential function never exceeds 2α.
- In each weight augmentation step, the potential function increases by at least 1.

The first two properties follow directly from the initial value and from the fact that no edge gets a weight of more than 2. Consider an iteration in which the adversary gives a connectivity demand (S, T). A weight augmentation of a cut $\mathcal{C}$ is performed as long as the connectivity between S and T is less than 1. The total weight assigned by

OPT' to edges in $\mathcal{C}$ is at least 1. Thus, the increase of the potential function is at least:

$$\begin{aligned}\Delta\Phi &= \sum_{e\in\mathcal{C}} c_e w_e^* \log_2\left(w_e\left(1+\frac{1}{c_e}\right)\right) \\ &\quad - \sum_{e\in\mathcal{C}} c_e w_e^* \log_2 w_e \\ &= \sum_{e\in\mathcal{C}} c_e w_e^* \log_2\left(1+\frac{1}{c_e}\right) \;\geq\; \sum_{e\in\mathcal{C}} w_e^* \;\geq\; 1\end{aligned}$$

□

THEOREM 3.1. *The above algorithm is $O(\log m)$-competitive for the fractional generalized connectivity problem.*

Proof. It suffices to show that the following is maintained throughout the algorithm:

$$\sum_{e\in E} w_e c_e \leq 6\alpha \log_2 m + 4\alpha + 1 = O(\alpha \log m).$$

Consider an iteration in which a connectivity demand (S,T) is given. Let $\mathcal{C}$ be a cut that has weight less than 1, i.e., $\sum_{e\in\mathcal{C}} w_e < 1$. The weight of each edge $e \in \mathcal{C}$ increases by w_e/c_e in each weight augmentation step. Thus, the total increase of the quantity $\sum_{e\in E} w_e c_e$ in a weight augmentation step does not exceed

$$\sum_{e\in\mathcal{C}} \frac{w_e}{c_e} c_e = \sum_{e\in\mathcal{C}} w_e < 1.$$

Initially, $\sum_{e\in E} w_e c_e \leq m \cdot \frac{1}{2m^3} \cdot 2m^2 = 1$, and the claim thus follows from Lemma 3.2 that bounds the number of weight augmentation steps. □

3.2 Generalized Cuts. We now present an $O(\log m)$-competitive algorithm for the generalized cuts problem. It is essentially the same as the algorithm for the generalized connectivity problem presented in the previous section. We highlight the changes needed.

Initially, the algorithm assigns each edge a length of $1/2m^3$. Assume now that the adversary gives the algorithm a new request (S,T). The algorithm is as follows.

1. If the length of the shortest path from S to T is already at least 1, then do nothing.
2. Else: While the length of the shortest path from S to T is less than 1 perform a *length augmentation*:
   - Compute the shortest path $\mathcal{P}$ between S and T.
   - For each edge $e \in \mathcal{P}$, $w_e \leftarrow w_e(1+\frac{1}{c_e})$.

Clearly, the above algorithm produces a feasible fractional solution to the problem. Proving the competitive factor in this case follows closely the proof in the case of generalized connectivity. Hence we conclude:

THEOREM 3.2. *The above algorithm is $O(logm)$-competitive for the fractional generalized cut problem.*

3.3 Lower Bounds. We now show that our algorithm is optimal for the fractional generalized connectivity and generalized cuts problems. In order to show that we prove two lemmas. The first one provides a lower bound on the competitive ratio of either a deterministic or a randomized algorithm for the generalized connectivity problem. The lemma also holds with respect to an integral optimal solution. The second lemma provides the same lower bound with respect to the generalized cuts problem.

LEMMA 3.3. *Any deterministic or randomized algorithm for the online fractional connectivity problem has a competitive ratio of at least $\Omega(\log m)$ with respect to both an integral optimal solution and a fractional optimal solution. This holds even when the graph is an undirected star.*

Proof. Let A be any deterministic or randomized online algorithm. Let T be a star with n leaves (and edges) $v_1, v_2, \ldots, v_n$, a root r, $n = 2^k$. We describe the strategy of the adversary. The adversary starts by asking the demand (S,T), $S = \{r\}$, $T = (v_1, v_2, \ldots, v_n)$. This is defined to be iteration zero. Algorithm A must increase the flow from the root to all the leaves to be equal to 1. This means that the expected flow to either the $n/2$ first terminals or to the $n/2$ last terminals is at least half. Thus, in the next iteration the adversary changes T to be either $(v_1, v_2, \ldots, v_{n/2})$ or $(v_{(n/2)+1}, \ldots, v_n)$, choosing the set with the smaller expected flow value. In the kth iteration, if the previous demand was $(\{r\}, \{v_i, v_{i+1}, \ldots, v_j\})$, $j > i$, then the next demand is either $(\{r\}, \{v_i, \ldots, v_{\frac{i+j}{2}}\})$ or $(\{r\}, \{v_{\frac{i+j}{2}+1}, \ldots, v_j\})$ choosing the set with the less expected flow. It is not hard to see that the expected cost of Algorithm A in the fractional case is at least,

$$\sum_{i=1}^{\log n} \frac{1}{2} = \Omega(\log n) = \Omega(\log m).$$

The optimal integral solution can assign a weight of 1 only to the edge adjacent to the last vertex asked, completing the proof of the lower bound. ◻

LEMMA 3.4. *Any deterministic or randomized algorithm for the online fractional cuts problem has a competitive ratio of at least $\Omega(\log m)$ with respect to both an integral optimal solution and a fractional optimal solution. This is true even when the graph is a line and the cuts demands groups are of size 1.*

Proof. Let A be any deterministic or randomized online algorithm. Let G be a line with vertices $v_1, v_2, \ldots, v_n$ ($n = 2^k + 1$). We describe the strategy of the adversary. The adversary starts with the demand $(\{v_1\}, \{v_n\})$. This is defined to be iteration zero. Algorithm A must increase the distance from v_1 to v_n to be 1. This means that the expected distance from either v_1 to $v_{(n+1)/2}$ or from $v_{(n+1)/2}$ to v_n is at least half. Thus, in the next iteration the adversary continues with either $(\{v_1\}, \{v_{(n+1)/2}\})$ or $(\{v_{(n+1)/2}\}, \{v_n\})$, choosing the path which has the shorter expected distance. The adversary can continue doing so until it asks two consecutive vertices. It is not hard to see that the expected cost of Algorithm A in the fractional case is at least,

$$\sum_{i=1}^{\log(n-1)} \frac{1}{2} = \Omega(\log n) = \Omega(\log m)$$

The optimal integral solution can assign a length of 1 to the edge separating the last two vertices, completing the proof of the lower bound. ◻

We remark that the proofs of the lower bounds in this section cannot be applied to the fractional online Steiner tree problem, as well as for the fractional online generalized Steiner tree. However, a lower bound on the competitive ratio for any deterministic or randomized online algorithm for these problems follows in a straightforward manner from the lower bound shown for the integral Steiner tree problem in [14].

4 Applications - Integral Connectivity and Cuts Problems

We can use the algorithms described in the previous section as a basis for an efficient randomized online algorithm for special cases of the integral connectivity and cuts problems as well. This can be done by rounding online the fractional solution generated by the algorithm in the previous section. This is the heart of our general approach to online network optimization problems. The rounding algorithms use the online algorithms for generating a fractional solution as a "black box". We present here four problems in which such a rounding is applicable.

In section 4.1 we consider the multicast problem and the non-metric facility location problem. In section 4.2 we consider the group Steiner problem on a tree. Then, in section 4.3 we consider the group Steiner tree problem in general graphs. We conclude with the online multi-cut problem in Section 4.4.

4.1 Multicast and Non-metric Facility Location Problems. We describe a randomized algorithm for the multicast problem. Following each request, we first compute an $O(\log m)$-competitive fractional solution. We now explain how the rounding of the fractional solution is performed.

Initially, the algorithm starts with an empty cover $\mathcal{C} = \emptyset$. The algorithm keeps for every tree $T_i \in \mathcal{T}$, $2\lceil \log(n'+1) \rceil$ random independent variables, $X(T_i, j)$, $1 \leq j \leq 2\lceil \log(n'+1) \rceil$, distributed uniformly in the interval $[0,1]$. The value of n' is the number of terminals asked so far by the adversary. As n' changes we gradually increase the number of random variables. Define the threshold of tree T_i to be:

$$\theta(T_i) = \min_{j=1}^{2\lceil\log(n'+1)\rceil} X(T_i, j).$$

The rounding method is very simple. Take to the solution $\mathcal{C}$ all edges e for which $w_e > \theta(T_e)$, where T_e is the tree containing edge e. That is, the weight of edge e has exceeded the threshold of the tree containing it. We note that increasing n' adds more random variables which can only decrease the thresholds of the trees, and hence increase the probability of taking edges to the solution. Thus, when increasing n', it is necessary to reconsider previously considered edges.

Let α be the value of an optimal fractional solution to the instance given so far. We now analyze the performance of the algorithm.

LEMMA 4.1. *The following holds throughout the algorithm:*

1. *The expected cost of the solution produced by the algorithm is $O(\alpha \log n' \log m)$.*
2. *For any terminal t, the probability that t is requested, yet it is not covered, is at most $1/n'^2$.*

Proof. We begin by proving (1). For each edge e and j, $1 \le j \le 2\log n$, let $Y(e,j)$ be the indicator random variable of the event that $w_e > X(T_e, j)$. Thus,

$$\begin{aligned}
\mathbf{Exp}\left[\sum_{e\in\mathcal{C}} c_e\right] &\le \sum_{e\in \mathrm{E}} \sum_{j=1}^{2\lceil\log(n'+1)\rceil} c_e \cdot \mathbf{Exp}[(Y(e,j)] \\
&= \sum_{e\in \mathrm{E}} \sum_{j=1}^{2\lceil\log(n'+1)\rceil} c_e w_e \\
&\le 2\lceil\log(n'+1)\rceil(2\alpha\log m + \alpha + 1) \\
&= O(\alpha \log n' \log m).
\end{aligned}$$

We now prove (2). Consider a terminal t. The fractional solution guarantees that the total amount of flow that can be sent from the roots of the trees in $\mathcal{T}$ to the vertices that are associated with t is at least 1. Let f_t^T be the flow to terminal t in tree T. Thus, the probability that terminal t is not covered is bounded from above by the probability that the threshold of each tree T containing t is larger than f_t^T. Recall that the weight of each edge on a path to t is at least the flow going to the terminal on the path.

For any tree T and j, $1 \le j \le 2\lceil\log(n'+1)\rceil$, the probability that the flow to t in T is at most $X(T,j)$ is $1 - f_t^T$. Thus, the probability that none of the flow paths to t exceeds $X(T,j)$ is

$$\prod_{T\in\mathcal{T}} (1 - f_t^T) \le e^{-\sum_{T\in\mathcal{T}} f_t^T} < \frac{1}{e},$$

where the last inequality follows from the fact that $\sum_T f_t^T = f_t \ge 1$. Thus, the probability that the terminal is not covered by any $1 \le j \le 2\lceil\log(n'+1)\rceil$ is less than $1/n'^2$. □

Lemma 4.1 suggests the following change to the algorithm to guarantee that a feasible solution is always computed. We run the algorithm. If at any time a terminal t is requested, but is not covered, then we choose the cheapest path from a root of a tree in $\mathcal{T}$ to t to the cover. The cost of this path is certainly a lower bound on the optimal solution. Since this event happens with probability at most $1/n'^2$ for each terminal, its effect on the expected cost of the algorithm is negligible. Thus, we obtain the following theorems.

THEOREM 4.1. *There exists a randomized algorithm for the online multi-cast problem in trees that is $O(\log n' \log m)$ competitive.*

THEOREM 4.2. *There exists an $O(\log n \log m)$ competitive randomized algorithm for the non-metric facility location, where m is the number of facilities and n is the number of clients.*

We remark that both the online multi-cast problem in trees and the online non-metric facility location are generalizations of the online set-cover problem introduced in [1]. Thus, the lower bound of $\Omega(\frac{\log n \log m}{\log\log m + \log\log n})$ proved in [1] for any deterministic algorithm for the online set-cover problem applies to these problems as well.

4.2 The Group Steiner Problem on Trees We describe a randomized algorithm for the group Steiner tree problem on trees. Following each request, we first compute an $O(\log m)$-competitive fractional solution. We now explain how the rounding of the fractional solution is performed. To this end, we use an online variation on the method of [9].

Initially, the algorithm starts with an empty cover $\mathcal{C} = \emptyset$. Applying the technique of [9] requires that the fractional weights on a path from the root to a terminal are monotonically decreasing. However, the fractional solution that we compute may not necessarily satisfy this property. Therefore, we define the weight of each edge to be the maximum flow through the edge going to a terminal in the edge's subtree. Thus, we abuse notation and let w_e denote the flow on edge e instead of the actual weight of e. Note that by doing so we can only decrease the value of the fractional solution that serves as our baseline for performing the competitive analysis, since the flow value on an edge can only be less than the actual weight of the edge.

Consider a weight augmentation iteration in the fractional algorithm. For each edge e, let w_e and $w'_e = w_e + \delta_e$ denote the weight of e before and after the weight augmentation iteration, respectively. For an edge e, let f be the edge adjacent to e and closer to the root r. The rounding algorithm processes the edges in topological order (from top to bottom). For each edge e, the following is performed:

- If $w'_e > 1$, then add e to $\mathcal{C}$.
- If e is incident on r, or $w'_f > 1$, then add e to $\mathcal{C}$ with probability $\delta_e/(1 - w_e)$.
- If $f \in \mathcal{C}$, then add e to $\mathcal{C}$ with probability $\delta_e/(w'_f - w_e)$.

Note that for each edge e, the probability $\delta_e/(w'_f - w_e) \leq \delta_e/(w'_e - w_e) = 1$, since $w'_e < w'_f$. The edges are considered in topological order so that $\mathcal{C}$ induces a (connected) subtree of T, since an edge is added to $\mathcal{C}$ only if the path connecting it to the root r already belongs to $\mathcal{C}$. We prove the following lemma:

LEMMA 4.2. *At any point of time t in the algorithm, the probability that an edge e belongs to $\mathcal{C}$ is $w_e(t)$, where $w_e(t)$ is the weight of e at time t. If $w_e > 1$, then $e \in \mathcal{C}$.*

Proof. The rounding algorithm adds each edge e to $\mathcal{C}$ for which $w_e > 1$.

Consider an edge e and let the path from the root r to e be $e_0, e_1, \ldots e_p = e$. The proof is by induction on the time t.

Induction Basis: At $t = 0$, the probability that e is added to $\mathcal{C}$ is equal to $w_{e_0} \cdot \prod_{i=1}^{p} w_{e_i}/w_{e_{i-1}} = w_{e_p}$.

Inductive Step: Consider a time $t > 0$ where w_e is raised to $w_e + \delta_e$. By the inductive hypothesis, each edge e_i, $0 \leq i \leq p-1$, belongs to $\mathcal{C}$ with probability $\min\{w'_{e_i}, 1\}$, and $e \in \mathcal{C}$ with probability w_e.

We need another (internal) induction on p, the length of the path from r to e. The base case is when e is incident on r. Then, the probability that e belongs to $\mathcal{C}$ is

$$w_e + \frac{\delta_e \cdot (1 - w_e)}{(1 - w_e)} = w_e + \delta_e = w'_e.$$

If $w'_e > 1$ then edge e belongs to $\mathcal{C}$ with probability 1.

Let e be an edge of depth p. By the (internal) inductive hypothesis, the probability that e_{p-1} is added to $\mathcal{C}$ is w'_{p-1}. The probability that e is added to $\mathcal{C}$ is equal to

$$w_e + \frac{\delta_e \cdot (w'_{p-1} - w_e)}{(w'_{p-1} - w_e)} = w_e + \delta_e = w'_e.$$

The above LHS is the probability that e was previously added to $\mathcal{C}$ plus the probability that e was not previously added to $\mathcal{C}$, but e_{p-1} was previously added to $\mathcal{C}$ and e is added to $\mathcal{C}$ in the current step.

If $w'_{p-1} > 1$, then the probability that e is added to $\mathcal{C}$ in the current step is

$$w_e + \frac{\delta_e \cdot (1 - w_e)}{(1 - w_e)} = w_e + \delta_e = w'_e.$$

If $w'_e > 1$, then edge e belongs to $\mathcal{C}$. Thus, the claim holds. □

LEMMA 4.3. *At any point of time t in the algorithm, the expected cost of the edges added to $\mathcal{C}$ is at most $\sum_{e \in \mathcal{T}} c_e w_e(t)$.*

Proof. From Lemma 4.2 it follows that the probability at any time t that edge e belongs to $\mathcal{C}$ is at most w_e. By linearity of the expectation the claim follows. □

We now analyze the probability that a group g is covered when $w_g > 1$.

LEMMA 4.4. *For any group g, consider the first time t such that $w_g > 1$. Then, the probability that a vertex belonging to g is in at time t is $\Omega(1/\log N)$, where N is the maximum size of a group.*

Proof. Our proof uses [9, Thm. 3.4, p. 72]. This requires proving that the probability of the "good" events remains the same and that the dependency between them remains the same. The first claim follows from lemma 4.2. In order to prove the second claim we need to show that the probability of two "good" events is the same as in Theorem 3.4 of [9]. This follows from the fact that the probability that e is chosen to the solution given that h, an edge closer to the root, is chosen is exactly w_e/w_h. The events for e and f are independent, given that h, their least common ancestor, is chosen. Thus, the probability for e and f is $w_e w_f / w_h$, and we are done.

Actually, in order to use the original proof of [9], we are required to prove a stronger assertion about the independence of the corresponding events. However, this is not needed, since the proof of [9] can be modified so that only the second moment of the variable which is the number of paths from the root to a vertex in g needs to be computed. This follows from the assertion of [2, Sec. 4.8, Ex. 1]. Therefore, the above independence result suffices. □

Lemma 4.4 suggests the following randomized online algorithm. Run $O(\log k \log N)$ independent trials in parallel using the randomized rounding described, where k is the number of groups asked by the adversary. This results in a randomized algorithm with competitive ratio $O(\log k \log N)$ that covers all the groups with probability at least $1 - 1/k$. In order to guarantee that the algorithm always produces a feasible solution, we can use the shortest path to a group in case the algorithm fails to cover it. The cost of this path is a lower bound on the optimal solution, and since this event happens with probability at most $1/k$, it changes the expected competitive ratio of the algorithm by a negligible factor. Since we do not know in advance the value of k we may increase the number of trials gradually as more groups are asked, similar to Section 4.1.

4.3 The Group Steiner Problem on General Graphs. We now consider the Group Steiner tree algorithm on general graphs. To this end, we use hierarchically well-separated trees (HST-s) [4, 8]. A set of metric spaces $\mathcal{S}$ over V is said to α-probabilistically approximate a metric space $\mathcal{M}$ over V, if: (1) for every $x, y \in V$ and $S \in \mathcal{S}$, $d_S(x,y) \geq d_{\mathcal{M}}(x,y)$ and (2) there exists a probability distribution D over the metric spaces in $\mathcal{S}$ such that for all $x, y \in V$, $E[d_D(x,y)] \leq \alpha d_{\mathcal{M}}(x,y)$. Recently, the following theorem was proved in [8], improving upon the basic approach of [4].

THEOREM 4.3. *Every weighted connected graph G on n vertices can be α-probabilistically approximated by a set of weighted trees, where $\alpha = O(\log n)$. The probability distribution can be computed in polynomial time.*

We use this theorem to obtain the following bounds.

THEOREM 4.4. *There is a randomized online algorithm for the group Steiner problem in general graphs with a competitive ratio of $O(\log^3 n \log k)$.*

Proof. We first use Theorem 4.3 to randomly choose a tree T from the distribution D. Then, we run the online algorithm from Section 4.2 on the tree T. When a new vertex v is being connected to the root r, we just connect it in the graph via its closest neighbor in the tree that is already connected to the root. Since the tree is an HST, the cost of this path in the tree is only twice the connection cost of v to the least common ancestor of v and its closest previously connected neighbor. Thus, on the average, we are paying at most twice the stretch factor of the paths, and the theorem follows directly from Theorem 4.3 and the guarantee on the performance of the algorithm in Section 4.2

4.4 The multi-cut problem In this section we consider the online multi-cut problem in undirected graphs. The online algorithm we present here does not fit the general framework developed in the paper, where a fractional solution is computed online and then rounded online into an integral solution.

In [16], Räcke describes a procedure for finding a hierarchical decomposition of any undirected graph $G = (V, E)$ with capacities on the edges. An efficient procedure for finding such a decomposition tree T_G appears in [6] and [12]. This remarkable decomposition enables us to transform the problem from a general graph to a tree. We later on present an online algorithm for the multi-cut problem on trees with competitive ratio α, where α may depend on the height of the tree.

The nodes of the decomposition tree T_G correspond to a laminar family of subsets of V. There is a 1-1 correspondence between V and the leaves of the tree. The edges of T_G correspond to cuts in G and each tree edge is associated with a capacity (or cost) which is equal to the capacity (or cost) of the corresponding cut in G. The tree T_G has the property that for any choice of source-sink pairs, any feasible multi-commodity flow function in T_G can be routed in G causing a congestion of at most β. The best value of β is $O(\log^2 n \log\log n)$ for general graphs and $O(\log n \log\log n))$ for planar graphs, and it is given by [12] together with a polynomial-time construction of T_G.

Thus, the multi-cut problem in G translates into a multi-cut problem in the decomposition tree T_G, where the goal is to separate between the leaves containing the source-sink pairs. We run an α-competitive online algorithm for the (online) multi-cut problem in T_G. A multi-cut in T_G is a set of edges which translate back in G into a set of cuts having at most the capacity of the multi-cut in T_G. Clearly, this translation can be done online.

THEOREM 4.5. *There is a deterministic polynomial-time algorithm for the online multi-cut problem that achieves a competitive ratio of:*

- $O(\log^3 n \log\log n)$ *for general graphs.*
- $O(\log^2 n \log\log n)$ *for planar graphs.*
- $O(\log^2 n)$ *for trees.*

Proof. (Sketch) Let $\mathcal{C}_{onl}(G)$ and $\mathcal{C}_{onl}(T_G)$ denote the multi-cuts found by the online algorithm in G and in T_G, respectively. Let $\mathcal{C}_{opt}(T_G)$ denote the optimal multi-cut in T_G, and let $\mathrm{MCF}_{opt}(T_G)$ be the maximum multi-commodity flow in T_G between the source-sink pairs. By [11], in a tree, $\mathcal{C}_{opt}(T_G) \leq 2 \cdot \mathrm{MCF}_{opt}(T_G)$. Hence,

$$\begin{aligned} \mathcal{C}_{onl}(G) \ \leq \ \mathcal{C}_{onl}(T_G) \ &\leq \ \alpha \cdot \mathcal{C}_{opt}(T_G) \\ &\leq \ 2\alpha \cdot \mathrm{MCF}_{opt}(T_G). \end{aligned}$$

Let f^* be a maximum multi-commodity flow between the source-sink pairs in T_G. Let $\mathrm{MCF}_{opt}(G)$ denote a maximum multi-commodity flow in G between the source-sink pairs. Since f^* can be routed in G with a congestion of at most β, we get that,

$$\mathrm{MCF}_{opt}(G) \geq \frac{1}{\beta}\mathrm{MCF}_{opt}(T_G),$$

yielding that

$$\mathcal{C}_{onl}(G) \leq 2\alpha\beta \cdot \mathrm{MCF}_{opt}(G).$$

Since $\mathrm{MCF}_{opt}(G)$ lower bounds the optimal multi-cut in G, we get that our algorithm is $(2\alpha\beta)$-competitive. Substituting the appropriate values for β, and setting $\alpha = O(\log n)$, the claimed bounds follow. ∎

We now proceed and show an online algorithm for the multi-cut problem in trees. First, note that there is a simple reduction from the online multi-cut problem in trees to the online set cover problem. Each pair of vertices in the tree corresponds to an element; each edge of the tree corresponds to a set. A set contains an element if the corresponding edge separates the two vertices corresponding to the element. Hence, by the main result of [1], (which follows the basic general approach developed here), there is a deterministic $O(\log^2 n)$-competitive algorithm for the online minimum multi-cut tree problem.

The above reduction applies to any tree. However, when considering the decomposition trees produced by [12], we observe that their height is only $O(\log n)$. We use this to improve on the competitive ratio by providing an $O(h)$-competitive online algorithm for any tree, where h denotes the height. The online algorithm essentially follows the primal-dual 2-approximation algorithm of [11]. However, in an online setting, we cannot choose the order of the source-sink pairs and we cannot apply the "cleaning" stage at the end. Thus, applying the standard primal-dual scheme on the multi-cut problem on a tree yields an $O(h)$-approximation factor that translates to an $O(h)$-competitive online algorithm. The $O(h)$-approximation factor follows since the primal complementary slackness condition is preserved, and a relaxed dual condition with a $2h$ factor is trivially preserved. An alternative description of the algorithm is via the local ratio technique: reduce from the cost of all the edges on the unique path between the new source-sink pair the minimum cost of an edge on the path, and then take into the cut all zero-cost edges.

References

[1] 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*, pp. 100-105, 2003.

[2] N. Alon and J. H. Spencer, **The probabilistic method**, Second Edition, Wiley, New York, 2000.

[3] B. Awerbuch, Y. Azar, and Y. Bartal. On-line generalized Steiner problem. In *Proc. of the 7th Ann. ACM-SIAM Symp. on Discrete Algorithms*,pp. 68-74, 1996.

[4] Y. Bartal Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th annual IEEE Symposium on Foundations of Computer Science*, pp. 184-193, 1996.

[5] P. Berman and C. Coulston. On-line algorithms for Steiner tree problems. In *Proc. of the 29th annual ACM Symp. on the Theory of Computation*, 1997.

[6] M. Bienkowski, M. Korzeniowski and H. Räcke. A practical algorithm for constructing oblivious routing schemes. In *Proceedings of the 15th SPAA*, 2003.

[7] Dimitris Fotakis. On the competitive ratio for online facility location. In *ICALP 2003*,pp. 637-652 2003.

[8] J. Fakcharoenphol, S. Rao, K. Talwar A tight bound on approximating arbitrary metrics by tree metrics In *Proceedings of the 35th annual ACM Symposium on the Theory of Computation*, pp. 448-455, 2003.

[9] N. Garg, G. Konjevod, R. Ravi. A Polylogarithmic approximation algorithm for the group Steiner tree problem. In *Journal of Algorithms*, 37(1):66-84, 2000.

[10] N. Garg, V. V. Vazirani and M. Yannakakis Approximate max-flow min-(multi)cut theorems and their applications In *SIAM J. on Computing*, 25:235-251, 1996.

[11] N. Garg, V. V. Vazirani and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. In *Algorithmica*, 18:3-20, 1997.

[12] C. Harrelson, K. Hidrum and S. Rao A polynomial time tree decomposition to minimize congestion. In *Proceedings of the 15th SPAA*, pp. 34-43, 2003.

[13] D. Hochbaum. **Approximation algorithms for NP-Hard problems**. PWS Publishing Company, Boston, MA, 1997.

[14] M. Imase and B.M. Waxman. Dynamic Steiner tree problem. In *SIAM Journal Discrete Math.*, 4:369-384, 1991.

[15] Adam Meyerson. Online facility location. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science, FOCS*,pp. 426-431 2001.

[16] H. Räcke. Minimizing congestion in general networks. In *Proceedings of the 43rd Annual Symposium on Foundations of Computer Science, FOCS*, 2002.

[17] V. V. Vazirani. **Approximation algorithms**. Springer Verlag, Berlin-New York, 2001.