TOWARDS A UNIFIED ANALYSIS FRAMEWORK FOR ONLINE NETWORK DESIGN

By

Seeun William Umboh

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2015

Date of final oral examination: 05/13/2015

The dissertation is approved by the following members of the Final Oral Committee:

Shuchi Chawla (Advisor), Associate Professor, Computer Sciences
Eric Bach (Coadvisor), Professor, Computer Sciences
Dieter van Melkebeek, Professor, Computer Sciences
Alberto Del Pia, Assistant Professor, Industrial and Systems Engineering
Mohit Singh, Researcher, Microsoft Research

To my love, Ka Yi.

Acknowledgments

My Ph.D. journey was full of challenges and I could not have made it without the support and encouragement of many people.

First and foremost, I am indebted to my advisor, Shuchi Chawla, for her guidance and insight. She also kindly took me along with her during her sabbatical at the University of Washington and Microsoft Research, Redmond. It was during this period that I developed the main ideas for this thesis and I am grateful that Shuchi gave me the freedom to pursue my own ideas. Moreover, I was spared the brutal polar vortex that blasted the midwest that winter! I also want to thank my coadvisor Eric, and the members of my dissertation committee Dieter, Mohit, and Alberto for their time and input. In particular, Mohit's comments greatly improved the exposition in my SODA paper and my thesis.

I am thankful to all my collaborators and coauthors: Shuchi Chawla, Siddharth Barman, David Malec, Mohit Singh, Debmalya Panigrahi, Anupam Gupta, R. Ravi, and David Williamson. It was a great pleasure working with you!

I am grateful to the UW theory students, current and past: Matt, Jeff, Siddharth, David, Dalibor, Tyson, Balu, Heng, Ben, and many others. In particular, I want to thank Siddharth. His excitement for network design heavily influenced my decision to work in this area. I learnt a ton from our collaborations and had a lot of fun (and coffee!) working together with him. The weekly theory lunches also provided a welcome respite from work.

Many thanks to my friends and family. I have fond memories of climbing and gorging on good food afterwards with Makrand during my one-year stay in Seattle. I am very grateful to have known the "Italianos": Alberto, Carla, Paola, Jeff, Lorenzo. The last few months of graduate school were some of the most stressful, and our

long weekly dinners and lively discussions about food and culture kept me sane during this period.

Last but not least, I want to thank my fiancee for her unconditional love and infinite patience during the past few years. I could not have made it without you!

Contents

List of Figures

Abstract

We consider network design problems in the online setting: the algorithm is given a graph with edge costs; requirements arrive over time, and the algorithm has to satisfy them as they arrive. The main contribution of this thesis is an analysis framework based on a novel application of tree embeddings that leads to a new approach for online network design: we use greedy-like algorithms but analyze them using tree embeddings. Using this approach, we improve upon previous work and obtain deterministic algorithms for a wide variety of problems that achieve the best possible competitive ratios, even when compared against randomized algorithms. We also apply our framework to two other settings of online network design: online multicast games where terminals represent selfish players, and to online network design problems where the requests arrive in random order.

1 Introduction

Network design concerns the following fundamental combinatorial optimization problem:

Problem 1.1. *Given a graph* G*, a set of requirements and an objective function, find an optimal network in* G *satisfying the requirements.*

A basic network design problem is the *Steiner tree problem*: given a graph G with edge costs and a distinguished subset of vertices called terminals, find a minimum-cost tree connecting all terminals. Network design problems often model optimization problems in real-life networks, such as telecommunication and transportation networks, capturing the essential combinatorial features and computational difficulties of their real-life counterparts. From a mathematical standpoint, the quest for efficient algorithms for network design has led to many beautiful theorems about the combinatorial structure of optimal solutions, such as the celebrated max-flow min-cut theorem.

This thesis focuses on online algorithms for network design problems. Typically, network design problems are studied in the *offline* setting in which all connectivity requirements are given in advance to the algorithm. However, in many practical situations, the requirements arrive over time and the network needs to grow to satisfy them as they arrive. For instance, telecommunications networks grow over time as new customers arrive, and the goal is to minimize the cost of adding new customers to the network. This is called the *online* setting.

An *online algorithm* receives the requirements over time, and constructs the network on-the-fly to satisfy the requirements (also called *requests*) as they arrive; furthermore, at any time, previous decisions are irrevocable—it is not possible

to recoup previously-incurred costs. *Competitive analysis* [ST84] is a standard approach to measure the quality of an online algorithm. In competitive analysis, the algorithm's solution is compared against the "optimal-in-hindsight" solution—the optimal solution found when all requirements are known in advance. An online algorithm has *competitive ratio* α if its solution costs no more than α of the optimal-in-hindsight solution. (For brevity, we will use the term "optimal solution" to refer to the optimal-in-hindsight solution.) A problem is said to have a competitive ratio of α if there exists an α-competitive online algorithm for it.

The most basic online network design problem is the *online Steiner tree problem* introduced by Imase and Waxman [IW91], who initiated the study of online network design. In this problem, the algorithm is given the underlying graph G and a distinguished terminal (called the root) in advance, but the rest of the terminals are revealed one-by-one. The algorithm maintains a subgraph H; when a terminal is revealed (we say that the terminal *arrives*), the algorithm must choose edges to add to H so that i is connected to the root in H. Furthermore, once the algorithm has chosen an edge, it cannot undo its decision in the future. After all the terminals have arrived, the algorithm's solution is compared against the optimal Steiner tree over those terminals.

In the online setting, the main challenge is having to commit to decisions without having complete information about the input. There are two fundamental questions in the study of any online problem.

Question 1. *What is the best-possible competitive ratio?*

It is natural to compare online algorithms against offline ones. Clearly, no online algorithm can produce a better solution than the best offline algorithm, which is given the entire input in advance. But can offline algorithms find the optimal solution

efficiently? It turns out that many interesting network design problems (including the Steiner tree problem) are NP-hard so it is unlikely—unless P = NP—that they admit efficient optimal algorithms. Thus, we compare online algorithms against approximation algorithms. An *approximation algorithm* (for cost-minimization problems) is a polynomial-time offline algorithm whose solution is guaranteed to cost no more than some factor of the optimal solution; this factor is called the *approximation ratio* of the algorithm. For any given problem, its approximation ratio is no worse than its competitive ratio since an online algorithm can be used as an approximation algorithm. The gap between the competitive ratio and the approximation ratio measures the handicap imposed by the online setting, and determining this gap is an important goal in the study of online algorithms.

Question 2. *What is the power of randomness?*

Randomness is a powerful resource for online algorithms. For many online problems, the best randomized algorithm has an expected competitive ratio that is much better than the competitive ratio of the best deterministic algorithm.[1] The main reason is that in the online setting, deterministic algorithms can be easily fooled into making decisions that look good at the time, but turns out to be highly suboptimal in hindsight.[2] On the other hand, truly random bits can be difficult to obtain, so we prefer good deterministic algorithms. Understanding the gap between deterministic and randomized online algorithms—the power of randomness—is also central to the study of online algorithms.

[1] One such example is the paging problem [FKL+91, ST84]. See [BEY05] for other examples.

[2] Roughly speaking, the adversary sets up an instance so that for each request, the algorithm has to choose from several options that appear equally good given previous decisions and requests; since the algorithm is deterministic, the adversary can simulate it to figure out which option it will choose and this allows the adversary to set up the remaining requests so that the algorithm's choice is the wrong one in hindsight. Randomized algorithms foil this adversarial strategy since the adversary cannot predict the algorithm's decisions.

Imase and Waxman [IW91] answered the above questions for the Steiner tree problem: the gap between online and offline algorithms is $\Theta(\log k)$, where k is the number of terminals, and randomization does not help. In the offline setting, there is a simple 2-approximation based on a reduction to the minimum spanning tree problem (c.f. [Vaz01, Chapter 3]), and the current best is a 1.39-approximation by Byrka et al. [BGRS13]. In the online setting, Imase and Waxman [IW91] showed that the natural greedy algorithm—connect the current terminal to the nearest previous terminal—is $O(\log k)$-competitive. They also proved a matching lower bound: every online algorithm, even randomized ones, has a competitive ratio of $\Omega(\log k)$.

Since the work of [IW91], other network design problems have been studied in the online setting. This thesis focuses on problems that admit an approximation factor preserving reduction to their metric versions. In the metric version of a problem, the input graph G is a complete graph and the edge costs form a metric—i.e. they satisfy the triangle inequality. These online problems include Steiner forest, prize-collecting Steiner forest, Steiner network with edge duplication, rent-or-buy, connected facility location. In each of these problems, each request is defined either over a terminal or a terminal pair. Henceforth, we use X to denote the set of terminals and $k = |X|$, while $n = |V|$, the total number of vertices.

These problems generalize the Steiner tree problem, so they do not admit algorithms with competitive ratio asymptotically better than $\log k$, even randomized ones. Previous work gave deterministic $O(\log k)$-competitive algorithms for Steiner forest [BC97] and prize-collecting Steiner forest [QW11]. However, for Steiner network with edge duplication, the current-best is a randomized $O(\log n)$-competitive algorithm is known (folklore); for rent-or-buy, the current-best is a randomized $O(\log k)$-competitive algorithm; and for connected facility, the current-best is a randomized $O(\log^2 k)$-competitive algorithm.

Problem	Previous Work	This Thesis
Steiner network with edge duplication	$O(\log n)$ rand.	$O(\log k)$ det.
Rent-or-Buy	$O(\log k)$ rand. [AAB04]	$O(\log k)$ det.
Connected facility location	$O(\log^2 k)$ rand. [SFWL14a]	$O(\log k)$ det.

Table 1.1: Our main results.

This thesis develops a new analysis framework for online network design problems. Using this framework, we improve upon previous work and devise deterministic $O(\log k)$-competitive algorithms for Steiner network with edge duplication, rent-or-buy, and connected facility location problems. Thus, our algorithms achieve the best possible competitive ratio, even when compared against randomized algorithms. Our analysis framework also yields simpler analyses of the algorithms of [BC97] and [QW11] for Steiner forest and prize-collecting Steiner forest, respectively.

Next, we discuss the dominant approaches in previous work to establish the context for our work. In the following, since we are focusing on metric versions of the above problems, we assume the input is a metric (V, d) instead of a graph G with edge costs.

1.1 Previous Approaches

The central task in designing approximation and online algorithms is to find an approximate characterization of the combinatorial structure of optimal solutions, and develop algorithmic techniques to exploit it. In particular, the goal is to identify a lower bound on OPT, the cost of the optimal solution, and design an algorithm whose solution costs at most some factor times the lower bound.

Over the past thirty years, researchers have developed a deep toolkit of approximation techniques for network design: greedy algorithms; linear programming

(LP) relaxation methods such as primal-dual, rounding, and dual-fitting; and metric embedding methods.[3] We now discuss how previous work adapted the primal-dual and metric embedding techniques to the online setting, and their strengths and weaknesses.

1.1.1 Primal-dual

The primal-dual method was originally used to solve linear programs and combinatorial optimization problems that can be solved in polynomial time. We discuss briefly how it works for minimization problems. The basic idea is to iteratively build the primal and dual solutions in tandem. Initially, both primal and dual solutions are empty. Each iteration of the primal-dual method reduces primal infeasibility while maintaining dual feasibility. In particular while the primal solution is infeasible, the unsatisfied primal constraints guides which dual variables to grow, and the dual constraints that become tight in turn suggests which primal variables to increase to reduce primal infeasibility. The analysis proceeds by showing that complementary slackness conditions hold and so the primal objective is equal to the dual objective. Since the dual is feasible, weak LP duality now implies that the primal solution is optimal. For approximation algorithms in network design, the primal-dual method was first used implicitly in the Steiner forest algorithm of Agrawal et al. [AKR95], and later formalized and generalized by Goemans and Williamson [GW95] to constrained forest problems. At a high level, the basic method outlined above is modified to show that approximate versions of the complementary slackness conditions hold, which imply that the primal objective is at most some α factor of the dual objective. (See [GW97] and [Wil02] for a fuller discussion of the primal-dual method.)

[3] See the survey by Gupta and Könemann [GK11] for a discussion of these techniques and others.

The challenge in adapting this method to the online setting is that it is sensitive to the order in which dual variables are grown. In particular, they need to be grown in a synchronized manner. However, in the online setting, the requirements, which define the primal constraints and dual variables, are not given in advance, but arrive one by one over time. For the online Steiner forest problem, Berman and Coulston [BC97] overcame this obstacle by simulating the offline primal-dual method over a family of $O(\log k)$ feasible dual solutions, resulting in a deterministic $O(\log k)$-competitive algorithm. This was later extended by Qian and Williamson [QW11] to give deterministic $O(\log k)$-competitive algorithms for the online constrained forest and prize-collecting Steiner forest problems.

1.1.2 Tree embedding

Over the past two decades, the technique of metric embedding has emerged as a powerful algorithmic tool. Metric embeddings allow us to approximate arbitrary metrics with simpler metrics. For optimization problems on metrics, metric embeddings can be used to reduce to instances on simpler metrics, for which it is easier to devise good approximation algorithms. Tree embeddings are particularly useful in network design since network design on a tree is usually easy, even in the online setting, essentially because there is only one way to connect any two vertices of a tree. For instance, in the Steiner tree problem, if G is a tree then the optimal solution is simply the subtree induced by the terminals.

We now sketch how previous work use tree embeddings to design algorithms. First, we need some definitions. A *tree metric* (V, T) is a tree with vertices V and equipped with lengths on edges; the distance $T(u, v)$ between any two vertices $u, v \in V$ is given by the length of the unique (u, v)-path in the tree. A *tree embedding*

of a metric (V, d) is a tree metric (V_T, T) whose vertices $V_T \supseteq V$, and distances $T(u, v) \geqslant d(u, v)$ for all $u, v \in V$ (this property is called "expanding"). The *distortion* of the embedding $\max_{u,v \in V} \frac{T(u,v)}{d(u,v)}$ measures how well it approximates (V, d). In the following, we abuse notation T to refer to the metric as well as the tree.

Algorithms based on tree embeddings employ the following template. Given a problem defined on (V, d), compute a tree embedding (V_T, T) of (V, d), solve the problem on (V_T, T) and translate the solution back into (V, d). The translation essentially maps edges over V_T to edges over V. If the distortion of the embedding is α, then this is an α-approximation. The analysis consists of two parts. Let $\mathrm{ALG}(T)$ be the cost of the algorithm. Since the distortion of the embeddings is α, we have $\mathrm{OPT}(T) \leqslant \alpha\,\mathrm{OPT}$, where $\mathrm{OPT}(T)$ is the cost of the optimal solution in (V_T, T). Furthermore, the expanding property of the embedding implies that the translation can only decrease the cost of the solution, so we have

$$\mathrm{ALG}(T) \leqslant \mathrm{OPT}(T) \leqslant \alpha\,\mathrm{OPT}\,.$$

How well can tree metrics approximate arbitrary metrics? Building on the work of Bartal [Bar96, Bar98], Fakcharoenphol et al. [FRT04] showed that any metric (V, d) admits a distribution of tree embeddings $\mathcal{T}$ with expected distortion $\max_{u,v \in V} \mathrm{E}_{T \in \mathcal{T}} \left[\frac{T(u,v)}{d(u,v)}\right] \leqslant O(\log n)$, and this is asymptotically tight. Using the distribution of [FRT04] in the above template gives a randomized algorithm whose cost is

$$\mathrm{E}_{T \in \mathcal{T}}[\mathrm{ALG}(T)] \leqslant \mathrm{E}_{T \in \mathcal{T}}[\mathrm{OPT}(T)] \leqslant O(\log n)\,\mathrm{OPT}\,.$$

Awerbuch and Azar [AA97] observed that the above approach can be easily adapted to the online setting: first compute a tree embedding (V_T, T) of the input metric (V, d) before the requests arrive, solve the online problem on (V_T, T), and

translate the solution in an online fashion back to (V, d). The translation can be online since the mapping between edges in the embedding and edges in the input metric can be computed at the beginning together with the embedding. This approach gives randomized online algorithms for these problems with expected competitive ratio $O(\log n)$.

1.1.3 Strengths and weaknesses

On the one hand, the embedding approach is highly versatile and can be easily applied to a wide range of problems. In contrast, the primal-dual approach has to be tailored closely to the specific LP relaxation used for each problem, and typically involves an intricate accounting scheme. This makes it difficult to extend to problems with complex connectivity requirements, as their duals are correspondingly more complex.

On the other hand, the embedding approach gives randomized algorithms whose competitive ratios scale with the size of the metric space, which can be much larger than the number of requests. In contrast, the primal-dual approach gives deterministic algorithms whose competitive ratios depend only on the number of requests and do not need to preprocess the entire input metric. Furthermore, unlike in the offline setting, online algorithms based on the embedding approach cannot be derandomized without significantly degrading their performance guarantees.

1.2 Our Contributions

The main contribution of this thesis is an analysis framework that leads to a new unified approach for online network design which addresses the drawbacks of the previous two approaches. At the heart of the analysis framework is the insight that

embeddings into a special class of tree metrics called *hierarchically separated trees* (HSTs) introduced by Bartal [Bar96, Bar98] give a useful approximate characterization of the combinatorial structure of optimal solutions and that we can devise algorithms to exploit it without having to compute the embedding.

We now present a high-level overview of our analysis framework. Suppose we are given an instance of an optimization problem with input metric (V, d) and an algorithm whose solution has cost ALG. For ease of exposition, we start with a simpler version that gives a competitive ratio of $O(\log n)$ and show later how to optimize it to get $O(\log k)$. The main goal in the analysis framework is to prove the following lemma.

Lemma 1.2. ALG $\leqslant O(1)\,\mathrm{OPT}(T)$ *for* every *HST embedding* T *of* (V, d).

Since every n-point metric can be embedded into a distribution of HSTs $\mathcal{T}$ with distortion $O(\log n)$ [Bar96, Bar98, FRT04], Lemma 1.2 implies

$$\mathrm{ALG} \leqslant O(1) \cdot \min_{T \in \mathcal{T}} \mathrm{OPT}(T) \leqslant O(1) \cdot \mathbb{E}_{T \in \mathcal{T}}[\mathrm{OPT}(T)] \leqslant O(\log n)\,\mathrm{OPT}, \tag{1.1}$$

as desired. We remark that no particular property of $\mathcal{T}$ is needed, it is only used to show the *existence* of a HST embedding T with $\mathrm{OPT}(T) \leqslant O(\log n)\,\mathrm{OPT}$. Furthermore, depending on the instance, it is possible that the embedding T^* which minimizes $\mathrm{OPT}(T^*)$ is much smaller than $O(\log n)\,\mathrm{OPT}$.

To prove Lemma 1.2, we use a charging scheme that leverages a special property of HST embeddings: a HST embedding T of (V, d) corresponds to a decomposition of (V, d) into bounded-diameter subsets. In particular, an edge e_T of T with length ℓ corresponds to a subset $C \subseteq V$ of diameter $\mathrm{diam}(C) = \max_{u,v \in C} d(u, v) < \ell$—the correspondence between the length of e_T and the diameter of C will be crucial for

us. The subsets are called *cuts* and the collection of cuts $\mathcal{C}(T)$ is called a *hierarchical decomposition*. Recall that we focus on problems where the objective is a linear combination of edge lengths in the underlying metric. Thus, the correspondence between edges and cuts allows us to express $\mathrm{OPT}(T)$ in terms of contributions from the cuts of $\mathcal{C}(T)$:

$$\mathrm{OPT}(T) = \sum_{C \in \mathcal{C}(T)} \beta(C)\,\mathrm{diam}(C).$$

Since the optimal solution on trees is easy to characterize, given T, it is easy to compute $\beta(C)$ for each $C \in \mathcal{C}(T)$. Another way to view this is that a HST embedding gives a hierarchical decomposition of the metric, which in turn gives a hierarchical decomposition of the optimal solution.

At a high level, the proof of Lemma 1.2 boils down to a argument that locally, in any small neighborhood, the cost incurred by the algorithm in serving the terminals in the neighborhood is not too large. In particular, for each terminal $v \in X$, our charging scheme charges the cost incurred in serving v to a cut of the hierarchical decomposition that contains it. The choice of the cut is mostly generic and problem-independent. For $C \in \mathcal{C}(T)$, let $\mathrm{ALG}(C)$ be the charge received by C. The crux of the analysis lies in proving that $\mathrm{ALG}(C) \leqslant O(1)\beta(C)\,\mathrm{diam}(C)$ for any cut $C \in \mathcal{C}(T)$. This would imply that

$$\mathrm{ALG} = \sum_{C \in \mathcal{C}(T)} \mathrm{ALG}(C) \leqslant O(1) \cdot \sum_{C \in \mathcal{C}(T)} \beta(C)\,\mathrm{diam}(C) = \mathrm{OPT}(T),$$

which proves Lemma 1.2.

Next, we describe how to get a $O(\log k)$ bound instead of $O(\log n)$. (Recall that X is the subset of terminals that arrived and $k = |X|$.) The key observation is that the charging scheme only involves terminals, so the analysis outlined above only

requires a hierarchical decomposition of the submetric (X, d). Thus, our analysis framework can be used to prove the following stronger lemma.

Lemma 1.3. $ALG \leqslant O(1)\,OPT(T)$ *for* every *HST embedding* T *of* (X, d).

Since the metric (X, d) has k points, it can be embedded into a distribution of HST metrics $\mathcal{T}$ with expected distortion $O(\log k)$. Plugging $\mathcal{T}$ into Equation 1.1 gives $ALG \leqslant O(\log k)\,OPT$.

Applying our framework to the online Steiner forest algorithm [BC97] and the online prize-collecting Steiner forest algorithm of [QW11] yields simpler analyses of their algorithms. The framework also leads to a new approach for online network design. Roughly speaking, given a problem, we devise a greedy-like algorithm that ensures that in any small neighborhood, the cost it incurs in serving terminals in the neighborhood is not too large. Using this approach, we obtain improved algorithms for rent-or-buy, connected facility location, and Steiner network. We emphasize that the algorithms themselves do not compute any embedding; the embedding is only used in the analysis.

Comparison with previous approaches. Our approach has several advantages over the previous two. First, HST embeddings enable us to systematically generate lower bounds for a wide variety of problems, providing a unified method to analyse algorithms for these problems. Thus, in contrast to the primal-dual approach, the analysis framework extends easily to problems with more general connectivity requirements. Furthermore, the structural properties of the HST embedding allow for a simpler analysis than using duals. In contrast to the embedding approach, our algorithms are simple and deterministic. Moreover, because we use the embedding only in the analysis and not in the algorithm itself, we are able to consider embed-

dings of just the terminals and prove a competitive ratio that scales with the number of requests instead of the size of the entire metric space.

Our work also highlights an interesting connection between HST embeddings and dual-fitting. We can interpret our charging scheme in terms of dual-fitting—given any expanding HST embedding of the terminals, we build a dual solution that is feasible for the HST and charge against it. These dual solutions can be highly infeasible for the original metric, but averaging over the probabilistic embeddings of [FRT04] gives a dual solution that is feasible after scaling by the embedding distortion.[4] However, our approach differs from dual-fitting in the usual sense—we charge against *multiple* dual solutions simultaneously (one per possible embedding), and the dual solutions are not built with respect to the original metric but with respect to the HST embeddings of the terminals. In a sense, our work uses HST embeddings as a black-box to generate good dual solutions.

Other applications. We also apply our framework to two other settings of online network design: online network games where terminals represent selfish players, and to online network design problems where the requests arrive in random order.

1.3 Overview of Results

Chapter 3: Steiner Problems. We demonstrate the basic techniques in this chapter. We warm up with a simple application to the online Steiner tree problem. Then we show how to apply it to the Berman-Coulston algorithm for online Steiner forest and obtain a simpler analysis. Finally, we show how this essentially allows us to reduce an online Steiner network instance to several online Steiner forest instances and

[4]This method of charging against randomized duals that are only feasible in expectation was first used by [DJK13] to analyze a randomized algorithm.

obtain a deterministic $O(\log k)$-competitive algorithm for online Steiner network with edge duplication. Previously, the best algorithm for this problem uses the standard reduction to tree instances via tree embeddings, and has a randomized $O(\log k)$ expected competitive ratio.

This chapter is based on [Umb15].

Chapter 4: Shared-vs-Individual Objectives. Next, we consider problems that have what we call a "shared-vs-individual" objective. These include the rent-or-buy, connected facility location, and prize-collecting Steiner problems. In each of these problems, part of the solution can be used to satisfy all requests—we call this the "shared" portion—and part of the solution can only be used for a single request—we call this the "individual" portion. In particular, the cost of the shared portion can be shared among several requests but each request is responsible for the cost of its own individual portion.

In this chapter, we use our analysis framework to obtain deterministic $O(\log k)$-competitive algorithms for these problems. For the rent-or-buy problem, Awerbuch et al. [AAB04] gave a randomized $O(\log k)$-competitive algorithm as well as a deterministic $O(\log^2 k)$-competitive algorithm and posed the existence of a deterministic $O(\log k)$-competitive algorithm as an open problem. Our result resolves this positively. The online connected facility location problem was recently proposed by San Felice et al. [SFWL14a] and they gave a randomized $O(\log^2 k)$-competitive algorithm. Our algorithm improves on both the competitive ratio as well as the use of randomness. For the prize-collecting Steiner forest problem, we obtain the same result as Qian and Williamson [QW11] with a simpler analysis.

This chapter is based on [Umb15].

Chapter 5: Network Design Games. Next, we apply our framework to analyze the dynamics of selfish behavior in online network design. In an online multicast game, the setup is identical to the online Steiner tree problem, except that each arriving terminal is associated with a selfish player that desires a path to the root. When a player arrives, it chooses a path that minimizes its cost share. The cost shares are defined by dividing the cost of each edge evenly among the players that use it. Note that the total cost share is the total cost of edges used by at least one player. We consider a generalization in which players can depart as well—this means that it no longer participates in the computation of cost shares. Observe that after an arrival or departure, some previously-arrived player might prefer to switch to a different path that reduces its cost share, i.e. make an improving move.

Our first result concerns a setting in which whenever the current players are not in an equilibrium—some player has an improving move—then a central authority can suggest a particular improving move to a particular player, and that players can only arrive and depart when we are in an equilibrium. We show that in this setting, after each arrival or departure, there exists a sequence of improving moves leading to an equilibrium, such that the total cost shares of the final equilibrium after N arrivals and departures is at most $O(\log N)$ of the minimum Steiner tree over the vertices which was associated with at least one player. Our second result shows that if players can arrive and depart at arbitrary times, even when the current players are not in equilibrium, then there exists a sequence of N arrivals and departures ending in an equilibrium whose total cost share is $\Omega(N^{1/6})$ of the minimum Steiner tree.

This chapter is based on joint work with Shuchi Chawla, Debmalya Panigrahi, and Mohit Singh.

Chapter 6: Random Arrivals In this chapter, we consider an input model called the "random permutation model," and study online network design in this model. Here, the adversary decides on the set of requests, but the algorithm is presented the requests in a (uniformly) random order σ. We consider a special case of the online constrained forest problem that we call the online Steiner tree with group arrivals problem. Under the random permutation model, an adversary decides on a collection of g terminal sets $X_1, \ldots, X_g$, that we call groups, with a common vertex $r \in X_1 \cup \cdots \cup X_g$. The algorithm is then presented the groups in a uniformly random order. First, we use a new type of embedding to show that there exists a simple deterministic algorithm whose expected competitive ratio is $O(\log g)$, where the expectation is over the random arrival order of the groups. Note that when g is much smaller than $k = |X_1 \cup \cdots \cup X_g|$, this is much better than $O(\log k)$. Second, we use the lower bound instance of Imase and Waxman [IW91] for the online Steiner tree problem to show that if the arrival order is adversarial, then no randomized algorithm can have expected competitive ratio asymptotically better than $\Omega(\min\{\log k, g\})$, which is significantly worse than our upper bound for the random permutation model when g is much smaller than k.

This chapter is based on work by the author.

2 Background on HST Embeddings

In this chapter, we present the necessary metric embedding definitions and results.

2.1 HST Embeddings

Let (X, d) be a metric over a set X of k terminals. Without loss of generality, we focus on metrics whose smallest distance $\min_{u,v \in X} d(u,v) = 1$. We are only interested in *expanding* embeddings, i.e. embeddings that do not contract distances.

Definition 2.1 (Embeddings)**.** *A metric* (X', d') *with* $X' \supseteq X$ *is an* embedding *of* (X, d) *if for all* $u, v \in X$*, we have* $d(u,v) \leqslant d'(u,v)$*. The* distortion *of the embedding is* α *if* $d'(u,v) \leqslant \alpha d(u,v)$ *for all* $u, v \in X$*.*

In this thesis, we will be concerned only with embeddings into hierarchically separated trees [Bar96].

Definition 2.2 (Tree metrics)**.** *A metric* (V_T, T) *is a* tree metric *if there exists a tree* $G = (V_T, E)$ *with vertices* V_T *and edge lengths such that for every* $u, v \in V_T$*,* $T(u,v) = d_G(u,v)$*, where* $d_G(u,v)$ *is the total length of the unique path between* u *and* v *in* G*.*

In the following, we abuse notation and use T to refer to the tree G as well.

Definition 2.3 (HST Embeddings)**.** *A tree metric* (V_T, T) *is a* hierarchically separated tree (HST) embedding *of* (X, d) *if* T *is a rooted tree with lengths on edges satisfying the following properties.*

1. *Leaf edges have length* 1*, and edge lengths double as one moves along a leaf-to-root path.*

2. *Every node is equidistant to each of its children.*

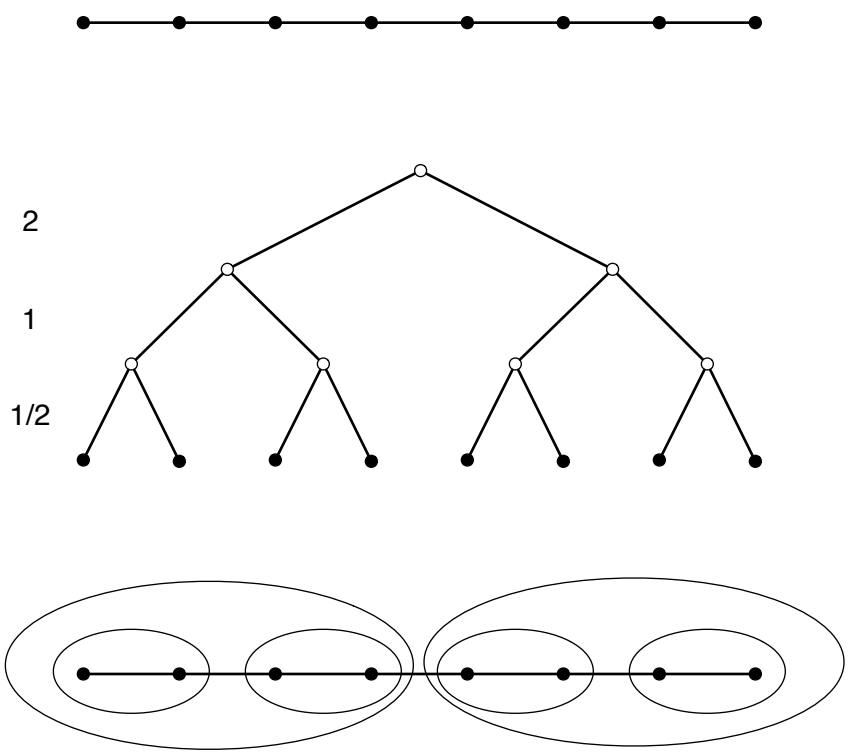

Figure 2.1: The first figure is a line graph with unit-length edges. The second figure is a HST embedding of the line metric, where the hollow tree vertices are extra vertices that do not belong to the line graph. The third figure shows the hierarchical decomposition corresponding to the HST embedding: the outer, and inner, ellipses correspond to the partiton given by the HST edges of length 2, and 1, respectively. The partition given by the edges of length $1/2$ are just the terminal singletons and are not shown.

3. *The leaves of* T *are exactly the terminals* X.

4. *For an edge* e *in* T, *let* $C_e \subseteq X$ *be the subset of terminals that are separated by* e *from the root. If* e *has length* 2^{j-1}, *then* $d(u, v) < 2^j$ *for any* $u, v \in C_e$.

The last property will be crucial to our analyses. See Figure 2.1 for an example of a HST embedding.

A HST embedding T of a metric space (X, d) defines a hierarchical decomposition of (X, d) in the following way. Call an edge e of length 2^{j-1} a *level-*j *edge* of T and C_e a *level-*j *cut*. Denote by $E_j(T)$ the set of level-j edges and define $\mathcal{C}_j(T) = \{C_e : e \in E_j(T)\}$

to be the set of all level-j cuts. Now since the leaves of T are exactly X, the level-j cuts $\mathcal{C}_j(T)$ partitions X into subsets of diameter less than 2^j. The family of partitions $\mathcal{C}(T) = \{\mathcal{C}_j(T)\}_j$ is called the *hierarchical decomposition* of (X, d) defined by T. Note that there are $\lfloor \log(\max_{u,v \in X} d(u,v)) \rfloor$ levels and the level-0 cuts $\mathcal{C}_0(T)$ are just the terminal singletons.

In the following, we will use the notation $\delta(C)$ to denote the set of vertex pairs with exactly one endpoint in C, i.e. $\delta(C) = \{(u,v) : |\{u,v\} \cap C| = 1\}$. Note that $(u,v) \in \delta(C_e)$ if and only if e lies on the path between u and v in T.

We denote by $T(u,v)$ the distance between u, v in T. Fakcharoenphol et al. [FRT04] showed that any metric can be embedded into HSTs with logarithmic expected distortion.

Theorem 2.4 ([FRT04])**.** *For any metric (X, d), there exists a distribution $\mathcal{D}$ over HST embeddings T such that $E_{T \sim \mathcal{D}}[T(u,v)] \leqslant O(\log k) d(u,v)$ for all $u, v \in X$.*

The following corollary follows by standard arguments. We will apply it to the Steiner network and rent-or-buy problems.

Corollary 2.5. *For any network design instance with terminals X whose objective is a linear combination of edge lengths $d(u,v)$, there exists a distribution $\mathcal{D}$ over HST embeddings T of (X, d) such that $E_{T \sim \mathcal{D}}[\mathrm{OPT}(T)] \leqslant O(\log k)\,\mathrm{OPT}$, where OPT (and $\mathrm{OPT}(T)$ resp.) is the cost of the optimal solution on (X, d) (and T resp.).*

Again, we emphasize that no particular property of the distribution $\mathcal{D}$ is required. We only need the existence of a HST embedding T^* such that $\mathrm{OPT}(T^*) \leqslant O(\log k)\,\mathrm{OPT}$.

3 Steiner Problems

In this chapter, we apply our analysis framework to the Steiner problems.

Our results. We start with formal definitions of these problems and a statement of the main result of this chapter.

- In the *online Steiner tree problem*, the algorithm is given a root terminal r at the beginning. Terminals i arrive online one-by-one and the algorithm maintains a subgraph H connecting terminals to the root.
- In the *online Steiner forest problem*, terminal pairs (s_i, t_i) arrive online one-by-one and the algorithm maintains a subgraph H in which each terminal pair is connected.
- In the *online Steiner network problem with edge duplication*, each (s_i, t_i) pair comes with a requirement R_i, and the algorithm maintains a multigraph H which contains R_i edge-disjoint (s_i, t_i)-paths. Note that allowing H to be a multigraph means that the algorithm is allowed to buy multiple copies of an edge. For brevity, we will simply call this the *online Steiner network problem*.

The goal in these problems is to maintain a minimum-cost (multi)-graph H satisfying the respective requirements, subject to the constraint that once an edge is added to H, it cannot be removed later on. The main result of this chapter is the following theorem.

Theorem 3.1. *There is a deterministic* $O(\log k)$*-competitive algorithm for the online Steiner network problem.*

For the online Steiner network problem, the previous-best algorithm (as far as we know) is given by the usual tree embedding approach: begin by embedding the input metric into a tree, solve the tree instance online and then translate the solution back to the input metric. This yields a randomized algorithm with $O(\log n)$ expected competitive ratio.

Chapter outline. In Section 3.1, we warm up with a simple application of the analysis framework to the online Steiner tree problem. In particular, we use the framework to obtain an alternative proof of Imase and Waxman's result that the greedy algorithm is $O(\log k)$-competitive for the online Steiner tree problem [IW91]. In Section 3.2, we apply our framework to the Berman-Coulston algorithm for online Steiner forest [BC97] and obtain a simpler analysis. In the final section, Section 3.3, we show how the analysis in Section 3.2 allows us to reduce any online Steiner network instance to several online Steiner forest instances, and obtain a deterministic $O(\log k)$-competitive algorithm for the online Steiner network problem.

3.1 Warmup: Steiner Tree

The greedy algorithm for the online Steiner tree problem is very natural: when terminal i arrives, connect it to the nearest previously-arrived terminal.

Analysis. Let $X \subseteq V$ be the set of k terminals that arrived and (X, d) be the sub-metric induced by X. The total cost of the greedy algorithm is $\sum_i a_i$ where a_i is the distance between terminal i and the nearest previously-arrived terminal. Our goal is to show that we can charge the cost of the greedy algorithm against the cost of the optimal solution on any HST embedding of the terminals.

We first define cost shares for each terminal such that the total cost share accounts for the cost of the algorithm. We classify terminals according to a_i — define $\text{class}(i) = j$ if $a_i \in [2^j, 2^{j+1})$ and $Z_j \subseteq X$ to be the set of class-j terminals. We define the cost share of each class-j terminal $i \in Z_j$ to be 2^{j+1}, i.e. i's cost share is a_i rounded up to the next power of 2. Thus, the total cost share is at least the cost of the algorithm.

Lemma 3.2. $\sum_i a_i \leqslant \sum_j 2^{j+1}|Z_j|$.

Before we proceed, we show that class-j terminals are at least 2^j-apart from each other.

Lemma 3.3. *For every* $i, i' \in Z_j$, *we have that* $d(i, i') \geqslant 2^j$.

Proof. Suppose i arrived later than i'. We have $a_i \leqslant d(i, i')$ since i could have connected to i', and $a_i \geqslant 2^j$ by definition of Z_j. Thus, we get $d(i, i') \geqslant 2^j$. □

Next, we show that we can charge the cost shares against the cost of the optimal solution on any HST embedding of the terminals.

Lemma 3.4. $\sum_j 2^{j+1}|Z_j| \leqslant O(1)\,\text{OPT}(T)$ *for any HST embedding* T *of* (X, d).

Proof. Let T be a HST embedding of (X, d). We begin by expressing $\text{OPT}(T)$ in terms of cuts from the hierarchical decomposition $\mathcal{C}(T)$. Since the terminals are exactly the leaves of T, the unique feasible solution on T is the entire tree. In particular, we have

$$\text{OPT}(T) = \sum_j 2^{j-1}|E_j(T)| = \sum_j 2^{j-1}|\mathcal{C}_j(T)|.$$

For each terminal $i \in Z_j$, charge i's cost share to the level-j cut containing it. The total charge received by a level-j cut $C \in \mathcal{C}_j(T)$ is $2^{j+1}|Z_j \cap C|$ since C is charged 2^{j+1}

for each class-j terminal in C. Thus we have

$$\sum_j 2^{j+1}|Z_j| = \sum_j 2^{j+1} \sum_{C \in \mathcal{C}_j(T)} |Z_j \cap C|.$$

Each level-j cut $C \in \mathcal{C}_j(T)$ has diameter less than 2^j so Lemma 3.3 implies $|Z_j \cap C| \leqslant 1$. Therefore, we have $\sum_j 2^{j+1}|Z_j| \leqslant \sum_j 2^{j+1}|\mathcal{C}_j(T)| = 4\,\mathrm{OPT}(T)$. □

We are now ready to bound the competitive ratio of the greedy algorithm. Lemmas 3.2 and 3.4 imply that the cost of the greedy algorithm is at most $O(1)\,\mathrm{OPT}(T)$ for any HST embedding T of (X, d). Furthermore, Corollary 2.5 implies that there exists a HST embedding T^* such that $\mathrm{OPT}(T^*) \leqslant O(\log k)\,\mathrm{OPT}$. Thus, the greedy algorithm is $O(\log k)$-competitive for the online Steiner tree problem.

3.2 Steiner Forest

Next, we apply our framework to the online Steiner forest problem and show that for every instance, the Berman-Coulston algorithm (Algorithm 3.1) finds a solution whose cost is at most a constant times the optimal solution on any HST embedding of the terminals. We remark that this also gives a simpler analysis of the algorithm.

Algorithm. Algorithm 3.1 proceeds as follows. When a terminal pair (s_i, t_i) arrives, its endpoints s_i and t_i are classified based on their distance: $\mathrm{class}(s_i) = \mathrm{class}(t_i) = \lfloor \log d(s_i, t_i) \rfloor$. The algorithm then proceeds in *levels*, starting from level $j = 0$ up to $\lfloor \log d(s_i, t_i) \rfloor$. At level j, for every previously-arrived terminal v with $\mathrm{class}(v) \geqslant j$, it adds the edge (s_i, v) to its solution H if $d(s_i, v) < 2^{j+1}$, and the edge (t_i, v) if $d(t_i, v) < 2^{j+1}$. We emphasize that the algorithm always considers distances according to the original metric d, it does not contract edges in H.

Algorithm 3.1 Berman-Coulston Algorithm for Steiner Forest

```
1: H ← ∅
2: while request (s_i, t_i) arrives do
3:    Assign class(s_i) ← ⌊log d(s_i, t_i)⌋ and class(t_i) ← ⌊log d(s_i, t_i)⌋
4:    for level j = 0 to ⌊log d(s_i, t_i)⌋ do
5:       for each terminal v with class(v) ⩾ j, d(s_i, v) < 2^{j+1}, and s_i and v not
         already connected in H do
6:          Add (s_i, v) to H
7:       end for
8:       for v such that class(v) ⩾ j, d(t_i, v) < 2^{j+1}, and t_i and v not already con-
         nected in H do
9:          Add (t_i, v) to H
10:      end for
11:   end for
12: end while
```

Analysis. Let X be the set of terminals that arrived and define X_j to be the set of terminals t with $\text{class}(t) \geqslant j$. Let A_j be the set of edges added in level j of some iteration. Note that $(u, v) \in A_j$ implies that $d(u, v) < 2^{j+1}$ and both $u, v \in X_j$.

We need the following lemma for our charging scheme.

Lemma 3.5. *For each class* j*, let* $\mathcal{S}_j$ *be a collection of disjoint subsets of* X *such that*

- $\mathcal{S}_j$ *covers* X_j *and*
- *for each* $S \in \mathcal{S}_j$*, we have* $d(u, v) < 2^j$ *for* $u, v \in S$*.*

Then we have $c(H) \leqslant \sum_j 2^{j+1}|\mathcal{S}_j|$.

Proof. We have that $c(H) \leqslant \sum_j 2^{j+1}|A_j|$ and so it suffices to prove that $|A_j| \leqslant |\mathcal{S}_j|$ for each j. Fix j and consider the following meta-graph: nodes correspond to $\mathcal{S}_j$; for each edge $(u, v) \in A_j$, there is a meta-edge between the nodes corresponding to sets containing u and v. This is well-defined since $u, v \in X_j$ and $\mathcal{S}_j$ covers X_j. The meta-edges correspond to A_j and the number of nodes is $|\mathcal{S}_j|$. We will show that the meta-graph is acyclic and so $|A_j| \leqslant |\mathcal{S}_j|$, as desired.

Suppose, towards a contradiction, that there is a cycle in the meta-graph. Thus, there exists edges $(u_1, v_1), \ldots, (u_\ell, v_\ell) \in A_j$ and sets $S_1, \ldots, S_\ell \in \mathcal{S}_j$ such that $v_i, u_{i+1} \in S_{i+1}$ for each $i < \ell$ and $v_\ell, u_1 \in S_1$. Suppose that (u_ℓ, v_ℓ) was the last among $(u_1, v_1), \ldots, (u_\ell, v_\ell)$ that was added by the algorithm and consider the state of the algorithm right before it added (u_ℓ, v_ℓ). The algorithm is currently in level j of some iteration because $(u_\ell, v_\ell) \in A_j$. So, for each $1 \leqslant m \leqslant \ell$, the terminals $X_j \cap S_m$ that have arrived by this iteration are already connected by now since $\operatorname{diam}(S_m) < 2^j$ and they have class at least j. Therefore u_ℓ and v_ℓ are actually already connected at this time, contradicting the algorithm's condition for adding the edge (u_ℓ, v_ℓ). Thus the meta-graph is acyclic, as desired. □

Lemma 3.6. *Define the function* $f : 2^X \to \{0, 1\}$ *as follows:* $f(S) = 1$ *if* $|S \cap \{s_i, t_i\}| = 1$ *for some* i. *For any HST embedding* T *of* (X, d), *we have* $c(H) \leqslant 4 \sum_j \sum_{C \in \mathcal{C}_j} 2^{j-1} f(C) = 4\,\mathrm{OPT}(T)$.

Proof. Let T be a HST embedding. Define $\mathcal{S}_j = \{C \in \mathcal{C}_j(T) : C \cap X_j \neq \emptyset\}$, i.e. $\mathcal{S}_j$ is the collection of level-j cuts that contain a terminal of class at least j. Since $\mathcal{S}_j$ satisfies the conditions of Lemma 3.5, we have that

$$c(H) \leqslant \sum_j 2^{j+1} |\mathcal{S}_j|.$$

Next, we lower bound $\mathrm{OPT}(T)$ in terms of its hierarchical decomposition $\mathcal{C}(T)$. For any level-j cut $C_e \in \mathcal{C}_j(T)$, if $(s_i, t_i) \in \delta(C_e)$ then e lies on the (s_i, t_i) path in T so the optimal solution on T contains e. Thus, we have that

$$\mathrm{OPT}(T) = \sum_j \sum_{C \in \mathcal{C}_j(T)} 2^{j-1} f(C).$$

Since $c(H) \leqslant \sum_j 2^{j+1}|\mathcal{S}_j|$ and $\mathcal{C}_j(T) \supseteq \mathcal{S}_j$, it suffices to prove that $f(C) = 1$ for $C \in \mathcal{S}_j$. Fix a cut $C \in \mathcal{S}_j$. By definition of $\mathcal{S}_j$, there exists a terminal $s_i \in C$ with $\text{class}(s_i) \geqslant j$. Since $\text{class}(s_i) \geqslant j$, we have that $d(s_i, t_i) \geqslant 2^j$ and so $t_i \notin C$ as the diameter of a level-j cut is less than 2^j. Thus, $C \cap \{s_i, t_i\} = \{s_i\}$ and so $f(C) = 1$, as desired. Now we have $\sum_{C \in \mathcal{C}_j(T)} f(C) \geqslant |\mathcal{S}_j|$ for each level j and this completes the proof of the lemma. □

Observe that this lemma together with Corollary 2.5 implies that Algorithm 3.1 is $O(\log k)$-competitive for the online Steiner forest problem. Lemma 3.6 says that the cost of Algorithm 3.1 is at most $O(1)\,\text{OPT}(T)$ for any HST embedding T of (X, d). Furthermore, Corollary 2.5 implies that there exists a HST embedding T^* such that $\text{OPT}(T^*) \leqslant O(\log k)\,\text{OPT}$. Thus, Algorithm 3.1 is $O(\log k)$-competitive for the online Steiner forest problem.

In the next section, we use Lemma 3.6 to analyze a greedy algorithm for the online Steiner network problem.

3.3 Steiner Network

Recall that in the online Steiner network problem, a terminal pair (s_i, t_i) with requirement R_i arrives at each online step and the algorithm maintains a multigraph H which contains R_i edge-disjoint (s_i, t_i)-paths for each arrived terminal pair. Let $R_{\max} = \max_i R_i$ be the maximum requirement. Note that the Steiner forest problem is a special case with $R_{\max} = 1$.

Intuition. Consider rooted Steiner network instances in which every terminal pair contains a common root and the requirements are uniform. We call such an instance a "scaled" Steiner tree instance. For such instances, the optimal solution

is simply R copies of the optimal Steiner tree over the terminals, where R is the common requirement. Goemans and Bertsimas [GB93] used this observation to obtain a $O(\log R_{max})$-approximation algorithm for rooted Steiner network instances with non-uniform requirements. At a high level, the algorithm decomposes the Steiner network instance into $O(\log R_{max})$ scaled Steiner tree subinstances, and then applies a good Steiner tree approximation algorithm to each subinstance. More formally, it rounds each requirement to the next higher power of 2, and then for each power $0 \leqslant \ell \leqslant \lceil \log R_{max} \rceil$, it buys $2^{\ell+1}$ copies of an approximate Steiner tree over the terminals with rounded requirement 2^{ℓ}. Later, Agrawal et al. [AKR95] used this approach with their 2-approximation algorithm for Steiner forest obtain a $O(\log R_{max})$-approximation algorithm for general Steiner network instances. In both cases, the resulting approximation loses a factor of $O(\log R_{max})$ over the approximation factor of the Steiner forest and Steiner tree problems, and there is a matching $\Omega(\log \min\{R_{max}, k\})$ lower bound as well. Intuitively, the $O(\log R_{max})$ loss makes sense since each of the $O(\log R_{max})$ subinstances are treated independently.

This approach can also be made to work for the online Steiner network problem by running the Berman-Coulston online Steiner forest algorithm on the subinstances. The above line of reasoning suggests that the competitive ratio should be $\Theta(\log R_{max} \cdot \log k)$, since the Berman-Coulston algorithm is $\Theta(\log k)$-competitive. Surprisingly, our framework shows that the competitive ratio is actually $O(\log k)$, losing only a constant factor over the competitive ratio of the Berman-Coulston algorithm.

The analysis consists of two ingredients. The first is that for any online Steiner forest instance, the cost of the Berman-Coulston algorithm is at most a constant times the cost of the optimal solution on any HST embedding of the terminals (Lemma 3.6 in the previous section). The second is the following decomposition property of Steiner network instances: for a Steiner network instance on a tree, the total cost of

the optimal solutions to the scaled Steiner forest subinstances is at most a constant times the cost of the optimal solution to the original Steiner network instance[1].

3.3.1 From Steiner Forest to Steiner Network

We now state our Steiner network algorithm formally and show that it is $O(\log k)$-competitive.

Algorithm. For ease of exposition, we first assume that we are given the maximum requirement R_{max} and remove this assumption later on. We run $\lfloor \log R_{max} \rfloor$ instantiations of Algorithm 3.1 (the Berman-Coulston Steiner forest algorithm); when we receive a terminal pair with requirement $R_i \in [2^\ell, 2^{\ell+1})$, we pass the pair to the ℓ-th instantiation and buy $2^{\ell+1}$ copies of each edge bought by that instantiation. The assumption that we are given R_{max} can be removed by starting a new instantiation of Algorithm 3.1 when we receive a terminal pair whose requirement is higher than all previous requirements.

Analysis. We maintain a feasible solution so it remains to bound its cost. Let X be the set of terminals that arrived and H_ℓ be the final subgraph of the ℓ-th instantiation of Algorithm 3.1. The cost of our solution is $\sum_\ell 2^{\ell+1} c(H_\ell)$. We now show that this is at most a constant times the cost of the optimal solution on any HST embedding of the terminals.

Lemma 3.7. $\sum_\ell 2^{\ell+1} c(H_\ell) \leqslant O(1)\,\mathrm{OPT}(T)$ *for any HST embedding* T *of* (X, d).

Proof. Fix a HST embedding T of (X, d). Define the function $f : 2^X \to \mathbb{N}$ as follows: for $S \subseteq X$, we have $f(S) = \max_{i:(s_i,t_i)\in\delta(S)} R_i$. For any level-j cut $C_e \in \mathcal{C}_j(T)$, if

[1] In general graphs, we lose a factor of $O(\log R_{max})$.

$(s_i, t_i) \in \delta(C_e)$ then e lies on the (s_i, t_i) path in T so the optimal solution on T buys $f(C_e)$ copies of e. Summing over all cuts of the hierarchical decomposition gives us

$$\mathrm{OPT}(T) = \sum_j \sum_{C \in \mathcal{C}_j(T)} 2^{j-1} f(C).$$

For each $0 \leqslant \ell \leqslant \lfloor \log R_{\max} \rfloor$, define the subset of terminal pairs $D_\ell = \{(s_i, t_i) : R_i \in [2^\ell, 2^{\ell+1})\}$ and the function $f_\ell : 2^X \to \{0, 1\}$ as follows: $f_\ell(S) = 1$ if $S \subseteq X$ separates a terminal pair of D_ℓ. Since H_ℓ is the output of Algorithm 3.1 when run on the subsequence of terminal pairs D_ℓ, Lemma 3.6 implies that

$$\begin{aligned}\sum_\ell 2^{\ell+1} c(H_\ell) &\leqslant 4 \sum_\ell 2^{\ell+1} \left(\sum_j \sum_{C \in \mathcal{C}_j(T)} 2^{j-1} f_\ell(C) \right) \\ &= 4 \sum_j \sum_{C \in \mathcal{C}_j(T)} 2^{j-1} \left(\sum_\ell 2^{\ell+1} f_\ell(C) \right).\end{aligned}$$

Fix a cut C. Since each terminal pair $(s_i, t_i) \in D_\ell$ has requirement $R_i \in [2^\ell, 2^{\ell+1})$ and $f_\ell(C) = 1$ if C separates a terminal pair of D_ℓ, we have

$$\sum_\ell 2^{\ell+1} f_\ell(C) \leqslant 4 \max_{i:(s_i,t_i) \in \delta(C)} R_i = 4 f(C).$$

Thus,

$$4 \sum_j \sum_{C \in \mathcal{C}_j(T)} 2^{j-1} \left(\sum_\ell 2^{\ell+1} f_\ell(C) \right) \leqslant 16 \sum_j \sum_{C \in \mathcal{C}_j(T)} 2^{j-1} f(C) = 16\, \mathrm{OPT}(T).$$

□

By Corollary 2.5, there exists a HST embedding T^* with $\mathrm{OPT}(T^*) \leqslant O(\log k)\, \mathrm{OPT}$. So Lemma 3.7 implies that the cost of our algorithm is at most $O(\log k)\, \mathrm{OPT}$. Thus,

our algorithm is $O(\log k)$-competitive for Steiner network and this proves Theorem 3.1.

Remarks. One can apply a dual-fitting analysis like that used by Berman and Coulston [BC97] to this algorithm, but the upper bound it gives is proportional to the number of relevant distance scales. This is $O(\log(kR_{max}))$ for this problem which can deteriorate to $O(k)$ since R_{max} can be as large as 2^k.

3.4 Notes

The work in this chapter appeared as part of [Umb15].

Related work. There is a long line of work on these problems in the offline setting. The current-best algorithm for the Steiner tree problem is a 1.39-approximation by Byrka et al. [BGRS13]. Agrawal et al. [AKR95] gave a primal-dual 2-approximation algorithm for the Steiner forest problem. This was later extended by Goemans and Williamson [GW95] to give a 2-approximation for the more general constrained forest problem. In a breakthrough result, Jain [Jai01] gave a 2-approximation algorithm using iterative LP rounding for the Steiner network problem. We remark that Jain's result holds even when edge duplication is not allowed.

In the online setting, Imase and Waxman [IW91] showed that the greedy algorithm is $O(\log k)$-competitive for the online Steiner tree problem and gave a matching lower bound. Awerbuch et al. [AAB04] showed that a greedy algorithm is $O(\log^2 k)$-competitive for the online Steiner forest problem; later, Berman and Coulston [BC97] gave a $O(\log k)$-competitive algorithm. Disallowing edge duplication makes the problem significantly harder, and in particular, it cannot be reduced to metric instances, so our techniques do not apply to this version of the problem.

Gupta et al. [GKR12] gave a randomized algorithm for it with expected competitive ratio $\tilde{O}(R_{max} \log^3 n)$.

4 Shared-vs-Individual Objectives

In this chapter, we extend the analysis framework to problems with what we call "shared-vs-individual" objectives: the rent-or-buy, connected facility location, and prize-collecting Steiner problems. These problems are said to have "shared-vs-individual" objectives because part of the solution can be used to satisfy all requests—we call this the "shared" portion—and part of the solution can only be used for a single request—we call this the "individual" portion. In particular, the cost of the shared portion can be shared among several requests but each request is responsible for the cost of its own individual portion.

Our results. First, we define the problems and describe our results.

- The rent-or-buy problem generalizes the Steiner forest problem. The algorithm is allowed to either *rent* or *buy* edges in order to satisfy a request. Buying an edge costs M times more than renting, but once an edge is bought, it can be used for free in the future. On the other hand, a rented edge can only be used once; future terminals have to either re-rent it or buy it in order to use it.

  More formally, in the *online rent-or-buy problem*, the algorithm is also given a parameter $M \geqslant 0$. The algorithm maintains a subgraph H of bought edges; when a terminal pair (s_i, t_i) arrives, the algorithm buys zero or more edges and rents edges Q_i such that $H \cup Q_i$ connects s_i and t_i. Both rent and buy decisions are irrevocable — edges cannot be removed from H later on and Q_i is fixed after the i-th step. The total cost of the algorithm is $Mc(H) + \sum_i c(Q_i)$. In the *online single-source rent-or-buy problem*, the algorithm is also given a root terminal r in advance; terminals i arrive online and i has to be connected to r

in the subgraph $H \cup Q_i$.

For this problem, the subgraph H is the shared portion and Q_i is the individual portion for the i-th request.

- In the *online connected facility location problem*, we call terminals *clients*. At the beginning, the algorithm is given a parameter $M \geqslant 0$, a set of *facilities* $F \subseteq V$ and facility opening costs f_x for each facility $x \in F$. There is also a designated root facility $r \in F$ with zero opening cost. The algorithm maintains a set of *open* facilities $F' \subseteq F$ and a Steiner tree H connecting F' and r. At any online step, the algorithm knows the submetric over the arrived clients and the facilities. When a client i arrives, the algorithm may open a new facility, and then assigns i to some open facility $\sigma(i) \in F'$. All decisions are irrevocable — edges cannot be removed from H later on, clients cannot be reassigned and opened facilities cannot be closed. The total cost of the algorithm is $\sum_{z \in F'} f_z + Mc(H) + \sum_i d(i, \sigma(i))$. When $M = 0$ (i.e. the facilities can be connected for free), this is called the *online facility location problem*. For these problems, we use k to denote the number of clients.

  For this problem, the open facilities F' and the subgraph H connecting them form the shared portion, and the connection cost $d(i, \sigma(i))$ is the individual portion for request i.

- In the *online prize-collecting Steiner forest* problem, each terminal pair (s_i, t_i) arrives with a penalty π_i and the algorithm either pays the penalty or connects the pair. The total cost of the algorithm is $c(H) + \sum_{i \in P} \pi_i$ where H is the subgraph maintained by the algorithm and $i \in P$ if the algorithm paid the penalty for (s_i, t_i). Note that penalties are irrevocable: once the algorithm

decides to pay the penalty π_i, the penalty does not get removed from the algorithm's cost even if later on H ends up connecting s_i and t_i.

For this problem, the subgraph H is the shared portion and the penalty π_i is the individual portion for request (s_i, t_i).

For the rent-or-buy problem, Awerbuch et al. [AAB04] were the first to consider the rent-or-buy problem.[1] They gave a randomized $O(\log k)$-competitive algorithm as well as a deterministic $O(\log^2 k)$-competitive algorithm and posed the existence of a deterministic $O(\log k)$-competitive algorithm as an open problem. We resolve this positively.

Theorem 4.1. *There is a deterministic $O(\log k)$-competitive algorithm for the online rent-or-buy problem.*

The online connected facility location problem was recently proposed by San Felice et al. [SFWL14a] and they gave a randomized $O(\log^2 k)$-competitive algorithm. Our next result improves on both the competitive ratio and the use of randomness.

Theorem 4.2. *There is a deterministic $O(\log k)$-competitive algorithm for the online connected facility location problem.*

For the prize-collecting Steiner forest problem, we obtain the same result as Qian and Williamson [QW11] with a simpler analysis.

Theorem 4.3 ([QW11])**.** *There is a deterministic $O(\log k)$-competitive algorithm for the online prize-collecting Steiner forest problem.*

High-level approach. We now demonstrate the basic ideas using the single-source rent-or-buy problem. Let us begin by considering instances of the following type,

[1] They use the name "network connectivity leasing problem" instead.

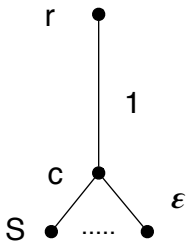

Figure 4.1: Counterexample for purely greedy algorithm.

illustrated in Figure 4.1. The metric (V, d) is given by a star graph with $n+2$ vertices. The root r is attached to the center of the star c with an edge of length 1 and the rest of the n vertices S are each connected to the center c with an edge of negligible length $\epsilon \ll 1$. Consider an online instance in which k vertices of S arrive one-by-one, in some arbitrary order. The cost of the optimal solution is essentially $\min\{k, M\}$: if $k \leqslant M$, the optimal solution rents the edges (r, c) and (c, i) for each terminal i; if $k > M$, the optimal solution buys the edge (r, c) and rents the edge (c, i) for each terminal i.

It is easy to see that a purely greedy algorithm which seeks to minimize the cost incurred per request has a poor competitive ratio when $k \gg M$. The algorithm rents the edges (r, c) and (c, i) for each terminal i, at a total cost of $k(1+\epsilon) \gg M$. Thus, the algorithm fails because it is too myopic. Even though buying the edge (r, c) is more expensive than renting, buying it might be a cheaper option overall because it can be used towards all future requests. For this particular type of instances, a better online algorithm is the following: for the first M terminals, rent the edge (r, c), and then for the $(M+1)$-th terminal, buy it. This algorithm is 2-competitive since it rents (r, c) at most $\min\{k, M\}$ times, and only buys it when it has rented it M times (i.e. when $k > M$).[2]

[2]Essentially, this type of instances captures the classic online ski rental problem [KMMO94], and the algorithm we describe is the "break-even" algorithm for that problem.

Our algorithm for single-source rent-or-buy is a natural generalization of the above. Every terminal is designated as either a *buy terminal* or a *rent terminal*, with the root being a buy terminal, and at each point in time, the algorithm's set of bought edges H is an online Steiner tree over the current set of buy terminals. When terminal i arrives, the algorithm does the following. Suppose z is the previous buy terminal that is nearest to i. If there are at least M rent terminals i' with high enough rent cost $c(Q_i)$ close to i, the algorithm buys the edge (z, i) and designates i as a buy terminal, otherwise it rents (z, i) and designates it a rent terminal. Essentially, this strategy allows us to charge the cost of every edge (z, i) we buy to the rent cost of terminals near the corresponding buy terminal z, and to show that the rent cost of terminals in any small neighborhood is not too large. The latter allows us to argue that the total rent cost is at most $O(1)\,\mathrm{OPT}(T)$ for any HST embedding T of the terminals. (See Section 4.1 for a formal description and analysis of the algorithm.)

In general, our algorithms use the following greedy-like strategy: for each request, if the individual cost of nearby requests are sufficiently high, then the algorithm augments the shared portion in a greedy manner to satisfy the request; otherwise, it augments the individual portion of the request. The analysis proceeds by charging the shared cost to the individual cost, and then charging the individual cost to $\mathrm{OPT}(T)$, for every HST embedding T.

Chapter outline. In Section 4.1, we formally describe and analyze our algorithm for the single-source rent-or-buy problem. Then we extend these ideas to the multiple-source setting in Section 4.2 using the Berman-Coulston online Steiner forest algorithm discussed in Section 3.2, and the connected facility location problem in Section 4.3. The objective of the prize-collecting Steiner tree (forest) problem is intimately related to the single-source (multiple-source) rent-or-buy problem, and a straight-

forward adaptation of the single-source (multiple-source) rent-or-buy algorithm gives a prize-collecting Steiner tree (Steiner forest) algorithm. In Section 4.4, we describe the prize-collecting Steiner tree algorithm. We omit a discussion of the prize-collecting Steiner forest algorithm as it is obtained via an identical adaptation of the multiple-source rent-or-buy algorithm.

4.1 Single-Source Rent-or-Buy

In this section, we elaborate on the description and analysis of the algorithm for the single-source rent-or-buy problem given earlier and prove the following theorem.

Theorem 4.4. *There exists a* $O(\log k)$*-competitive algorithm for the online single-source rent-or-buy problem.*

Recall that the total cost of the algorithm is $Mc(H) + \sum_i c(Q_i)$. We call $Mc(H)$ the *buy cost*, and $c(Q_i)$ the *rent cost* of terminal i.

Algorithm. Our algorithm (Algorithm 4.1) designates each terminal as either a *buy terminal* or a *rent terminal*, with the root r being a buy terminal, and maintains H to be an online Steiner tree connecting the buy terminals. When a terminal i arrives, the algorithm does the following. Let Z denote the current set of buy terminals and R the current set of rent terminals. Suppose $z \in Z$ is the closest buy terminal to i. Define $a_i = d(i, z)$ and $\text{class}(i) = \lfloor \log d(i, z) \rfloor$. Suppose $\text{class}(i) = j$. Denote by R_j the set of rent terminals $v \in R$ with $\text{class}(v) = j$. The *witness set* of i is defined to be $W(i) = \{v \in R_j : d(i, v) < 2^{j-1}\}$. If $|W(i)| \geqslant M$, the algorithm buys the edge (i, z) and designates i a buy terminal; otherwise, it rents (i, z) and designates i a rent terminal.

Algorithm 4.1 Algorithm for Online Single-Source Rent-or-Buy

$Z \leftarrow \{r\}; R_j \leftarrow \emptyset; H \leftarrow \emptyset$
while terminal i arrives **do**
  Let $z \in Z$ be closest buy terminal to i, and $j = \lfloor \log d(i,z) \rfloor$
  Assign class$(i) \leftarrow j$
  $W(i) \leftarrow \{i' \in R_j : d(i,i') < 2^{j-1}\}$
  if $|W(i)| \geqslant M$ **then**
    Add i to Z and buy (i,z), i.e. add (i,z) to H
  else
    Add i to R_j and rent (i,z), i.e. assign $Q_i \leftarrow \{(i,z)\}$
  end if
end while

Analysis. Let X be the set of terminals that arrived. For each buy terminal $z \in Z$, the algorithm incurs a buy cost of Ma_z; for each rent terminal $i \in R$, the algorithm incurs a rent cost of a_i. Thus, the total cost of the algorithm is $\sum_{z\in Z} Ma_z + \sum_{i\in R} a_i$. Our goal in the analysis is to show that this is at most a constant times $OPT(T)$ for any HST embedding T of (X,d).

First, we define cost shares for each terminal. For each rent terminal $i \in R_j$, we define i's cost share to be 2^{j+1}, i.e. a_i rounded up to the next power of 2. The total cost share is $\sum_j 2^{j+1}|R_j|$. We now show that the cost of the algorithm is at most twice the total cost share.

Lemma 4.5. $\sum_{z\in Z} Ma_z + \sum_{i\in R} a_i \leqslant 2\sum_j 2^{j+1}|R_j|$.

Proof. Define $Z_j = \{z \in Z : \text{class}(z) = j\}$, the set of buy terminals with class j. We have $\sum_{i\in R} a_i \leqslant \sum_j 2^{j+1}|R_j|$ and $\sum_{z\in Z} Ma_z \leqslant \sum_j M2^{j+1}|Z_j|$. We now show that $M|Z_j| \leqslant |R_j|$ for each class j.

Fix a class j. The witness set $W(z)$ of $z \in Z_j$ satisfies the following properties: (1) $|W(z)| \geqslant M$; (2) $W(z) \subseteq R_j$; (3) $d(i,z) < 2^{j-1}$ for $i \in W(z)$. The first implies that $M|Z_j| \leqslant \sum_{z\in Z_j} |W(z)|$. We claim that $d(z,z') \geqslant 2^j$ for $z, z' \in Z_j$. This completes the

proof since together with the second and third properties, it implies that the witness sets of Z_j are disjoint subsets of R_j and so $\sum_{z \in Z_j} |W(z)| \leqslant |R_j|$.

Now we prove the claim. Observe that H is the subgraph produced by the greedy online Steiner tree algorithm if it were run on the subsequence of buy terminals Z and for each $z \in Z$, we have that a_z is exactly the distance from z to the nearest previously-arrived buy terminal. Thus, we can apply Lemma 3.3 and get that $d(z, z') \geqslant 2^j$ for any $z, z' \in Z_j$, proving the claim. Putting all of the above together, we get $\sum_{z \in Z} M a_z \leqslant \sum_j 2^{j+1} |R_j|$. This finishes the proof of the lemma. □

Next, we show that the total cost share is at most a constant times the cost of the optimal solution on any HST embedding of the terminals.

Lemma 4.6. $\sum_j 2^{j+1} |R_j| \leqslant O(1)\, OPT(T)$ *for any HST embedding* T *of* (X, d).

Proof. Let T be a HST embedding of (X, d). We begin by lower bounding $OPT(T)$ in terms of T's cuts. Fix a level-j cut $C_e \in \mathcal{C}_j(T)$. If $r \notin C_e$, then e lies on the path between i and r for each terminal $i \in C_e$. Since $e \in E_j(T)$ and has length 2^{j-1}, the optimal solution on T either rents e for each terminal in C at a cost of $2^{j-1}|C|$ or buys it at a cost of $2^{j-1}M$. Summing over all cuts of the hierarchical decomposition gives us the following lower bound

$$OPT(T) \geqslant \sum_j \sum_{C \in \mathcal{C}_j(T) : r \notin C} 2^{j-1} \min\{M, |C|\}.$$

For each rent terminal $i \in R_j$, charge 2^{j+1} to the unique level-$(j-1)$ cut[3] containing i. The total charge received by each level-j cut $C \in \mathcal{C}_j(T)$ is $2^{j+2}|R_j \cap C|$ since C is

[3] We can extend T with an additional level of terminal singletons to accomodate charging against level $j = -1$. This only increases $OPT(T)$ by at most a constant.

charged 2^{j+2} for each class-$(j+1)$ rent terminal in C. Thus we have

$$\sum_j 2^{j+1}|R_j| = \sum_j \sum_{C \in \mathcal{C}_j(T)} 2^{j+2}|R_{j+1} \cap C|.$$

It remains to prove the following claim: for each level-j cut $C \in \mathcal{C}_j(T)$, we have $|R_{j+1} \cap C| = 0$ if $r \in C$ and $|R_{j+1} \cap C| \leqslant \min\{M, |C|\}$ if $r \notin C$. Suppose $r \in C$. Since $a_i < d(i,r) < 2^j$ for all $i \in C$, there cannot be any class-$(j+1)$ terminal in C and so $R_{j+1} \cap C = \emptyset$. Now consider the case $r \notin C$. Since $|R_{j+1} \cap C| \leqslant |C|$, it suffices to prove that $|R_{j+1} \cap C| \leqslant M$. Suppose, towards a contradiction, that $|R_{j+1} \cap C| > M$ and let i be the last-arriving terminal of $R_{j+1} \cap C$. The terminals of $R_{j+1} \cap C \setminus \{i\}$ arrive before i, are each of distance less than 2^j from i (diameter of C is less than 2^j) and of the same class as i, so they belong to i's witness set $W(i)$. Since $|R_{j+1} \cap C| > M$, we have that $|W(i)| \geqslant |R_{j+1} \cap C \setminus \{i\}| \geqslant M$. Thus, i would have been a buy terminal but this contradicts the assumption that $i \in R_{j+1}$. Therefore, we have $|R_{j+1} \cap C| \leqslant M$ as desired. This completes the proof of the claim and so we get $\sum_j 2^{j+1}|R_j| \leqslant 8\,\mathrm{OPT}(T)$. □

Now we put all of the above together. Lemmas 4.5 and 4.6 imply that the cost of Algorithm 4.1 is at most $O(1)\,\mathrm{OPT}(T)$ for any HST embedding T of (X, d). Furthermore, by Corollary 2.5, there exists a HST embedding T^* such that $\mathrm{OPT}(T^*) \leqslant O(\log k)\,\mathrm{OPT}$. Thus the algorithm is $O(\log k)$-competitive for single-source rent-or-buy and this proves Theorem 4.4.

4.2 Multiple-Source Rent-or-Buy

We move on to the multiple-source setting. Recall that in this setting, terminal pairs (s_i, t_i) arrive one-by-one and algorithm has to update H (buy edges) and rent

edges Q_i so that s_i and t_i are connected in $H \cup Q_i$. The total cost of the algorithm is $Mc(H) + \sum_i c(Q_i)$. As in the single-source setting in the previous section, we call $Mc(H)$ the *buy cost*, and $c(Q_i)$ the *rent cost* of terminal pair (s_i, t_i).

Intuition. The high-level idea is similar to the single-source rent-or-buy algorithm in Section 4.1. Our algorithm (Algorithm 4.2) designates each terminal pair to be either a *buy pair* or a *rent pair*. It maintains H to be an online Steiner forest over the buy pairs and rents the direct edge (s_i, t_i) for each rent pair (s_i, t_i). Essentially, the algorithm designates terminal pair (s_i, t_i) to be a buy pair if there are at least M endpoints of rent pairs with sufficiently large rent costs near each of s_i and t_i. The algorithm uses the Berman-Coulston algorithm for online Steiner forest [BC97] (described in Section 3.2) to determine H.

Algorithm. When a terminal pair (s_i, t_i) arrives, the algorithm does the following. Define $a_i = d(s_i, t_i)$ and $\text{class}(s_i) = \text{class}(t_i) = \lfloor \log a_i \rfloor$. Suppose $\text{class}(s_i) = \text{class}(t_i) = j$. Denote by R_j the set of rent terminals v with $\text{class}(v) = j$. The *witness sets* of s_i and t_i are defined as follows: $W(s_i) = \{v \in R_j : d(s_i, v) < 2^{j-2}\}$ and $W(t_i) = \{v \in R_j : d(t_i, v) < 2^{j-2}\}$. If $|W(s_i)| \geqslant M$ and $|W(t_i)| \geqslant M$, then (s_i, t_i) is a buy pair and the algorithm augments H so that s_i and t_i are connected in H; otherwise, (s_i, t_i) is a rent pair, and the algorithm rents the direct edge (s_i, t_i) and designates exactly one of s_i or t_i to be a rent terminal. To determine H, the algorithm runs Algorithm 3.1 on the subsequence of buy pairs and defines H to be the online Steiner forest maintained by Algorithm 3.1.

Analysis. Let X be the set of terminals that arrived, R be the set of all rent terminals and Z be the set of terminals that were passed to Algorithm 3.1 (i.e. Z is the set of endpoints of buy pairs). We call Z the set of *buy terminals*. Note that R contains

Algorithm 4.2 Algorithm for Online Multiple-Source Rent-or-Buy

$H \leftarrow \emptyset$; $R_j \leftarrow \emptyset$
while request (s_i, t_i) arrives **do**
  Let $j = \lfloor \log d(s_i, t_i) \rfloor$
  Assign class$(s_i) \leftarrow j$ and class$(t_i) \leftarrow j$
  $W(s_i) \leftarrow \{v \in R_j : d(s_i, v) < 2^{j-2}\}$
  $W(t_i) \leftarrow \{v \in R_j : d(t_i, v) < 2^{j-2}\}$
  if $|W(s_i)| < M$ **then**
    Rent (s_i, t_i), i.e. $Q_i \leftarrow \{(s_i, t_i)\}$
    Add s_i to R_j
  else if $|W(t_i)| < M$ **then**
    Rent (s_i, t_i), i.e. $Q_i \leftarrow \{(s_i, t_i)\}$
    Add t_i to R_j
  else
    Pass (s_i, t_i) to Algorithm 3.1
    Update H to be the online Steiner forest maintained by Algorithm 3.1
  end if
end while

exactly one endpoint of each rent pair; we rename the terminals so that if (s_i, t_i) is a rent pair, then $t_i \in R$. The cost of the algorithm is $Mc(H) + \sum_{t_i \in R} d(s_i, t_i)$. The analysis proceeds by charging the cost of the algorithm against the cost of the optimal solution on any HST embedding of the terminals.

We first define cost shares for each terminal. For each $t_i \in R_j$, we define t_i's cost share to be 2^{j+1}. The total cost share is $\sum_j 2^{j+1}|R_j|$. We now show that the cost of the algorithm is at most twice the total cost share.

Lemma 4.7. $Mc(H) + \sum_{t_i \in R} d(s_i, t_i) \leqslant 2 \sum_j 2^{j+1}|R_j|$.

Proof. Since t_i's cost share is at least $d(s_i, t_i)$, we have $\sum_{t_i \in R} d(s_i, t_i) \leqslant \sum_j 2^{j+1}|R_j|$. Next, we bound $Mc(H)$. Let $Z_j \subseteq Z$ be the set of buy terminals with class at least j. Define $Z'_j \subseteq Z_j$ to be the maximal subset of Z_j such that $d(u, v) \geqslant 2^{j-1}$ for all $u, v \in Z'_j$. We will show that $Mc(H) \leqslant \sum_j 2^{j+1}|R_j|$ in two steps: first we prove that $Mc(H) \leqslant \sum_j 2^{j+1} M|Z'_j|$ and then prove that $M|Z'_j| \leqslant |R_j|$ for each j.

We partition Z_j as follows: assign each $v \in Z_j$ to the closest terminal in Z'_j, breaking ties arbitrarily, and define S_u to be the set of terminals assigned to u. Observe that the diameter of S_u is less than 2^j for all $u \in Z'_j$. Define $\mathcal{S}_j = \{S_u\}_{u \in Z'_j}$. Since $\mathcal{S}_j$ satisfies the conditions of Lemma 3.5, we get that $Mc(H) \leqslant \sum_j 2^{j+1} M|\mathcal{S}_j| = \sum_j 2^{j+1} M|Z'_j|$.

Next we show that $M|Z'_j| \leqslant 2^{j+1}|R_j|$ for each j. Fix j. The witness set $W(z)$ of $z \in Z'_j$ satisfies the following properties: (1) $|W(z)| \geqslant M$; (2) $W(z) \subseteq R_j$; (3) $d(i,z) < 2^{j-2}$ for $i \in W(z)$. The first implies that $M|Z_j| \leqslant \sum_{z \in Z_j} |W(z)|$. We have $d(z,z') \geqslant 2^{j-1}$ for $z, z' \in Z'_j$ by definition, so with the second and third properties, we get that the witness sets of Z'_j are disjoint subsets of R_j, and so $\sum_{z \in Z'_j} |W(z)| \leqslant |R_j|$. Putting all of the above together, we have $Mc(H) \leqslant \sum_j 2^{j+1}|R_j|$. □

Next, we show that we can charge the cost shares against the optimal solution on any HST embedding T.

Lemma 4.8. $\sum_j 2^{j+1}|R_j| \leqslant O(1)\,OPT(T)$ *for all HST embeddings* T *of* (X, d).

Proof. Let T be a HST embedding of (X, d). We begin by expressing $OPT(T)$ in terms of T's cuts. Define $D(S) = \{(s_i, t_i) : (s_i, t_i) \in \delta(S)\}$ for each vertex subset $S \subseteq X$. Consider a level-j cut $C_e \in \mathcal{C}_j(T)$. By definition, $e \in E_j(T)$ and has length 2^{j-1}. If $(s_i, t_i) \in \delta(C_e)$, then e lies on the (s_i, t_i) path in T. Thus, the optimal solution on T either rents e for each terminal pair in $D(C_e)$ at a cost of $2^{j-1}|D(C_e)|$ or buys it at a cost of $2^{j-1}M$ and so

$$OPT(T) = \sum_j \sum_{C \in C_j(T)} 2^{j-1} \cdot \min\{M, |D(C)|\}.$$

For each rent terminal $t_i \in R_j$, charge its cost share 2^{j+1} to the level-$(j-2)$ cut $C \in \mathcal{C}_{j-2}(T)$ containing i. Each level-j cut $C \in \mathcal{C}_j(T)$ receives a charge of $2^{j+3}|R_{j+2} \cap C|$

and so

$$\sum_j 2^{j+1}|R_j| = \sum_j 2^{j+3} \sum_{C \in \mathcal{C}_j(T)} |R_{j+2} \cap C|.$$

It remains to prove the following claim: for each level-j cut $C \in \mathcal{C}_j(T)$, we have $|R_{j+2} \cap C| \leqslant \min\{M, |D(C)|\}$. Since C has diameter less than 2^j and $d(s_i, t_i) \geqslant 2^{j+2}$ for each $t_i \in R_{j+2}$, we have $|R_{j+2} \cap C| \leqslant |D(C)|$. So now we prove that $|R_{j+2} \cap C| \leqslant M$. Suppose, towards a contradiction, that $|R_{j+2} \cap C| > M$ and let t_i be the last-arriving terminal of $R_{j+2} \cap C$. The terminals of $R_{j+2} \cap C \setminus \{t_i\}$ arrive before t_i, are each of distance less than 2^j from t_i (diameter of C is less than 2^j) and of the same class as t_i, so they are part of t_i's witness set $W(t_i)$. Since $|R_{j+2} \cap C| > M$, we have that $|W(t_i)| \geqslant |R_{j+2} \cap C \setminus \{t_i\}| \geqslant M$. Thus, t_i would have been a buy terminal but this contradicts the assumption that $t_i \in R_{j+2}$. Therefore, we have $|R_{j+2} \cap C| \leqslant M$ as desired. This completes the proof of the claim and so $\sum_j 2^{j+1}|R_j| \leqslant 16\,\mathrm{OPT}(T)$. □

Finally, Lemmas 4.7 and 4.8 imply that the cost of Algorithm 4.2 is at most $O(1)\,\mathrm{OPT}(T)$ for any HST embedding T of (X, d). Furthermore, by Corollary 2.5, there exists a HST embedding T^* such that $\mathrm{OPT}(T^*) \leqslant O(\log k)\,\mathrm{OPT}$. Thus the algorithm is $O(\log k)$-competitive for the online rent-or-buy problem and this proves Theorem 4.1.

4.3 Connected Facility Location

In this section, we consider the connected facility location problem and prove Theorem 4.2. Recall the problem statement. At the beginning, the algorithm is given a parameter $M \geqslant 0$, a set of *facilities* $F \subseteq V$ and facility opening costs f_x for each facility $x \in F$. There is a designated root facility $r \in F$ with opening cost $f_r = 0$. The algorithm maintains a set of open facilities F' and a subgraph H

connecting F' and r. When a client i arrives, the algorithm may open a new facility, and then assigns i to some open facility $\sigma(i) \in F'$. The cost of the algorithm is $\sum_{x \in F'} f_x + Mc(H) + \sum_i d(i, \sigma(i))$. We call $Mc(H)$ the *Steiner cost* and $d(i, \sigma(i))$ the *assignment cost* of client i. The special case when $M = 0$ (i.e. open facilities can be connected for free) is called the online facility location problem.

Intuition. The connected facility location problem shares a similar cost structure with single-source rent-or-buy. Here, the "buy cost" consists of the facility opening cost and the Steiner cost, the "rent cost" is the assignment cost. At a high-level, we use a similar strategy: a facility is only opened if its opening cost and the additional Steiner cost to connect it to the other open facilities can be paid for using the assignment costs of clients near it. In order to do this, we use an online facility location algorithm together with an adaptation of our single-source rent-or-buy algorithm. The former indicates whether the opening cost can be paid for, and the latter indicates whether the additional Steiner cost can be paid for.

In the analysis, we will not be able to show that the cost of the algorithm is at most $O(1)\,OPT(T)$ for any HST embedding T of the clients because there is a lower bound of $\Omega(\frac{\log k}{\log \log k})$ for the facility location problem even on HSTs [Fot08]. Essentially, routing is easy on trees but choosing a good set of facilities to open online remains difficult. Instead, we will use the fact that the connected facility location instance induces a rent-or-buy instance and a facility location instance on the same metric, and that there is a $O(\log k)$-competitive facility location algorithm [Fot07]. Our analysis charges part of the cost to the cost of the facility location algorithm and the rest to the optimal rent-or-buy solution.

Algorithm. Since the root facility has zero opening cost, it can be assumed to be open. Our algorithm (Algorithm 4.3) runs Fotakis's $O(\log k)$-competitive algorithm [Fot07]—call it OFL-ALG—in parallel. Denote by $\hat{F}$ its open facilities by $\hat{F}$ and by $\hat{\sigma}$ its assignments, and call them *virtual facilities* and *virtual assignments*, respectively. Our algorithm only opens a facility if it was already opened by OFL-ALG. i.e. its open facilities F' is a subset of the virtual facilities $\hat{F}$.

Our algorithm classifies clients into one of 3 types: virtual, buy, and rent. When a client i arrives, our algorithm does the following. Let $x \in F'$ be the nearest open facility; define $a_i = d(i, x)$ and $\text{class}(i) = \lfloor \log a_i \rfloor$. Suppose $\text{class}(i) = j$. The client is first passed to OFL-ALG, which may open a new virtual facility and then it assigns i to the nearest virtual facility $\hat{\sigma}(i)$. If x is not much further than $\hat{\sigma}(i)$—in particular, if $d(i, x) \leqslant 4d(i, \hat{\sigma}(i))$—then i is called a *virtual client* and is assigned to x. Otherwise, our algorithm will consider opening $\hat{\sigma}(i)$ and assigning i to it. Denote by R_j the set of rent clients i' with $\text{class}(i') = j$. The *witness set* of i is defined to be $W(i) = \{v \in R_j : d(i, v) \leqslant 2^{j-2}\}$. If $|W(i)| \geqslant M$, then i is called a *buy client*, the facility $\hat{\sigma}(i)$ is opened and i is assigned to it, and the edge $(\hat{\sigma}(i), x)$ is added to H (i.e. $\hat{\sigma}(i)$ is connected to the other open facilities by the edge $(\hat{\sigma}(i), x)$). Otherwise, i is called a *rent client* and is assigned to x.

In the description of Algorithm 4.3 below, Q_j and R_j are the sets of virtual and buy clients i' with $\text{class}(i') = j$, respectively.

Analysis. Let X be the set of clients and the root, and k be the number of clients. Consider a facility location instance over metric (V, d) with the same facilities F and clients X as well as a rent-or-buy instance over metric (V, d) with terminals X and root r. Let OPT_{FL} and OPT_{ROB} be the cost of the respective optimal solutions. A feasible solution to the connected facility location instance is feasible for both the

Algorithm 4.3 Algorithm for Online Connected Facility Location

Initialize $F' \leftarrow \{r\}; H \leftarrow \emptyset; Q_j \leftarrow \emptyset; Z_j \leftarrow \emptyset; R_j \leftarrow \emptyset;$
while client i arrives **do**
  Pass i to OFL-ALG and update virtual solution $\hat{F}, \hat{\sigma}$
  Let $x \in F'$ be nearest open facility to i and $j = \lfloor \log d(i, x) \rfloor$
  Assign class$(i) \leftarrow j$
  $W(i) \leftarrow \{i' \in R_j : d(i', i) < 2^{j-2}\}$
  if $a_i \leqslant 4d(i, \hat{\sigma}(i))$ **then**
    Assign i to x and add i to Q_j
  else if $|W(i)| \geqslant M$ **then**
    Open $\hat{\sigma}(i)$ and add $\hat{\sigma}(i)$ to F'
    Add edge $(\hat{\sigma}(i), x)$ to H
    Assign i to $\hat{\sigma}(i)$ and add i to Z_j
  else
    Assign i to x and add i to R_j
  end if
end while

facility location and rent-or-buy instances so we have the following lemma.

Lemma 4.9. $\mathrm{OPT}_{\mathrm{FL}} \leqslant \mathrm{OPT}$ *and* $\mathrm{OPT}_{\mathrm{ROB}} \leqslant \mathrm{OPT}$.

Let Q, Z, R be the set of virtual, buy, and rent clients, respectively. Note that a virtual or rent client $i \in Q \cup R$ has assignment cost $d(i, \sigma(i)) = a_i$ and a buy client $i \in Z$ has assignment cost $d(i, \sigma(i)) = d(i, \hat{\sigma}(i))$. Thus the cost of the algorithm is

$$\sum_{x \in F'} f_x + Mc(H) + \sum_{i \in Q \cup R} a_i + \sum_{i \in Z} d(i, \hat{\sigma}(i)).$$

We first charge the opening cost as well as the assignment cost of virtual and buy clients to the cost of the virtual solution. Then we charge the Steiner cost and the assignment cost of rent clients to $\mathrm{OPT}_{\mathrm{ROB}}$.

Lemma 4.10. $\sum_{x \in F'} f_x + \sum_{i \in Q} a_i + \sum_{i \in Z} d(i, \hat{\sigma}(i)) \leqslant O(\log k)\,\mathrm{OPT}$.

Proof. Our algorithm only opens a facility if it was already opened by OFL-ALG so $\sum_{x \in F'} f_x \leqslant \sum_{x \in \hat{F}} f_x$. Furthermore, the assignment cost of a virtual client $i \in Q$

is $a_i \leqslant 4d(i, \hat{\sigma}(i))$, and for each buy client $i \in Z$ we have $d(i, \sigma(i)) = d(i, \hat{\sigma}(i))$. Therefore, we have

$$\sum_{x \in F'} f_x + \sum_{i \in Q} a_i + \sum_{i \in Z} d(i, \hat{\sigma}(i)) \leqslant \sum_{x \in \hat{F}} f_x + 4 \sum_{i} d(i, \hat{\sigma}(i)).$$

Since OFL-ALG is a $O(\log k)$-competitive algorithm for facility location, the cost of its virtual solution is $\sum_{x \in \hat{F}} f_x + \sum_i d(i, \hat{\sigma}(i)) \leqslant O(\log k)\, OPT_{FL}$. The lemma now follows from Lemma 4.9. □

Next, we show that the Steiner cost and the assignment cost of rent clients $Mc(H) + \sum_{i \in R} a_i$ is at most a constant times the cost of the optimal rent-or-buy solution on any HST embedding of the terminals. This part of the analysis is analogous to that in Section 4.1. For each rent client $i \in R_j$, we define i's cost share to be 2^{j+1}. We now show that the Steiner cost and the assignment cost of rent clients is at most thrice the total cost share $\sum_j 2^{j+1}|R_j|$.

We will need the following lemma which says that class-j buy clients are at least 2^{j-1}-apart from each other.

Lemma 4.11. $d(z, z') \geqslant 2^{j-1}$ *for* $z, z' \in Z_j$.

Proof. By triangle inequality, we have $d(z, z') \geqslant d(z, \sigma(z')) - d(z', \sigma(z'))$. Suppose z arrived after z'. Since a_z is defined to be the distance from z to the nearest open facility when z arrived and the facility $\sigma(z')$ that z' was assigned to is open at this time, we have $d(z, \sigma(z')) \geqslant a_z$. Moreover, z' is a buy client and so we assigned it to a facility $\sigma(z')$ such that $d(z', \sigma(z')) < \frac{1}{4} a_{z'}$. We now have that $d(z, z') \geqslant a_z - \frac{1}{4} a_{z'}$. Since z, z' are of class j, we have $a_z \geqslant 2^j$ and $a_{z'} < 2^{j+1}$. Thus, $d(z, z') \geqslant 2^{j-1}$. □

Lemma 4.12. $Mc(H) + \sum_{i \in R} a_i \leqslant 3 \sum_j 2^{j+1}|R_j|$.

Proof. Since the cost share of a rent terminal $i \in R$ is at least a_i, it suffices to show that $Mc(H) \leqslant 2\sum_{i \in R} a_i$. We will prove this using the following two claims.

Claim 4.13. $c(H) \leqslant \sum_{z \in Z} 2a_z$.

Proof. We proceed by charging the increase in $c(H)$ due to an open facility to the buy client that opened it. For each open facility $y \in F'$, define $z(y) \in Z$ to be the buy client that opened it and $x(y) \in F'$ to be the open facility that y was connected to when it was opened. So $c(H) = \sum_{y \in F'} d(y, x(y))$. We now show that $d(y, x(y)) \leqslant 2a_{z(y)}$ for each $x \in F'$.

Fix an open facility $y \in F'$. For brevity, we write z, and x, in place of $z(y)$, and $x(y)$, respectively. By triangle inequality, $d(y, x) \leqslant d(y, z) + d(z, x)$ so now we bound the right-hand side in terms of a_z. By definition of the algorithm, at the time when z just arrived, before it opened y, we have that x was the nearest open facility and y was the nearest virtual facility. So $a_z = d(z, x)$ and $y = \hat{\sigma}(z)$. Furthermore, $d(z, \hat{\sigma}(z)) < \frac{1}{4}a_z$ since z is not a virtual client. Thus, $d(y, x) \leqslant d(y, z) + d(z, x) \leqslant 2a_z$, as desired.

Each open facility was opened by a distinct buy client so $c(H) = \sum_{y \in F'} d(y, x(y)) \leqslant \sum_{z \in Z} 2a_z$. □

Claim 4.14. $\sum_{z \in Z} Ma_z \leqslant \sum_j 2^{j+1}|R_j|$.

Proof. We have $\sum_{z \in Z} Ma_z \leqslant \sum_j 2^{j+1} M|Z_j|$. We now prove that $M|Z_j| \leqslant |R_j|$ for each class j. The witness set $W(z)$ of $z \in Z_j$ satisfies the following properties: (1) $|W(z)| \geqslant M$; (2) $W(z) \subseteq R_j$; (3) $d(i, z) < 2^{j-2}$ for $i \in W(z)$. The first implies that $M|Z_j| \leqslant \sum_{z \in Z_j} |W(z)|$. Lemma 4.11 together with the second and third properties imply that the witness sets of Z_j are disjoint subsets of R_j and so $\sum_{z \in Z_j} |W(z)| \leqslant |R_j|$. These two inequalities imply that $\sum_{z \in Z} Ma_z \leqslant \sum_j 2^{j+1}|R_j|$. □

Combining these claims, we have $Mc(H) \leqslant \sum_{z \in Z} 2M a_z \leqslant 2 \sum_j 2^{j+1}|R_j|$. Therefore, $Mc(H) + \sum_{i \in R} a_i \leqslant 3 \sum_j 2^{j+1}|R_j|$. □

Now we show that the total cost share is at most a constant times the cost of the optimal rent-or-buy solution on any HST embedding of the terminals.

Lemma 4.15. $\sum_j 2^{j+1}|R_j| \leqslant O(1)\, OPT_{ROB}(T)$ *for any HST embedding* T *of* (X, d).

Proof. Let T be a HST embedding. We charge the cost share of class-j clients to the level-$(j-2)$ cuts. So overall, we have

$$\sum_j 2^{j+1}|R_j| = \sum_j \sum_{C \in \mathcal{C}_j(T)} 2^{j+3}|R_{j+2} \cap C|.$$

The rest of the proof proceeds as in the proof of Lemma 4.6. □

Now we have all the required ingredients to bound the competitive ratio of Algorithm 4.3. By Corollary 2.5 and Lemma 4.9, there exists a HST embedding T^* such that $OPT_{ROB}(T^*) \leqslant O(\log k)\, OPT_{ROB} \leqslant O(\log k)\, OPT$. So Lemmas 4.12 and 4.15 imply that the Steiner cost and the assignment cost of rent clients is $Mc(H) + \sum_{i \in R} a_i \leqslant O(\log k)\, OPT$. Lemma 4.10 says that the remainder of the algorithm's cost is at most $O(\log k)\, OPT$ as well. Thus it is $O(\log k)$-competitive for connected facility location and this proves Theorem 4.2.

4.4 Prize-Collecting Steiner Tree

In this section, we give a simple algorithm and analysis for the prize-collecting Steiner tree problem and prove the following theorem.

Theorem 4.16 ([QW11])**.** *There is a deterministic* $O(\log k)$*-competitive algorithm for the online prize-collecting Steiner tree problem.*

Recall the problem statement. The algorithm is given a root terminal r initially and maintains a subgraph H online. At each online step, a terminal i with *penalty* π_i arrive and the algorithm can either pay the penalty or augment H such that H connects i to the root. We say that i is a *penalty terminal* if the algorithm chose to pay i's penalty, and a *buy terminal* otherwise. Denote by P the set of penalty terminals. The total cost of the algorithm is $c(H) + \sum_{i \in P} \pi_i$.

Algorithm. Our algorithm (Algorithm 4.4) associates to each terminal i a cost share ρ_i. When a terminal i with penalty π_i arrives, our algorithm does the following. Let $z \in Z$ be the closest buy terminal; define $a_i = d(i, z)$ and define $\text{class}(i) = \lfloor \log a_i \rfloor$. Suppose $\text{class}(i) = j$. Denote by X_j the set of terminals i' with $\text{class}(i') = j$. The *witness set* of i is $W(i) = \{i' \in X_j : d(i, i') \leqslant 2^{j-1}\}$. Our algorithm initializes i's cost share ρ_i to 0 and raises ρ_i until either $\rho_i + \sum_{i' \in W(i)} \rho_{i'} = 2^{j+1}$ or $\rho_i = \pi_i$. In the first case, the edge (i, z) is added to H; in the second case, the penalty is paid.

Algorithm 4.4 Algorithm for Online Prize-Collecting Steiner Tree

$Z \leftarrow \{r\}$; $H \leftarrow \emptyset$; $X_j \leftarrow \emptyset$ for all j
while terminal i with penalty π_i arrives **do**
  Let z be closest terminal in Z to i, set $j = \lfloor \log d(i, z) \rfloor$ and add i to X_j
  $W(i) \leftarrow \{i' \in X_j : d(i, i') < 2^{j-1}\}$
  Initialize $\rho_i \leftarrow 0$ and increase ρ_i until $\rho_i + \sum_{i' \in W(i)} \rho_{i'} \geqslant 2^{j+1}$ or $\rho_i = \pi_i$
  if $\sum_{i' \in W(i)} \rho_{i'} \geqslant 2^{j+1}$ **then**
    Add i to Z and buy (i, z), i.e. $H \leftarrow H \cup \{(i, z)\}$
  else
    Pay penalty π_i
  end if
end while

Analysis. Let X be the set of terminals that arrived. For each buy terminal $z \in Z$, the algorithm incurs a cost of a_z. Thus, the total cost of the algorithm is $\sum_{z \in Z} a_z +$

$\sum_{i \in P} \pi_i$. Our goal in the analysis is to show that this is at most $\mathrm{OPT}(T)$ for any HST embedding of (X, d).

First, we show that the cost of the algorithm is at most twice the total cost share $\sum_i \rho_i$.

Lemma 4.17. *We have that* $\sum_{z \in Z} a_z + \sum_{i \in P} \pi_i \leqslant 2 \sum_i \rho_i$.

Proof. The algorithm pays the penalty π_i for terminal i only if $\rho_i = \pi_i$ so $\sum_{i \in P} \pi_i \leqslant \sum_i \rho_i$. Let $Z_j \subseteq Z$ be the set of class-j buy terminals. Since $a_z \in [2^j, 2^{j+1})$ for $z \in Z_j$, we have $\sum_{z \in Z} a_z \leqslant \sum_j 2^{j+1} |Z_j|$. We now show that $2^{j+1}|Z_j| \leqslant \sum_{i \in X_j} \rho_i$ for each class j.

Fix a class j. The witness set $W(z)$ of $z \in Z_j$ satisfies the following properties: (1) $\sum_{i \in W(z)} \rho_i \geqslant 2^{j+1}$; (2) $W(z) \subseteq X_j$; (3) $d(i, z) < 2^{j-1}$ for $i \in W(z)$. The first implies that $2^{j+1}|Z_j| \leqslant \sum_{z \in Z_j} \sum_{i \in W(z)} \rho_i$. We claim that $d(z, z') \geqslant 2^j$ for $z, z' \in Z_j$. This completes the proof since together with the second and third properties, we have that the witness sets of Z_j are disjoint subsets of X_j and so $\sum_{z \in Z_j} \sum_{i \in W(z)} \rho_i \leqslant \sum_{i \in X_j} \rho_i$.

Now we prove the claim. Observe that H is the subgraph produced by the online greedy Steiner tree algorithm if it were run on the subsequence of buy terminals Z, and for each $z \in Z$ we have that a_z is exactly the distance from z to the nearest previously-arrived buy terminal. Thus, we can apply Lemma 3.3 and get that $d(z, z') \geqslant 2^j$ for any $z, z' \in Z_j$, proving the claim. Putting all of the above together, we get $\sum_{z \in Z} a_z \leqslant \sum_j \sum_{i \in X_j} \rho_i = \sum_i \rho_i$. This finishes the proof of the lemma. □

Next, we show that the total cost share is at most a constant times the cost of the optimal solution on any HST embedding.

Lemma 4.18. $\sum_i \rho_i \leqslant 8\,\mathrm{OPT}(T)$ *for any HST embedding* T *of* (X, d).

Proof. Let T be a HST embedding of (X, d). First, we lower bound $\mathrm{OPT}(T)$ in terms of T's cuts. Define $R_j \subseteq X_j$ to be the subset of class-j terminals i with $\rho_i > 0$. Consider a level-j cut $C_e \in \mathcal{C}_j(T)$ and associate the terminals $R_{j+1} \cap C_e$ to C_e. (Note that a terminal can only be associated to one cut.) By definition, $e \in E_j(T)$ and has length 2^{j-1}. If $r \notin C_e$ then e lies on the (i, r) path in T for each associated terminal $i \in R_{j+1} \cap C_e$ so the optimal solution on T either pays the penalty for each associated terminal at a cost of $\sum_{i \in R_{j+1} \cap C_e} \pi_i$ or buys e at a cost of 2^{j-1}. Since each terminal is associated to a unique cut, we have

$$\mathrm{OPT}(T) \geqslant \sum_j \sum_{C \in C_j(T) : r \notin C} \min \left\{ \sum_{i \in R_{j+1} \cap C} \pi_i, 2^{j-1} \right\}.$$

We have $\sum_i \rho_i = \sum_j \sum_{i \in R_j} \rho_i$. For each terminal $i \in R_j$, we charge ρ_i to the level-$(j-1)$ cut $C \in \mathcal{C}_{j-1}$ containing i. Each level-j cut $C \in \mathcal{C}_j(T)$ is charged $\sum_{i \in R_{j+1} \cap C} \rho_i$. Thus, we have

$$\sum_i \rho_i = \sum_j \sum_{C \in \mathcal{C}_j(T)} \left(\sum_{i \in R_{j+1} \cap C} \rho_i \right).$$

Since $\rho_i \leqslant \pi_i$ for each terminal i, it suffices to prove the following claim: for each level-j cut $C \in \mathcal{C}_j(T)$, we have $\sum_{i \in R_{j+1} \cap C} \rho_i = 0$ if $r \in C$, and $\sum_{i \in R_{j+1} \cap C} \rho_i \leqslant 2^{j+2}$ if $r \notin C$. Suppose $r \in C$. Since $a_i < d(i, r) < 2^j$ for each $i \in C$, there cannot be any class-$(j+1)$ terminal in C and so $\sum_{i \in R_{j+1} \cap C} \rho_i = 0$. Now consider the case $r \notin C$. Suppose, towards a contradiction, that $\sum_{i \in R_{j+1} \cap C} \rho_i > 2^{j+2}$. Let i^* be the last-arriving terminal of $R_{j+1} \cap C$, i.e. i^* is the last-arriving class-$(j+1)$ terminal of C with $\rho_{i^*} > 0$. Since the diameter of C is less than 2^j, we have $W(i^*) \supseteq R_{j+1} \cap C$. We also have $\sum_{i \in R_{j+1} \cap C \setminus \{i^*\}} \rho_i < 2^{j+2}$ because otherwise the algorithm would not have increased ρ_{i^*}. But the algorithm increased ρ_{i^*} ensuring that $\sum_{i \in W(i^*)} \rho_i \leqslant 2^{j+1}$, contradicting the assumption that $\sum_{i \in R_{j+1} \cap C} \rho_i > 2^{j+1}$. This completes the proof of

the claim and so we get $\sum_i \rho_i \leqslant 8\,\mathrm{OPT}(T)$. □

Now, Lemmas 4.17 and 4.18 imply that the cost of Algorithm 4.4 is at most $O(1)\,\mathrm{OPT}(T)$ for any HST embedding T of (X, d). Furthermore, by Theorem 2.4, there exists a HST embedding T^* such that $\mathrm{OPT}(T^*) \leqslant O(\log k)\,\mathrm{OPT}$. Thus, the algorithm is $O(\log k)$-competitive for prize-collecting Steiner tree and this proves Theorem 4.16.

4.5 Notes

The work in this chapter appeared as part of [Umb15].

Related work. In the offline setting, Swamy and Kumar [SK04] gave a primal-dual 4.55-approximation algorithm for the single-source rent-or-buy problem and a 8.55-approximation algorithm for the connected facility location problem. Later, Gupta et al. [GKPR07] developed an elegant randomized sample-and-augment framework and improved the constants. Their approach was then refined by Eisenbrand et al. [EGRS08] and Grandoni and Rothvoss [GR10].

In the online setting, Awerbuch et al. [AAB04] was the first to consider the rent-or-buy problem (which they call the *network connectivity leasing problem*). They gave a randomized $O(\log k)$-competitive algorithm as well as a deterministic $O(\log^2 k)$-competitive algorithm and posed the existence of a deterministic $O(\log k)$-competitive algorithm as an open problem. The online connected facility location problem was recently proposed by San Felice et al. [SFWL14a] and they gave a randomized $O(\log^2 k)$-competitive algorithm based on the offline connected facility location algorithm of Eisenbrand et al. [EGRS08]. Subsequent to our work, the authors of [SFWL14a] independently improved their results and obtained a randomized

$O(\log k)$-competitive algorithm for the problem [SFWL14b]. For the special case of the facility location problem ($M = 0$), Fotakis [Fot08] showed that, surprisingly, there is a lower bound of $\Omega(\frac{\log k}{\log \log k})$ even when the underlying metric space is a HST. Qian and Williamson [QW11] were the first to consider the online prize-collecting Steiner forest problem, and they obtained a deterministic $O(\log k)$-competitive algorithm.

5 Online Multicast Games

In previous chapters, we studied a model of online network design in which the algorithm could dictate how each request is to be satisfied. For instance, in the online Steiner tree problem, we could connect each arriving terminal to any previous terminal. We now turn to a decentralized model in which each terminal is an autonomous selfish player.

5.1 Introduction

In many practical scenarios, users in a communications network are free to make routing decisions according to their own selfish interests. To study selfish behavior in networks, Anshelevich et al. [ADK+08] introduced the following *multicast game*. We are given a graph with a root and edge costs, and a set of terminals that want to connect to the root. Each terminal is associated with a selfish player. A state of the game consists of the paths taken by the players, and its cost is the cost of all edges used. The cost of a state is distributed among the players by dividing the cost of each edge evenly among the players using it. A player's total payment for the edges in its path is called its *cost share*. The concept of Nash equilibrium is often used to predict the outcome of selfish behavior in games. A *Nash equilibrium* in the multicast game is a stable state of the game in which no player can unilaterally switch to a different path and lower its cost share, i.e. make an improving move.

For the study of equilibria to be useful, we need to define what kind of initial states are allowed and how players can converge to an equilibrium. Chekuri et al. [CCLE+07] introduced an online model in which the players join the game one by one. We start in an empty state; at each time step, either a player not in the

game joins the game and chooses a path selfishly, or a player already in the game performs an improving move. We are interested in equilibria that can be reached by any sequence of arrivals and improving moves. So far, we only know of bounds in certain special settings. For the setting in which players first arrive one by one, and are allowed to make improving moves only after all players have joined the game, Chekuri et al. [CCLE+07] proved an upper bound of $O(\sqrt{N}\log^2 N)$ on the ratio of the worst equilibrium and the optimum (the minimum Steiner tree over the players) and a lower bound of $\Omega(\log N/\log\log N)$, where N is the number of players. These bounds were later refined by Charikar et al. [CKM+08] to $O(\log^3 N)$ and $\Omega(\log N)$, respectively.

In this chapter, we make progress on understanding equilibria in online multicast games. We study online multicast games on complete graphs[1], where players can also depart from the game; when a player leaves, its cost share is redistributed among players still in the game. We show that if arrivals and departures only happen at equilibria, and players can take turns making improving moves in between, then it is possible to reach an equilibrium, after all the arrivals and departures have happened, whose cost is at most $O(\log N)$ OPT, the cost of the optimal Steiner tree over the root and the set of vertices on which a player arrived at some point. This is the best possible in light of the lower bound in [CKM+08]. On the other hand, if arrivals and departures can happen anytime, then there exists a sequence of arrivals and departures such that after the final arrival, the state is an equilibrium and costs $N^{1/6} \cdot \mathrm{OPT}$.

[1] This is not without loss of generality, unlike in previous chapters.

5.2 Model

We now formally define the model of online multicast games with departures. Since the underlying graph is complete, we can assume without loss of generality that the game is taking place over a metric (V, d) instead. Players arrive and depart one at a time. When a player a arrives at a vertex v, it becomes *active* and chooses a path P_a from v to r that minimizes its *cost share*:

$$\sum_{e \in P_a} \frac{d(e)}{x(e)}$$

the player pays $d(e)/x_e$, where x_e is the number of active players using e (including a itself); we say that P_a is a *best-response* path for a. When a player departs, it becomes *inactive* and the cost of the state is redistributed among the remaining active players. At any point in time, the *state* S of the game is defined to be the set of paths used by currently-active players. Let $x_e(S)$ be the number of players using e in S. The cost $c(S)$ of state S is defined to be the cost of the edges used by active players. Our benchmark OPT will be the cost of the minimum Steiner tree over $X \cup \{r\}$, where X is the set of vertices on which at least one player arrived. In the following, we will use N to denote the number of players.

Our results are the following two theorems.

Theorem 5.1. *If arrivals and departures can only happen at equilibrium, and active players can take turns making improving moves ending in an equilibrium after each arrival and departure, then after* N *arrivals and departures, it is possible to reach an equilibrium* S^f *that costs* $c(S^f) \leqslant O(\log N)\,\mathrm{OPT}$.

Theorem 5.2. *If arrivals and departures can happen at any state, then there exists a sequence* σ *of* N *arrivals and departures such that after the final arrival, the state* S^f *is an*

equilibrium and $c(S^f) \geqslant \Omega(N^{1/6})\,\mathrm{OPT}(\sigma)$.

We prove Theorem 5.1 in the following section and Theorem 5.2 in Section 5.4.

5.3 Upper Bound

We assume, without loss of generality, that each player arrives at a distinct vertex. If there is a player at vertex v, we will use "player v" to refer to the player, and say that v is "active."

We begin with some notation. Let S be a state. Define $A(S)$ to be the set of active players in S, $E(S)$ to be the set of edges used by $A(S)$, and $V(S)$ to be the set of vertices incident to $E(S)$. In the following, we will restrict ourselves to states S such that $E(S)$ is acyclic. Since $E(S)$ is acyclic, each vertex $v \in V(S)$ has a unique path S_v to the root in $E(S)$; define $e_v(S) \in E(S)$ to be the first edge on S_v. If there is an active player at vertex v, i.e. $v \in A(S)$, then its path is S_v.

At a high level, after each arrival, we will perform sequences of improving moves, called *tree-follow* sequences, until we reach an equilibrium. Our goal is to construct tree-follow sequences such that when we reach an equilibrium S, such that any two edges that are used by some players and are of roughly equal length are not too close together. This then implies the desired bound on $c(S)$. More precisely, we will use a charging scheme that relies on a HST embedding T^* of (V, d) obtained from the HST embedding result of Fakcharoenphol et al. [FRT04].

Lemma 5.3. *There exists a HST embedding* T^* *of* (V, d) *with* $\mathrm{OPT}(T^*) \leqslant O(\log N)\,\mathrm{OPT}$, *where* $\mathrm{OPT}(T^*)$ *is the cost of the minimum Steiner tree over* $X \cup \{r\}$ *in* T^* *and* OPT *is the cost of the minimum Steiner tree over* $X \cup \{r\}$ *in* (V, d).

Note that since each player arrives at a distinct vertex, we have $|X| = N$.

Charging scheme. We now describe a charging scheme that charges the cost of a state S to T^*. We begin by expressing $\mathrm{OPT}(T^*)$ in terms of cuts from the hierarchical decomposition $\mathcal{C}(T^*)$:

$$\mathrm{OPT}(T^*) = \sum_j 2^{j-1}|E_j(T^*)| = \sum_j 2^{j-1}|\mathcal{C}_j(T^*)|.$$

Let $X_j(S) = \{v : d(e_v) \in [2^j, 2^{j+1})$, and note that $c(S) \leqslant \sum_j 2^{j+1}|X_j(S)|$. For each player $v \in X_j(S)$, we charge the cost of its first edge to the unique level-$(j-2)$ cut containing it. For a cut $C \in \mathcal{C}_{j-2}$, define $\mathrm{Charge}(S, C) = X_j(S) \cap C$, the set of vertices whose first edges are charged to C. For states S such that $E(S)$ is acyclic, we say that S is *good* if $|\mathrm{Charge}(S, C)| = 1$ for each cut C. Furthermore, S is *almost-good* if $|\mathrm{Charge}(S, C')| = 2$ for one cut C' and $|\mathrm{Charge}(S, C)| = 1$ for every other cut C; we call the two vertices $\mathrm{Charge}(S, C') = \{u, v\}$ the *conflicting vertices* of S. We also say that u conflicts with v and vice versa, and that their first edges, $e_u(S)$ and $e_v(S)$, are conflicting edges.

Proposition 5.4. *If u and v are conflicting vertices of a state S, then their distance is $d(u, v) < \min\{d(e_u(S)), d(e_v(S))\}/4$.*

Proof strategy. For a good state S, we have $c(S) \leqslant O(1)\,\mathrm{OPT}(T^*) \leqslant O(\log N)\,\mathrm{OPT}$. Since the initial empty state is (trivially) good, to prove Theorem 5.1, it suffices to show that if the current state is good and is an equilibrium, then after the next arrival or departure, there exists a sequence of improving moves that results in a state that is good and an equilibrium. This ensures that our final state is good, as desired. The proof consists of two parts. First, we show that arrivals affect the charging scheme minimally: if a player arrives at vertex u in state S' and the resulting state (after it chooses its best-response path) is S, then $E(S) = E(S') \cup \{(u, v)\}$ for some vertex

$v \in V(S')$—the best-response path connects directly to some vertex v via the edge (u, v) and then follow v's path S_v to the root (Lemma 5.5). Therefore, after an arrival, we are in either a good state or an almost-good state. Observe that if the current state is a good equilibrium, then if a player departs, the resulting state is still a good state (but not necessarily an equilibrium).

Next, we use an algorithm Find-Equilibrium to construct a sequence of improving moves that ends in a good equilibrium. While the current state S is not an equilibrium, it executes a sequence of improving moves called a tree-follow sequence, to be formally defined later; essentially, a tree-follow sequence swaps an edge of $E(S)$ with another edge not in $E(S)$. The algorithm maintains the invariant that the current state S is either good or almost-good. We will show that if S is good but not an equilibrium, then there exists an improving tree-follow sequence; if S is an almost-good state, then there exists an improving tree-follow sequence that swaps out one of the conflicting edges. In either case, the resulting state is either good or almost-good. Therefore, when the current state S is an equilibrium, and the algorithm terminates, S must be a good state.

Tree-follow moves. We now formally define follow and tree-follow moves. Fix a state S. For vertices $u, v \in V(S)$, we define $Q_{uv}(S)$ to be the path consisting of the edge (u, v) and the path S_v. If u is active, then we say that switching to $Q_{uv}(S)$ is a *follow move* for player u. Player u's current cost share on S_u is $\sum_{e \in S_u} \frac{d(e)}{x_e(S)}$; its cost share if it switched to $Q_{uv}(S)$ would be $\sum_{e \in Q_{uv}(S) \cap S_u} \frac{d(e)}{x_e(S)} + \sum_{e \in Q_{uv}(S) \setminus S_u} \frac{d(e)}{x_e(S)+1}$. So, if u were to switch to $Q_{uv}(S)$, its cost share would decrease by

$$\Delta_{uv}(S) = \sum_{e \in S_u \setminus Q_{uv}(S)} \frac{d(e)}{x_e(S)} - \sum_{e \in Q_{uv}(S) \setminus S_u} \frac{d(e)}{x_e(S)+1}.$$

Define $\Delta^*_u(S) = \max_{v \in V(S)} \Delta_{uv}(S)$, and let $Q^*_u(S)$ be the corresponding path, i.e. switching to $Q^*_u(S)$ is the best follow move for u. We will use the notation Follow$^*(S, u)$ to refer to the move itself as well as the state after executing the move.

Tree-follow moves generalize follow moves. Let $A(S, u)$ be the set of active players v whose paths S_v contain u and let S_{vu} denote the segment of S_v from v to u. The *tree-follow move* Tree-Follow$^*(S, u)$ consists of each player $v \in A(S, u)$ switching, in arbitrary order, to the path that consists of S_{vu} and $Q^*_u(S)$; in particular, v's new path follows its old path S_v up to u and then follows $Q^*_u(S)$. Observe that the cost share of the first player in $A(S, u)$ to switch decreases by $\Delta^*_u(S)$, and for every other player in $v \in A(S, u)$, v's cost share decreases by more than $\Delta^*_u(S)$ right after switching. Therefore, Tree-Follow$^*(S, u)$ corresponds to a sequence of improving moves if and only if $\Delta^*_u(S) > 0$. As before, we will use the notation Tree-Follow$^*(S, u)$ to refer to the move itself as well as the state after executing the move. Note that the move Tree-Follow$^*(S, u)$ swaps $e_u(S)$, the first edge of u, for the first edge of $Q^*_u(S)$.

Algorithm 5.1 FIND-EQUILIBRIUM

Input: State $S = \text{Arrival}(S', v)$ or $S = \text{Depart}(S', v)$

if $S = \text{Arrival}(S', v)$ **then**
  $\hat{v} \leftarrow u$
end if
while S is not an equilibrium **do**
  if S is good **then**
    Find vertex $u \in V(S)$ such that $\Delta^*_u(S) > 0$
    $S \leftarrow$ Tree-Follow$^*(S, u)$
    $\hat{v} \leftarrow u$
  else if S is almost-good **then**
    Let $u \in V(S)$ be the vertex that $\hat{v}$ that conflicts with
    $S \leftarrow$ Tree-Follow$^*(S, u)$
    $\hat{v} \leftarrow u$
  end if
end while
return S

In the following, we will use the notation Arrival(S', v) (Depart(S', v)) to denote the state after a player arrives at (departs from, resp.) v in state S'. The algorithm Find-Equilibrium is specified formally as Algorithm 5.1.

5.3.1 Analysis of Find-Equilibrium

First, we show that if S' is a good state and an equilibrium, then when a player arrives at u, its best-response is a follow move.

Lemma 5.5. *Let S' be a good state and an equilibrium. We have that* Arrival$(S', u) =$ Follow$^*(S', u)$.

Proof. Let $P^* = S'_u$ and (u, v) be the next first edge on P^*. Note that $-\Delta_{uv}(S)$ is exactly the cost share of player v if $P^* = Q_{uv}(S)$. We can assume without loss of generality that $v \in V(S)$. It suffices to show that $P^* = Q_{uv}(S)$; in other words, P^* consists of the edge (u, v) and the path S_v.

Suppose, towards a contradiction, that $P^* \neq Q_{uv}(S)$; in particular, suppose P^* consists of the edge (u, v) and some other path $R \neq S_v$ from v to the root. Player u's cost share on $Q_{uv}(S)$ would have been $d(u, v) + \sum_{e \in S_v} \frac{d(e)}{x_e(S)+1}$, while its cost share on P^* is $d(u, v) + \sum_{e \in R} \frac{d(e)}{x_e(S)+1}$. Since P^* is a best-response path for u, we have that the former cost share is at least the latter. Subtracting the contributions to both from (u, v) and $S_v \cap R$, we have that

$$\sum_{e \in S_v \setminus R} \frac{d(e)}{x_e(S) + 1} \geqslant \sum_{e \in R \setminus S_v} \frac{d(e)}{x_e(S) + 1}. \tag{5.1}$$

On the other hand, state S is an equilibrium, and so no player $u' \in A(S, v)$ can improve its cost share by switching to a path P that follows $S_{u'}$ to v and then R. Since

$S_{u'}$ and P differs in $S_v \setminus R$ and $R \setminus S_v$, we have

$$\sum_{e \in S_v \setminus R} \frac{d(e)}{x_e(S)} \leqslant \sum_{e \in R \setminus S_v} \frac{d(e)}{x_e(S)+1},$$

and so $\sum_{e \in S_v \setminus R} \frac{d(e)}{x_e(S)+1} < \sum_{e \in R \setminus S_v} \frac{d(e)}{x_e(S)+1}$. However, this contradicts Equation (5.1) and thus, $P^* = Q_{uv}(S)$, as desired. □

Now we analyze Find-Equilibrium.

Lemma 5.6. Find-Equilibrium *executes improving moves and returns a good equilibrium.*

Proof. We begin by showing that if S good but not an equilibrium, then there exists a vertex $u \in V(S)$ such that Tree-Follow$^*(S, u)$ is a sequence of improving moves.

Claim 5.7. *If S is a good state but not an equilibrium, then there exists a vertex $w \in V(S)$ such that $\Delta^*_w(S) > 0$.*

Proof. Since S is not an equilibrium, there exists a player with an improving alternate path. Among the set of paths that is an improving path for some player, define P^* to be one with the minimum number of new edges $|P^* \setminus E(S)|$. Suppose P^* is an improving path for player u. Let (w, z) be the edge of $P^* \setminus E(S)$ closest to the root along P^*, with w being the nearer vertex to u on P^*. Observe that the segment of P^* from w to the root is just the path $Q_{wz}(S)$. To prove that $\Delta^*_w(S) > 0$, we will now show that $\Delta_{wz}(S) > 0$.

Define path P' as the path that follows P^* from u to w and then follows S_w, instead of $Q_{wz}(S)$, from w to the root. Since $S_w \subseteq E(S)$ and $Q_{wz}(S) \setminus E(S) = \{(w, z)\}$, we have $|P' \setminus E(S)| < |P^* \setminus E(S)|$. By definition of P^*, we have that P' is not an improving path for u. In particular, u's cost share if it were to switch to P^* is less than its cost share if it were to switch to P'. Since P^* and P' differs in exactly $Q_{wz}(S) \setminus S_w$ and

$S_w \setminus Q_{wz}(S)$, we get

$$\sum_{e \in (Q_{wz}(S) \setminus S_w) \cap S_u} \frac{d(e)}{x_e(S)} + \sum_{e \in (Q_{wz}(S) \setminus S_w) \setminus S_u} \frac{d(e)}{x_e(S)+1}$$
$$< \sum_{e \in (S_w \setminus Q_{wz}(S)) \cap S_u} \frac{d(e)}{x_e(S)} + \sum_{e \in (S_w \setminus Q_{wz}(S)) \setminus S_u} \frac{d(e)}{x_e(S)+1}.$$

The LHS is at least $\sum_{e \in Q_{wz}(S) \setminus S_w} \frac{d(e)}{x_e(S)+1}$ and the RHS is at most $\sum_{e \in S_w \setminus Q_{wz}(S)} \frac{d(e)}{x_e(S)}$. Therefore, we have $\Delta_{wz} > 0$, as desired. □

Next, we show that if the current state S is almost-good, then there exists an improving tree-follow sequence that swaps out a conflicting edge.

Claim 5.8. *In any iteration of* Find-Equilibrium, *if the current state S is almost-good, then $\Delta^*_u(S) > 0$, where u is the vertex that $\hat{v}$ conflicts with. In particular, S cannot be an equilibrium.*

Proof. We have that $S = \text{Arrival}(S', \hat{v})$ or $S = \text{Tree-Follow}^*(S', \hat{v})$. We will show that in both cases, $\Delta_{u\hat{v}}(S') > 0$. Consider the first case, when $S = \text{Tree-Follow}^*(S', \hat{v})$. We can assume without loss of generality that $|A(S, \hat{v})| = 1$. This is because as $|A(S, \hat{v})|$ increases, $x_e(S')$ increases for $e \in S'_{\hat{v}}$ and $x_e(S')$ can only decrease for $e \notin S'_{\hat{v}}$, which in turn increases $\Delta_{u\hat{v}}(S')$.

Let $e_{\hat{v}}(S') = (\hat{v}, w)$. Since $S' = \text{Tree-Follow}^*(S, \hat{v})$, we have that $\text{Tree-Follow}^*(S', \hat{v})$ is not an improving sequence and so $\Delta_{\hat{v}u}(S') \leqslant 0$. Furthermore, since $|A(S, \hat{v})| = 1$, there is only one player using the edge $(\hat{v}, w)$ in S'. Our goal is to show that $\Delta_{u\hat{v}}(S') > 0$, i.e.

$$d(u, \hat{v}) + \frac{d(\hat{v}, w)}{2} + \sum_{e \in S'_w \setminus S'_u} \frac{d(e)}{x_e(S')+1} < \sum_{e \in S'_u \setminus S'_w} \frac{d(e)}{x_e(S')}. \tag{5.2}$$

Since $\Delta_{\hat{v}u}(S') \leqslant 0$, we have

$$d(\hat{v}, w) + \sum_{e \in S'_w \setminus S'_u} \frac{d(e)}{x_e(S')} \leqslant d(u, \hat{v}) + \sum_{e \in S'_u \setminus S'_w} \frac{d(e)}{x_e(S') + 1}. \tag{5.3}$$

Furthermore, since u conflicts with $\hat{v}$, by Proposition 5.4, $d(u, \hat{v}) < d(\hat{v}, w)/4$ and so $d(\hat{v}, w) - d(u, \hat{v}) < d(u, \hat{v}) + \frac{d(\hat{v}, w)}{2}$. Together with Equation (5.3), we get Equation (5.2), and so $\Delta_{u\hat{v}}(S') > 0$, as desired.

Finally, the same argument applies to the case when $S = \text{Arrival}(S', \hat{v})$ since $S = \text{Follow}^*(S', \hat{v})$ by Lemma 5.5. □

Finally, we need to show that the algorithm terminates eventually. The multicast game is part of a wide class of games that admit a potential function [Ros73, MS96]. For the multicast game, the *potential* $\Phi(S)$ of a state S is

$$\Phi(S) = \sum_e d(e) \sum_{i=1}^{x_e(S)} \frac{1}{i},$$

and it satisfies the following properties:

- For any state S, $c(S) \leqslant \Phi(S) \leqslant O(\log N)c(S)$.
- If a player makes an improving move, then amount that the potential decreases by is the same as the decrease in the player's cost share.

The first property says that the potential of the initial state after an arrival or departure is bounded. Since at least one player's cost share improves in every iteration, the second property implies that the algorithm must terminate eventually. By Claim 5.8, the algorithm cannot terminate when the current state is almost-good. Thus, when the algorithm terminates, it returns a good state. This completes the proof of the lemma. □

Lemmas 5.5 and 5.6 give us Theorem 5.1.

5.4 Lower Bound

We now turn to proving Theorem 5.2. In the previous section, it is convenient to assume that players arrive at distinct vertices. For this section, it is convenient to assume that multiple players can arrive at the same vertex.

We will construct a family of lower bound instances. We now describe the m-th instance. The instance uses the metric induced by a graph $G = (V, E)$ with edge lengths. The vertex set consists of the root r and $m + 1$ *layers* $V^0, \ldots, V^m$. For $0 < i \leqslant m$, layer V^i consists of m *clusters* $C^i_1, \ldots, C^i_m$; each cluster is a clique of m vertices where each clique edge is of length $1/m$. We use $v^i_{j,k}$ to denote the k-th vertex of C^i_j. Layer V^0 consists of m^2 vertices, which are also labeled $v^0_{j,k}$ for $j, k \in [m]$. The vertices of V^m are called *end* vertices, and those of V^0 are called *auxiliary* vertices. In addition to the clique edges, there are also the following *inter-layer* edges E_L. Each auxiliary vertex $v^0_{j,k}$ has edges $(r, v^0_{j,k})$ and $(v^0_{j,k}, v^1_{j,k})$. For $0 < i < m$, we have an edge $(v^i_{j,k}, v^{i+1}_{k,j})$ for each $j, k \in [m]$. Each inter-layer edge has unit length. Note that there are $n = m^2(m + 1) + 1$ vertices.

Observe that each end vertex $v^n_{j,k}$ has a unique path $P_{j,k}$ to the root that consists of only inter-layer edges; furthermore, each inter-layer edge belongs to exactly one such path. In other words, the set of inter-layer edges is a disjoint union of all the paths $P_{j,k}$.

Final state. We will construct a sequence σ of $N = m^6$ arrivals and departures and show that the cost of the final state S^f is $c(S^f) = \Omega(m)\,\mathrm{OPT}$. Each of the vertex will have an arrival at some point in the sequence. First, we describe S^f and show that it

is an expensive equilibrium. The state S^f is as follows: at each end vertex $v^m_{j,k}$, there are m players using the path $P^m_{j,k}$.

Lemma 5.9. *State S^f is an equilibrium and $c(S^f) = \Omega(m)\,\mathrm{OPT}$.*

Proof. First, we prove that S^f is an equilibrium. Consider a player at end vertex $v^m_{j,k}$ with path $P^m_{j,k}$ and an alternative path P'. For every clique edge e, we have $x_e(S^f) = 0$, and for every inter-layer edge e, we have $x_e(S^f) = m$. So the player's current cost share is 1. The path P' contains at least one clique edge and at least m inter-layer edges. Thus, the player's cost share when it switches to P' is at least $\frac{1}{m} + \frac{m}{m+1} > 1$. Therefore, S^f is an equilibrium.

Next, we prove that $c(S^f) = \Omega(m)\,\mathrm{OPT}$. We have $E(S) = E_L$ and $d(E_L) = m^2(m-1) = \Omega(m^3)$. Since every vertex has at least one arrival at some point in the sequence, OPT is just the cost of the minimum spanning tree of G. Here is one minimum spanning tree: connect the vertices in each cluster using the clique edges, connect each cluster C^i_j to C^{i+1}_j via the inter-layer edge $(v^i_{j,j}, v^{i+1}_{j,j})$, and take all edges incident to V^0. There are a total of m^2 clusters so the clique edges cost $m^2((m-1)/m)$, the inter-layer edges used cost $m(m+1)$, so the total cost is $O(m^2)$. Thus, we have $c(S^f) = \Omega(m)\,\mathrm{OPT}$. □

Sequence of arrivals and departures. We now describe the sequence σ. It is constructed in m *phases*, each phase consisting of m^2 *rounds*, one per end vertex $v^m_{j,k}$, and indexed by (j,k). Let $\prec$ be an arbitrary total order on the pairs (j,k). The sequence σ is constructed to maintain the following invariant: at the end of round (j,k) of phase ℓ, there will be ℓ players on $v^m_{j',k'}$ for $(j',k') \prec (j,k)$, and $\ell - 1$ players on the remaining end vertices. Furthermore, each player on $v^m_{j,k}$ uses the path $P_{j,k}$.

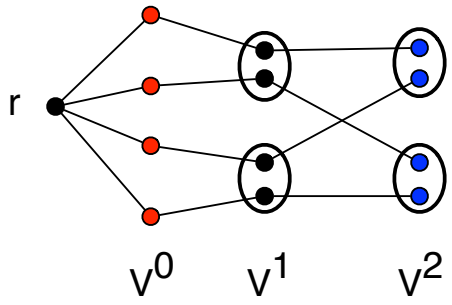

Figure 5.1: Example for $n = 2$. Auxiliary vertices are in red, end vertices are in blue. Ovals represent clusters. Only inter-layer edges are shown.

We now specify the subsequence for each round. Consider round (j, k) of phase ℓ. For simplicity of notation, we use v^i to denote the vertex of V^i on $P_{j,k}$. We also use P^i to denote the segment of $P_{j,k}$ starting at v^i and ending at the root. The round consists of $m+2$ iterations. In iterations $i = 0$ up to $m-1$, m^2 players arrive at v^i. In iteration $i = m$, one player arrives at v^m. Finally, the players on $v^0, \ldots, v^{m-1}$ depart.

We now show that the sequence σ maintains the above invariant and thus leads to the desired final state S^f.

Lemma 5.10. *At the end of round (j, k) of phase ℓ, there will be ℓ players on $v^1_{j',k'}$ for $(j', k') \prec (j, k)$, and $\ell - 1$ players on the remaining end vertices. Furthermore, each player on $v^1_{j,k}$ uses the path $P_{j,k}$.*

Proof. We prove by induction that the invariant is maintained. It is easy to see that the invariant holds at the end of round 1 of phase 1. Suppose we are currently at the beginning of round (j, k) of phase ℓ, and the invariant holds for the previous round. Let S be the state at the start of the round. We have that $x_e(S) \leqslant \ell$ for $e \notin P_{j,k}$, and $x_e(S) = \ell - 1$ for $e \in P_{j,k}$.

We will prove the following claim.

Claim 5.11. *For iterations $0 \leqslant i \leqslant m$, the unique best-response path for each player arriving at v^i is P^i.*

Proof. Observe that if the unique best-response path of the *first* player of iteration i arriving at v^i is P^i, then P^i is the unique best-response path of every other player arriving at v^i. Consider the base case when $i = 0$. For the first player arriving at v^0, its cost share on $P^0 = (r, v^0)$ is at most $1/\ell$. Any other path Q contains the edge (v^0, v^1) and at least one other edge, so the cost share on Q is more than $1/\ell$. Therefore, each player arriving at v^0 will choose the path P^0. This proves the base case.

Suppose that the claim holds for all iterations up to $i-1$, where $i \geqslant 1$. Let $e_i \in P^i$ be the first edge on the path P^i. Note that P^i consists of e_i followed by P^{i-1}. Let S^i be the state at the start of iteration i. Consider the first player arriving at v^i; call it player a. We have

- $x_{e_i}(S^i) = \ell - 1$,
- $x_e(S^i) \geqslant m^2$ for $e \in P^{i-1}$,
- $x_e(S^i) \leqslant \ell$ for $e \notin P^{i-1}$, and
- $x_e(S^i) = 0$ for each clique edge e.

Therefore, player a's cost share on P^i is at most $1/\ell + m/m^2 = 1/\ell + 1/m$. Any other path Q for a contains at least two inter-layer edges that do not belong to P^{i-1} and at least one clique edge, so a's cost share on Q is at least $2/(\ell+1) + 1/m > 1/\ell + 1/m$. Thus, player a's unique best-response path is P^i. A similar analysis shows that P^i is also the unique best-response path for each of the other players arriving at v^i □

This completes the proof of the lemma. □

Lemma 5.10 shows that the sequence σ of arrivals and departures end in the final state S^f, which costs $\Omega(m)$ OPT by Lemma 5.9. This completes the proof of Theorem 5.2.

5.5 Notes

This chapter is based on joint work with Shuchi Chawla, Debmalya Panigrahi, and Mohit Singh.

6 Random Permutation Model

In the final chapter, we study online network design in a model where the input is not adversarially chosen.

6.1 Introduction

In this chapter, we consider an input model called the "random permutation model," and study online network design in this model. Here, the adversary decides on the set of requests $\mathcal{I}$, but the algorithm is presented the requests in a (uniformly) random order σ. For instance, in the online Steiner tree problem, this would mean that the adversary decides on the set of terminals X but they appear in a random order to the algorithm. The quality of the algorithm is measured by its *expected competitive ratio*:

$$\max_{\mathcal{I}} \frac{\mathrm{E}_{\sigma}[\mathrm{ALG}(\mathcal{I}, \sigma)]}{\mathrm{OPT}(\mathcal{I})},$$

where the maximum is taken over all sets of requests $\mathcal{I}$, and $\mathrm{ALG}(\mathcal{I}, \sigma)$ is the cost of the algorithm when it is presented requests of $\mathcal{I}$ in the order given by σ.

When the terminals arrive in adversarial order, Imase and Waxman [IW91] showed that no algorithm for online Steiner tree can have an asymptotically better competitive ratio than $O(\log k)$, where k is the number of terminals. Garg et al. [GGLS08] showed that even when the terminals arrive in a random order, there is a lower bound of $\Omega(\log k)$ on the expected competitive ratio.

We consider a generalization of the online Steiner tree problem, called the online constrained forest problem. As before, the algorithm is given a metric (V, d) in advance. Each request is a proper function $h : 2^V \to \{0, 1\}$; this means that $h(S) = h(V \setminus S)$, and $h(S_1 \cup S_2) \leqslant \max\{h(S_1), h(S_2)\}$ for all disjoint sets S_1 and S_2. When

request h arrives, the algorithm has to update its subgraph F so that it *satisfies* h: for every $S \subseteq V$, $|F \cap \delta(S)| \geqslant h(S)$. The vertices v such that $h(\{v\}) = 1$ are called the *terminals* of h. The online constrained forest problem was first studied by Qian and Williamson [QW11] and they gave a deterministic $O(\log k)$-competitive algorithm[1].

Our results. We study the special case when each proper function h_i encodes a Steiner tree instance over terminals X_i and that there exists a common vertex $r \in \bigcap_i X_i$. We call this the *online Steiner tree with group arrivals problem*; each X_i is called a *group*. The optimal solution is the minimum Steiner tree on $\bigcup_i X_i$. We show that if the arrival order is adversarial, then no deterministic algorithm can have a competitive ratio asymptotically better than linear in the number of requests.

Theorem 6.1. *In the adversarial model, there is a lower bound of* $\Omega(\log k)$ *on the expected competitive ratio of any randomized algorithm for online Steiner tree with group arrivals.*

Our lower bound result holds for the so-called *oblivious adversary*, which has to construct the request sequence without knowing the random bits used by the algorithm. (In contrast, for deterministic algorithms, the adversary is allowed to construct the sequence adaptively, i.e. it can choose the next request in response to the algorithm's decisions on previous requests. See [BEY05] for a discussion of adversary models.)

On the other hand, in the random permutation model, there exists a deterministic algorithm that can perform much better, in terms of the number of requests.

Theorem 6.2. *In the random permutation model, there is a deterministic algorithm for online Steiner tree with group arrivals that achieves an expected competitive ratio* $O(\log g)$*, where the expectation is over the random permutation.*

[1] There is a 2-approximation for the offline constrained forest problem [GW95])

6.2 Upper Bound

We now turn to proving Theorem 6.2.

Algorithm. The algorithm is very simple and essentially runs Prim's minimum spanning tree algorithm over the new group. The algorithm maintains a subgraph F and the set Q of terminals that is connected to the root in F. In each time step, it receives a group $X_i \subseteq V$. Until $X_i \subseteq Q$, it adds the smallest edge between Q and an unconnected terminal $u \in X_i \setminus Q$. (See Algorithm 6.1).

Algorithm 6.1 Algorithm for Online Steiner Problem with Group Arrivals

```
1: F ← ∅
2: Q ← {r}
3: while group X_i arrives do
4:    while X_i ⊈ Q do
5:       Let (u, v) be the edge of smallest length such that u ∈ X_i \ Q and v ∈ Q.
6:       Add (u, v) to F and u to Q
7:    end while
8: end while
```

It is clear that Algorithm 6.1 produces a feasible solution, so we proceed to the cost analysis.

6.2.1 Analysis

Define $X = X_1 \cup \cdots \cup X_g$ and $\mathcal{X} = \{X_1, \ldots, X_g\}$. Let F_σ denote the algorithm's solution on arrival order σ, i.e. the i-th request is $X_{\sigma(i)}$. Our goal is to prove the following bound on the expected competitive ratio.

$$E_\sigma[c(F_\sigma)] \leqslant O(\log g)\,\mathrm{OPT}\,.$$

First, we show that for any arrival order σ and any HST embedding T of (X, d) of

a particular type called a $(\mathcal{X}, \sigma)$-embedding, we have $c(F_\sigma) \leqslant O(1)\,\mathrm{OPT}(T)$, where $\mathrm{OPT}(T)$ is the cost of the optimal solution on T (Lemma 6.4). Second, we will show that for each arrival order σ, there exists a distribution $\mathcal{D}_\sigma$ of $(\mathcal{X}, \sigma)$-embeddings such that $E_{\sigma, T\sim\mathcal{D}_\sigma}[\mathrm{OPT}(T)] \leqslant O(\log g)\,\mathrm{OPT}$ (Lemma 6.5 in the next section). These lemmas imply the desired bound on the expected competitive ratio since we have

$$E_\sigma[c(F_\sigma)] \leqslant E_\sigma[E_{T\sim\mathcal{D}_\sigma}[\mathrm{OPT}]] \leqslant O(\log g)\,\mathrm{OPT},$$

where the first inequality follows from Lemma 6.4 and the second from Lemma 6.5.

We begin by defining $(\mathcal{X}, \sigma)$-embeddings. We use the notation $i \prec_\sigma i'$ to denote that $\sigma^{-1}(i) < \sigma^{-1}(i')$, i.e. i comes before i' in the arrival order σ.

Definition 6.3 ($(\mathcal{X}, \sigma)$-Embeddings)**.** *A tree metric (X', T) is an* $(\mathcal{X}, \sigma)$-embedding *of (X, d) if T is a rooted tree and equipped with lengths on edges satisfying the following properties.*

1. *Leaf edges have length 1, and edge lengths double as one moves along a leaf-to-root path.*
2. *Every node is equidistant to each of its children.*
3. *The leaves of T are exactly the terminals X.*
4. *Each edge e in T is associated with a group $X_{f(e)}$. If e has length $T(e) = 2^{j-1}$, then its leaves C satisfies the following.*
   a) *We have $C \cap X_i = \emptyset$ for any i such that $i \prec_\sigma f(e)$.*
   b) *For any $v \in C \setminus X_i$, we have $d(v, X_{f(e)}) < 2^{j-1}$.*
   c) *For any $u, v \in C$, there exists a path between u and v whose vertices belong to X_i, and whose edges are of length less than 2^j each.*

We remark that an $(\mathcal{X}, \sigma)$-embedding T is not an embedding as defined in Chapter 2 since it is not expanding, i.e. there could be $u, v \in X$ such that $T(u, v) < d(u, v)$. This is actually a feature that we need since there is a lower bound of $\Omega(\log k)$ on the expected distortion of expanding embeddings into trees, and we want expected distortion of $O(\log g)$, where g could be much smaller than k.

As before, we will use the fact that an $(\mathcal{X}, \sigma)$-embedding T defines a decomposition of (X, d). Call an edge e of length 2^{j-1} a level-j edge and its leaves C a level-j cut. Denote by $E_j(T)$ the set of level-j edges and define $\mathcal{C}_j(T)$ to be the set of all level-j cuts. We call the family of partitions $\mathcal{C}_j(T)$ across all levels j an $(\mathcal{X}, \sigma)$-*decomposition*.

Let $\mathcal{T}_\sigma$ be the collection of $(\mathcal{X}, \sigma)$-embeddings. We now show that F_σ can be charged against any $T \in \mathcal{T}_\sigma$.

Lemma 6.4. *For all orderings σ and $T \in \mathcal{T}_\sigma$, we have $c(F_\sigma) \leqslant \mathrm{OPT}(T)$.*

Proof. Fix σ and $T \in \mathcal{T}_\sigma$. We will show that $\mathrm{ALG}(\sigma) \leqslant 4\,\mathrm{OPT}(T)$. For each terminal $v \in X$, define its *augmentation cost* a_v to be the length of the edge used by the algorithm to connect it to Q. Define $\mathrm{class}(v) = j$ if $a_v \in [2^j, 2^{j+1})$ and $Z_j \subseteq X$ to be the set of class-j terminals. So we have $c(F_\sigma) \leqslant \sum_j 2^{j+1}|Z_j|$.

First, we express $\mathrm{OPT}(T)$ in terms of its hierarchical decomposition. Since the terminals are exactly the leaves of T, the unique feasible solution on T is the entire tree. We have

$$\mathrm{OPT}(T) \geqslant \sum_j 2^{j-1}|\{C \in \mathcal{C}_j(T)|. \tag{6.1}$$

We charge the augmentation cost of v to the unique level-j cut containing it. Fix a level-j cut $C \in \mathcal{C}_j(T)$. It receives a charge of $2^{j+1}|Z_j \cap C|$. On the other hand, it contributes 2^{j-1} to the RHS of Equation (6.1). So to prove that $\mathrm{ALG}(\sigma) \leqslant 4\,\mathrm{OPT}(T)$, it suffices to show that $|Z_j \cap C| \leqslant 1$.

Suppose, towards a contradiction, that $|Z_j \cap C| > 1$. Let X_i be the group associated with C. Consider two distinct terminals $u, v \in Z_j \cap C$. By property (4a) of Definition 6.3, there are two cases: either both $u, v \in X_i$ or at least one of them belongs to $X_{i'}$ for $i \prec_\sigma i'$. Consider the first case and suppose u was added to Q before v. By property (4c), there exists a path P between u and v consisting of terminals of X_i whose edges are of length less than 2^j each. This implies that when the algorithm connects v to the root, it does so via an edge of length less than 2^j and so $a_v < 2^j$. But this contradicts the fact that $\text{class}(v) = j$, so this case cannot happen.

Now we turn to the second case, in which either u or v belongs to $X_{i'}$ for $i \prec_\sigma i'$. Suppose it is u. Right before u was added to Q, we have $X_i \subseteq Q$ since X_i appeared before $X_{i'}$ in the ordering σ. So $X_i \subseteq Q$. Since $u \in C$, we have that $d(u, X_i) < 2^j$ by property (4b) of Definition 6.3. Therefore, the algorithm could have connected u to a vertex of X_i and so $a_u < 2^j$. But this contradicts the fact that $\text{class}(u) = j$, so this case cannot happen either. Since neither case can happen, we must have $|Z_j \cap C| = 1$, as desired. This completes the proof of the lemma. □

6.2.2 Existence of good embeddings

Next, we prove the existence of a good distribution $\mathcal{D}_\sigma$ of $(\mathcal{X}, \sigma)$-embeddings for each arrival order σ.

Lemma 6.5. *For each σ, there exists a distribution $\mathcal{D}_\sigma$ such that for all $u, v \in X$, we have*

$$\mathbb{E}_\sigma[\mathbb{E}_{T \sim \mathcal{D}_\sigma}[\text{OPT}(T)]] \leqslant O(\log g)\,\text{OPT}\,.$$

We will use a randomized algorithm (Algorithm 6.2) that takes as input an ordering σ and produces a random $(\mathcal{X}, \sigma)$-embedding. Then we show that the distribution $\mathcal{D}_\sigma$ produced by the algorithm satisfies the desired properties.

The high level idea is similar to that of finding low-distortion HST embeddings in [Bar96, Bar98, FRT04]. A HST embedding of (X, d) is defined by its hierarchical decomposition, so it suffices to define a hierarchical decomposition of (X, d). The decomposition can be defined recursively as follows. The highest level partition of the decomposition is just $\{X\}$. The level-j partition $\mathcal{C}_j$ is obtained from the level-$(j+1)$ partition $\mathcal{C}_{j+1}$ in the following manner: start with $\mathcal{C}_j = \emptyset$; for each cut $C \in \mathcal{C}_{j+1}$, use a cutting scheme to partition C into smaller cuts and add them to $\mathcal{C}_j$. The cutting scheme chooses a random radius $r_j \in [2^{j-1}, 2^j)$, goes through the terminals X and iteratively cuts an r-radius ball around each terminal. We use a generalization of this cutting scheme—instead of iterating over terminals, we iterate over groups and (essentially) cut terminals that are within a random radius r of the group.

Our cutting scheme uses clusterings. Given a group X_i and a radius r, the *clustering* $\mathcal{S}_i(r)$ is a partitioning of X_i into clusters defined as follows. Let H_i be a complete graph with vertex set X_i and edge lengths $d(u, v)$ for $u, v \in X_i$, and $H_i(r)$ be a subgraph obtained by removing all edges of length at least $2r$ from H_i. The clustering $\mathcal{S}_i(r)$ is defined to be the set of connected components of $H_i(r)$.

For a subset $S \subseteq X$ and terminal $v \in X$, we define $d(S, v) = \min_{u \in S} d(u, v)$. Furthermore, for a non-negative real r, we define $B(S, r) = \{v \in X : d(S, v) < r\}$.

Lemma 6.6. *The clustering $\mathcal{S}_i(r)$ satisfies the following properties. For every $u, v \in X_i$,*

1. *if u and v belong to different clusters, then $d(u, v) \geqslant 2r$;*
2. *if u and v belong to the same cluster, then there exists a (u, v) path whose vertices belong to X_i and every edge along the path has length at most $2r$.*

As a consequence of the first property, we have $B(S, r) \cap B(S', r) = \emptyset$ for any two distinct clusters $S, S' \in \mathcal{S}_i(r)$.

We are now ready to define our cutting scheme formally. For level j, we choose a random radius $r_j \in [2^{j-1}, 2^j)$; iterate over the groups according to the arrival order σ and the clusters $S \in \mathcal{S}_i(r)$ of each group, and cut terminals in $B(S, r)$. Our procedure for finding random $(\mathcal{X}, \sigma)$-embeddings is formally defined in Algorithm 6.2.

Algorithm 6.2 Algorithm for Finding Random $(\mathcal{X}, \sigma)$-Embeddings

Input: Arrival order σ

$\mathcal{C}_{\lceil \log \Delta \rceil} \leftarrow \{X\}$
Create node of HST T corresponding to $\{X\}$
for $j = \lceil \log \Delta \rceil - 1$ down to 1 **do**
  Choose $r_j \in [2^{j-1}, 2^j)$ uniformly at random
  $\mathcal{C}_j \leftarrow \emptyset$
  for $C \in \mathcal{C}_{j+1}$ **do**
    $U \leftarrow C$
    for $i = 1$ to g **do**
      for $S \in \mathcal{S}(X_{\sigma(i)}, r_j)$ **do**
        Define the cut $C' = B(S, r_j) \cap U$
        if $C' \neq \emptyset$ **then**
          Add C' to $\mathcal{C}_j$ and remove it from U
          Add a node in T corresponding to C' and an edge e of length 2^j between this node and the node corresponding to C
          Associate group $X_{\sigma(i)}$ with e
        end if
      end for
    end for
  end for
end for

Let $\mathcal{D}_\sigma$ be the distribution of $(\mathcal{X}, \sigma)$-embeddings produced by Algorithm 6.2. We show that the expected distortion over uniformly random permutations σ and embeddings $T \sim \mathcal{D}_\sigma$ is at most $O(\log g)$.

Lemma 6.7. *For all* $u, v \in X$,

$$E[T(u, v)] \leqslant O(\log g) d(u, v),$$

where the expectation is over uniformly random arrival orders σ *and embeddings* $T \sim \mathcal{D}_\sigma$.

Proof. Fix $u, v \in X$. We have

$$T(u,v) = \sum_j 2^{j-1} \cdot |\{C \in \mathcal{C}_j : (u,v) \in \delta(C)\}| = \sum_j 2^j \cdot 1\{\exists C \in \mathcal{C}_j : (u,v) \in \delta(C)\}.$$

The first equality is because the unique (u,v)-path P in T consists of every edge e in T whose leaves C contain exactly one of u and v. The second equality is because T is a HST and X are its leaves, so for every level j of the tree, either P contains two edges of that level or none.

Let Y_{ij} be the event that $u \in \delta(B(S, r_j))$ but $v \notin \delta(B(S, r_j))$ for some $S \in \mathcal{S}_i(r_j)$, and Z_{ij} be the event that the first cut $C \in \mathcal{C}_j$ to contain u is associated with X_i. There exists a level-j cut that separates u and v if and only if $\sum_i 1\{Y_{ij} \wedge Z_{ij}\} = 1$, thus we have

$$T(u,v) \leqslant \sum_j 2^j \sum_i 1\{Y_{ij} \wedge Z_{ij}\}$$

and so

$$\mathbb{E}[T(u,v)] \leqslant \sum_j 2^j \sum_i \Pr[Z_{ij} \mid Y_{ij}] \cdot \Pr[Y_{ij}].$$

For each group, define $t_i \in X_i$ to be the terminal in X_i closest to u. Reindex the groups according to their distance from u, i.e. $d(t_i, u) \leqslant d(t_{i'}, u)$ for $i < i'$. Because $B(S, r_j) \cap B(S', r_j) = \emptyset$ for distinct clusters $S, S' \in \mathcal{S}_i(r_j)$, event Y_{ij} happens only if $u \in \delta(B(t_i, r_j))$, and event Z_{ij} happens only if X_i was the first in the arrival order σ among groups $X_{i'}$ with $u \in B(t_i, r_j)$. The event Y_{ij} implies that $u \in B(t_{i'}, r_j)$ for groups $i' < i$. Therefore, conditioned on Y_{ij}, the event Z_{ij} only happens if $i \prec_\sigma i'$ for all $i' < i$ which happens with probability $\frac{1}{i}$. Thus $\Pr[Z_{ij} \mid Y_{ij}] = \frac{1}{i}$ for all i and j,

and so

$$\mathrm{E}[T(u,v)] \leqslant \sum_i \frac{1}{i} \sum_j 2^j \Pr[Y_{ij}].$$

Event Y_{ij} happens if and only if $d(t_i,u) < r_j \leqslant d(t_i,v)$. Since r_j is uniformly distributed in the interval $[2^{j-1}, 2^j)$, we have

$$\begin{aligned}\sum_j 2^j \Pr[Y_{ij}] &= \sum_j 2^j \frac{|(d(t_i,u), d(t_i,v)] \cap [2^{j-1}, 2^j)|}{2^{j-1}} \\ &= 2\sum_j |[d(t_i,u), d(t_i,v)) \cap [2^{j-1}, 2^j)| \\ &= 2(d(t_i,v) - d(t_i,u)) \leqslant 2d(u,v),\end{aligned}$$

where the triangle inequality is used in the last step.

We conclude that

$$\mathrm{E}[T(u,v)] \leqslant \sum_i \frac{1}{i} \cdot 2d(u,v) \leqslant O(\log g) d(u,v),$$

as desired. □

We are now ready to prove Lemma 6.5.

Proof of Lemma 6.5. Let F^* be a minimum spanning tree of (X,d). This is a 2-approximate minimum Steiner tree of X in (V,d) (see, e.g. [Vaz01, Chapter 3]), so $c(F^*) \leqslant 2\,\mathrm{OPT}$. For an arrival order σ, and $(\mathcal{X},\sigma)$-embedding T, define the set of paths $\mathcal{P}(T) = \{P_{uv}(T) : (u,v) \in F^*\}$, where $P_{uv}(T)$ is the (u,v)-path in T. Since F^* spans X, which are the leaves of T, the paths $\mathcal{P}(T)$ covers all edges of T. Furthermore, the length of a path $P_{uv}(T)$ is exactly $T(u,v)$. So,

$$\mathrm{OPT}(T) = \sum_j 2^{j-1} |E_j(T)| \leqslant \sum_{(u,v)\in F^*} T(u,v).$$

In expectation over uniformly random arrival order σ and $(\mathcal{X}, \sigma)$-embedding $T \sim \mathcal{D}_\sigma$, we have

$$\begin{aligned} \mathbb{E}[\mathrm{OPT}(T)] &\leqslant \sum_{(u,v)\in F^*} \mathbb{E}[T(u,v)] \\ &\leqslant O(\log g) \sum_{(u,v)\in F^*} d(u,v) \\ &= O(\log g) c(F^*) \leqslant O(\log g)\,\mathrm{OPT}, \end{aligned}$$

where the last inequality uses the fact that $c(F^*) \leqslant 2\,\mathrm{OPT}$. □

Putting all of this together, Lemmas 6.4 and 6.5 imply that Algorithm 6.1 has an expected competitive ratio of $O(\log g)$ for the online Steiner tree with random group arrivals problem. This completes the proof of Theorem 6.2.

6.3 Lower Bound

We now turn to proving the lower bound, Theorem 6.1. We begin by showing that we only need to consider lazy algorithms. An algorithm ALG is *lazy* if for every step of the online process, its subgraph F is a minimal Steiner tree for the terminals seen so far i.e. no subgraph strictly contained in F is a Steiner tree for these terminals.

Lemma 6.8. *For any randomized algorithm* ALG*, there exists a randomized lazy algorithm* LAZY *whose expected competitive ratio is at most that of* ALG*'s.*

Proof. Given an algorithm ALG, the desired lazy algorithm LAZY is one that simulates ALG and maintains a subgraph of ALG's solution that spans the terminals seen so far. Let F_i be the subgraph maintained by ALG after it is presented the first

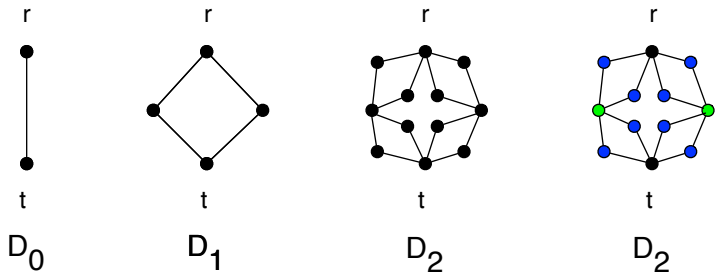

Figure 6.1: Diamond graphs D_0, D_1, and D_2. In the last diagram, the black vertices represent W_0, green represent W_1, and blue represent W_2

i groups. The algorithm LAZY maintains a subgraph F'_i that is a minimal subset of F_i that spans $X_1 \cup \cdots \cup X_i$. Therefore, for any set of groups $\mathcal{X}$ and arrival order σ, the cost of LAZY is at most that of ALG. This implies that the expected competitive ratio of LAZY is at most that of ALG's. □

Next, we construct a lower bound instance against lazy algorithms. The underlying graph of the instance is called a *diamond* graph. All edges are of unit length The family of rooted diamond graphs $D_N = (V_N, E_N)$ are defined recursively. The base case D_0 is simply an edge (r, t). We say that r is a *mate* with t, and vice versa. To go from D_{i-1} to D_i, we copy the vertex set of V_{i-1}, and for each edge $(u, v) \in E_{i-1}$, introduce two parallel (u, v)-paths consisting of two edges each. More specifically, we introduce two new vertices x, x' and four new edges (u, x), (u, x'), (v, x), (v, x'); we also define x and x' to be mates of each other. Every edge of E_i. Let $W_i = V_i \setminus V_{i-1}$, the new vertices introduced in going from D_{i-1} to D_i. Note that each vertex v has a unique mate, which we denote by $m(v)$.

Let us first construct a lower bound instance against deterministic lazy algorithms. We will then adapt it to prove a lower bound against randomized lazy algorithms.

Lemma 6.9. *Every deterministic lazy algorithm has competitive ratio* $\Omega(\min\{\log k, g\})$.

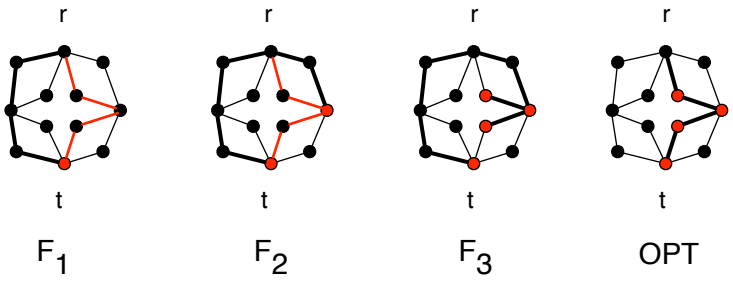

Figure 6.2: Example for $N = 3$. The thick edges represent F_1, F_2, F_3 and OPT, respectively. The red vertices represent terminals. In the first two diagrams, the red path represents P_2 and P_3, respectively.

Proof. We will show that for any lazy algorithm LAZY and any positive integer g, there exists a sequence of g groups X_i on the g-th diamond graph D_g with $k = |X_1 \cup \cdots \cup X_g| = 2^g + 1$ terminals, such that the competitive ratio of LAZY is at least $\Omega(g)$. Note that $g = O(\log k)$.

Fix a lazy algorithm LAZY and a positive integer g. Recall that for deterministic algorithms, the adversary can construct the request sequence adaptively and choose the next request of the sequence based on the algorithm's decisions so far. The instance is constructed such that $X = X_1 \cup \cdots \cup X_g$ lies along a (r, t)-path. The first group is $X_1 = \{r, t\}$. Let $X_{<i} = X_1 \cup \cdots \cup X_{i-1}$, and F_{i-1} is the minimal Steiner tree of $X_{<i}$ bought by LAZY. Among the (r, t)-paths that contain $X_{<i}$, we claim that there exists a path P such that $V(P) \cap W_i \cap V(F_{i-1}) = \emptyset$, where $V(H)$ denotes the set of vertices in subgraph H. In other words, there exists an (r, t)-path containing $X_{<i}$ such that every vertex of W_i on the path is not in F_{i-1}. To see why this claim is true, let Q be some (r, t)-path containing $X_{<i}$; then for every vertex $w \in V(Q) \cap W_i$, either $w \notin V(F_{i-1})$ or its mate $m(w) \notin V(F_{i-1})$, since F_{i-1} is a minimal Steiner tree for $X_{<i}$. The i-th group is $X_i = (V(P) \cap W_i) \cup \{r\}$. This defines the g groups of the request sequence. We now analyze the cost of LAZY.

Since $V(P) \cap W_i \cap V(F_{i-1}) = \emptyset$, LAZY has to incur a cost of

$$|V(P) \cap W_i| \cdot d(w, X_{<i}) = 2^{i-1} \cdot 2^{g-i} = 2^{g-1}$$

to connect group X_i. This is because there are 2^{i-1} vertices of W_i along any (r, t)-path, and for any $w \in W_i$, we have $d(w, X_{<i}) \geqslant 2^{g-i}$. Therefore, the total cost of LAZY over all g requests is $g \cdot 2^{g-2}$. On the other hand, the terminals X lie on a (r, t)-path, which is of length 2^{g-1}. Thus, the competitive ratio of LAZY is $\Omega(g)$, as desired. □

Finally, we prove a lower bound on the expected competitive ratio of randomized lazy algorithms.

Lemma 6.10. *The expected competitive ratio of any randomized lazy algorithm is at least* $\Omega(\min\{\log k, g\})$.

Proof. Yao's minimax principle [Yao77] states that the expected cost of any randomized algorithm on an input that is worst-case for that algorithm is at least, over any distribution of the input, the expected cost of a deterministic algorithm that is tailored for that distribution. In particular,

$$\min_{\text{rand. alg. ALG}} \max_{(\mathcal{I},\sigma)} \mathbb{E}[\text{ALG}(\mathcal{I}, \sigma)] \geqslant \max_{\text{input dist. } \mathcal{D}} \min_{\text{det. alg. ALG}} \mathbb{E}_{(\mathcal{I},\sigma)\sim\mathcal{D}}[\text{ALG}(\mathcal{I}, \sigma)].$$

Recall that an online instance consists of a request set $\mathcal{I}$ and an arrival order σ. Note that the expectation on the LHS is over the random coin flips of the algorithm while the expectation on the RHS is over the input distribution. Thus, we only need to construct a distribution of online instances such that, in expectation, every deterministic lazy algorithm has a competitive ratio at least $\Omega(\min\{\log k, g\})$.

Consider the following random input. The underlying graph is the g-th diamond graph D_g. We will ensure that the groups lie along some (r,t)-path and that X_i consists of the root and a subset of W_i. The first group is $X_1 = \{r,t\}$. We construct the i-th group X_i as follows. Write the vertices of X_{i-1} as $v_1, \ldots, v_{2^{i-1}}$ where v_j is the j-th closest to r. Start with $X_i = \emptyset$. By the construction of the diamond graph, for each $1 \leqslant j < 2^{i-1}$, there are exactly two vertices $y_j^i, z_j^i \in W_i$ that lie on a (v_j, v_{j+1})-path and they are mates of each other; choose one of them at random to add to X_i. Finally add r and t to X_i.

Fix a deterministic lazy algorithm LAZY. Let F_{i-1} be its minimal Steiner tree over $X_{<i} = X_1 \cup \cdots \cup X_{i-1}$. The algorithm incurs a cost of

$$|V(P) \cap W_i \setminus F_{i-1}| \cdot d(w, X_{<i}) = |V(P) \cap W_i \setminus F_{i-1}| \cdot 2^{g-2}$$

to connect group X_i.

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^{g} |V(P) \cap W_i \setminus F_{i-1}| \cdot 2^{g-i}\right] \\
&= \sum_{i=1}^{g} 2^{g-i} \sum_{Y_1,\ldots,Y_{i-1} \subseteq V} \mathbb{E}\left[|V(P) \cap W_i \setminus F_{i-1}| \,\middle|\, \bigwedge_{j=1}^{i-1} X_i = Y_i\right] \cdot \Pr\left[\bigwedge_{j=1}^{i-1} X_i = Y_i\right] \\
&= \sum_{i=1}^{g} 2^{g-i} \cdot \frac{|V(P) \cap W_i|}{2} \\
&= \sum_{i=1}^{g} 2^{g-i} \cdot 2^{i-2} = g \cdot 2^{g-3},
\end{aligned}$$

where the second equality follows from the fact that F_{i-1} is a minimal Steiner tree over $X_{<i}$, and $y_j^i, z_j^i \in W_i$ are mates so they cannot be both in F_{i-1}; thus, the probability that, in the construction of X_i, we chose the one that is in F_{i-1} is $1/2$.

Since the terminals X lie on a (r,t)-path, which is of length 2^{g-1}, the competitive

ratio of LAZY is $\Omega(g)$, as claimed. □

Combining Lemmas 6.8 and 6.10 gives us Theorem 6.1.

6.4 Notes

This chapter is based on work by the author.

Bibliography

[AA97] Baruch Awerbuch and Yossi Azar. Buy-at-bulk network design. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 542–547, Oct 1997.

[AAB04] Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized Steiner problem. *Theoretical Computer Science*, 324(2-3), September 2004.

[ADK+08] Elliot Anshelevich, Anirban Dasgupta, Jon Kleinberg, Éva Tardos, Tom Wexler, and Tim Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008.

[AKR95] Ajit Agrawal, Philip Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, June 1995.

[Bar96] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 184–193. IEEE Comput. Soc. Press, 1996.

[Bar98] Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 161–168, New York, NY, USA, 1998. ACM.

[BC97] Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems (extended abstract). In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 344–353, New York, NY, USA, 1997. ACM.

[BEY05] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 2005. Reissue edition (first published 1998).

[BGRS13] Jarosław Byrka, Fabrizio Grandoni, Thomas Rothvoss, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, February 2013.

[CCLE+07] C. Chekuri, J. Chuzhoy, L. Lewin-Eytan, J. Naor, and A. Orda. Non-cooperative multicast and facility location games. *Selected Areas in Communications, IEEE Journal on*, 25(6):1193–1206, August 2007.

[CKM+08] Moses Charikar, Howard Karloff, Claire Mathieu, Joseph (Seffi) Naor, and Michael Saks. Online multicast with egalitarian cost sharing. In *Proceedings of the 20th Annual Symposium on Parallelism in Algorithms and Architectures*, SPAA '08, pages 70–76, New York, NY, USA, 2008. ACM.

[DJK13] Nikhil R Devanur, Kamal Jain, and Robert D Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 101–107. SIAM, 2013.

[EGRS08] Friedrich Eisenbrand, Fabrizio Grandoni, Thomas Rothvoß, and Guido Schäfer. Approximating connected facility location problems via random facility sampling and core detouring. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1174–1183. Society for Industrial and Applied Mathematics, 2008.

[FKL+91] Amos Fiat, Richard M Karp, Michael Luby, Lyle A McGeoch, Daniel D Sleator, and Neal E Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

[Fot07] Dimitris Fotakis. A primal-dual algorithm for online non-uniform facility location. *Journal of Discrete Algorithms*, 5(1):141–148, 2007.

[Fot08] Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

[Fot11] Dimitris Fotakis. Online and incremental algorithms for facility location. *SIGACT News*, 42(1), March 2011.

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, November 2004.

[GB93] Michel X Goemans and Dimitris J Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Mathematical Programming*, 60(1-3):145–166, 1993.

[GGLS08] Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 942–951, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[GK11] Anupam Gupta and Jochen Könemann. Approximation algorithms for network design: A survey. *Surveys in Operations Research and Management Science*, 16(1):3 – 20, 2011.

[GKPR07] Anupam Gupta, Amit Kumar, Martin Pál, and Tim Roughgarden. Approximation via cost sharing: Simpler and better approximation algorithms for network design. *Journal of the ACM (JACM)*, 54(3):11, 2007.

[GKR12] Anupam Gupta, Ravishankar Krishnaswamy, and R. Ravi. Online and stochastic survivable network design. *SIAM Journal on Computing*, 41(6):1649–1672, 2012.

[GR10] Fabrizio Grandoni and Thomas Rothvoß. Network design via core detouring for problems without a core. In *Automata, Languages and Programming*, pages 490–502. Springer, 2010.

[GW95] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[GW97] Michel X Goemans and David P Williamson. The primal-dual method for approximation algorithms and its application to network design problems. *Approximation algorithms for NP-hard problems*, pages 144–191, 1997.

[IW91] Makoto Imase and Bernard M. Waxman. Dynamic steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991.

[Jai01] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[KMMO94] Anna R Karlin, Mark S Manasse, Lyle A McGeoch, and Susan Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

[Mey01] Adam Meyerson. Online facility location. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 426–431. IEEE Comput. Soc, 2001.

[MS96] Dov Monderer and Lloyd S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124 – 143, 1996.

[QW11] Jiawei Qian and David P. Williamson. An $O(\log n)$-competitive algorithm for online constrained forest problems. In Luca Aceto, Monika Henzinger, and Jiří Sgall, editors, *Automata, Languages and Programming*, volume 6755 of *Lecture Notes in Computer Science*, pages 37–48. Springer Berlin Heidelberg, 2011.

[Ros73] Robert W Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[SFWL14a] Mário César San Felice, David P. Williamson, and Orlando Lee. The online connected facility location problem. In Alberto Pardo and Alfredo Viola, editors, *LATIN 2014: Theoretical Informatics*, volume 8392 of *Lecture Notes in Computer Science*, pages 574–585. Springer Berlin Heidelberg, 2014.

[SFWL14b] Mário César San Felice, David P. Williamson, and Orlando Lee. A randomized $o(\log n)$-competitive algorithm for the online connected facility location problem. In Submission, 2014.

[SK04] Chaitanya Swamy and Amit Kumar. Primal–Dual Algorithms for Connected Facility Location Problems. *Algorithmica*, 40(4), September 2004.

[ST84] Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update rules. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC)*, pages 488–492. ACM, 1984.

[Umb15] Seeun Umboh. Online network design algorithms via hierarchical decompositions. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1373–1387. Society for Industrial and Applied Mathematics, 2015.

[Vaz01] Vijay V. Vazirani. *Approximation algorithms*. Springer-Verlag Berlin Heidelberg, 2001.

[Wil02] David P. Williamson. The primal-dual method for approximation algorithms. *Mathematical Programming*, 91(3):447–478, 2002.

[Yao77] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '77, pages 222–227, Oct 1977.