A General Approach to Online Network Optimization Problems

NOGA ALON
Schools of Mathematics and Computer Science, Tel Aviv University, Tel Aviv, Israel
BARUCH AWERBUCH
Computer Science Dept., Johns Hopkins University, Baltimore, MD
YOSSI AZAR
School of Computer Science, Tel Aviv University, Tel Aviv, Israel
NIV BUCHBINDER
Computer Science Dept., Technion, Haifa, Israel
and
JOSEPH (SEFFI) NAOR
Computer Science Dept., Technion, Haifa, Israel

A preliminary version of this paper appeared in the Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), New Orleans, Louisiana (2004), pp. 570-579.
The work of the first author was 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. The work of the second author was supported in part 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. The work of the third author was supported in part by the Israel Science Foundation and by the IST Program of the EU. The work of the fifth author was supported in part by US-Israel BSF grant 2002276 and by EU contract IST-1999-14084 (APPOL II).

Authors' addresses: N. Alon, Schools of Mathematics and Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel, email: nogaa@post.tau.ac.il; B. Awerbuch, Computer Science Dept., Johns Hopkins University, Baltimore, MD 21218. email: baruch@blaze.cs.jhu.edu; Y. Azar, School of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel, email: azar@post.tau.ac.il; Niv Buchbinder, Computer Science Dept., Technion, Haifa 32000, Israel, email: nivb@cs.technion.ac.il; J. (S) Naor, Computer Science Dept., Technion, Haifa 32000, Israel, email: naor@cs.technion.ac.il.

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 connectivity (bandwidth) or cut demands between vertices in the network which arrive online.

We develop a unified framework for designing online algorithms for problems involving connectivity and cuts. We first present a general $O(\log m)$-competitive deterministic algorithm for generating a 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 both directed and undirected graphs. We then show how to obtain integral solutions via an online rounding of the fractional solution. This part of the framework is problem dependent, and applies various tools including results on approximate max-flow min-cut for multicommodity flow, the Hierarchically Separated Trees (HST) method and its extensions, certain rounding techniques for dependent variables, and Räcke's new hierarchical decomposition of graphs.

Specifically, our results for the integral case include an $O(\log m \log n)$-competitive randomized algorithm for the online non-metric facility location problem and for a generalization of the problem called the multicast problem. In the non-metric facility location problem, 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.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; F.1.2 [**Modes of Computation**]: Online computation

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Online network optimization, competitive analysis, facility location, group Steiner, multi-cuts, randomized rounding

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 $c : E \rightarrow \mathbb{R}^+$, and a requirement function f (to be defined for each problem separately). The goal is to find a minimum cost subgraph that satisfies the requirement function. 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 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 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 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 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 the multiway cut problem and the multicut problem (see e.g., [Vazirani 2001]).

There is a natural linear programming 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 programming relaxation is very useful for computing (offline) an approximate solution. Please refer to [Vazirani 2001] for more details. We note that fractional solutions have a motivation of their own in certain network design problems and bandwidth allocation problems (see, for example, [Plotkin et al. 1995]).

1.1 Previous work

Network optimization problems in an online setting have been studied extensively. The online Steiner problem was considered in [Imase and Waxman 1991] 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 [Awerbuch et al. 1996], where an $O(\log^2 n)$-competitive algorithm is given. This was improved to an $O(\log n)$-competitive ratio algorithm by [Berman and Coulston 1997]. The online version of the *metric* facility location problem was also considered recently. Meyerson [Meyerson 2001] 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 [Fotakis 2003]. Recently, a deterministic $O(\log n \log m)$-competitive algorithm for the online set cover problem was given by [Alon et al. 2003] where n is the number of elements and m is the number of sets. An almost matching lower bound of $\Omega(\frac{\log n \log m}{\log\log m + \log\log n})$ on the competitive factor was also shown for any deterministic algorithm for the online set cover problem [Alon et al. 2003].

There is a vast literature on efficient (offline) approximation algorithms for problems involving connectivity and cuts. The reader is referred to [Hochbaum 1997; Vazirani 2001] for more details.

1.2 Results

We study generalized connectivity and cuts problems in a unified framework. The basic idea is to first compute a fractional solution online and then round this solution to an integral one in an online fashion. We provide a general deterministic procedure that computes a near-optimal fractional solution to any problem belonging to our class of problems. The fractional solution is competitive with respect to an optimal fractional solution. 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, where the cost function is defined on the vertices of the graph rather than the edges, and to directed graphs. We also show a matching lower bound of $\Omega(\log m)$ on the competitive ratio of any deterministic or randomized algorithm for this problem.

We next describe our results on converting a fractional solution into an integral solution. This rounding is problem dependent and we describe the rounding for each of the special cases considered. Note that the rounding phase should also be done in an online fashion. Since the fractional solution produced by our algorithm is competitive with respect to an optimal fractional solution, the competitive ratio of the algorithms we propose for all the online integral graph problems is also analyzed with respect to an optimal fractional solution. The cost of the optimal fractional solution bounds from below the cost of any integral solution.

The first problem we consider is the *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 version of the non-metric facility location, clients arrive online. The online algorithm may choose to open additional facilities in order to serve all currently asked clients. However, the online algorithm is not allowed to close any previously opened facility. 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 clients. Each client is associated with at most one vertex in each tree. The goal is to find a minimum weight set of subtrees that contain all the clients, where a subtree must contain the root of the tree it belongs to. In the online case, the clients arrive online, and upon arrival of a client it is necessary to connect the client to some root of a tree containing it. The online algorithm may choose additional edges to its solution, but may not remove any previously chosen edges from the solution. 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 as well as on general graphs [Garg et al. 2003]. 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 weight 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 and, again, the algorithm may not remove any previously chosen edges from its solution. 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 vertices 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 multicut 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. Again, the algorithm is not allowed to cancel the choice of any previously chosen edge. We show an online algorithm for the multicut problem using the constructive version of a remarkable result of Räcke [Räcke 2002] for the hierarchical decomposition of graphs ([Bienkowski et al. 2003] and [Harrelson et al. 2003]) together with an approximate max-flow min-cut theorem on trees [Garg et al. 1997]. This decomposition is used along with an online primal-dual algorithm for the problem on trees. We note that our algorithm for the online multicut problem is not based on an online rounding of a fractional solution.

We summarize the results obtained:

—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 [Bartal 1996; Fakcharoenphol et al. 2003]

—A deterministic $O(\log^3 n \log\log n)$ competitive algorithm for the online multicut problem in general graphs. Improved bounds are obtained for planar graphs and for trees.

Our algorithms draw on ideas taken from the algorithm of [Alon et al. 2003] for the online set cover problem. We note that the idea of generating a fractional solution online and then rounding it is implicit in the work of [Alon et al. 2003]. However, unlike [Alon et al. 2003], we provide a tighter analysis of the algorithms comparing their performance to an optimal fractional solution as opposed to an optimal integral solution.

1.3 Organization

In Section 2 we formally define the problems and the online setting. In Section 3 we provide a deterministic algorithm for computing a near-optimal fractional solution for both generalized connectivity (Section 3.1) and generalized cuts (Section 3.2). Matching lower bounds for both problems are given in Section 3.3.

In Section 4 we present a general approach for solving online integral connectivity and cuts problems. In the context of connectivity problems we study the online non-metric facility location problem and the multicast problem in Section 4.1. We then study the group Steiner problem on trees in Section 4.2. The group Steiner problem on general graphs is considered in Section 4.3. In section 4.4 we design a deterministic algorithm for the online multicut problem . Finally, in Section 5 we conclude and discuss a few open questions.

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. We say that there is a flow from A to B of value α if there exists a valid flow function 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 $S_i \subset V$, $T_i \subset V$ and $S_i \cap T_i = \emptyset$.

We first define the *generalized connectivity* problem. The input for the problem is a graph $G = (V, E)$ with cost function $c : E \to \mathbb{R}^+$ and a requirement function. 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 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 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 for this problem is again a graph $G = (V, E)$ with cost function $c : E \to \mathbb{R}^+$ and a requirement function. 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. However, the algorithm is not allowed to decrease the weight of an edge. Thus, previous demands remain satisfied. The performance of the algorithm is measured with respect to the ratio between the cost of the solution produced by the algorithm and the cost of an optimal fractional solution that satisfies the given demands with minimum cost. The *competitive ratio* of the algorithm is defined to be the supremum of this ratio, taken over all possible input sequences. The optimal fractional solution, which bounds from below the optimal integral solution, is also used to analyze the competitive ratio of the algorithms we propose for integral problems. We next define the special cases that we consider in the context of generalized connectivity.

The *multicast* problem in trees is defined as follows: Let $X = \{c_1, c_2, \ldots, c_n\}$ be a ground set of clients, 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 of each tree in $\mathcal{T}$. Each tree leaf is associated with a subgroup of the clients, where any client c_i belongs to at most one leaf in each of the trees. Note that we may assume without loss of generality that only leaves of the trees are associated with clients. This can be achieved easily by connecting each vertex in the trees which is associated with a client to a new leaf with a zero cost edge. To see that the multicast problem is a special case of the generalized connectivity problem, connect all the roots of the trees in $\mathcal{T}$ to a joint root r with zero cost edges. A request for a client c is of the form (r, U_c), where U_c is the set of leaves associated with client c.

Let $X' \subseteq X$ be the set of requested clients. A *cover* is a collection of rooted subtrees $\mathcal{T}' = \{T'_1, T'_2, \ldots, T'_m\}$, where $T'_i \subseteq T_i$ $(1 \leq i \leq m)$, 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. In the online setting the algorithm is allowed at any step to choose additional edges to its solution in order to serve the clients. It may not, however, remove any previously chosen edge from its solution. We note that the multicast problem in trees has an (offline) $O(\log n)$-approximation algorithm.

The multicast problem generalizes the set cover problem. To see this, think of each set as being represented by a tree containing only one edge, where one vertex is a root and the other vertex is a leaf. The cost of each edge equals the cost of the set and the leaf is associated with the elements belonging to the set. The *non-metric facility location* problem is also 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 $C = \{c_1, c_2, \ldots, c_n\}$ and a connection cost function $c : F \times C \to \mathbb{R}^+$ denoting the cost of connecting each client to each facility. A feasible 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 sum of the connection cost of the clients, as defined by the mapping of clients to the open facilities. The online algorithm is allowed to open new facilities, but it cannot close any previously opened facility. The non-metric facility location problem is a special case of the multicast problem. Think of each facility as being represented by a tree of depth two. The cost of the "root" edge is equal to the setup cost of the facility, and the cost of each "leaf" edge corresponding to a client equals 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 $g_1, g_2, \ldots g_k \subset V$. Let r denote the root of the tree T. The objective is to find a minimum cost rooted subtree T' that contains at least one vertex from each of the groups g_i, $1 \leq i \leq k$. That is, using the terminology of the generalized connectivity problem, each request is of the form (r, g_i). The group steiner problem is a generalization of the multicast problem. Given an instance of the multicast problem, as before, we connect the roots of all the multicast trees using zero cost edges to a joint root r. The group g_i, $1 \leq i \leq k$, is defined to be the set of leaves associated with the client c_i. Note 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 online setting of the group steiner problem the groups arrive one by one in an online fashion. The algorithm has to choose additional edges to its solution such that the solution contains at least one vertex from each group. The algorithm may not cancel the choice of any previously chosen edge.

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 [Garg et al. 2003]. In general (i.e., undirected) graphs, the best approximation factor known for the group Steiner problem is $O(\log^2 n \log k)$ by combining [Garg et al. 2003] with [Fakcharoenphol et al. 2003].

The *multicut* problem in undirected graphs is a special case of the previously defined generalized cuts problem, where the requirement function consists of source-sink pairs $\{s_i, t_i\}$, $1 \leq i \leq k$. The pairs arrive one by one in an online fashion, and the algorithm has to disconnect each pair upon arrival. The online algorithm is not allowed to cancel the choice of any previously chosen edge. The best offline approximation factor for this problem is $O(\log k)$ [Garg et al. 1996].

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, during the run of the algorithm, it turns out that the value of the optimal solution is larger than our current guess for α, (that is, the cost of the fractional solution exceeds $\Theta(\alpha \log m)$), or we are unable to satisfy some demand, then we can "forget" about all weights given so far to the edges, update the value of α by doubling it, and continue on. The constants hidden inside the $\Theta(\alpha \log m)$ are known through the competitive analysis of the performance of the algorithm that uses the value α (Section 3.1). Recall that the online algorithm is not allowed to decrease the weights already given to the edges. Thus, by "forgetting" the weights of the edges, we mean that in any iteration the algorithm only uses the current set of weights for the edges in order to satisfy the demands. However, at any point of time, the actual weight of each edge is the maximal weight that it has received in any iteration up to the current one. We note that the fractional cost of the weights 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 next claim that since the value α is known, we may also assume that all edges have cost between 1 and m. To achieve this property we observe that in a fractional optimal solution, no edge with cost larger than α gets a positive weight. Assume to the contrary that in an optimal solution, edge e has weight $w(e) > 0$ while $c(e) > \alpha$. Note that $w(e) < 1$, since otherwise the cost of the solution is larger than α. We now show how to obtain a new feasible solution w' with cost strictly less than α, contradicting the optimality of OPT. For edge e, let $w'(e) \leftarrow 0$, and for $e' \neq e$, let $w'(e') \leftarrow \frac{w(e')}{1-w(e)}$. It is not hard to see that this yields a feasible fractional solution having cost:

$$\frac{\alpha - c(e)w(e)}{1 - w(e)} < \frac{\alpha(1 - w(e))}{1 - w(e)} = \alpha.$$

Thus, our algorithm may "ignore" all edges with cost greater than α without increasing the value of the optimum. For the generalized connectivity problem, ignoring means that we do not use these edges to connect vertices, and hence these edges are removed from the graph. For the generalized cuts problem ignoring means that such edges do not participate in a cut, and hence we merge the two endpoints of any such edge. Next we take to our solution all edges having cost less than α/m paying at most an additive factor of α. Thus, the cost of all the edges that are left in the graph is between α/m and α, and the costs can further be normalized so that the minimum cost is 1 and the maximum cost is at most m.

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 $1/(m^2)$, and thus the total initial cost is less than 1. Assume now that the algorithm is given 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 weight 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 1. *When the algorithm terminates, all connectivity demands are satis-*

fied.

Proof. Follows immediately from the algorithm. □

Lemma 2. *The number of weight augmentation steps performed during the run of the algorithm is at most*

$$2\alpha \log_2 m + \alpha = O(\alpha \log m).$$

Proof. Obviously, for each edge $e \in E$, $w_e \leq 1 + \frac{1}{c_e}$ always holds, since an edge with weight exceeding 1 cannot be part of a minimum weight cut having 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 show three properties of Φ:

—The initial value of the potential function is: $-2\alpha \log_2 m$.

—The potential function never exceeds α.

—In each weight augmentation step, the potential function increases by at least 1.

The first property follow directly from the initial value of the variables. The second property follows by the observation that no edge gets a weight of more than 2. Consider an iteration in which the adversary gives a connectivity demand (S, T) and a weight augmentation of a cut $\mathcal{C}$ is performed. The total weight assigned by OPT to edges in $\mathcal{C}$ is at least 1. Thus, the increase of the potential function in a single weight augmentation 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. *The algorithm is $O(\log m)$-competitive for the fractional generalized connectivity problem.*

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

$$\sum_{e \in E} w_e c_e \leq 2\alpha \log_2 m + \alpha + 1 = O(\alpha \log m).$$

Consider an iteration in which a connectivity demand (S, T) is given. Let $\mathcal{C}$ be a cut whose weight is augmented. The weight of $\mathcal{C}$ is 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 single 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, $\sum_{e \in E} w_e c_e \leq m \cdot \frac{1}{m^2} \cdot m = 1$, and the theorem thus follows from Lemma 2 that bounds the number of weight augmentation steps. $\square$

We remark again that the above analysis assumes that the value of α is fixed. To get the total cost of the algorithm, taking into account the doubling process, one should multiply the cost by 2.

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 $1/(m^2)$ and thus the total initial cost is less than 1. Assume now that the algorithm is given a new request (S, T). The following is performed in this case.

```
(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 of Theorem 3. Hence we conclude:

THEOREM 4. *The algorithm is $O(\log m)$-competitive for the fractional generalized cut problem.*

3.3 Lower Bounds

In this section we show that our algorithm for the fractional generalized connectivity and generalized cuts problems is optimal up to constant factors. To this end 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 on the generalized cuts problem. In the randomized case, our lower bounds hold with respect to an oblivious adversary. An oblivious adversary must construct the request sequence in advance based on the description of the randomized algorithm, but without knowing the actual random choices made by the algorithm.

LEMMA 5. *For any deterministic or randomized algorithm for the online fractional connectivity problem, the competitive ratio is 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. Consider a star with $n = 2^k$ leaves (and edges) $v_1, v_2, \ldots, v_n$ and a root r. We describe the strategy of the adversary. In iteration 0 the demand is $S = \{r\}$, $T = (v_1, v_2, \ldots, v_n)$. The algorithm must increase the flow from the root to all the leaves to at least 1. Therefore, the expected flow to either the first $n/2$ vertices or the last $n/2$ vertices is at least $1/2$. In the next iteration the adversary changes T to be the set with the smaller expected flow value between $(v_1, v_2, \ldots, v_{n/2})$ and $(v_{(n/2)+1}, \ldots, v_n)$. 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 smaller expected flow. Thus, it is not hard to see that the expected cost of the algorithm is at least,

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

An 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 6. *For any deterministic or randomized algorithm for the online fractional cuts problem, the competitive ratio is 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 a line and the cut demands are sets 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 next describe the strategy of the adversary. In iteration 0 the adversary asks the demand $(\{v_1\}, \{v_n\})$. The algorithm must increase the distance from v_1 to v_n to be 1. Thus, 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. 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 with 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 the algorithm 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. □

Note that the adversary we described in the proof of Lemma 5 produces demands that are sets of vertices. Indeed, the proof of the lower bound actually applies to any generalization of the fractional set cover problem, e.g., online fractional versions of the non-metric facility location problem, the multicast problem, and the group Steiner problem. However, the fractional online Steiner tree problem, as well as the fractional online generalized Steiner tree problem, do not generalize the set cover problem, and therefore our lower bounds are not applicable to these problems. 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) online Steiner tree problem in [Imase and Waxman 1991].

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. This can be done by an online rounding of the fractional solution generated by the algorithm described in the previous section. This is the heart of our general approach to online network optimization problems. The algorithms we propose in this section use the online algorithms for generating a fractional solution as a "black box". We present here four problems for which such an online 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 multicut problem in Section 4.4. In the online multicut problem we only use the above approach implicity to solve the problem for trees. For general graphs we propose a different algorithm that leads to a better competitive ratio.

4.1 Multicast and Non-metric Facility Location Problems.

In this section we describe a randomized algorithm for the non-metric facility location problem and the multicast problem. As the non-metric facility location problem is a special case of the multicast problem, we only describe the algorithm for the multicast problem. Our algorithm first uses the algorithm presented in section 3.1 to generate a fractional solution in an online fashion. A fractional solution to the multicast problem is an assignment of weights to the edges of the trees. For each client that is requested, a feasible fractional solution guarantees that the total amount of flow that can be sent separately from the roots of the trees in $\mathcal{T}$ to the vertices that are associated with the client is at least 1. Furthermore, the total cost of the solution it generates is at most $O(\log m)$ times the optimal fractional cost. The fractional weights given to the edges by the online algorithm may only increase during the run of the algorithm.

To produce a feasible integral solution, we show how to round the fractional solution in an online fashion. We propose a randomized rounding method having a small expected cost that will cover with high probability every requested client. Specifically, each requested client will be covered by our randomized rounding method with probability at least $1 - 1/(n')^2$, where n' is the number of clients. In case some client is not covered by the randomized algorithm, we serve the client by the cheapest path from a root of any tree in $\mathcal{T}$ to the client. This path is a lower bound on the value of the optimum. Since this event happens with probability at most $1/n'^2$ for each client, the total effect on the expected cost of the algorithm is negligible. In the following we propose an online randomized rounding method and analyze its performance.

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 number of clients that have arrived so far is denoted by n'. Define the threshold of a tree T_i to be:

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

The algorithm includes in the solution $\mathcal{C}$ all edges e with $w_e > \theta(T_e)$, where T_e is the tree containing edge e. That is, we take to the solution an edge e if its weight exceeds the threshold of the tree containing it. As the value of n' increases, the algorithm also increases the number of random variables. Thus, the threshold of a tree may decrease during the run of the algorithm. In the latter case, the algorithm reconsiders all previously considered edges in the trees, and adds edges that now exceed the (new) threshold. Let α be the value of an optimal fractional solution to the instance given so far. The next lemma analyzes the expected cost of the randomized algorithm and its success probability.

LEMMA 7. *The following holds throughout the run of the algorithm:*

(1) The expected cost of the solution produced by the algorithm is $O(\alpha \log n' \log m)$.

(2) Any client that is covered fractionally is also covered by the randomized algorithm with probability at least $1 - 1/n'^2$.

PROOF. We start with the first part of the lemma. For each edge e and index j, $1 \le j \le 2\log n$, let $Y(e, j)$ be the indicator of the event that $w_e > X(T_e, j)$, i.e., edge e is chosen to the solution due to the random variable $X(T_e, j)$. Since $X(T_e, j)$ is chosen uniformly in the interval $[0, 1]$, the probability and the expectation of the indicator $Y(e, j)$ is at most w_e. Note that each edge could be chosen to the solution due to several random variables. Thus, we can bound the expected cost of the solution as follows:

$$\begin{aligned}
\mathbf{Exp}\left[\sum_{e\in\mathcal{C}} c_e\right] &\le \sum_{e\in \mathrm{E}} \sum_{j=1}^{2\lceil\log(n'+1)\rceil} c_e \cdot \mathbf{Exp}[(Y(e,j)] && (1)\\
&\le \sum_{e\in \mathrm{E}} \sum_{j=1}^{2\lceil\log(n'+1)\rceil} c_e w_e && (2)\\
&\le 2\lceil\log(n'+1)\rceil(4\alpha\log m + 2\alpha + 2) && (3)\\
&= O(\alpha \log n' \log m).
\end{aligned}$$

Where, Inequality (1) follows by a simple union bound, Inequality (2) follows by the bound we stated on the expectation of the indicator $Y(e, j)$, and Inequality (3) follows from the guarantee on the performance of the online fractional algorithm.

We now prove the second part of the lemma. Consider a client c. 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 leaves that are associated with c is at least 1. Let $f^T(c)$ be the flow to the leaf associated with client c in tree T. Note that the weight of each edge on the path to the leaf is at least $f^T(c)$. Thus, the probability that client c is not covered is bounded from above by the probability that the threshold of each of the trees T that contain c is larger than $f^T(c)$.

For each tree T and index j, $1 \le j \le 2\lceil\log(n'+1)\rceil$, the probability that the flow to the leaf associated with c in T is at most $X(T, j)$ is $1 - f^T(c)$. Since the random variables $X(T, j)$, for fixed j, are independent, the probability that client c is not covered by any of the trees due to index j is at most

$$\prod_{T \in \mathcal{T}} (1 - f^T(c)) \leq e^{-\sum_{T \in \mathcal{T}} f^T(c)} \leq \frac{1}{e}.$$

The last inequality follows from the fact that $\sum_T f^T(c) \geq 1$ for each requested client. Since the random variables $X(T, j)$, for fixed T and $1 \leq j \leq 2\lceil \log(n' + 1) \rceil$, are independent, the probability that client c is not covered due to any of these random variables is less than $\left[\frac{1}{e}\right]^{2 \log(n'+1)} \leq 1/n'^2$. $\square$

By the above analysis, including the change made to the algorithm to guarantee its feasibility, we get the following theorem:

THEOREM 8. *There is an $O(\log n' \log m)$ competitive randomized algorithm for the online multicast problem in trees.*

Since the non-metric facility location problem is a special case of the multicast problem we also get:

THEOREM 9. *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 multicast problem in trees and the online non-metric facility location are generalizations of the online set-cover problem introduced in [Alon et al. 2003]. Thus, the lower bound of $\Omega(\frac{\log n \log m}{\log \log m + \log \log n})$ proved in [Alon et al. 2003] for any deterministic algorithm for the online set-cover problem applies to these problems as well.

4.2 The Group Steiner Problem on Trees

In this section we describe a randomized algorithm for the online group Steiner problem on trees. We first generate a fractional solution to the problem online. We now explain how the rounding of the fractional solution is performed in an online fashion. To this end, we use an online variation on the method of [Garg et al. 2003].

The randomized rounding method we propose covers each group with probability $\Omega(1/\log N)$, where N is the maximum size of any group. In addition, its expected cost is at most the cost of the fractional solution. We then run $O(\log k \log N)$ independent trials of this randomized rounding method in parallel, where k is the number of groups asked by the adversary. The algorithm takes to the solution each edge that was selected in any of the trials. Using simple probabilistic analysis we get that our algorithm has a competitive ratio of $O(\log n \log k \log N)$ and each of the groups is covered with probability at least $1 - 1/k$. In order to guarantee that the algorithm produces a feasible solution, we can use the shortest path to a group in case the algorithm fails to cover the group. The cost of this path is certainly 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, similarly to Section 4.1. Next, we propose an online randomized rounding method and analyze its performance.

Initially, the algorithm starts with an empty cover $\mathcal{C} = \emptyset$. Applying the technique of [Garg et al. 2003] requires that the fractional weights on a path from the root to any vertex are monotonically decreasing. However, the fractional solution that our algorithm computes may not necessarily satisfy this property. Therefore, we define the weight of each edge to be the maximum flow that can be routed through this edge to any vertex in in its subtree. In the following we abuse notation and let w_e denote the flow on edge e, instead of the actual weight of e. Since the flow routed on each edge is at most its weight, we note that this can only decrease the fractional value of the solution serving as our baseline for bounding the competitive analysis.

Consider an iteration in which the fractional weight of some edges is augmented. Since the weight of an edge is the maximum flow that can be routed through it, the fractional weight of an edge can be augmented when the algorithm augments the weights of other edges as well. If the weight of several edges is augmented at the same iteration, the rounding algorithm considers the edges one by one, according to a topological ordering, starting from the edges closer to the root. Let w_e and $w'_e = w_e + \delta_e$ be the weight of edge e before and after the weight augmentation, respectively. Let δ_e be the change in the weight of e. Let $e(p)$ be the edge adjacent to e and closer to the root r. This definition is, of course, only relevant if the edge e is not incident on the root r. The rounding algorithm randomly chooses the edges to the solution by the following scheme.
Consider all edges for which $\delta_e > 0$ in any topological order:

—If $w'_e > 1$, add e to $\mathcal{C}$.

—If e is incident on r, or $w'_{e(p)} > 1$, add e to $\mathcal{C}$ with probability $\delta_e/(1 - w_e)$.

—If $e(p) \in \mathcal{C}$, add e to $\mathcal{C}$ with probability $\delta_e/(w'_{e(p)} - w_e)$.

Note that for each edge e that is not incident on the root, $\delta_e/(w'_{e(p)} - w_e) \leq \delta_e/(w'_e - w_e) = 1$, since $w'_e \leq w'_{e(p)}$. Thus, the probabilities are well defined. Furthermore, note that $\mathcal{C}$ induces a connected subtree of T. This follows since the edges that were augmented at the same iteration are considered in topological order and each edge may be added to $\mathcal{C}$ only if the path connecting it to the root r is already in $\mathcal{C}$. The following lemma proves a basic important property of the randomized rounding method.

LEMMA 10. *For each edge e, at the end of each iteration, the probability that $e \in \mathcal{C}$ is w'_e. If $w_e > 1$, then $e \in \mathcal{C}$ with probability* 1.

PROOF. The second part of the lemma is trivial since the rounding algorithm adds each edge e with weight $w_e > 1$ to $\mathcal{C}$. We prove the first part of the lemma by induction on the iterations of the algorithm.
Induction Basis: Before the first iteration the weight of all edges is zero and $\mathcal{C}$ is empty.
Inductive Step: Suppose the lemma holds for the first i iterations and consider now iteration $i + 1$. Any edge whose weight did not change during iteration $i + 1$ is added to the solution with probability 0, and so the lemma holds for such edges by the induction hypothesis. Consider an edge e whose weight changed during iteration $i+1$. Let $\delta_e = w'(e) - w(e)$. If $w'(e) \geq 1$, then the edge e is chosen to the

solution with probability 1. Assume the weight of edge e at the end of iteration $i+1$ is strictly less than 1. The weights of edges on each path emanating from the root are monotonically decreasing, therefore the edges with weight exceeding 1 form a connected subtree A. The subtree A is taken to the solution, by the topological ordering, before any other edge (with weight strictly less than 1) is considered by the algorithm in iteration $i+1$. Let $e(0), e(1), \dots e(k) = e$ be the path from subtree A to edge e. All the edges on the path have weights $w'(e(i)) < 1$ and their weights are monotonically decreasing.

The algorithm process the edges on this path according to a topological order starting from the edges closer to the root. By the induction hypothesis, the probability that each edge $e(i)$ $(1 \leq i \leq k)$was chosen to the solution before iteration $i+1$ is $w(e(i))$. We wish to prove that each edge on the path is chosen to the solution at the end of iteration $i+1$ with probability $w'(e(i))$. We can prove this claim by a direct argument, but it is easier to prove the claim by induction on the length of the path.

According to the algorithm, the probability that $e(0) \in \mathcal{C}$ at the end of iteration $i+1$ is:

$$w_{e(0)} + \frac{\delta_{e(0)} \cdot (1 - w_{e(0)})}{(1 - w_{e(0)})} = w_{e(0)} + \delta_{e(0)} = w'_{e(0)}.$$

The first term is the probability that $e(0)$ is already in the solution before iteration $i+1$, while the second term is the probability that edge $e(0)$ did not belong to the solution before iteration $i+1$, yet was chosen to the solution in iteration $i+1$. We now assume inductively that edge $e(k-1)$ is in the solution at the end of iteration $i+1$ with probability $w'(e(k-1))$. Since the algorithm considers the edges in topological order this assumption holds at the time edge $e(k)$ is considered in iteration $i+1$. Thus, just before edge $e(k)$ is considered in iteration $i+1$, the probability that $e(k) \notin \mathcal{C}$ and $e(k-1) \in \mathcal{C}$ is $w'_{e(k-1)} - w_{e(k)}$. The probability that $e(k)$ belongs to the solution at the end of iteration $i+1$ is therefore:

$$w_{e(k)} + \frac{\delta_{e(k)} \cdot (w'_{e(k-1)} - w_{e(k)})}{(w'_{e(k-1)} - w_{e(k)})} = w_{e(k)} + \delta_{e(k)} = w'_{e(k)}.$$

The first term is the probability that $e(k)$ was added to $\mathcal{C}$ before iteration $i+1$, while the second term is the probability that $e(k)$ did not belong to the solution before iteration $i+1$, yet e_{k-1} belongs to $\mathcal{C}$, and $e(k)$ is added to $\mathcal{C}$ in iteration $i+1$. $\square$

The next lemma follows from linearity of expectation.

LEMMA 11. *At the end of each iteration, the expected cost of the edges in $\mathcal{C}$ is at most $\sum_{e \in \mathcal{T}} c_e w'_e$, where w'_e is the weight of edge e at the end of the iteration.*

Let N be the maximum size of a group $g = \{\{v_1, v_2, \dots, v_k\}$. Let w_g be the total flow that can be routed to the vertices in g simultaneously. The next lemma bounds from below the probability that any group g with $w_g > 1$ is covered.

LEMMA 12. *In any iteration, if, for a group $g = \{v_1, v_2, \dots, v_k\}$, $w_g \geq 1$, then the probability that there exists $v_i \in \mathcal{C}$ $(1 \leq i \leq k)$ is $\Omega(1/\log N)$.*

PROOF. Our proof uses [Garg et al. 2003, Thm. 3.4, p. 72]. This requires proving that in our rounding, the probability of the "good" events and the dependency between them remains the same as in the offline rounding of [Garg et al. 2003]. The first claim follows from Lemma 10. 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 [Garg et al. 2003]. Let $e(1)$ and $e(2)$ be two edges in the tree. Let e be the least common ancestor edge of $e(1)$ and $e(2)$ (with respect to the root). If there is no such edge e, then the events that $e(1)$ and $e(2)$ are chosen to the solution are independent. Otherwise, the probability that $e(1) \in \mathcal{C}$ given that $e \in \mathcal{C}$ is $w_{e(1)}/w_e$, and the probability that $e(2) \in \mathcal{C}$ given that $e \in \mathcal{C}$ is $w_{e(2)}/w_e$. Given that e is chosen to the solution, the events $e(1) \in \mathcal{C}$ and $e(2) \in \mathcal{C}$ are independent. Thus, the probability that $e(1)$ and $e(2)$ are in $\mathcal{C}$ is $w_{e(1)}w_{e(2)}/w_e$, and we are done.

Actually, in order to use the original proof of [Garg et al. 2003], we are required to prove a stronger assertion about the independence of the corresponding events. However, this is not needed, since the proof of [Garg et al. 2003] 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 [Alon and Spencer 2000, Sec. 4.8, Ex. 1]. Therefore, the above independence result suffices. $\square$

To conclude, we state the performance of the randomized algorithm for the online group steiner on trees.

THEOREM 13. *There is a randomized online algorithm for the group Steiner problem in trees with a competitive ratio of* $O(\log^2 n \log k)$.

4.3 The Group Steiner Problem on General Graphs

In this section we consider the group Steiner tree problem on general graphs. The algorithm for the group Steiner tree problem on general graphs uses hierarchically well-separated trees (HST-s) [Bartal 1996; Fakcharoenphol et al. 2003]. 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 $x, y \in V$, $E\left[d_D(x,y)\right] \leq \alpha d_{\mathcal{M}}(x,y)$. We define α to be the stretch factor. Recently, the following theorem was proved in [Fakcharoenphol et al. 2003], improving upon the initial result of [Bartal 1996].

THEOREM 14. *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 an online randomized algorithm for the group Steiner tree problem on general graphs with the following competitive ratio.

THEOREM 15. *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 14 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 α of the paths, and the theorem follows directly from Theorem 14, and the guarantee on the performance of the algorithm in Section 4.2. □

4.4 The multicut problem

In this section we consider the online multicut problem in undirected graphs. The algorithm we propose for the online multicut problem on trees fits our general approach. However, for general graphs we propose a different algorithm that deviates from the general framework developed in the paper.

In [Räcke 2002], 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 [Bienkowski et al. 2003] and [Harrelson et al. 2003]. 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 multicut problem on trees with competitive ratio α, where α may depend on the height of the tree.

The vertices of the decomposition tree T_G correspond to a laminar family of subsets of V [1]. 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 in T_G can be routed in G causing a congestion of at most β. The current 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 [Harrelson et al. 2003] together with a polynomial-time construction of T_G.

Thus, the multicut problem in G translates into a multicut 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) multicut problem in T_G. A multicut 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 multicut in T_G. Clearly, this translation can be done online.

THEOREM 16. *There is a deterministic polynomial-time algorithm for the online multicut 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. Let $\mathcal{C}_{onl}(G)$ and $\mathcal{C}_{onl}(T_G)$ denote the multicut found by the online algorithm in G and in T_G, respectively. Let $\mathcal{C}_{opt}(T_G)$ denote the optimal multicut in T_G, and let $\text{MCF}_{opt}(T_G)$ be the maximum multi-commodity flow in T_G between

[1]A collection of subsets of V forms a laminar family if no two sets in the collection cross.

the source-sink pairs. By [Garg et al. 1997], 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 multicut 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. $\square$

We now proceed and show an online algorithm for the multicut problem in trees. First, note that there is a simple reduction from the online multicut 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 [Alon et al. 2003], (which follows the basic general approach developed here), there is a deterministic $O(\log^2 n)$-competitive algorithm for the online minimum multicut tree problem.

The above reduction applies to any tree. However, when considering the decomposition trees produced by [Harrelson et al. 2003], 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 [Garg et al. 1997]. 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 multicut 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.

5. CONCLUDING REMARKS

We described a general method for generating fractional solution for connectivity and cuts problems in an online fashion. This, together with online versions of randomized rounding, yields competitive online algorithms for many integral connectivity problems.

Specifically, we described randomized online algorithms for the non-metric facility location problem, the multicast problem, and the group Steiner problem. The question of providing deterministic algorithms for these problems remains open. In the offline case, all the randomized rounding methods that were proposed for these problems can be derandomized using the method of conditional expectations. However, in the online setting we do not know whether it is possible to obtain deterministic algorithms for these problems. We believe that this should be possible, at least for the non-metric facility location problem.

For many offline problems, randomized rounding methods were not studied since they involve high computational time complexity, mainly due to the necessity to solve a linear program. For example, generating an $O(\log n)$ approximation to the set cover problem via randomized rounding is possible, but it is certainly "cheaper" to apply the usual greedy approach. Our approach for the design of online algorithms motivates the study of randomized rounding methods for such problems since they might serve as a basis for online algorithms.

REFERENCES

Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N. and Naor, J. 2003. The online set cover problem. In *35th Annual ACM Symposium on the Theory of Computation.* 100–105.

Alon, N. and Spencer, J. H. 2000. *The Probabilistic Method.* John Wiley and Sons, Inc., 2nd Edition, 2000.

Awerbuch, B., Azar, Y., and Bartal, Y. 2001. On-line generalized Steiner problem. In *7th Annual ACM-SIAM Symposium on Discrete Algorithms.* 68–74.

Bartal, Y. 1996. Probabilistic approximation of metric spaces and its algorithmic applications. In *37th Annual IEEE Symposium on Foundations of Computer Science.* 184–193.

Berman, P. and Coulston, C. 1997. On-line algorithms for Steiner tree problems. In *29th Annual ACM Symposium on the Theory of Computation.* 344–353.

Bienkowski, M., Korzeniowski, M., and Räcke, H. 2003. A practical algorithm for constructing oblivious routing schemes. In *15th ACM Symposium on Parallelism in Algorithms and Architectures.* 24–33.

Fotakis, D. 2003. On the competitive ratio for online facility location. In *30th International Colloquium on Automata, Languages and Programming.* 637–652.

Fakcharoenphol, J., Rao, S., and Talwar, K. 2003. A tight bound on approximating arbitrary metrics by tree metrics. In *35th annual ACM Symposium on Theory of Computation.* 448–455.

Garg, N., Konjevod, G., and Ravi, R. 2003. A Polylogarithmic approximation algorithm for the group Steiner tree problem. *Journal of Algorithms 37*, 66–84.

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

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

Harrelson, C., Hidrum, K., and Rao, S. 2003. A polynomial time tree decomposition to minimize congestion. In *15th ACM Symposium on Parallelism in Algorithms and Architectures.* 34–43.

Hochbaum, D. S. 1997. *Approximation Algorithms.* PWS Publishing Company, 1997.

Imase, M. and Waxman, B. M. 1991. Dynamic Steiner tree problem. *SIAM J. on Discrete Math 4*, 369–384.

Meyerson, A. 2001. Online facility location. In *42nd Annual IEEE Symposium on Foundations of Computer Science.* 426–431.

Plotkin, S. A., Shmoys, D., and Tardos, E. 1995. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research 20*, 257–301.

Räcke, H. 2002. Minimizing congestion in general networks. In *43rd Annual IEEE Symposium on Foundations of Computer Science*. 43–52.

Vazirani, V. V. 2001. *Approximation Algorithms*. Springer-Verlag, 2001.

RECEIVED ; REVISED ; ACCEPTED