ABSTRACT

Title of dissertation: PRIMAL-DUAL TECHNIQUES FOR ONLINE ALGORITHMS AND MECHANISMS

Vahid Liaghat, Doctor of Philosophy, 2015

Dissertation directed by: Professor MohammadTaghi Hajiaghayi
Department of Computer Science

An offline algorithm is one that knows the entire input in advance. An online algorithm, however, processes its input in a serial fashion. In contrast to offline algorithms, an online algorithm works in a local fashion and has to make irrevocable decisions without having the entire input. Online algorithms are often not optimal since their irrevocable decisions may turn out to be inefficient after receiving the rest of the input. For a given online problem, the goal is to design algorithms which are competitive against the offline optimal solutions.

In a classical offline scenario, it is often common to see a *dual analysis* of problems that can be formulated as a linear or convex program. Primal-dual and dual-fitting techniques have been successfully applied to many such problems. Unfortunately, the usual tricks come short in an online setting since an online algorithm should make decisions

without knowing even the whole program. In this thesis, we study the competitive analysis of fundamental problems in the literature such as different variants of online matching and online Steiner connectivity, via *online* dual techniques.

Although there are many generic tools for solving an optimization problem in the offline paradigm, in comparison, much less is known for tackling online problems. The main focus of this work is to design generic techniques for solving *integral* linear optimization problems where the solution space is restricted via a set of linear constraints. A general family of these problems are online packing/covering problems. Our work shows that for several seemingly unrelated problems, primal-dual techniques can be successfully applied as a unifying approach for analyzing these problems. We believe this leads to generic algorithmic frameworks for solving online problems.

In the first part of the thesis, we show the effectiveness of our techniques in the stochastic settings and their applications in Bayesian mechanism design. In particular, we introduce new techniques for solving a fundamental linear optimization problem, namely, *the stochastic generalized assignment problem* (GAP). This packing problem generalizes various problems such as online matching, ad allocation, bin packing, etc. We furthermore show applications of such results in the mechanism design by introducing Prophet Secretary, a novel Bayesian model for online auctions.

In the second part of the thesis, we focus on the covering problems. We develop the framework of *Disk Painting* for a general class of *network design problems* that can be characterized by proper functions. This class generalizes the node-weighted and edge-

weighted variants of several well-known Steiner connectivity problems. We furthermore design a generic technique for solving the prize-collecting variants of these problems when there exists a dual analysis for the non-prize-collecting counterparts. Hence, we solve the online prize-collecting variants of several network design problems for the first time.

Finally we focus on designing techniques for online problems with mixed packing/covering constraints. We initiate the study of *degree-bounded* graph optimization problems in the online setting by designing an online algorithm with a tight competitive ratio for *the degree-bounded Steiner forest problem*. We hope these techniques establishes a starting point for the analysis of the important class of online degree-bounded optimization on graphs.

PRIMAL-DUAL TECHNIQUES FOR ONLINE ALGORITHMS AND MECHANISMS

by

Vahid Liaghat

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:
Professor MohammadTaghi Hajiaghayi, Chair/Advisor
Professor Samir Khuller
Professor Aravind Srinivasan
Professor Hector Corrada Bravo
Professor Subramanian Raghavan, Robert H. Smith School of Business and Institute for Systems Research

Dedication

To my lovely wife Ferreshteh for bringing life to my life;

And to my parents Soghra and Masood for all their love and support.

Acknowledgments

First and foremost I'd like to thank my advisor, Prof. MohammadTaghi Hajiaghayi for his excellent scientific guidance over the last 5 years. He has been a wonderful source of research problems and ideas. Looking back, I can see his imprint on every aspect of my scientific development. He has always made himself available for help and advice and there has never been an occasion when I've knocked on his door and he hasn't given me time. Thanks Mohammad for EVERYthing, and I look forward to our continued collaboration.

It is doubtless that I owe the greatest part of my academic achievements to each and every one of my teachers and mentors. Mohammad gave me an amount of support and guidance unheard-of. Moreover, in grad school I was particularly fortunate to seek advise from Aravind Srinivasan and Samir Khuller at UMD and from my internship mentors Howard Karloff, Matthew Andrews, Qiong Wang, Mohit Singh, Debmalya Panigrahi, and Harry Raecke. Their generosity with their time and knowledge helped me to set sail in a sea of wonderful ideas that will greatly support my future careers. Many thanks to Samir Khuller, Aravind Srinivasan, Hector Corrada Bravo, and Subramanian Raghavan for serving on my dissertation committee and providing me with informative comments.

My interest in theoretical computer science dates back to summer camps of Iran National Olympiad in Informatics (INOI). I owe the root of many of my skills to my INOI mentors, who quickly became my dear friends. Thank you Amin Sayedi, Arash Asadpour, Shayan Oveis Gharan, Kian Mirjalali, MohammadHossein Bateni, Morteza Zadimoghadam, Aidin NasiriShargh, Hamid Mahini, Ali Sharifi, Mohammad Norouzi, Iman Hajirasouliha, Farshid Jafarpour, and many others who greatly helped me in that time. I would like to specially thank Prof. Mohammad Ghodsi, the head of INOI committee and also my BSc advisor for his support and commitment.

I believe that grad school is indeed one of the most interesting pages of one's life. I would like to thank all my friends Afshin Nikzad, Ahmad Khajehnejad, Ali Shafahi, Mohammad Alizadeh, Mercedeh Tariverdi, Rajesh Chitnis, MohammadReza Khani, Ioana Bercea, Hamid Mahini, Faezeh Dorri, Hossein Esfandiari, Milad Gholami, Melika Abolhassani, Morteza Monemizadeh, Kiana Roshanzamir, Saeed Seddighin, Sina Dehghani, Soheil Ehsani, and many others for filling this page of my life with the most wonderful memories. Thank you all!

Leaving the most important for the last, I would like to thank my parents, my wife, and my siblings, whom I can never even begin to thank enough for their love and support. I would not be wrong if I say everything good I have, I owe it to the most wonderful parents in the world, Soghra and Masood. Among the wonders my parents have shown me, Honesty and Mathematics, are two that will surely continue to fascinate me all my life. Getting to know my wife, was the most amazing thing that happened to me. Thank you Ferreshteh for showing me the world in a light that I never even knew it exists. Thank you for being the peace of my heart, for changing my life forever...

I also gratefully acknowledge the support of the following grants during my PhD studies: NSF CAREER award 1053605, NSF grant CCF-1161626, NSF CAREER award 0844872, and Google US/Canada PhD Fellowship.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Overview

1.1 Introduction

In the real world algorithmic problems, it often happens that at each step we only have the knowledge of a part of the input. For example consider a physical or virtual server which can provide a service to a set of clients while the goal is to optimize a global objective. The clients send the requests one by one and the server does not have the time to wait for the rest of requests, thus the server faces an online problem. Indeed in recent decades applications of online problems have been expanded as rapid as the use of World Wide Web. In this thesis, we design competitive algorithms for fundamental problems in the literature such as online Steiner connectivity problems and online generalized assignment problem.

It is well known that in offline settings, heuristic algorithms often do much better in practice than sophisticated algorithms with theoretical guarantees. However, for

many heuristic approaches such as genetic algorithms, no suitable generalization to online settings is known. Thus even in practice, the most efficient algorithms are often those guaranteed by a competitive analysis. On the other hand, the unpredictable nature of online problems leads to strong lower bounds for the worst-case analysis. Indeed the competitive ratio obtained from the worst case analysis is simply not acceptable in many applications. The real world inputs contain a lot of structure and are hardly adversarial. In an online setting, this can be modeled by assuming stochastic information about the arriving input. In recent years the effectiveness of stochastic algorithms has been shown in a few problems, a major one being the ad allocation problems in search engines. Despite their applications in the industry, for many fundamental problems such as k-server and Steiner connectivity, the theoretical guarantees for stochastic settings are still not better than that of the worst case analysis. This has been the main motivation for me to consider stochastic settings as well in various problems.

Advertising on search engines is a famous example of the category of online stochastic optimization problems. Motivated by this framework, we consider different models for various online problems: the adversarial model, i.i.d models, and the prophet inequality model. In the adversarial model the input is fed to the algorithm by an adversary while in the two latter models the input is coming from independent distributions. In i.i.d models the distributions are identical though in the prophet inequality model they may change over time. We consider different models for the aforementioned problems and develop dual-based techniques for designing competitive algorithms.

One major obstacle in applying (theoretical) algorithms on the real world problems is that many algorithms are not flexible: the analysis may fail by adding a simple extra detail or constraint. Thus finding solutions for different problems using the same framework is much more desirable than a set of (even stronger) independent solutions. For example the problems mentioned at the beginning are drastically different. An ambitious question is "Can we design a framework which leads to a uniform solution for all of these problems"? In this thesis, we take one step towards this goal under stochastic assumptions.

Packing/Covering Linear Programming.

Linear programming (LP) is the well-studied problem of minimizing or maximizing a linear objective function over a feasible space described by a set of linear inequalities. It is well-known that any LP can be formulated as follows.

$$\begin{aligned} &\text{Min.} && \sum_{i=1}^{n} c_i \mathbf{x}_i && \text{(P)} \\ &\forall j \in [m] && \sum_{i=1}^{n} a_{ij} \mathbf{x}_i \geq 1 \\ & && \mathbf{x}_i \geq 0 \end{aligned}$$

where $c_i, a_{ij} \in \mathbb{R}$ for every $i \in [n]$[1] and $j \in [m]$. We refer to P as the *primal* problem. For every primal LP one can derive a *dual* problem formulated as the following maximization problem.

[1]For an integer α, let $[\alpha]$ denote the set of integers $1, \ldots, \alpha$.

$$\begin{aligned} &\text{Max.} && \sum_{j=1}^{m} \mathbf{y}_j && \text{(D)} \\ &\forall i \in [n] && \sum_{j=1}^{m} a_{ij}\mathbf{y}_j \leq c_i \\ & && \mathbf{y}_j \geq 0 \end{aligned}$$

A well-studied class of LP problems are those in which all coefficients are non-negative $c_i, b_j, a_{ij} \geq 0$. In this case, intuitively, a feasible solution of the primal problem covers a requirement for every $j \in [m]$. On the other hand, a feasible solution of the dual problem does not violate the limit c_i for every $i \in [n]$. Therefore the primal problem is called a *covering problem* while the dual problem is called a *packing problem*.

In the online variant of a covering problem, the objective function is known in advance. However, the constrains (with their corresponding coefficients) arrive online. After receiving a new constraint, the online algorithm may only increase the values of variables to satisfy the new constraint. In an online packing problem, the linear objective function and all the linear constraints are known, but only partially. The packing limits (i.e., c_i's) are known in advance while the variables with their corresponding coefficients arrive online. The online algorithm has one shot to set the value of a variable at the time of its arrival.

Among the problems which fall in this general framework are the generalized assignment problem, the problem of allocating ad-auctions, weighted caching problem, generalized catching, the set cover problem, Steiner connectivity problems, routing, and load

balancing. Indeed in a series of papers, primal-dual approaches have been developed for solving online packing/covering LPs fractionally (e.g. see a survey in [BN07]). In these papers, the primal-dual method is extended to the setting of online algorithms, and its applications to a wide variety of problems are shown. As pointed out by Buchbinder and Naor in their survey, the primal-dual method has many advantages over the existing methods. Primal-dual algorithms are often clean and efficient, and the competitive ratio analysis is often direct without a problem-specific potential function. Unfortunately, it is (provably) impossible to solve the general problem when restricted to integral solutions.

In this thesis, our work shows that by assuming stochastic information, achieving online competitive integral solutions is still possible. Indeed when a framework captures various different problems, it embeds barriers of different flavors. Thus to push our understanding of the problem, an important step is to analyze the special cases of the general framework which are seemingly different, and try to come up with uniform solutions for those problems. We describe the techniques we have designed for each set of problems separately. The first set can be considered as maximization problems related to packing LPs, while the second set can be considered minimization problems related to covering LPs. In the next section, we begin by showing the application of primal-dual methods in solving *fractional* variants of the problems.

1.2 Background on Packing/Covering Linear Programs

In this section we consider the general framework of packing-covering linear formulations. Buchbinder and Naor [BN07] give an extensive survey of the primal-dual approaches for solving related problems. To see a taste of how a primal-dual framework helps in tackling online variants of the problem, we briefly present the two algorithms solving fractional covering and packing linear programs online. Indeed even online factional solutions have several applications in tackling real-world problems, see for example [LLZ+13].

1.2.1 The Online Fractional Packing Problem

Buchbinder and Naor [BN09b] give an online scheme for computing a near-optimal fractional solution for an online packing problem. For a given value $B > 0$, the scheme returns a solution within a factor of B of optimal solution while violating the packing constraint by a factor of roughly $O(\frac{\log m}{B})$ which also depends on other input factors (to be defined precisely). Recall that in a packing problem, given a set of m variables y_j's, a linear objective function $\sum_{j=1}^{m} \mathbf{y}_j$ is to be maximized where the feasible space is defined by a set of n constraints in the form $\sum_{j=1}^{m} a_{ij}\mathbf{y}_j \leq c_i$ for every $i \in [n]$ (see D). In the online variant of the problem, at a time j, the coefficients of $\mathbf{y}_j$ in the packing constraints are revealed. Next, the online algorithm outputs a (fractional) value for y_j which cannot be changed in the future.

Algorithm.

The algorithm simultaneously maintains a primal (covering) and a dual (packing) solution as defined in P and D. The primal variables are initializes to zero. At round j, the coefficients a_{ij}'s are revealed for every $i \in [n]$. Thus in the primal instance, the constraint $\sum_{i=1}^{n} a_{ij}\mathbf{x}(i) \geq 1$ is introduced. The algorithm increases the value of the new variable $\mathbf{y}(j)$ and the primal variables $\mathbf{x}_i$'s until the new primal constraint is satisfied. The primal variables are augmented via a monotonically increasing function of $\mathbf{y}(j)$. See Algorithm 1 for the precise function.

Algorithm 1 Online Fractional Packing Problem

Input: The packing limit c_i for every $i \in [n]$ and an online stream of the coefficients.
Output: A feasible primal vector $\mathbf{x}$ and a dual vector $\mathbf{y}$.
Offline Process:

1: For every $i \in [n]$ initialize $\mathbf{x}(i) = 0$.

Online Scheme; assuming for a $j \in [m]$ the coefficients a_{ij} are arrived for every $i \in [n]$:

1: Set $\mathbf{y}(j) = 0$.
2: **while** $\sum_{i=1}^{n} a_{ij}\mathbf{x}(i) < 1$ **do**
3: Increase $\mathbf{y}(j)$ continuously [2].
4: For every $i \in [n]$, increase $\mathbf{x}(i)$ to the following $\max\left\{\mathbf{x}(i), \frac{\exp\left(\frac{B}{2c_i}\sum_{k=1}^{j} a_{ik}\mathbf{y}(k)\right)-1}{n \cdot \max_{k=1}^{j}\{a_{ik}\}}\right\}$.

Buchbinder and Naor [BN09b] show that the objective value of the output of the algorithm is within a factor B of that of the optimum. More formally,

Theorem 1.1 (Theorem 3.1 of [BN09b]). *Algorithm 1 is a B-competitive algorithm for the online packing problem where the i^{th} constraint is violated by at most*

$$\frac{2\log(1+n\cdot\frac{a_i(max)}{a_i(min)})}{B}$$

where $a_i(max) = \max_{j=1}^{m}\{a_{ij}\}$ and $a_i(min) = \min_{j=1}^{m}\{a_{ij}|a_{ij} > 0\}$.

Observe that Theorem 1.1 gives a $O(\log(n))$-competitive fractional algorithm when the coefficients are either zero or one (or more generally when the ratio between the maximum and minimum non-zero coefficient is constant).

1.2.2 The Online Fractional Covering Problem

In the online covering problem, the objective function of the primal problem is known in advance. At iteration j, the j^{th} constraint is revealed. The online algorithm can only increase the value of primal variables. Buchbinder and Naor [BN09b] give an online algorithm with a logarithmic competitive ratio which is tight up to a constant factor.

Theorem 1.2 (Theorem 4.1 of [BN09b]). *For any given $B > 0$, there exists an $O\left(\frac{\log(n)}{B}\right)$-competitive fractional algorithm (see Al-*

gorithm 2) for the online covering problem where a primal constraint may be violated by a factor of at most B.

Algorithm.

Similar to the previous section, the algorithm simultaneously maintains a primal (covering) and a dual (packing) solution. First, we assume that the value of the optimal solution *OPT* is known within a two factor. This is without loss of generality: one can guess *OPT* using a standard doubling technique and lose at most a constant factor in the competitive ratio. The algorithm initializes the primal variables to zero. Similar to Algorithm 1, we continuously increase the value of the dual variable $\mathbf{y}(j)$ corresponding to the new constraint while increasing the primal variables correspondingly. See Algorithm 2 for the formal description.

Algorithm 2 Online Fractional Covering Problem

Input: The objective function (i.e., c_i's) and an online stream of the primal constraints.
Output: A primal vector $\mathbf{x}$ and a feasible dual vector $\mathbf{y}$.
Offline Process:

1: For every $i \in [n]$ initialize $\mathbf{x}(i) = 0$.

Online Scheme; assuming for a $j \in [m]$ the j^{th} constraint is arrived:

1: Set $\mathbf{y}(j) = 0$.
2: **while** $\sum_{i=1}^{n} a_{ij}\mathbf{x}(i) < \frac{1}{B}$ **do**
3: Increase $\mathbf{y}(j)$ continuously.
4: For every $i \in [n]$, set $x(i) = \frac{OPT}{2n \cdot c_i} \exp\left(\frac{\log(2n)}{c_i} \sum_{k=1}^{j} a_{ik}\mathbf{y}(k)\right)$.

1.3 Online Maximization Problems

We first give a brief description of our techniques for the following family of packing problems. Next, we will introduce Prophet Secretary as a new Bayesian auction setting and we demonstrate the applications of online optimization techniques in developing near-optimal mechanisms.

1.3.1 From Matching to Online Packing

As noted by Lovasz and Plummer in their classic book [LP86], "[Matching Theory] is a central part of graph theory, not only because of its applications, but also because it is the source of important ideas developed during the rapid growth of combinatorics during the last several decades". Hence, we begin by considering online matching. Online bipartite matching problem first was introduced by Karp, Vazirani, and Vazirani [KVV90]. They proved that a simple randomized online algorithm achieves a (1-1/e)-competitive ratio and this factor is the best possible in the adversarial model. Online bipartite matching has been considered under stochastic assumptions in [GM08, FMMM09, MOS11], where most recent of them is the work of Manshadi et al. [MOS11] that presents an online algorithm with a competitive ratio of 0.702. They also show that no online algorithm can achieve a competitive ratio better than 0.823.

An important generalization of online matching is the Google AdWord problem. Online advertising alongside search results is a multi-billion dollar business [Lah06] and

is a major source of revenue for search engines like Google, Yahoo and Bing. A related ad allocation problem is the AdWords assignment problem [MSVV07] that was motivated by sponsored search auctions. When modeled as an online bipartite assignment problem, each edge has a weight, and there is a budget on each advertiser representing the upper bound on the total weight of edges that might be assigned to it. In the online setting, it is typical to assume that edge weights (i.e., bids) are much smaller than the budgets. Devanur and Hayes [DH09] provide an algorithm with competitive ratio (1-1/e) in the stochastic setting where the sequence of arrivals is a random permutation. In most of the previous work it is assumed that the arriving requests are drawn from identical distributions. However, by studying the data obtained from search engines of Yellow Pages of AT&T, we showed this assumption is not quite true. Indeed the distribution of search terms observed by a search engine is highly dependent on the time and date. Thus in our paper [AHL+11], we used a prophet inequality model for the first time in an online matching setting which generalizes several previous advertising models such as online secretary problem and Google adword. In contrast to i.i.d. model, this allows us to model the change in the distribution of items throughout the time. We call this setting the Prophet-Inequality Matching because of the possibility of having a different distribution for each time. In our paper [AHL12] we generalize the classic prophet inequality by presenting an algorithm with asymptotically optimum approximation ratio. We have recently generalized our methods to the stochastic generalized assignment problem which provides a uniform solution for several fundamental optimization problems including the

online variants of multiple knapsack, bin packing, Google adword, and banner advertisement. The proposed algorithm initially computes an optimal solution for a linear program corresponding to a fractional expected instance. By using this solution as a guideline, we use a novel randomized technique for assigning arriving requests without violating the problem constraints. This potentially provides us with a tool for attacking other online problems. We believe our methods are robust enough to be adopted for more generalized constraints, hoping to give efficient algorithms applicable to real world problems.

1.3.2 Prophet Secretary

Optimal stopping theory is a powerful tool for analyzing scenarios such as online auctions in which we generally require optimizing an objective function over the space of stopping rules for an allocation process under uncertainty. Perhaps the most classic problems of stopping theory are the prophet inequality problem (first studied in TCS by Hajiaghayi, Kleinberg, and Sandholm [HKS07]) and the secretary problem (Hajiaghayi, Kleinberg and Parkes [HKP04]); both instances of packing problems when formulated as optimization problems. The classical prophet inequality states that by choosing the same threshold $OPT/2$ for every step, one can achieve the tight competitive ratio of 0.5. On the other hand, for the basic secretary problem, the optimal strategy is to ignore the first $\frac{n}{e}$ elements of the sequence while using the maximum of the first $\frac{n}{e}$ elements as the threshold for the rest of sequence. This strategy achieves the optimal competitive ratio of $1/e = 0.36$.

In this thesis, we introduce *prophet secretary*, a natural combination of the prophet inequality problem and the secretary problem. An example of motivations for our problem is as follows. Consider a seller that has an item to sell on the market to a set of arriving customers. The seller knows the types of customers that may be interested in the item and he has a price distribution for each type: the price offered by a customer of a type is anticipated to be drawn from the corresponding distribution. However, the customers arrive in a random order. Upon the arrival of a customer, the seller makes an irrevocable decision to whether sell the item at the offered price. We address the question of finding a strategy for selling the item at a high price.

We show that by using a single uniform threshold one cannot break the 0.5 barrier of the prophet inequality for the prophet secretary problem. However, we show that

- using n distinct non-adaptive thresholds one can indeed obtain the competitive ratio of $(1 - 1/e \approx 0.63)$; and
- no online algorithm can achieve a competitive ratio better than 0.73.

Our results improve the approximation guarantee of single-item sequential posted pricing mechanisms from 0.5 to $(1 - 1/e)$ when the order of agents (customers) are chosen randomly.

We also consider the minimization variants of the prophet inequality problem. In particular, we show that, even for the simple case in which numbers are drawn from identical and independent distributions (i.i.d.), there is no constant competitive online algorithm for the minimization variants of the prophet inequality and prophet secretary

problems. We refer the reader to Section 2.2 for a detailed description of our model and techniques.

1.4 Online Minimization Problems

We now give a brief description of our framework for solving a major family of Steiner Connectivity problems.

1.4.1 From Steiner Connectivity to Online Covering

Network design problems deal with settings where the goal is to design a network (i.e., find a subgraph of a given graph) that satisfies certain connectivity requirements. Edges of the given graph describe the possible links the network may have, and each requirement is in the form of connecting a pair of vertices of the graph. The Steiner tree problem is among the most fundamental problems in the literature; given a set of vertices called terminals, the goal is to connect them with the minimum cost to a special vertex root (see recent achievements on the offline topic in [BHL13] and the references therein). In the online variant of the problem where the terminals are revealed in an online manner, sophisticated primal-dual methods are known for the problem when only the edges have costs. However, no primal-dual competitive algorithm is known for the more general version where the vertices (nodes) have costs. One obstacle is that the node-weighted variant has set cover as a special case which makes the problem inherently more difficult. Indeed the node-weighted variant has many applications; especially with

the growth in the usage of wireless systems, an edge-weighted graph cannot be a proper model in many applications. In our recent work [BHL13], we have given a simple and efficient algorithm for the offline node-weighted (prize-collecting) Steiner forest problem. In the Steiner forest (SF) problem each terminal is requested to be connected to a different root. In another work, we have shown the applications of our primal-dual framework by giving the first algorithm with poly-logarithmic competitive ratio for the online variant of Steiner forest problem (and more generally network-design problems characterized by proper functions). We also design the first node-weighted Steiner forest algorithm for planar graphs with asymptotically tight competitive ratio. This is indeed the first instance of an online graph optimization problem in which the structure of planar graphs are successfully exploited to achieve better results. All our solutions can be interpreted as instances of a single primal-dual framework. We call this framework *Disk painting* which we will describe in the next chapters.

1.4.2 Prize-collecting Variants

Over the last two decades, network design problems have been a cornerstone of algorithmic research. The Steiner tree problem and its various generalizations have been central to this research effort. One branch of Steiner problems that has attracted substantial attraction are the so called prize-collecting problems, where the algorithm is permitted to violate one or more connection constraints but must pay corresponding *penalties* in the objective function. Prize collecting (PC) Steiner problems were originally moti-

vated by applications in network planning for service providers (see e.g., [JMP00]). They have since become a well-studied branch of approximation algorithms (see the work of Archer *et al* [ABHK11] and Goemans and Williamson [GW95]). In this thesis, we focus on the online model, where the connectivity demands appear over time and must be immediately satisfied. The study of online prize-collecting network design was initiated recently by Qian and Williamson [QW11]. This is somewhat surprising on two counts: first, online versions of Steiner problems appear prominently in the algorithmic literature and prize-collecting variants are a natural generalization; and second, the online model is well-motivated in the context of prize-collecting Steiner problems since in practice, new customers appear over time and network providers must upgrade their networks according to the new demands or lose these customers, thereby paying the corresponding penalty in the revenue.

We provide a simple generic approach to online prize-collecting Steiner problems that reduces these problems to their fractional non-prize-collecting counterparts losing a logarithmic factor in the competitive ratio. Using known results for the online edge-weighted (EW) and node-weighted (NW) Steiner tree problems, this reduction yields algorithms with competitive ratios of $O(\log^2 n)$ and $O(\log^4 n)$ respectively for their prize-collecting variants. Exploring further, we give improved algorithms for both these problems: for the EW problem, we match the competitive ratio of $O(\log n)$ obtained by Qian and Williamson [QW11], whereas for the NW problem, we obtain a competitive ratio of $O(\log^3 n)$. Both these results are obtained by employing a novel online dual-fitting

approach. Our result represents the first algorithm for online NW-PCST that achieves a poly-logarithmic competitive ratio.

1.5 Mixed Packing/Covering: Online Degree-bounded SF

The problem of satisfying connectivity demands on a graph while respecting given constraints has been a pillar of the area of network design since the early seventies [Chv73, CGM80, CG82, PY82]. The problem of DEGREE-BOUNDED SPANNING TREE, introduced in Garey and Johnson's *Black Book* of NP-Completeness [MD79], was first investigated in the pioneering work of Fürer and Raghavachari [FR90] (Allerton'90). In the DEGREE-BOUNDED SPANNING TREE problem, the goal is to construct a spanning tree for a graph $G = (V, E)$ with n vertices whose maximal degree is the smallest among all spanning trees. Let b^* denote the maximal degree of an optimal spanning tree. Fürer and Raghavachari [FR90] give a parallel approximation algorithm which produces a spanning tree of degree at most $O(\log(n)b^*)$.

Agrawal, Klein, and Ravi ([AKR91]) consider the following generalizations of the problem. In the DEGREE-BOUNDED STEINER TREE problem we are only required to connect a given subset $T \subseteq V$. In the even more general DEGREE-BOUNDED STEINER FOREST problem the demands consist of vertex pairs, and the goal is to output a subgraph in which for every demand there is a path connecting the pair. They design an algorithm that obtains a multiplicative approximation factor of $O(\log(n))$. Their main technique is to reduce the problem to minimizing congestion under

integral concurrent flow restrictions and to then use the randomized rounding approach due to Raghavan and Thompson ([RT85], STOC'85).

Shortly after the work of Agrawal *et al.*, in an independent work in SODA'92 and later J. of Algorithms'94, Fürer and Raghavachari [FR94] significantly improved the result for DEGREE-BOUNDED STEINER FOREST by presenting an algorithm which produces a Steiner forest with maximum degree at most $b^* + 1$. They show that the same guarantee carries over to the *directed* variant of the problem as well. Their result is based on reducing the problem to that of computing a sequence of maximal matchings on certain auxiliary graphs. This result settles the approximability of the problem, as computing an optimal solution is NP-hard even in the spanning tree case.

We initiate the study of degree-bounded network design problems *in an online setting*, where connectivity demands appear over time and must be immediately satisfied. We first design a deterministic algorithm for ONLINE DEGREE-BOUNDED STEINER FOREST with a logarithmic competitive ratio. Then we show that this competitive ratio is asymptotically best possible by proving a matching lower bound for randomized algorithms that already holds for the Steiner tree variant of the problem.

1.6 Outline

In the first part of the thesis, we develop techniques for solving a major family of online packing problems. In Section 2.1, we apply these techniques to the online stochastic generalized assignment problem. In Section 2.2, we introduce the applications

of online techniques in designing mechanisms for our novel Bayesian model, Prophet Secretary.

In the second part of the thesis, we first introduce the Disk Painting framework in Section 3.1. We continue by demonstrating the applications of this framework for solving various Steiner connectivity problems in Sections 3.5.3 and 3.5.

Finally in the last part of thesis, we solve the first degree-bounded connectivity problem via a dual analysis in Section 4.1. This is indeed one of the very few problems with both packing and covering constraints that have been efficiently solved in an online setting.

CHAPTER 2

Packing Problems

2.1 The Generalized Assignment Problem (GAP)

We start by considering a general subclass of packing problems. Here for simplicity we describe our analytic tools as a randomized process. However, a dual interpretation of these methods can be found in [AHL+11, AHL12].

Online Generalized Assignment Problem

The generalized assignment problem (GAP) and its special cases *multiple knapsack*[1] and *bin packing*[2] capture several fundamental optimization problems and have many

[1] In the multiple knapsack problem, we are given a set of items and a set of bins (knapsacks) such that each item j has a profit v_j and a size s_j, and each bin i has a capacity C_i. The goal is to find a subset of items of maximum profit such that they have a feasible packing in the bins.

[2] In the bin packing problem, given a set of items with different sizes, the goal is to find a packing of items into unit-sized bins that minimizes the number of bins used.

practical applications in computer science, operations research, and related disciplines. The (offline) GAP is defined as follows:

Definition 2.1 (Generalized Assignment Problem)**.** *There is a set of n items and m bins. Each bin has a hard capacity where the total size of items placed in a bin cannot exceed its capacity. Each item has a value and a size if placed in a bin; both might depend on the bin. The goal is to find a maximum valued assignment of items to the bins which respects the capacities of the bins.*

For example GAP can be viewed as a scheduling problem on parallel machines, where each machine has a capacity (or a maximum load) and each job has a size (or a processing time) and a profit, each possibly dependent on the machine to which it is assigned, and the objective is to find a feasible scheduling which maximizes the total profit. Though multiple knapsack and bin packing have a fully polynomial-time approximation scheme (asymptotic for bin packing) in the offline setting [CK00], GAP is APX-hard and the best known approximation ratio is $1 - 1/e + \epsilon$ where $\epsilon \approx 10^{-180}$ [FV06], which improves on a previous $(1 - 1/e)$-approximation [FGMS06].

In this section we consider the *online stochastic* variant of the problem:

Definition 2.2 (Online Stochastic Generalized Assignment Problem)**.** *There are n items arriving in an online manner which can be of different types. There are m (static) bins each with a capacity limit on the total size of items that can be placed in it. A type of an item is associated with a value and a size distribution which may depend on the bin to which the item is placed. Stochastic information is known about the type of an item*

and sizes/values of the types. Upon arrival of an item, the type of that item is revealed. However the realization of the size of the item is revealed only after it has been placed in a bin. The goal is to find a maximum valued assignment of items to the bins. We consider a large-capacity *assumption: no item takes up more than $\frac{1}{k}$ fraction of the capacity of any bin.*

We emphasize that there are two sources of uncertainty in our model: the type of an item and the size of the item. The type of an item (which contains a size-distribution) is revealed before making the assignment, however the actual size of an item is revealed after the assignment.

2.1.1 Previous Work

To the best of our knowledge, Feldman et al. [FKM+09] were the first to consider the generalized assignment problem in an online setting, albeit with deterministic sizes. In the adversarial model where the items and the order of arrivals are chosen by an adversary, there is no competitive algorithm. Consider the simple case of one bin with capacity one and two arriving items each with size one. The value of the first item is 1. The value of the second item would be either $\frac{1}{\epsilon}$ or 0 based on whether we place the first item in the bin. Thus the online profit cannot be more than ϵ factor of the offline profit. Indeed one can show a much stronger hardness result for the adversarial model: no algorithm can

be competitive for the two special cases of GAP, namely the adword problem[3] and the display ad problem[4] even under the large-capacity assumption [FKM+09, MSVV07].

Since no algorithm is competitive for online GAP in the adversarial model, Feldman et al. consider this model with *free disposal*. In the free disposal model, the total size of items placed in a bin may exceed its capacity, however, the profit of the bin is the maximum-valued subset of the items in the bin which does not violate the capacity. Feldman et al. give a $(1 - \frac{1}{e} - \epsilon)$-competitive primal-dual algorithm for GAP under the free disposal assumption and the additional large-capacity assumption by which the capacity of each bin is at least $O(\frac{1}{\epsilon})$ times larger than the maximum size of a single item. Although the free disposal assumption might be counter-intuitive in time-sensitive applications such as job scheduling where the machine may start doing a job right after the job assignment, it is a very natural assumption in many applications including applications in economics like ad allocation – a buyer does not mind receiving more items.

Dean, Goemans, and Vondrak [DGV04] consider the (offline) stochastic knapsack problem which is closely related to GAP. In their model, there is only one bin and the value of each item is known. However, the size of each item is drawn from a known distribution only after it is placed in the knapsack. We note that this is an offline setting in the sense that we may choose any order of items for allocation. This model is motivated by job scheduling on a single machine where the actual processing time required

[3]The adword problem is a special case of GAP where the size and the value of placing an item in a bin is the same.

[4]The display ad problem is a special case of GAP where all sizes are uniform.

for a job is learned only after the completion of the job. Dean et al. give various adaptive and non-adaptive algorithms for their model where the best one has a competitive ratio $\frac{1}{3} - \epsilon$. Recently Bhalgat improved the competitive ratio to $\frac{1}{2} - \epsilon$ [Bha12]. Other variations, such as soft capacity constraints, have also been considered for which we refer the reader to [BGK11, GI99, KRT97]. Dean et al. [DGV04] also introduce an *ordered model* where items must be considered in a specific order, which can be seen as a version of the the online model with a known order. Dean et al. [DGV04] present a $\frac{1}{9.5}$-competitive algorithm. In general, the online model can be considered as a more challenging variation of the models proposed by Dean et al, and we show that the large-capacity assumption is enough to overcome this challenge.

To the best of our knowledge, the current variation of the online stochastic GAP has not been considered before. We note that since the distributions are not necessary i.i.d., this model generalizes the well-known *prophet inequalities*. [5] Even with stochastic information about the arriving queries, no online algorithm can achieve a competitive ratio better than $\frac{1}{2}$ [Ala11, HKS07, KS77, KS78]. Consider the simple example from before where the value of the first item is 1 with probability one and the value of the second item is $\frac{1}{\epsilon}$ with probability ϵ, and 0 with probability $1 - \epsilon$. The algorithm can only select one item. No online (randomized) algorithm can achieve a profit more than $\max\{1, \epsilon(\frac{1}{\epsilon})\} = 1$ in expectation. However, the expected profit of the optimum offline assignment is $(1 -$

[5] In the classic prophet inequality problem, given the distribution of a sequence of random variables, an onlooker has to choose from the succession of these values. The onlooker can only choose a certain number of values and cannot choose a past value. The goal is to maximize the total sum of selected numbers.

$\epsilon)1 + \epsilon(\frac{1}{\epsilon}) = 2 - \epsilon$. Therefore without any additional assumption one cannot get a competitive ratio better than $1/2$. We overcome this difficulty by considering the natural large-capacity assumption which arises in many applications such as online advertising.

There has been various $\frac{1}{2}$-approximation algorithms for offline GAP [CK00, FNW78, ST93]. Fleischer et al. [FGMS06] developed the first $(1 - 1/e)$-approximation algorithm for GAP. It has been observed for more general packing constraints, improving over the ratio $1 - 1/e$ is not possible unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log \log n)})$ [FGMS06]. However, for GAP with simple knapsack constraints, Feige and Vondrak [FV06] improve the approximation ratio to $1 - 1/e + \epsilon$ where $\epsilon \approx 10^{-180}$. GAP is a special case of the problem of maximizing a monotone submodular function subject to a matroid constraint [FNW78]. Vondrak [Von08] improves the approximation ratio for this problem from $1/2$ to $1 - 1/e$ and thus solves a long-standing open problem.

Recently special cases of online GAP have been extensively studied, mostly motivated by ad allocation frameworks. In the most basic setting, namely *online matching*, all the sizes, values, and capacities are one. In the worst case model, the celebrated result of Karp et al. [KVV90] gives a $(1-\frac{1}{e})$-approximation algorithm. In the stochastic random arrival model, a sequence of papers [FMMM09, KMT11, MY11] improve this ratio to 0.696. In the full-information model, the best known approximation ratio is 0.703 [MZO12] improving on a previous ratio 0.702 [MOS11].

A generalization of online matching is the display ad problem where the values are arbitrary but all item sizes are one. In the worst case analysis by assuming free disposal,

Feldman et al. [FKM+09] give a tight $(1-\frac{1}{e})$-approximation algorithm. In the stochastic full-information setting, Haeupler et al. [HMZ11] give a 0.66-approximation algorithm. Under the large-capacity assumption, Feldman et al. [FHK+10] give a $(1-o(1))$-approximation algorithm where the contribution of any single item is at most roughly $O(\frac{\epsilon}{m \log n})$. Devanur et al. [DJSW11] give a $(1-\epsilon)$-algorithm for the case of unknown i.i.d. distribution where the profit contributed by a single item is at most $O(\frac{\epsilon^2}{\log(n/\epsilon)})$ of the total profit. Recently Alaei et al. [AHL12] give a $(1-\epsilon)$-approximation for a more general Prophet Inequality setting where the capacity of each bin is $\Omega(\frac{1}{\epsilon^2})$.

Another generalization of online matching is the adword problem [MSVV07] which can be considered as a special case of GAP where the values and sizes are equal. In the worst case analysis, by assuming large capacities one can obtain a $(1-\frac{1}{e})$-approximation algorithm [BJN07, MSVV07]. Devanur and Hayes [DH09] give a $(1-O(\frac{1}{\sqrt{k}}))$-approximation algorithm in the random order model which depends on other input variables too. Here k is the ratio of a bidder's budget to his maximum bid. Alaei et al. [AHL12] give a $O(1-\frac{1}{\sqrt{2\pi k}})$-competitive algorithm for the Prophet Inequality setting. Recently Devanur et al. [DSA12] give an algorithm with the same competitive ratio for the unknown distribution model.

GAP generalizes adword, display ad, and stochastic knapsack problems. For the case of full-information, we give a natural algorithm which achieves a $(1-\frac{1}{\sqrt{k}})$-competitive algorithm where the size of each item is at most $\frac{1}{k}$ of a bin capacity. The closest work to this result may be that of [FHK+10] where Feldman et al. give a $(1-o(1))$-

approximation algorithm for a more general setting and under the random order model (though the size of an item is known before the allocation). However, the capacity to the item size ratio also depends on the input variables. More precisely, the ratio of the maximum value of a single item to *OPT* should be at most $O(\frac{\epsilon}{m \log n})$ and the ratio of the maximum size of a single item to a bin capacity should be at most $O(\frac{\epsilon^3}{m \log n})$. A summary of the latest results is shown in Table 2.1.

2.1.2 Current Achievements for Stochastic GAP

The *loss factor* of an online algorithm is the ratio α such that the profit of the algorithm is at least $1 - \alpha$ fraction of the optimal offline profit. The main result of this section can be summarized in the following theorem [AHL13].

Theorem 2.1. *For the stochastic generalized assignment problem, there exists a randomized online algorithm (see 2.6) with the loss factor at most* $\frac{1}{\sqrt{k}}$ *in expectation.*

The proposed algorithm initially computes an optimal solution for a linear program corresponding to a fractional expected instance. In the online stage, the algorithm tentatively assigns each item upon arrival to one of the bins at random with probabilities proportional to the fractional LP solution. This ensures that the expected total size of items assigned tentatively to each bin does not exceed its capacity. However, once a bin becomes full, any item which gets tentatively assigned to that bin will have to be dis-

carded. In general, a straightforward randomized assignment based on the LP solution could be arbitrarily far from optimal; that is because the probability of an item being discarded due to a bin being full could be arbitrarily close to 1 for items that arrive towards the end. To overcome this problem, we incorporate an adaptive threshold based strategy for each bin so that an item tentatively assigned to a bin is only placed in the bin if the remaining capacity of the bin is more than a certain threshold. This ensures the online algorithm discards a tentatively assigned item with a probability at most $\frac{1}{\sqrt{k}}$ of the fractional LP assignment. The thresholds are computed adaptively based on previously observed items.

Indeed by using the fractional solution as a guideline, it is possible to achieve a non-adoptive competitive algorithm. One may scale down the fractional solution by a factor $1 - O(\frac{\log k}{\sqrt{k}})$ and assign the items with the modified probabilities, thus achieving a loss factor $O(\frac{\log k}{\sqrt{k}})$. By using the Chernoff bound it can be shown that the probability of exceeding bin capacities is very small. This simple algorithm gives an asymptotically optimum solution, however there are two drawbacks. The first issue is that the constants in various implementations of this idea are large. Thus unless the value k is very large, this algorithm cannot guarantee a reasonable competitive ratio; in contrast, the loss factor of our algorithm is exactly $\frac{1}{\sqrt{k}}$. On the other hand, in the applications of online GAP such as the Adword problem, the factor $O(\frac{\log k}{\sqrt{k}})$ is the loss factor of the millions of dollars. Therefore our algorithm saves a logarithmic factor in the loss of revenue. Indeed these drawbacks were also the motivation for improving the loss factor in the special cases of

online GAP in previous papers [AHL12, AHL+11].

The threshold based strategy of the online algorithm is presented in Section 2.1.4 in the form of a generalization of the magician's problem of [Ala11]. The original magician's problem can be interpreted as a stochastic knapsack problem with unit size items and a knapsack of size k and such that each item arrives in one of two possible states (e.g. good/bad) with known probabilities; the objective being to maximize – simultaneously for all items – the probability of picking every item that is good. On the other hand, in the generalized variant presented here, the size of each item can vary according to an arbitrary (but known) distribution; in this version k is an integer lower bound on the ratio of the total size of the knapsack to the maximum possible size of a single item. Although the bound we obtain for the generalized magician is similar to that of [Ala11], they are incomparable for small k; in particular for $k = 1$, one can easily achieve a $\frac{1}{2}$-competitive algorithm for the magician's problem with fixed size items, whereas for the generalized version with variable size items, no constant competitive algorithm is possible for $k = 1$.

Recently, Alaei et al. [AHL+11, AHL12][6] use a combination of expected linear programming approach and dynamic programming to achieve a $1 - \epsilon$-competitive algorithm for adword and display ad. They use a relatively simple dynamic programming in combination with the LP solution to check whether they should assign an item to a bidder or discard it. Using an approach similar to "dual fitting" [JMM+03], they demonstrate an analysis of the combination of a dynamic programming approach with an online LP-based

[6] In an independent work, Devanur et al. [DJSW11] also consider the expected linear program of a similar problem.

algorithm and prove a loss factor $\frac{1}{\sqrt{k+3}}$ for the display ad problem. They use the sand theorem of [Ala11] as a black-box in their analysis to derive proper dual variables in their dual fitting analysis of the algorithm. A dynamic programming approach needs to know the stochastic information about the remaining items, while a threshold-based approach needs to know the past, i.e., they are complements of each other. However, analysis of the dynamic programming even with uniform sizes is involved. Furthermore, it is not easy to generalize the approach of [AHL12] to the model of Goemans et al. [DGV04] where the given sizes show only the expected size of an item.

It is worthwhile to compare our stochastic model against other popular models, i.e., random order model[7] and unknown distribution model[8]. While both the random order and the unknown distribution models require less stochastic information, they both treat items uniformly; hence they are more suitable for applications where items are more symmetric[9]. The model considered in this chapter is more suitable when there is a high degree of distribution asymmetry across the items. In particular, the extra stochastic information allows us to obtain practical bounds even for small values of k whereas in other stochastic models the obtained bounds often become meaningful only asymptotically in k.

[7] In the random arrival model items are chosen by an adversary, however they arrive in a random order.

[8] In the unknown distribution model the items are chosen i.i.d. from a fixed but unknown distribution.

[9] At a first glance, the random order model may appear to allow for asymmetry, however note that for any i and j, the i^{th} arriving item and the j^{th} arriving item have the same ex ante distribution in the random order model.

The all-or-nothing model

The online algorithm of this section can be applied to a slightly different model in which each item should still be either fully assigned or discarded, however in case of assignment, unlike GAP, an item can be fractionally split across multiple bins (i.e., the all-or-nothing assignment model). Note that the LP for the expected instance is still the same for the all-or-nothing model, therefore our online algorithm still obtains the same bound in expectation compared to the optimal offline solution. The all-or-nothing model is suitable for *subscription-based advertisement* and *banner advertisement*.

The subscription-based advertisement problem is an example of an offline ad allocation setting motivated by the banner advertisement. In this problem, there is a set of contracts proposed by the advertisers and the goal is to accept the contracts of a subset of the advertisers which maximizes the revenue. The contract proposed by an advertiser specifies a collection of webpages which are relevant to his product, a desired total quantity of impressions on these pages, and a price. Each webpage has an ad inventory with a certain number of banner ads. The problem of selecting a feasible subset of advertisers with the maximum total value does not have any non-trivial approximation. This can be shown by a reduction from the Independent Set problem on a graph; advertisers represent the vertices of the graph and webpages represent the edges of the graph. Advertisers desire all the impressions of the relevant webpages. Thus any feasible subset of advertisers would denote an independent set in the graph. This shows maximizing the total value does not have a non-trivial approximation. Different pricing models have also been intro-

duced by Feige et al. [FIMN08]. The proof of the following corollary is by a reduction from the all-or-nothing model.

Corollary 2.1. *There exists a randomized algorithm for the subscription-based advertisement problem which obtains a loss factor* $\frac{1}{\sqrt{k}}$ *in expectation where the number of available impressions on each website is at least* k *times the required impressions of each relevant advertiser.*

Proof. One can show that this is an offline version of the all-or-nothing model where bins denote the webpages and items denote the advertisers. The size of an item is the required number of ads of an advertiser and the value of an item is the proposed price. By Theorem 2.5, we can achieve at least $1 - \frac{1}{\sqrt{k}}$ fraction of the optimal profit in expectation. □

2.1.3 Preliminaries

Model

We consider the problem of assigning n items to m bins; items arrive online and stochastic information is known about the size/value of each item; the objective is to maximize the total value of the assignment. The item $i \in [n]$ (arriving at time i) has r_i possible types with each type $t \in [r_i]$ having a probability of p_{it}, a value of $v_{itj} \in \mathbb{R}_+$, and a size of $S_{itj} \in [0, 1]$ if placed in bin j (for each $j \in [m]$); S_{itj} is a random variable which is drawn from a distribution with a CDF of F_{itj} if the item is placed in bin j. Each bin $j \in [n]$ has a capacity of $c_j \in \mathbb{N}_0$ which limits the total size of the items placed in that

bin[10]. The type of each item is revealed upon arrival and the item must be either placed in a bin or discarded; this decision cannot be changed later. The size of an item is revealed only after it has been placed in a bin, furthermore an item can be placed in a bin only if the bin has at least one unit of capacity left. We assume that n, m, c_j, v_{itj} and F_{itj} are known in advance.

Benchmark

Consider the following linear program in which $\widetilde{s}_{itj} = \mathbf{E}_{S_{itj} \sim F_{itj}}[S_{itj}]$ [11].

$$\begin{aligned}
\text{maximize} \quad & \sum_i \sum_t \sum_j v_{itj} x_{itj} && && (\overline{OPT}) \\
\text{subject to} \quad & \sum_i \sum_t \widetilde{s}_{itj} x_{itj} \leq c_j, && \forall j \in [m] \\
& \sum_j x_{itj} \leq p_{it}, && \forall i \in [n], \forall t \in [r_i] \\
& x_{itj} \in [0, 1],
\end{aligned}$$

The optimal value of this linear program, which corresponds to the expected instance, is an upper bound on the expected value of the optimal offline assignment.

[10] Our results hold for non-integer capacitates, however we assume integer capacities to simplify the exposition.

[11] Throughout the rest of this chapter, we often omit the range of the sums whenever the range is clear from the context (e.g., $\sum_i$ means $\sum_{i \in [n]}$, and $\sum_j$ means $\sum_{j \in [m]}$, etc).

Theorem 2.2. *The optimal value of the linear program ($\overline{OPT}$) is an upper bound on the expected value of the offline optimal assignment.*

Proof. Let x^*_{itj} denote the ex ante probability that item i is of type t and is assigned to bin j in the optimal offline assignment. It is easy to see that x^*_{itj} is a feasible assignment for the linear program. Furthermore, the expected value of the optimal offline assignment is exactly $\sum_i \sum_t \sum_j v_{itj} x^*_{itj}$ which is equal to the value of the linear program for x^*_{itj} which is itself no more than the optimal value of the linear program. Note that the optimal value of the linear program may be strictly higher since a feasible assignment of the linear program does not necessarily correspond to a feasible offline assignment policy. □

Section 2.1.5 presents an online adaptive algorithm which obtains a loss factor $\frac{1}{\sqrt{k}}$ w.r.t. the optimal value of the above linear program, where $k = \min_j c_j$. We emphasize that our adaptive algorithm saves a logarithmic factor in the loss of the outcome compared to the non-adaptive methods. Next section presents a stochastic toy problem and its solution which is used in our online algorithm.

2.1.4 The Generalized Magician's Problem

We present a generalization of the magician's problem, which was originally introduced in [Ala11]; we also present a near-optimal solution for this generalization.

Definition 2.3 (The Generalized Magician's Problem)**.** *A magician is presented with a*

series of boxes one by one, in an online fashion. There is a prize hidden in one of the boxes. The magician has a magic wand that can be used to open the boxes. The wand has k units of mana [Wik12]. If the wand is used on box i and has at least 1 unit of mana, the box opens, but the wand loses a random amount of mana $X_i \in [0, 1]$ drawn from a distribution specified on the box by its cumulative distribution function F_{X_i} (i.e., the magician learns F_{X_i} upon seeing box i). The magician wishes to maximize the probability of obtaining the prize, but unfortunately the sequence of boxes, the distributions written on the boxes, and the box containing the prize have been arranged by a villain; the magician has no prior information (not even the number of the boxes); however, it is guaranteed that $\sum_i \mathbf{E}[X_i] \leq k$, and that the villain has to prepare the sequence of boxes in advance (i.e., cannot make any changes once the process has started).

The magician could fail to open a box either because (a) he might choose to skip the box, or (b) his wand might run out of mana before getting to the box. Note that once the magician fixes his strategy, the best strategy for the villain is to put the prize in the box which, based on the magician's strategy, has the lowest ex ante probability of being opened. Therefore, in order for the magician to obtain the prize with a probability of at least γ, he has to devise a strategy that guarantees an ex ante probability of at least γ for opening each box. Notice that allowing the prize to be split among multiple boxes does not affect the problem. We present an algorithm parameterized by a probability $\gamma \in [0, 1]$ which guarantees a minimum ex-ante probability of γ for opening each box while trying to minimize the mana used. We show that for $\gamma \leq 1 - \frac{1}{\sqrt{k}}$ this algorithm never requires

more than k units of mana.

Definition 2.4 (γ-Conservative Magician)**.** *The magician adaptively computes a sequence of thresholds $\theta_1, \theta_2, \ldots \in \mathbb{R}_+$ and makes a decision about each box as follows: let W_i denote the amount of mana lost prior to seeing the i^{th} box; the magician makes a decision about box i by comparing W_i against θ_i; if $W_i < \theta_i$, it opens the box; if $W_i > \theta_i$, it does not open the box; and if $W_i = \theta_i$, it randomizes and opens the box with some probability (to be defined). The magician chooses the smallest threshold θ_i for which $\mathbf{Pr}[W_i \leq \theta_i] \geq \gamma$ where the probability is computed ex ante (i.e., not conditioned on $X_1, \ldots, X_{i-1}$). Note that γ is a parameter that is given. Let $F_{W_i}(w) = \mathbf{Pr}[W_i \leq w]$ denote the ex ante CDF of random variable W_i, and let Y_i be the indicator random variable which is 1 iff the magician opens the box i. Formally, the probability with which the magician should open box i condition on W_i is computed as follows*[12].

$$\mathbf{Pr}\left[Y_i = 1 \middle| W_i\right] = \begin{cases} 1 & W_i < \theta_i \\ (\gamma - F^-_{W_i}(\theta_i))/(F_{W_i}(\theta_i) - F^-_{W_i}(\theta_i)) & W_i = \theta_i \\ 0 & W_i > \theta_i \end{cases} \tag{Y}$$

$$\theta_i = \inf\{w | F_{W_i}(w) \geq \gamma\} \tag{θ}$$

In the above definition, $F^-_{W_i}$ is the left limit of F_{W_i}, i.e., $F^-_{W_i}(w) = \mathbf{Pr}[W_i < w]$.

Note that $F_{W_{i+1}}$ and $F^-_{W_{i+1}}$ are fully determined by F_{W_i} and F_{X_i} and the choice of γ (see Theorem 2.4). Observe that θ_i is in fact computed before seeing box i itself.

[12] Assume $W_0 = 0$

A γ-conservative magician may fail for a choice of γ unless all thresholds θ_i are less than or equal to $k-1$. The following theorem states a condition on γ that is sufficient to guarantee that $\theta_i \leq k-1$ for all i.

Theorem 2.3 (γ-Conservative Magician)**.** *For any $\gamma \leq 1 - \frac{1}{\sqrt{k}}$, a γ-conservative magician with k units of mana opens each box with an ex ante probability of γ exactly.*

Proof. See Section 2.1.6. □

Definition 2.5 (γ_k)**.** *We define γ_k to be the largest probability such that for any $k' \geq k$ and any instance of the magician's problem with k' units of mana, the thresholds computed by a γ_k-conservative magician are no more than $k'-1$. In other words, γ_k is the optimal choice of γ which works for all instances with $k' \geq k$ units of mana. By Theorem 2.3, we know that γ_k must be*[13] *at least $1 - \frac{1}{\sqrt{k}}$.*

Observe that γ_k is a non-decreasing function in k and approaches 1 as $k \to \infty$. However $\gamma_1 = 0$ which is in contrast to the bound of $\frac{1}{2}$ obtained for $k = 1$ in [Ala11] in which all X_i are Bernoulli random variables. It turns out that when X_i are arbitrary random variables in $[0, 1]$, no algorithm exists for the magician that can guarantee a constant non-zero probability for opening each box.

Proposition 2.1. *For the generalized magician's problem for $k = 1$, no algorithm for the magician (online or offline) can guarantee a constant non-zero probability for opening*

[13] Because for any $k' \geq k$ obviously $1 - \frac{1}{\sqrt{k}} \leq 1 - \frac{1}{\sqrt{k'}}$.

each box.

Proof. Suppose there is an algorithm for the magician that is guaranteed to open each box with a probability of at least $\gamma \in (0, 1]$. We construct an instance in which the algorithm fails. Let $n = \lceil \frac{1}{\gamma} \rceil + 1$. Suppose all X_i are (independently) drawn from the distribution specified below.

$$X_i = \begin{cases} \frac{1}{2n} & \text{with prob. } 1 - \frac{1}{2n} \\ 1 & \text{with prob. } \frac{1}{2n} \end{cases}, \qquad \forall i \in [n]$$

As soon as the magician opens a box, the remaining mana will be less than 1, so he will not be able to open any other box, i.e., the magician can open only one box at every instance. Let Y_i denote the indicator random variable which is 1 iff the magician opens box i. Since $\sum_i Y_i \leq 1$, it must be $\sum_i \mathbf{E}[Y_i] \leq 1$. On the other hand, $\mathbf{E}[Y_i] \geq \gamma$ because the magician has guaranteed to open each box with a probability of at least γ. However $\sum_i \mathbf{E}[Y_i] \geq n\gamma > 1$ which is a contradiction. Note that $\sum_i \mathbf{E}[X_i] < 1$ so it satisfies the requirement of 2.3. □

Computation of $F_{W_i}(\cdot)$

For every $i \in [n]$, the equation $W_{i+1} = W_i + Y_i X_i$ relates the distribution of W_{i+1} to those of W_i and X_i [14]. The following lemma shows that the distribution of W_{i+1} is fully determined by the information available to the magician before seeing box $i + 1$.

[14] Note that the distribution of Y_i is dependent on/determined by W_i.

Theorem 2.4. *In the algorithm of γ-conservative magician (2.4), the choice of γ and the distributions of $X_1, \ldots, X_i$ fully determine the distribution of W_{i+1}, for every $i \in [n]$. In particular, $F_{W_{i+1}}$ can be recursively defined as follows.*

$$F_{W_{i+1}}(w) = F_{W_i}(w) - G_i(w) + \mathbf{E}_{X_i \sim F_{X_i}}\left[G_i(w - X_i)\right] \quad \forall i \in [n], \forall w \in \mathbb{R}_+ \qquad (F_W)$$

$$G_i(w) = \min(F_{W_i}(w), \gamma) \qquad \forall i \in [n], \forall w \in \mathbb{R}_+ \qquad (G)$$

Proof. See Section 2.1.6. □

As a corollary of Theorem 2.4, we show how F_{W_i} can be computed using dynamic programming, assuming X_i can only take discrete values that are proper multiples of some minimum value.

Corollary 2.2. *If all X_i are proper multiple of $\frac{1}{D}$ for some $D \in \mathbb{N}$, then $F_{W_i}(\cdot)$ can be computed using the following dynamic program.*

$$F_{W_{i+1}}(w) = \begin{cases} F_{W_i}(w) - G_i(w) + \sum_\ell \mathbf{Pr}[X_i = \frac{\ell}{D}] G_i(w - \frac{\ell}{D}) & i \geq 1, w \geq 0 \\ 1 & i = 0, w \geq 0 \\ 0 & \text{otherwise.} \end{cases},$$

$$\forall i \in [n], \forall w \in \mathbb{R}_+$$

In particular, the γ-conservative magician makes a decision for each box in time $O(D)$.

Note that it is enough to compute F_{W_i} only for proper multiples of $\frac{1}{D}$ because $F_{W_i}(w) = F_{W_i}(\frac{\lfloor Dw \rfloor}{D})$ for any $w \in \mathbb{R}_+$.

2.1.5 An Algorithm for Stochastic GAP

We present an online algorithm which obtains at least $1 - \frac{1}{\sqrt{k}}$-fraction of the optimal value of the linear program ($\overline{OPT}$). The algorithm uses, as a black box, the solution of the generalized magician's problem.

Definition 2.6 (Online Stochastic GAP Algorithm)**.**

1. *Solve the linear program ($\overline{OPT}$) and let x be an optimal assignment.*

2. *For each $j \in [m]$, create a γ-conservative magician (2.4) with c_j units of mana for bin j. γ is a parameter that is given.*

3. *Upon arrival of each item $i \in [n]$, do the following:*

   (a) *Let t denote the type of item i.*

   (b) *Choose a bin at random such that each bin $j \in [m]$ is chosen with probability $\frac{x_{itj}}{p_{it}}$. Let j^* denote the chosen bin.*

   (c) *For each $j \in [m]$, define the random variable X_{ij} as $X_{ij} \leftarrow S_{itj}$ if $j^* = j$, and $X_{ij} \leftarrow 0$ otherwise*[15]*. For each $j \in [m]$, write the CDF of X_{ij} on a*

[15]Note that S_{itj} is learned only after item i is placed in bin j which implies that X_{ij} may not be known at this point, however the algorithm does not use X_{ij} until after it is learned.

box and present it to the magician of bin j. The CDF of X_{ij} is $F_{X_{ij}}(s) = (1 - \sum_{t'} x_{it'j}) + \sum_{t'} x_{it'j} F_{it'j}(s)$.

(d) If the magician for bin j^* opened his box in step 3c, then assign item i to bin j^*, otherwise discard the item. For each $j \in [m]$, if the magician of bin j opened his box in step 3c, decrease the mana of that magician by X_{ij}. In particular, $X_{ij} = 0$ for all $j \neq j^*$, and $X_{ij^*} = S_{itj^*}$.

Theorem 2.5. *For any $\gamma \leq \gamma_k$, the online algorithm of 2.6 obtains in expectation at least a γ-fraction of the expected value of the optimal offline assignment (recall that $\gamma_k \geq 1 - \frac{1}{\sqrt{k}}$).*

Proof. By Theorem 2.2, it is enough to show that the online algorithm obtains in expectation at least a γ-fraction of the optimal value of the linear program ($\overline{OPT}$). Let x be an optimal assignment for the LP. The contribution of each item $i \in [n]$ to the value of bin $j \in [m]$ in the LP is exactly $\sum_t v_{itj} x_{itj}$. We show that the online algorithm obtains in expectation $\gamma \sum_t v_{itj} x_{itj}$ from each item i and each bin j.

Consider an arbitrary item $i \in [n]$ and an arbitrary bin $j \in [m]$. WLOG, suppose the items are indexed in the order in which they arrive. Observe that

$$\mathbf{E}[X_{ij}] = \sum_t p_{it} \frac{x_{itj}}{p_{it}} \mathbf{E}[S_{itj}] = \sum_t x_{itj} \widetilde{s}_{itj}.$$

Consequently, $\sum_i \mathbf{E}[X_{ij}] = \sum_i \sum_t x_{itj} \widetilde{s}_{itj} \leq c_j$.

The last inequality follows from the first set of constraints in the LP of ($\overline{OPT}$).

Given that $\sum_i \mathbf{E}[X_{ij}] \leq c_j$ and $\gamma \leq \gamma_k \leq \gamma_{c_j}$, Theorem 2.3 implies that the magician of bin j opens each box with a probability of γ. Therefore, the expected contribution of item i to bin j is exactly $\sum_t \gamma p_{it} \frac{x_{itj}}{p_{it}} v_{itj} = \gamma \sum_t x_{itj} v_{itj}$. Consequently, the online algorithm obtains $\gamma \sum_i \sum_j \sum_t x_{itj} v_{itj}$ in expectation which is at least a γ-fraction of the expected value of the optimal offline assignment. Furthermore, each magician guarantees that the total size of the items assigned to each bin does not exceed the capacity of that bin. □

2.1.6 Analysis of Generalized γ-Conservative Magician

We present the proof of Theorem 2.3. We prove the theorem in two parts. In the first part, we show that the thresholds computed by the γ-conservative magician indeed guarantee that each box is opened with an ex-ante probability of γ, assuming there is enough mana. In the second part, we show that for any $\gamma \leq 1 - \frac{1}{\sqrt{k}}$, the thresholds θ_i are less than or equal to $k-1$, for all i, which implies that the magician never requires more than k units of mana. Below, we repeat the formulation of the threshold based strategy of the magician.

$$\mathbf{Pr}\left[Y_i = 1 \middle| W_i\right] = \begin{cases} 1 & W_i < \theta_i \\ (\gamma - F^-_{W_i}(\theta_i))/(F_{W_i}(\theta_i) - F^-_{W_i}(\theta_i)) & W_i = \theta_i \\ 0 & W_i > \theta_i \end{cases} \tag{Y}$$

$$\theta_i = \inf\{w | F_{W_i}(w) \geq \gamma\} \tag{θ}$$

Part 1

We show that the thresholds computed by a γ-conservative magician guarantee that each box is opened with an ex ante probability of γ (i.e., $\mathbf{Pr}[Y_i = 1] = \gamma$), assuming there is enough mana.

$$\begin{aligned}\mathbf{Pr}\left[Y_i \leq w\right] &= \mathbf{Pr}\left[Y_i = 1 \cap W_i < \theta_i\right] + \mathbf{Pr}\left[Y_i = 1 \cap W_i = \theta_i\right] \\ &\quad + \mathbf{Pr}\left[Y_i = 1 \cap W_i > \theta_i\right] \\ &= \mathbf{Pr}\left[W_i < \theta_i\right] + \frac{\gamma - F^{-}_{W_i}(\theta_i)}{F_{W_i}(\theta_i) - F^{-}_{W_i}(\theta_i)} \mathbf{Pr}\left[W_i = \theta_i\right] = \gamma\end{aligned}$$

Part 2

Assuming $\gamma \leq 1 - \frac{1}{\sqrt{k}}$, we show that the thresholds computed by a γ-conservative magician are no more than $k-1$ (i.e., $\theta_i \leq k-1$ for all i). First, we present an interpretation of how $F_{W_i}(\cdot)$ evolves in i in terms of a sand displacement process.

Definition 2.7 (Sand Displacement Process)**.** *Consider one unit of infinitely divisible sand which is initially at position* 0 *on the real line. The sand is gradually moved to the right and distributed over the real line in* n *rounds. Let* $F_{W_i}(w)$ *denote the total amount of sand in the interval* $[0, w]$ *at the beginning of round* $i \in [n]$*. At each round* i *the following happens.*

(I) The leftmost γ*-fraction of the sand is selected by first identifying the smallest threshold* $\theta_i \in \mathbb{R}_+$ *such that* $F_{W_i}(\theta_i) \geq \gamma$ *and then selecting all the sand in*

the interval $[0, \theta_i)$ *and selecting a fraction of the sand at position* θ_i *itself such that the total amount of selected sand is equal to* γ*. Formally, if* $G_i(w)$ *denotes the total amount of sand selected from* $[0, w]$*, the selection of sand is such that* $G_i(w) = \min(F_{W_i}(w), \gamma)$*, for every* $w \in \mathbb{R}_+$*. In particular, this implies that only a fraction of the sand at position* θ_i *itself might be selected, however all the sand to the left of position* θ_i *is selected.*

(II) The selected sand is moved to the right as follows. Consider the given random variable $X_i \in [0, 1]$ *and let* $F_{X_i}(\cdot)$ *denote its CDF. For every point* $w \in [0, \theta_i]$ *and every distance* $\delta \in [0, 1]$*, take a fraction proportional to* $\mathbf{Pr}[X_i = \delta]$ *out of the sand which was selected from position* w *and move it to position* $w + \delta$*.*

It is easy to see that θ_i and $F_{W_i}(w)$ resulting from the above process are exactly the same as those computed by the γ-conservative magician.

Lemma 2.1. *At the end of the* i^{th} *round of the sand displacement process, the total amount of sand in the interval* $[0, w]$ *is given by the following equation.*

$$F_{W_{i+1}}(w) = F_{W_i}(w) - G_i(w) + \mathbf{E}_{X_i \sim F_{X_i}}\left[G_i(w - X_i)\right] \quad \forall i \in [n], \forall w \in \mathbb{R}_+ \qquad (F_W)$$

Proof. According to definition of the sand displacement process, $F_{W_{i+1}}(w)$ can be defined

as follows.

$$\begin{aligned} F_{W_{i+1}}(w) &= (F_{W_i}(w) - G_i(w)) + \iint_{\omega+\delta \le w} dG_i(\omega)\, dF_{X_i}(\delta) \\ &= F_{W_i}(w) - G_i(w) + \int G_i(\omega - \delta)\, dF_{X_i}(\delta) \\ &= F_{W_i}(w) - G_i(w) + \mathbf{E}_{X_i \sim F_{X_i}}\left[G_i(w - X_i)\right] \end{aligned}$$

□

Proof of Theorem 2.4. The claim follows directly from 2.1 □

Consider a conceptual barrier which is at position $\theta_i + 1$ at the beginning of round i and is moved to position $\theta_{i+1} + 1$ for the next round, for each $i \in [n]$. It is easy to verify (i.e., by induction) that the sand never crosses to the right side of the barrier (i.e., $F_{W_{i+1}}(\theta_i + 1) = 1$). In what follows, the sand theorem implies that the sand remains concentrated near the barrier throughout the process. The barrier theorem implies that the barrier never passes k.

Theorem 2.6 (Sand)**.** *Throughout the sand displacement process (2.7), at the beginning of round $i \in [n]$, the following inequality holds.*

$$F_{W_i}(w) < \gamma F_{W_i}(w+1), \qquad \forall i \in [n], \forall w \in [0, \theta_i) \qquad (F_W\text{-ineq})$$

Furthermore, at the beginning of round $i \in [n]$, the average distance of the sand from the barrier, denoted by d_i, is upper bounded by the

following inequalities in which the first inequality is strict except for $i = 1$.

$$d_i \le (1 - \{\theta_i\})\frac{1-\gamma^{\lfloor\theta_i\rfloor+1}}{1-\gamma} + \{\theta_i\}\frac{1-\gamma^{\lceil\theta_i\rceil+1}}{1-\gamma}$$
$$\le \frac{1-\gamma^{\lceil\theta_i\rceil+1}}{1-\gamma} < \frac{1}{1-\gamma}, \qquad \forall i \in [n] \quad (d)$$

Proof. We start by proving the inequality (F_W-ineq). The proof is by induction on i. The case of $i = 1$ is trivial because all the sand is at position 0 and so $\theta_1 = 0$. Suppose the inequality holds at the beginning of round i for all $w \in [0, \theta_i)$; we show that it holds at the beginning of round $i + 1$ for all $w \in [0, \theta_{i+1})$. Note that $\theta_i \le \theta_{i+1} \le \theta_i + 1$, so there are two possible cases:

Case 1. $w \in [0, \theta_i)$. Observe that $G_i(w) = F_{W_i}(w)$ in this interval, so:

$$F_{W_{i+1}}(w) = F_{W_i}(w) - G_i(w) + \mathbf{E}_{X_i}\left[G_i(w - X_i)\right] \qquad \text{by } (F_W).$$

$$= \mathbf{E}_{X_i}\left[F_{W_i}(w - X_i)\right] \qquad \text{by } G_i(w) = F_{W_i}(w), \text{ for } w \in [0, \theta_i).$$

$$< \mathbf{E}_{X_i}\left[\gamma F_{W_i}(w - X_i + 1)\right] \qquad \text{by induction hypothesis.}$$

$$= \gamma\,\mathbf{E}_{X_i}\left[F_{W_i}(w - X_i + 1) - G_i(w - X_i + 1) + G_i(w - X_i + 1)\right]$$

$$\le \gamma \quad \left(F_{W_i}(w+1) - G_i(w+1) + \mathbf{E}_{X_i}\left[G_i(w - X_i + 1)\right]\right)$$

$$\text{by monotonicity of } F_{W_i}(\cdot) - G_i(\cdot).$$

$$= \gamma F_{W_{i+1}}(w+1) \qquad \text{by } (F_W).$$

Case 2. $w \in [\theta_i, \theta_{i+1}]$. We prove the claim by showing that $F_{W_{i+1}}(w) < \gamma$ and $F_{W_{i+1}}(w+1) = 1$. Observe that $F_{W_{i+1}}(w) < \gamma$ because $w < \theta_{i+1}$ and because of the definition of θ_{i+1} in (θ). Furthermore, observe that $F_{W_{i+1}}(w+1) \geq F_{W_{i+1}}(\theta_i + 1) = 1$ both before and after round i all the sand is still contained in the interval $[0, \theta_i + 1]$.

Next, we prove inequality (d) which upper bounds the average distance of the sand from the barrier at the beginning of round $i \in [n]$.

$$
\begin{aligned}
d_i &= \int_0^{\theta_i+1} (\theta_i + 1 - w)\, dF_{W_i}(w) \\
&= \int_0^{\theta_i+1} F_{W_i}(w)\, dw && \text{by integration by part.} \\
&= \sum_{\ell=0}^{\lceil \theta_i \rceil} \int_{\theta_i - \ell}^{\theta_i+1-\ell} F_{W_i}(w)\, dw
\end{aligned}
$$

$$
\begin{aligned}
&\leq \sum_{\ell=0}^{\lfloor \theta_i \rfloor} \int_{\theta_i}^{\theta_i+1} \gamma^\ell F_{W_i}(w)\, dw \; + \int_{\lfloor \theta_i \rfloor + 1}^{\theta_i+1} \gamma^{\lceil \theta_i \rceil} F_{W_i}(w)\, dw && \text{by (F_W-ineq).} \\
&\leq \sum_{\ell=0}^{\lfloor \theta_i \rfloor} \gamma^\ell \; + \{\theta_i\}\, \gamma^{\lceil \theta_i \rceil} && \text{by } F_{W_i}(w) \leq 1. \\
&= (1 - \{\theta_i\}) \sum_{\ell=0}^{\lfloor \theta_i \rfloor} \gamma^\ell \; + \{\theta_i\} \sum_{\ell=0}^{\lceil \theta_i \rceil} \gamma^\ell \\
&= (1 - \{\theta_i\}) \frac{1 - \gamma^{\lfloor \theta_i \rfloor + 1}}{1 - \gamma} + \{\theta_i\} \frac{1 - \gamma^{\lceil \theta_i \rceil + 1}}{1 - \gamma} \leq \frac{1 - \gamma^{\lceil \theta_i \rceil + 1}}{1 - \gamma}
\end{aligned}
$$

The last inequality follows because $(1-\beta)L + \beta H \leq H$ for any $\beta \in [0,1]$ and any L, H with $L \leq H$. Note that at least one of the first two inequalities is strict except for $i = 1$ which proves the claim. □

Theorem 2.7 (Barrier). *If $\sum_{i=1}^{n} \mathbf{E}_{X_i \sim F_{X_i}}[X_i] \leq k$ for some $k \in \mathbb{N}$, and $\gamma \leq 1 - \frac{1}{\sqrt{k}}$, then the distance of the barrier from the origin is no more than k throughout the process, i.e., $\theta_i \leq k - 1$ for all $i \in [n]$.*

Proof. At the beginning of round i, let d_i and d'_i denote the average distance of the sand from the barrier and from the origin respectively. Recall that the barrier is defined to be at position $\theta_i + 1$ at the beginning of round i. Observe that $d_i + d'_i = \theta_i + 1$. Furthermore, $d'_{i+1} = d'_i + \gamma\, \mathbf{E}[X_i]$, i.e., the average distance of the sand from the origin is increased exactly by $\gamma\, \mathbf{E}[X_i]$ during round i (because the amount of selected sand is exactly γ and the sand selected from every position $w \in [0, \theta_i]$ is moved to the right by an expected distance of $\mathbf{E}[X_i]$). By applying Theorem 2.6 we get the following inequality for all $i \in [n]$.

$$\begin{aligned}\theta_i + 1 = d'_i + d_i &< \gamma \sum_{r=1}^{i-1} \mathbf{E}\left[X_i\right] + d_i \\ &\leq \gamma k + (1 - \{\theta_i\}) \frac{1 - \gamma^{\lfloor \theta_i \rfloor + 1}}{1 - \gamma} + \{\theta_i\} \frac{1 - \gamma^{\lceil \theta_i \rceil + 1}}{1 - \gamma}\end{aligned}$$

In order to show that distance of the barrier from the origin is no more than k throughout the process, it is enough to show that the above inequality cannot hold for $\theta_i > k-1$. In fact it is just enough to show that it cannot hold for $\theta_i = k-1$; alternatively, it is enough to show that the complement of the above inequality holds for $\theta_i = k-1$, i.e., $k \geq \gamma k + \frac{1-\gamma^k}{1-\gamma}$. To complete the proof, consider the the stronger inequality $k \geq \gamma k + \frac{1}{1-\gamma}$ which is quadratic in γ and can be solved to get a bound of $\gamma \leq 1 - \frac{1}{\sqrt{k}}$. □

Theorem 2.7 implies that a γ-conservative magician requires no more than k units of mana, assuming that $\gamma \leq 1 - \frac{1}{\sqrt{k}}$. That completes the proof of Theorem 2.3.

2.2 Applications to Mechanism Design: Prophet Secretary

Optimal stopping theory is a powerful tool for analyzing scenarios in which we generally require optimizing an objective function over the space of stopping rules for an allocation process under uncertainty. One such a scenario is the online auction which is the essence of many modern markets, particularly networked markets where information about goods, agents, and outcomes is revealed over a period of time and the agents must make irrevocable decisions without knowing future information. Combining optimal stopping theory with game theory allows us to model the actions of rational agents applying competing stopping rules in an online market.

Perhaps the most classic problems of stopping theory are the *prophet inequality* and the *secretary problem*. Research investigating the relation between online auction mechanisms and prophet inequalities was initiated by Hajiaghayi, Kleinberg, and Sandholm [HKS07]. They observed that algorithms used in the derivation of prophet inequalities, owing to their monotonicity properties, could be interpreted as truthful online auction mechanisms and that the prophet inequality in turn could be interpreted as the mechanism's approximation guarantee. Later Chawla, Hartline, Malec, and Sivan [CHMS10] showed the applications of prophet inequalities in Bayesian optimal mechansim design problems. The connection between the secretary problem and online auction mechanisms has been explored by Hajiaghayi, Kleinberg and Parkes [HKP04] and initiated several follow-up papers (see e.g. [BIKK07, BIKK08, BIK07, IKM06, Kle05]).

Prophet Inequality.

The classical prophet inequality has been studied in the optimal stopping theory since the 1970s when introduced by Krengel and Sucheston [Ken78, KS77, KS78] and more recently in computer science Hajiaghayi, Kleinberg and Sandholm [HKS07]. In the prophet inequality setting, given (not necessarily identical) distributions $\{D_1, \ldots, D_n\}$, an online sequence of values $X_1, \cdots, X_n$ where X_i is drawn from D_i, an onlooker has to choose one item from the succession of the values, where X_k is revealed at step k. The onlooker can choose a value only at the time of arrival. The onlooker's goal is to maximize her revenue. The inequality has been interpreted as meaning that a prophet with complete foresight has only a bounded advantage over an onlooker who observes the random variables one by one, and this explains the name *prophet inequality*.

An algorithm for the prophet inequality problem can be described by setting a threshold for every step: we stop at the first step that the arriving value is higher than the threshold of that step. The classical prophet inequality states that by choosing the same threshold $\mathit{OPT}/2$ for every step, one achieves the competitive ratio of $1/2$. Here the optimal solution OPT is defined as $\mathrm{E}[\max X_i]$. Naturally, the first question is whether one can beat $1/2$. Unfortunately, this is not possible: let $q = \frac{1}{\epsilon}$, and $q' = 0$. The first value X_1 is always 1. The second value X_2 is either q with probability ϵ or q' with probability $1 - \epsilon$. Observe that the expected revenue of any (randomized) online algorithm is $\max(1, \epsilon(\frac{1}{\epsilon})) = 1$. However the prophet, i.e., the optimal offline solution would choose q' if it arrives, and he would choose the first value otherwise. Hence, the optimal offline

revenue is $(1-\epsilon) \times 1 + \epsilon(\frac{1}{\epsilon}) \approx 2$. Therefore we cannot hope to break the $1/2$ barrier using any online algorithm.

Secretary Problem.

Imagine that you manage a company, and you want to hire a secretary from a pool of n applicants. You are very keen on hiring only the best and brightest. Unfortunately, you cannot tell how good a secretary is until you interview him, and you must make an irrevocable decision whether or not to make an offer at the time of the interview. The problem is to design a strategy which maximizes the probability of hiring the most qualified secretary. It is well-known since 1963 by Dynkin in [Dyn63] that the optimal policy is to interview the first $t-1$ applicants, then hire the next one whose quality exceeds that of the first $t-1$ applicants, where t is defined by $\sum_{j=t+1}^{n} \frac{1}{j-1} \leq 1 < \sum_{j=t}^{n} \frac{1}{j-1}$. As $n \to \infty$, the probability of hiring the best applicant approaches $1/e \approx 0.36$, as does the ratio t/n. Note that a solution to the secretary problem immediately yields an algorithm for a slightly different objective function optimizing the expected value of the chosen element. Subsequent papers have extended the problem by varying the objective function, varying the information available to the decision-maker, and so on, see e.g., [AMW01, GHB83, Van80, Wil91].

2.2.1 Further Related Work

Prophet Inequality.

The first generalization of the basic prophet inequality introduced by Krengel and Sucheston [Ken78, KS77, KS78] is *the multiple-choices* prophet inequality [AGSc01] in which both the onlooker and the prophet have $k > 1$ choices. Currently, the best algorithm for this setting is due to Alaei [Ala11], who gave an online algorithm with $(1 - \frac{1}{\sqrt{k+3}})$-competitive ratio for k-choice optimal stopping. Besides this, we have two generalizations for the (multiple-choices) prophet inequality that are *matroid* prophet inequality [KW12] and *matching* prophet inequality [AHL12].

In the matroid prophet inequality, we are given a matroid whose elements have random weights sampled independently from (not necessarily identical) probability distributions on $\mathcal{R}^+$. We then run an online algorithm with knowledge of the matroid structure and of the distribution of each element's weight. The online algorithm must then choose irrevocably an independent subset of the matroid by observing the sampled value of each element (in a fixed, prespecified order). The online algorithm's payoff is defined to be the sum of the weights of the selected elements. Kleinberg and Weinberg [KW12] show that for every matroid, there is an online algorithm whose expected payoff is at least half of the expected weight of the maximum-weight basis. Observe that the original prophet inequality introduced by Krengel and Sucheston [Ken78, KS77, KS78] corresponds to the special case of rank-one matroids.

The matching prophet inequality is due to Alaei, Hajiaghayi, and Liaghat [AHL12]. Indeed, they study the problem of online prophet-inequality matching in bipartite graphs. There is a static set of bidders and an online stream of items. The interest of bidders in items is represented by a weighted bipartite graph. Each bidder has a capacity, i.e., an upper bound on the number of items that can be allocated to him. The weight of a matching is the total weight of edges matched to the bidders. Upon the arrival of an item, the online algorithm should either allocate it to a bidder or discard it. The objective is to maximize the weight of the resulting matching. Here we assume we know the distribution of the incoming items in advance and we may assume that the t^{th} item is drawn from distribution $\mathcal{D}_t$. They generalize the $\frac{1}{2}$-competitive ratio of Krengel and Sucheston [KS77] by presenting an algorithm with an approximation ratio of $1 - \frac{1}{\sqrt{k+3}}$ where k is the minimum capacity. Oberve that the classical prophet inequality is a special case of this model where we have only one bidder with capacity one, i.e., $k = 1$ for which they get the same $\frac{1}{2}$-competitive ratio.

Secretary Problem.

The first generalization of the basic secretary problem [Dyn63] is the *multiple-choice secretary problem* [BIKK08] (see a survey by Babaioff et al. [BIKK08])) in which the interviewer is allowed to hire up to $k \geq 1$ applicants in order to maximize performance of the secretarial group based on their overlapping skills (or the joint utility of selected items in a more general setting). More formally, assuming applicants

of a set $S = \{a_1, a_2, \cdots, a_n\}$ (applicant pool) arriving in a uniformly random order, the goal is to select a set of at most k applicants in order to maximize a non-negative profit function $f : 2^S \mapsto \mathcal{R}^{\geq 0}$. For example, when $f(T)$ is the maximum individual value [Fre83, GM66], or when $f(T)$ is the sum of the individual values in T [Kle05], the problem has been considered thoroughly in the literature. Beside this, two generalizations for the (multiple-choices) secretary problem are *submodular secretary* [BHZ13] and *matroid secretary* [BIK07].

The submodular secretary problem is introduced by Bateni, Hajiaghayi, and Zadimoghaddam [BHZ13]. Indeed, both of the maximum individual value [Fre83, GM66] and the sum of the individual values [Kle05] aforementioned are special monotone non-negative submodular functions. Bateni, Hajiaghayi, and Zadimoghaddam [BHZ13] give an online algorithm with $(\frac{7}{1-1/e})$-competitive ratio for the submodular secretary problem. We should mention that there are more recent results with better constant competitive ratio (See for example the references in [BHZ13]).

In the matroid secretary problem considered by Babaioff et al. [BIK07], we are given a matroid with a ground set $\mathcal{U}$ of elements and a collection of independent (feasible) subsets $\mathcal{I} \subseteq 2^{\mathcal{U}}$ describing the sets of elements which can be simultaneously accepted. The goal is to design online algorithms in which the structure of $\mathcal{U}$ and $\mathcal{I}$ is known at the outset (assume we have an oracle to answer whether a subset of $\mathcal{U}$ belongs to $\mathcal{I}$ or not), while the elements and their values are revealed one at a time in a random order. As each element is presented, the algorithm must make an irrevocable decision to select

or reject it such that the set of selected elements belongs to $\mathcal{I}$ at all times. Babaioff et al. [BIK07] present an $O(\log r)$-competitive algorithm for general matroids, where r is the rank of the matroid. However, they leave as a main open question the existence of constant-competitive algorithms for general matroids. On the other hand, there are several follow-up works including the recent FOCS 2014 paper due to Lachish [Lac14] which gives $O(\log \log \text{rank})$-competitive algorithm.

2.2.2 Our Contributions

In this paper, we introduce *prophet secretary* as a natural combination of the prophet inequality problem and the secretary problem with applications to the Bayesian optimal mechanism design. Consider a seller that has an item to sell on the market to a set of arriving customers. The seller knows the types of customers that may be interested in the item and he has a price distribution for each type: the price offered by a customer of a type is anticipated to be drawn from the corresponding distribution. However, the customers arrive in a random order. Upon the arrival of a customer, the seller makes an irrevocable decision to whether sell the item at the offered price. We address the question of maximizing the seller's gain.

More formally, in the prophet secretary problem we are given a set $\{D_1, \ldots, D_n\}$ of (not necessarily identical) distributions. A number X_i is drawn from each distribution D_i and then, after applying a random permutation $\pi_1, \ldots, \pi_n$, the numbers are given to us in an online fashion, i.e., at step k, π_k and X_{π_k} are revealed. We are allowed to choose only

one number, which can be done only upon receiving that number. The goal is to maximize the expectation of the chosen value, compared to the expectation of the optimum offline solution that knows the drawn values in advance (i.e., $\mathit{OPT} = \mathrm{E}\left[\max_i X_i\right]$). For the ease of notation, in what follows the index i iterates over the distributions while the index k iterates over the arrival steps.

An algorithm for the prophet secretary problem can be described by a sequence of (possibly adaptive) thresholds $\langle \tau_1, \ldots, \tau_n \rangle$: we stop at the first step k that $X_{\pi_k} \geq \tau_k$. In particular, if the thresholds are non-adaptive, meaning that they are decided in advance, the following is a generic description of an algorithm. The competitive ratio of the following algorithm is defined as $\frac{\mathrm{E}[Y]}{\mathit{OPT}}$.

Algorithm Prophet Secretary

Input: A set of distributions $\{D_1, \ldots, D_n\}$; a randomly permuted stream of numbers $(X_{\pi_1}, \ldots, X_{\pi_n})$ drawn from the corresponding distributions.

Output: A number Y.

1. Let $\langle \tau_1, \ldots, \tau_n \rangle$ be a sequence of thresholds.
2. For $k \leftarrow 1$ to n
   (a) If $X_{\pi_k} \geq \tau_k$ then let $Y = X_{\pi_k}$ and exit the For loop.
3. Output Y as the solution.

Recall that when the arrival order is adversarial, the classical prophet inequality states that by choosing the same threshold $OPT/2$ for every step, one achieves the tight competitive ratio of $1/2$. On the other hand, for the basic secretary problem where the distributions are not known, the optimal strategy is to let $\tau_1 = \cdots = \tau_{\frac{n}{e}} = \infty$ and $\tau_{\frac{n}{e}+1} = \cdots = \tau_n = \max(X_{\pi_1}, \ldots, X_{\pi_{\frac{n}{e}}})$. This leads to the optimal competitive ratio of $\frac{1}{e} \simeq 0.36$. Hence, our goal in the prophet secretary problem is to beat the $1/2$ barrier.

We first show that unlike the prophet inequality setting, one cannot obtain the optimal competitive ratio by using a single uniform threshold. Indeed, Theorem 2.11 in Section2.2.4 shows that $1/2$ is the best competitive ratio one can achieve with uniform

thresholds. To beat the $\frac{1}{2}$ barrier, as a warm up we first show that by using two thresholds one can achieve the competitive ratio of $5/9 \simeq 0.55$. This can be achieved by choosing the threshold $\frac{5}{9} \cdot \mathit{OPT}$ for the first half of the steps and then decreasing the threshold to $\frac{\mathit{OPT}}{3}$ for the second half of the steps. Later in Section 2.2.6, we show that by setting n distinct thresholds one can obtain almost-tight $(1 - 1/e \approx 0.63)$-competitive ratio for the prophet secretary problem.

Theorem 2.8. *Let $\langle \tau_1, \ldots, \tau_n \rangle$ be a non-increasing sequence of n thresholds, such that (i) $\tau_k = \alpha_k \cdot OPT$ for every $k \in [n]$; (ii) $\alpha_n = \frac{1}{n+1}$; and (iii) $\alpha_k = \frac{n\alpha_{k+1}+1}{n+1}$ for $k \in [n-1]$. The competitive ratio of Algorithm* Prophet Secretary *invoked by thresholds τ_k's is at least α_1. When n goes to infinity, α_1 quickly converges to $1 - 1/e \approx 0.63$.*

The crux of the analysis of our algorithm is to compute the probability of picking a value x at a step of the algorithm with respect to the threshold factors α_k's. Indeed one source of difficulty arises from the fundamental dependency between the steps: for any step k, the fact that the algorithm has not stopped in the previous steps leads to various restrictions on what we expect to see at the step k. For example, consider the scenario that D_1 is 1 with probability one and D_2 is either 2 or 0 with equal probabilities. Now if the algorithm chooses $\tau_1 = 1$, then it would never happen that the algorithm reaches step two and receives a number drawn from D_2! That would mean we have received a value from

D_1 at the first step which is a contradiction since we would have picked that number. In fact, the optimal strategy for this example is to shoot for D_2! We set $\tau_1 = 2$ so that we can ignore the first value in the event that it is drawn from D_1. Then we set $\tau_2 = 1$ so that we can always pick the second value. Therefore in expectation we get $5/4$ which is slightly less than $\mathit{OPT} = 6/4$.

To handle the dependencies between the steps, we first distinguish between the events for $k \in [n]$ that we pick a value between τ_{k+1} and τ_k. We show that the expected value we pick at such events is indeed highly dependent on $\theta(k)$, the probability of passing the first k elements. We then use this observation to analyze competitive ratio with respect to $\theta(k)$'s and the thresholds factors α_k's. We finally show that the competitive ratio is indeed maximized by choosing the threshold factors described in Theorem 2.8. In Section 2.2.5, we first prove the theorem for the simple case of $n = 2$. This enables us to demonstrate our techniques without going into the more complicated dependencies for general n. We then present the full proof of Theorem 2.8 in Section 2.2.6.

As mentioned before, Bayesian optimal mechanism design problems provide a compelling application of prophet inequalities in economics. In such a Bayesian market, we have a set of n agents with private types sampled from (not necessary identical) known distributions. Upon receiving the reported types, a seller has to allocate resources and charge prices to the agents. The goal is to maximize the seller's revenue in equilibrium. Chawla et al. [CHMS10] pioneered the study the approximability of a special class of such mechanisms, *sequential posted pricing* (SPM): the seller makes a sequence

of take-it-or-leave-it offers to agents, offering an item for a specific price. They show although simple, SPMs approximate the optimal revenue in many different settings. Therefore prophet inequalities directly translate to approximation factors for the seller's revenue in these settings through standard machineries. Indeed one can analyze the so-called *virtual values* of winning bids introduced by Roger Myerson [Mye79], to prove via prophet inequalities that the expected virtual value obtained by the SPM mechanism approximates an offline optimum that is with respect to the exact types. Chawla et al. [CHMS10] provide a type of prophet inequality in which one can choose the ordering of agents. They show that under matroid feasibility constraints, one can achieve a competitive ratio of 0.5 in this model, and no algorithm can achieve a ratio better 0.8. Kleinberg and Weinberg [KW12] later improved there result by giving an algorithm with the tight competitive ratio of 0.5 for an adversarial ordering. Our result can be seen as improving their approximation guarantees to 0.63 for the case of single-item SPMs when the order of agents are chosen randomly.

On the other hand, from the negative side we prove in Section 2.2.7 that no online algorithm for the prophet secretary problem can achieve a competitive ratio 0.73.

Theorem 2.9. *For any arbitrary small positive number ϵ, there is no online algorithm for the prophet secretary problem with competitive ratio $\frac{11}{15} + \epsilon \approx 0.73 + \epsilon$.*

We also consider the minimization variants of the prophet inequality problem. In

particular, in Section 2.2.8.1 we show that, even for the simple case in which numbers are drawn from *identical and independent distributions (i.i.d.)*, there is no constant competitive online algorithm for the minimization variants of the prophet inequality and prophet secretary problems.

Theorem 2.10. *The competitive ratio of any online algorithm for the minimization prophet inequality with n identical and independent distribution is bounded by $\frac{(1.11)^n}{6}$.*

2.2.3 Preliminaries

We first define a few notations. For every $k \in [n]$, let z_k denote the random variable that shows the value we pick at the k^{th} step. Observe that for a fixed sequence of drawn values and a fixed permutation, at most one of z_k's is non-zero since we only pick one number. Let z denote the value chosen by the algorithm. By definition, $z = \sum_{k=1}^{n} z_k$. In fact, since all but one of z_k's are zero, we have the following proposition. We note that since the thresholds are deterministic, the randomness comes from the permutation π and the distributions.

Proposition 2.2. $\Pr[z \geq x] = \sum_{k\in[n]} \Pr[z_k \geq x]$.

For every $k \in [n]$, let $\theta(k)$ denote the probability that Algorithm Prophet Secretary does not choose a value from the first k steps. For every $i \in n$ and $k \in [n-1]$, let $q_{-i}(k)$ denote the probability that the following two events concurrently happen:

1. Algorithm Prophet Secretary does not choose a value from the first k elements.

2. None of the first k values are drawn from D_i.

Proposition 2.3. *If the thresholds of Algorithm* Prophet Secretary *are non-increasing, then for every $i \in [n]$ and $k \in [n-1]$, we have $\theta(k+1) \leq q_{-i}(k)$.*

Proof. In what fallows, let $i \in [n]$ be a fixed value. The claim is in fact very intuitive: $q_{-i}(k)$ is the probability of the event that the algorithm passes k values chosen from all distributions but D_i. On the other hand, $\theta(k+1)$ corresponds to the event that the algorithm passes $k+1$ values chosen from all distributions. Intuitively, in the latter we have surely passed k values chosen from all but D_i. Therefore $\theta(k+1)$ cannot be more than $q_{-i}(k)$.

Formalizing the intuition above, however, requires an exact formulation of the probabilities. For a permutation s of size k of $[n]$, let $s(j)$, for $j \in [k]$, denote the number at position j of s. For $k \in [n]$, let $S(k)$ denote the set of permutations of size k of $[n]$. Similarly, let $S_{-i}(k)$ denote the set of permutations of size k of $[n]\backslash\{i\}$. Observe that

$$|S(k)| = \frac{n!}{(n-k)!} \qquad\qquad |S_{-i}(k)| = \frac{(n-1)!}{(n-1-k)!}$$

In particular, we note that $|S(k+1)| = n|S_{-i}(k)|$. We can now write down the exact formula for $q_{-i}(k)$ and $\theta(k+1)$.

$$\theta(k+1) = \frac{1}{|S(k+1)|} \sum_{s \in S(k+1)} \prod_{j \in [k+1]} \Pr\left[X_{s(j)} < \tau_j\right] \tag{2.1}$$

$$q_{-i}(k) = \frac{1}{|S_{-i}(k)|} \sum_{s \in S_{-i}(k)} \prod_{j \in [k]} \Pr\left[X_{s(j)} < \tau_j\right] \tag{2.2}$$

We now show that $\theta(k+1)$ can be written down as a convex combination of terms that are less than $q_{-i}(k)$. For every $\ell \in [k]$, let $S(k+1,\ell) = \{s \in S(k+1) | s(\ell) = i\}$. We have:

$$\begin{aligned}
&\sum_{s:S(k+1,\ell)} \prod_{j\in[k+1]} \Pr\left[X_{s(j)} \leq \tau_j\right] \\
&\leq \sum_{s:S(k+1,\ell)} \prod_{j\in[k+1]\setminus\{\ell\}} \Pr\left[X_{s(j)} \leq \tau_j\right] \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Pr\left[X_i < \tau_\ell\right] \leq 1 \\
&\leq \sum_{s:S(k+1,\ell)} \prod_{j\in[k]\setminus\{\ell\}} \Pr\left[X_{s(j)} \leq \tau_j\right] \times \Pr\left[X_{s(k+1)} \leq \tau_\ell\right] \\
&\qquad\qquad\qquad\qquad\qquad\qquad \Pr\left[X_{s(k+1)} \leq \tau_{k+1}\right] \leq \Pr\left[X_{s(k+1)} \leq \tau_\ell\right] \textit{ since } \tau_{k+1} \leq \tau_\ell \\
&= \sum_{s:S_{-i}(k)} \prod_{j\in[k]} \Pr\left[X_{s(j)} \leq \tau_j\right] \\
&= |S_{-i}(k)| q_{-i}(k) \qquad (2.3) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{By Eq. 2.2}
\end{aligned}$$

Eq. 2.3 establishes the relation to $q_{-i}(k)$ for members of $S(k+1)$ that contain i in one of the first k positions.

Let $\overline{S(k+1)} = S(k+1) \setminus \bigcup_{\ell \in [k]} S(k+1, \ell)$.

$$\begin{aligned}
&\sum_{s:\overline{S(k+1)}} \prod_{j \in [k+1] \setminus \{\ell\}} \Pr\left[X_{s(j)} \le \tau_j\right] \\
&= \sum_{s:\overline{S(k+1)}} \prod_{j \in [k] \setminus \{\ell\}} \Pr\left[X_{s(j)} \le \tau_j\right] \times \Pr\left[X_{s(k+1)} \le \tau_{k+1}\right] \\
&\le (n-k) \sum_{s:S_{-i}(k+1)} \prod_{j \in [k] \setminus \{\ell\}} \Pr\left[X_{s(j)} \le \tau_j\right] && \Pr\left[X_{s(k+1)} \le \tau_{k+1}\right] \le 1 \\
&= (n-k)|S_{-i}(k)|q_{-i}(k) && (2.4)
\end{aligned}$$

Finally by Eq. 2.1 we have

$$\begin{aligned}
\theta(k+1) &= \frac{1}{|S(k+1)|} \sum_{s \in S(k+1)} \prod_{j \in [k+1]} \Pr\left[X_{s(j)} < \tau_j\right] \\
&= \frac{1}{|S(k+1)|} \left(\sum_{\ell \in [k]} \sum_{s \in S(k+1,\ell)} \prod_{j \in [k+1]} \Pr\left[X_{s(j)} < \tau_j\right] \right. \\
&\quad \left. + \sum_{s \in \overline{S(k+1)}} \prod_{j \in [k+1]} \Pr\left[X_{s(j)} < \tau_j\right] \right) \\
&\le \frac{1}{|S(k+1)|} \left(k|S_{-i}(k)|q_{-i}(k) + (n-k)|S_{-i}(k)|q_{-i}(k)\right) && \textit{By Eq. 2.3 and 2.4} \\
&= q_{-i}(k) && |S(k+1)| = n|S_{-i}(k)|
\end{aligned}$$

□

2.2.4 One Threshold Cannot Break $\frac{1}{2}$ Barrier for Prophet Secretary

To illustrate that considering at least 2 thresholds is necessary to beat $\frac{1}{2}$ barrier for the prophet secretary problem, we first give an example that shows achieving better than

$\frac{1}{2}$-competitive ratio for any online algorithm that uses only one threshold for the prophet secretary problem is not possible.

Theorem 2.11. *There is no online algorithm for the prophet secretary problem that uses one threshold and can achieve competitive ratio better than* $0.5 + \frac{1}{2n}$.

Proof. Suppose we have $n + 1$ distributions where the first n distributions always gives $\frac{1}{1-1/n}$ and the $(n+1)^{th}$ distribution gives n with probability $\frac{1}{n}$ and gives 0 with probability $1 - \frac{1}{n}$. Therefore, with probability $\frac{1}{n}$, the maximum is n, and with probability $1 - \frac{1}{n}$, the maximum is $\frac{1}{1-1/n}$. Thus, the expected outcome of the offline optimum algorithm is $\frac{1}{n} \times n + (1 - \frac{1}{n}) \times \frac{1}{1-1/n} = 2$.

Now, suppose we have an online algorithm that uses one threshold, say T for a number T, that is the online algorithm accepts the first number greater or equal to a threshold T. We consider two cases for T. The first case is if $T > \frac{1}{1-1/n}$ for which the algorithm does not accept $\frac{1}{1-1/n}$ and thus, the expected outcome of such an algorithm is $\frac{1}{n} \times n = 1$.

The second case is if $T \leq \frac{1}{1-1/n}$. Observe that, with probability $\frac{n}{n+1}$, the first number is $\frac{1}{1-1/n}$ and the online algorithm accepts it. And, with probability $\frac{1}{n+1}$, the distribution that gives n with probability $\frac{1}{n}$ will be the first and the outcome of the algorithm is 1. Thus, the expected outcome of the online algorithm is

$$\frac{n}{n+1} \times \frac{1}{1-1/n} + \frac{1}{n+1} \times 1 = \frac{n^2}{n^2-1} + \frac{1}{n+1} \leq 1 + \frac{1}{n}.$$

Therefore, the competitive ratio of the online algorithms that uses only one threshold is bounded by $\frac{1+1/n}{2} = 0.5 + \frac{1}{2n}$. □

2.2.5 Two Thresholds Breaks $\frac{1}{2}$ Barrier

Since using one threshold is hopeless, we now try using two thresholds. More formally, for the first half of steps, we use a certain threshold, and then we use a different threshold for the rest of steps. We note that similar to the one-threshold algorithm, both thresholds should be proportional to *OPT*. Furthermore, at the beginning we should be optimistic and try to have a higher threshold, but if we cannot pick a value in the first half, we may need to lower the bar! We show that by using two thresholds one can indeed achieve the competitive ratio of $\frac{5}{9} \simeq 0.55$. In fact, this improvement beyond $1/2$ happens even at $n = 2$. Thus as a warm up before analyzing the main algorithm with n thresholds, we focus on the case of $n = 2$.

Let $\tau_1 = \alpha_1 \mathit{OPT}$ and $\tau_2 = \alpha_2 \mathit{OPT}$ for some $1 \geq \alpha_1 \geq \alpha_2 \geq 0$ to be optimized later. Recall that z_1 and z_2 are the random variables showing the values picked up by the algorithm at step one and two, respectively. We are interested in comparing $\mathrm{E}[z]$ with *OPT*. By Proposition 2.2 we have

$$\mathrm{E}[z] = \int_0^\infty \Pr[z \geq x]\, dx = \int_0^\infty \Pr[z_1 \geq x]\, dx + \int_0^\infty \Pr[z_2 \geq x]\, dx$$

Observe that z_1 (resp. z_2) is either zero or has a value more than τ_1 (resp. τ_2). In fact, since $\tau_1 \geq \tau_2$, z is either zero or has a value more than τ_2. Recall that $\theta(1)$ is the probability of $z_1 = 0$ while $\theta(2)$ is the probability of $z_1 = z_2 = 0$. This observation leads

to the following simplification:

$$\begin{aligned}
\mathrm{E}[z] &= \int_0^{\tau_2} \Pr[z_1 \geq x]\,dx + \int_{\tau_2}^{\tau_1} \Pr[z_1 \geq x]\,dx + \int_{\tau_1}^{\infty} \Pr[z_1 \geq x]\,dx \\
&\quad + \int_0^{\tau_2} \Pr[z_2 \geq x]\,dx + \int_{\tau_2}^{\infty} \Pr[z_2 \geq x]\,dx \\
&= \int_0^{\tau_2} \Pr[z \geq x]\,dx + \int_{\tau_2}^{\tau_1} \Pr[z_1 \geq x]\,dx \\
&\quad + \int_{\tau_1}^{\infty} \Pr[z_1 \geq x]\,dx + \int_{\tau_2}^{\infty} \Pr[z_2 \geq x]\,dx \\
&= \tau_2(1-\theta(2)) + (\tau_1 - \tau_2)(1-\theta(1)) + \int_{\tau_1}^{\infty} \Pr[z_1 \geq x]\,dx + \int_{\tau_2}^{\infty} \Pr[z_2 \geq x]\,dx
\end{aligned}$$

Let us first focus on $\Pr[z_1 \geq x]$. The first value may come from any of the two distributions, thus we have

$$\Pr[z_1 \geq x] = \frac{1}{2}\Pr[X_1 \geq x] + \frac{1}{2}\Pr[X_2 \geq x]$$

On the other hand, z_2 is non-zero only if we do not pick anything at the first step. For $i \in \{1, 2\}$, we pick a value of at least x drawn from D_i at step two, if and only if: (i) the value drawn from D_i is at least x; and (ii) our algorithm does not pick a value from the previous step which is drawn from the other distribution. By definitions, the former happens with probability $\Pr[X_i \geq x]$, while the latter happens with probability $q_{-i}(1)$. Since these two events are independent we have

$$\Pr[z_2 \geq x] = \frac{1}{2}\sum_{i \in \{1,2\}} q_{-i}(1)\Pr[X_i \geq x] \geq \frac{\theta(2)}{2}\sum_i \Pr[X_i \geq x]$$

where the last inequality follows from Proposition 2.3, although the proposition is trivial

for $n = 2$. We can now continue analyzing $\mathrm{E}[z]$ from before:

$$\begin{aligned}\mathrm{E}[z] &= \tau_2(1-\theta(2)) + (\tau_1-\tau_2)(1-\theta(1)) + \int_{\tau_1}^{\infty} \Pr[z_1 \geq x]\,dx + \int_{\tau_2}^{\infty} \Pr[z_2 \geq x]\,dx \\ &\geq \tau_2(1-\theta(2)) + (\tau_1-\tau_2)(1-\theta(1)) \\ &\quad + \frac{\theta(1)}{2}\int_{\tau_1}^{\infty} \sum_i \Pr[X_i \geq x]\,dx + \frac{\theta(2)}{2}\int_{\tau_2}^{\infty} \sum_i \Pr[X_i \geq x]\,dx\end{aligned}$$

We note that although the $\theta(1)$ factor is not required in the third term of the last inequality, we include it so that the formulas can have the same formation as in the general formula of the next sections.

It remains to bound $\int_{\tau_k}^{\infty} \sum_i \Pr[X_i \geq x]$ for $k \in \{1,2\}$. Recall that $\mathit{OPT} = \mathrm{E}[\max_i X_i]$. Hence for every $k \in \{1,2\}$ we have

$$\begin{aligned}\mathit{OPT} &= \int_0^{\tau_k} \Pr[\max X_i \geq x]\,dx + \int_{\tau_k}^{\infty} \Pr[\max X_i \geq x] \\ &\leq \tau_k + \int_{\tau_k}^{\infty} \Pr[\max X_i \geq x] && \Pr[\max X_i \geq x] \leq 1\end{aligned}$$

$$\begin{aligned}(1-\alpha_k)\mathit{OPT} &\leq \int_{\tau_k}^{\infty} \Pr[\max X_i \geq x] && \tau_k = \alpha_k \mathit{OPT} \\ &\leq \int_{\tau_k}^{\infty} \sum_i \Pr[X_i \geq x]\,dx && \Pr[\max X_i \geq x] \leq \sum_i \Pr[X_i \geq x]\,dx\end{aligned}$$

Therefore we get

$$\begin{aligned}
\mathrm{E}[z] &\geq \tau_2(1-\theta(2)) + (\tau_1-\tau_2)(1-\theta(1)) \\
&\quad + \frac{\theta(1)}{2}\int_{\tau_1}^{\infty}\sum_i \Pr[X_i \geq x]\,dx + \frac{\theta(2)}{2}\int_{\tau_2}^{\infty}\sum_i \Pr[X_i \geq x]\,dx \\
&\geq (\alpha_2 \mathit{OPT})(1-\theta(2)) + (\alpha_1-\alpha_2)\mathit{OPT}(1-\theta(1)) \\
&\quad + \frac{\theta(1)}{2}(1-\alpha_1)\mathit{OPT} + \frac{\theta(2)}{2}(1-\alpha_2)\mathit{OPT} \\
&= \mathit{OPT}\left(\alpha_1 + \theta_1(\frac{1+2\alpha_2-3\alpha_1}{2}) + \theta_2(\frac{1-3\alpha_2}{2})\right)
\end{aligned}$$

Therefore by choosing $\alpha_2 = 1/3$ and $\alpha_1 = 5/9$, the coefficients of θ_1 and θ_2 become zero, leading to the competitive ratio of $5/9 \simeq 0.55$. In the next section, we show how one can generalize the arguments to the case of n thresholds for arbitrary n.

2.2.6 $(1-\frac{1}{e} \approx 0.63)$-Competitive Ratio Using n Thresholds

In this section we prove our main theorem. In particular, we invoke Algorithm Prophet Secretary with n distinct thresholds $\tau_1, \ldots, \tau_n$. The thresholds $\tau_1, \ldots, \tau_n$ that we consider are *non-adaptive* (i.e., Algorithm Prophet Secretary is oblivious to the history) and *non-increasing*. Intuitively, this is because as we move to the end of stream we should be more pessimistic and use lower thresholds to catch remaining higher values.

Formally, for every $k \in [n]$, we consider threshold $\tau_k = \alpha_k \cdot OPT$ where the sequence $\alpha_1, \ldots, \alpha_n$ is non-increasing that is, $\alpha_1 \geq \alpha_1 \geq \ldots \geq \alpha_n$. We invoke Algorithm Prophet Secretary with these thresholds and analyze the competitive ratio of Algorithm Prophet Secretary with respect to coefficients α_k. Theorem 2.8 shows that there exists

a sequence of coefficients α_k that leads to the competitive ratio of $(1 - 1/e) \approx 0.63$.

Proof of Theorem 2.8.

We prove the theorem in two steps: First, we find a lower bound on $\mathrm{E}[z]$ in terms of OPT and coefficients α_i. Second, we set coefficients α_k so that (1) α_1 becomes the competitive ratio of Algorithm Prophet Secretary and (2) α_1 converges to $1-1/e$, when n goes to infinity.

Next we give the proof of the theorem in details. Here we first prove few auxiliary lemmas.

In the first lemma, we find a lower bound for $\int_{\tau_k}^{\infty} \Pr[\max X_i \geq x]\, dx$ based on $\mathit{OPT} = \mathrm{E}[\max_i X_i]$.

Lemma 2.2. $\int_{\tau_k}^{\infty} \Pr[\max X_i \geq x]\, dx \geq (1 - \alpha_k)\mathit{OPT}$.

Proof. For an arbitrary $k \in [n]$ and since $\Pr[\max X_i \geq x] \leq 1$, we have

$$\begin{aligned}
\mathit{OPT} = \mathrm{E}\left[\max_i X_i\right] &= \int_0^{\infty} \Pr[\max X_i \geq x]\, dx \\
&= \int_0^{\tau_k} \Pr[\max X_i \geq x]\, dx + \int_{\tau_k}^{\infty} \Pr[\max X_i \geq x]\, dx \\
&\leq \tau_k + \int_{\tau_k}^{\infty} \Pr[\max X_i \geq x]\, dx.
\end{aligned}$$

Since, by definition $\tau_k = \alpha_k \cdot \mathit{OPT}$, we have $(1 - \alpha_k) \cdot \mathit{OPT} \leq \int_{\tau_k}^{\infty} \Pr[\max X_i \geq x]\, dx$. □

In the next lemma we split $\mathrm{E}[z]$ into two terms. Later, we find lower bounds for each one of these terms based on $\mathit{OPT} = \mathrm{E}[\max_i X_i]$ separately.

Lemma 2.3. *Let $z = \sum_{k=1}^{n} z_k$ denote the value chosen by Algorithm* Prophet Secretary. *For z we have*

$$\mathrm{E}[z] = \sum_{k=1}^{n} \int_0^{\tau_k} \Pr[z_k \geq x]\, dx + \sum_{k=1}^{n} \int_{\tau_k}^{\infty} \Pr[z_k \geq x]\, dx.$$

Proof. By Proposition 2.2 we have

$$\begin{aligned}\mathrm{E}[z] &= \int_0^{\infty} \Pr[z \geq x]\, dx = \sum_{k=1}^{n} \int_0^{\infty} \Pr[z_k \geq x]\, dx \\ &= \sum_{k=1}^{n} \int_0^{\tau_k} \Pr[z_k \geq x]\, dx + \sum_{k=1}^{n} \int_{\tau_k}^{\infty} \Pr[z_k \geq x]\, dx,\end{aligned}$$

where we use this fact that $z = \sum_{k=1}^{n} z_k$ because we only pick one number for a fixed sequence of drawn values and a fixed permutation and therefore, at most one of z_k's is non-zero. □

Now, we find a lower bound for the first term of $\mathrm{E}[z]$ in Lemma 2.3.

Lemma 2.4. $\sum_{k=1}^{n} \int_0^{\tau_k} \Pr[z_k \geq x]\, dx \geq \mathit{OPT} \sum_{k=1}^{n} (1 - \theta(k))(\alpha_k - \alpha_{k+1})$.

Proof. Suppose $x \leq \tau_k$. Observe that event $z_k \geq x$ occurs when Algorithm Prophet Secretary chooses a value at step k. In fact, since the thresholds are non-increasing, whatever we pick at the first k steps would be at least x. Recall that for every $k \in [n]$, $\theta(k)$ is the probability that Algorithm Prophet Secretary does not choose a value from the first k steps. Hence, for every $k \in [n]$ and $x \leq \tau_k$ we have

$$\sum_{j \leq k} \Pr[z_j \geq x] = 1 - \theta(k). \tag{2.5}$$

To simplify the notation, we assume that $\alpha_0 = \infty$ which means $\tau_0 = \infty$ and we let $\alpha_{n+1} = 0$ which means $\tau_{n+1} = 0$. Therefore we have

$$\sum_{k=1}^{n} \int_{0}^{\tau_k} \Pr\left[z_k \geq x\right] dx = \sum_{k=1}^{n} \int_{\tau_{n+1}}^{\tau_k} \Pr\left[z_k \geq x\right] dx.$$

Next, we use Equation (2.5) to prove the lemma as follows.

$$\begin{aligned}
\sum_{k=1}^{n} \int_{0}^{\tau_k} \Pr\left[z_k \geq x\right] dx &= \sum_{k=1}^{n} \int_{\tau_{n+1}}^{\tau_k} \Pr\left[z_k \geq x\right] dx = \sum_{k=1}^{n} \sum_{r=k}^{n} \int_{\tau_{r+1}}^{\tau_r} \Pr\left[z_k \geq x\right] dx \\
&= \sum_{r=1}^{n} \int_{\tau_{r+1}}^{\tau_r} \sum_{k=1}^{r} \Pr\left[z_k \geq x\right] dx \geq \sum_{r=1}^{n} \int_{\tau_{r+1}}^{\tau_r} (1 - \theta(r)) dx \\
&= \sum_{r=1}^{n} (1 - \theta(r))(\tau_r - \tau_{r+1}) = \mathit{OPT} \cdot \sum_{k=1}^{n} (1 - \theta(k))(\alpha_k - \alpha_{k+1})
\end{aligned}$$

□

Then, we find a lower bound for the second term of $\mathrm{E}\left[z\right]$ in Lemma 2.3.

Lemma 2.5. $\sum_{k=1}^{n} \int_{\tau_k}^{\infty} \Pr\left[z_k \geq x\right] dx \geq \mathit{OPT} \sum_k \frac{\theta(k)}{n} (1 - \alpha_k)$.

Proof. Recall that for every distribution D_i we draw a number X_i. Later, we randomly permute numbers $X_1, \cdots, X_n$. Let the sequence of indices after random permutation be $\pi_1, \ldots, \pi_n$ that is, at step k, number X_{π_k} for $\pi_k \in [n]$ is revealed.

Suppose $x \geq \tau_k$. We break the event $z_k > 0$ to n different scenarios depending on which index of the distributions $D_1, \cdots, D_n$ is mapped to index π_k in the random permutation. Let us consider the scenario in which Algorithm Prophet Secretary chooses the value drawn from a distribution i at step k. Such a scenario happens if (i) Algorithm

Prophet Secretary does not choose a value from the first $k-1$ steps which are not drawn from i, and (ii) $X_i \geq \tau_k$. Observe that the two events are independent. Therefore, we have $\Pr[z_k \geq x] = \sum_i \Pr[\pi_k = i] \cdot \Pr[X_i \geq x] \cdot q_{-i}(k-1)$, where $q_{-i}(k)$ for every $i \in n$ and $k \in [n-1]$ is the probability that the following two events concurrently happen: (i) Algorithm Prophet Secretary does not choose a value from the first k elements, and (ii) none of the first k values are drawn from D_i. Since π_k is an index in the random permutation we obtain

$$\begin{aligned}\Pr[z_k \geq x] &= \sum_i \Pr[\pi_k = i] \cdot \Pr[X_i \geq x] \cdot q_{-i}(k-1) \\ &= \frac{1}{n} \cdot \sum_i \Pr[X_i \geq x] \cdot q_{-i}(k-1)\end{aligned}$$

Using Proposition 2.3 and an application of the union bound we then have

$$\begin{aligned}\Pr[z_k \geq x] &= \sum_i \Pr[\pi_k = i] \cdot \Pr[X_i \geq x] \cdot q_{-i}(k-1) \\ &= \frac{1}{n} \sum_i \Pr[X_i \geq x] \cdot q_{-i}(k-1) \\ &\geq \frac{\theta(k)}{n} \cdot \sum_i \Pr[X_i \geq x] \\ &\geq \frac{\theta(k)}{n} \cdot \Pr\left[\max_i X_i \geq x\right]\end{aligned}$$

Therefore, we obtain the following lower bound on $\sum_{k=1}^{n} \int_{\tau_k}^{\infty} \Pr[z_k \geq x]\, dx$.

$$\sum_{k=1}^{n} \int_{\tau_k}^{\infty} \Pr\left[z_k \geq x\right] dx \geq \sum_{k} \int_{\tau_k}^{\infty} \frac{\theta(k)}{n} \Pr\left[\max X_i \geq x\right] dx$$
$$= \sum_{k} \frac{\theta(k)}{n} \int_{\tau_k}^{\infty} \Pr\left[\max X_i \geq x\right] dx$$

Finally, we use the lower bound of Lemma 2.2 for $\int_{\tau_k}^{\infty} \Pr\left[\max X_i \geq x\right] dx$ to prove the lemma.

$$\sum_{k=1}^{n} \int_{\tau_k}^{\infty} \Pr\left[z_k \geq x\right] dx \geq \sum_{k} \frac{\theta(k)}{n} \int_{\tau_k}^{\infty} \Pr\left[\max X_i \geq x\right] dx$$
$$\geq \sum_{k} \frac{\theta(k)}{n} \cdot (1 - \alpha_k) \cdot \mathit{OPT}$$
$$= \mathit{OPT} \cdot \sum_{k} \frac{\theta(k)}{n} \cdot (1 - \alpha_k)$$

□

We use the lower bounds of Lemmas 2.4 and 2.5 in Lemma 2.3 to obtain a lower bound for $\mathrm{E}\left[z\right]$.

Lemma 2.6. *Let $z = \sum_{k=1}^{n} z_k$ denote the value chosen by Algorithm* Prophet Secretary. *For z we have*

$$\mathrm{E}\left[z\right] \geq \mathit{OPT} \cdot (\alpha_1 + \sum_{k=1}^{n} \theta(k)(\frac{1}{n} - \frac{\alpha_k}{n} - \alpha_k + \alpha_{k+1})).$$

Proof. By using the lower bounds of Lemmas 2.4 and 2.5 for the terms of Lemma 2.3 we

have

$$\begin{aligned}\mathrm{E}[z] &\geq \mathit{OPT}\sum_{k=1}^{n}((1-\theta(k))(\alpha_k-\alpha_{k+1})+\frac{\theta(k)}{n}(1-\alpha_k))\\ &= \mathit{OPT}(\alpha_1+\sum_{k=1}^{n}\theta(k)(\frac{1}{n}-\frac{\alpha_k}{n}-\alpha_k+\alpha_{k+1})).\end{aligned}$$

□

We finish the theorem by proving the following claim.

Lemma 2.7. *The competitive ratio of Algorithm* Prophet Secretary *is* α_1 *which quickly converges to* $1-1/e\approx 0.63$ *when* n *goes to infinity.*

Proof. Using Lemma 2.6, for z we have

$$\mathrm{E}[z] \geq \mathit{OPT}\left(\alpha_1+\sum_{k=1}^{n}\theta(k)\left(\frac{1}{n}-\frac{\alpha_k}{n}-\alpha_k+\alpha_{k+1}\right)\right)$$

which means that the competitive ratio depends on the probabilities $\theta(k)$'s. However, we can easily get rid of the probabilities $\theta(k)$'s by choosing α_k's such that for every k, $\left(\frac{1}{n}-\frac{\alpha_k}{n}-\alpha_k+\alpha_{k+1}\right)=0$.

More formally, by starting from $\alpha_{n+1}=0$ and choosing $\alpha_k=\frac{1+n\alpha_{k+1}}{1+n}$ for $k\leq n$, the competitive ratio of the algorithm would be α_1. In below, we show that when n goes to infinity, α_1 quickly goes to $1-1/e$ which means that the competitive ratio of Algorithm Prophet Secretary converges to $1-1/e\approx 0.63$.

First, by the induction we show that $\alpha_k=\sum_{i=0}^{n+1-k}\frac{n^i}{(1+n)^{i+1}}$. For the base case we have

$$\alpha_n = \frac{1 + n\alpha_{n+1}}{1+n} = \frac{1 + n \times 0}{1+n} = \frac{n^0}{(1+n)^1}.$$

Given $\alpha_{k+1} = \sum_{i=0}^{n+1-(k+1)} \frac{n^i}{(1+n)^{i+1}}$ we show the equality for α_k as follows.

$$\alpha_k = \frac{1 + n\alpha_{k+1}}{1+n} = \frac{1 + n \times \alpha_{k+1}}{1+n} = \frac{1 + n(\sum_{i=0}^{n+1-(k+1)} \frac{n^i}{(1+n)^{i+1}})}{1+n}$$
$$= \frac{n^0}{(1+n)^1} + \sum_{i=0}^{n+1-(k+1)} \frac{n^{i+1}}{(1+n)^{i+2}} = \sum_{i=0}^{n+1-k} \frac{n^i}{(1+n)^{i+1}}.$$

Now we are ready to show $\alpha_0 \geq 1 - 1/e$ when n goes to infinity.

$$\lim_{n\to\infty} \alpha_0 = \lim_{n\to\infty} \sum_{i=0}^{n+1} \frac{n^i}{(n+1)^{i+1}} = \lim_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n+1} (1 - \frac{1}{n+1})^i$$
$$\approx \lim_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n+1} e^{-i/n} \approx \int_0^1 e^{-x} dx = 1 - 1/e.$$

□

2.2.7 Lower Bounds for Prophet Secretary and Minimization Variants of Classical Stopping Theory Problems

In this section we give our lower bounds for the prophet secretary problem and the minimization variants of the prophet inequality and prophet secretary problems. First, in Section 2.2.8 we show that no online algorithm for the prophet secretary problem can achieve a competitive ratio of 0.73. Later, in Section 2.2.8.1 we consider the minimization variant of the prophet inequality problem. We show that, even for the simple case in

which numbers are drawn from *identical and independent distributions (i.i.d.)*, there is no constant competitive online algorithm for the minimization variants of the prophet inequality and prophet secretary problems.

2.2.8 0.73-Lower Bound of Prophet Secretary Problem

Here we first give a simple example which shows that no algorithm for the prophet secretary problem can achieve competitive ratio 0.75. Later, we improve this lower bound by giving an example for which no online algorithm for the prophet secretary problem can achieve a competitive ratio 0.73.

Lemma 2.8. *There is no online algorithm for the prophet secretary problem with competitive ratio* $0.75 + \epsilon$*, where* ϵ *is an arbitrary small positive number less than* 1.

Proof. It is known that no algorithm can guarantee a competitive ratio better than $0.5 + \epsilon$, where ϵ is an arbitrary small positive number less than 1. A hard example that shows this upper bound is as follow. We have two distributions. The first distribution always gives 1, and the second distribution gives $\frac{1}{\epsilon}$ with probability ϵ and 0 with probability $1 - \epsilon$. Observe that if we either accept the first number or reject it our expected outcome is 1. On the other hand, the offline optimum algorithm takes $\frac{1}{\epsilon}$ with probability ϵ and 1 with probability $1 - \epsilon$. Therefore, the expected outcome of the offline optimum algorithm is $\epsilon \cdot \frac{1}{\epsilon} + 1 \cdot (1 - \epsilon) = 2 - \epsilon$, and the competitive ratio is at most $\frac{1}{2-\epsilon} \leq 0.5 + \epsilon$.

The above example contains exactly two distributions. Thus, if the drawn numbers from these distributions arrive in a random order, with probability 0.5 the arrival order

is the worst case order. This means that in the prophet secretary, with probability 0.5 the expected outcome of any algorithm on this example is at most 1, while the offline optimum algorithm is always $2-\epsilon$. Therefore, there is no algorithm for the prophet secretary problem with competitive ratio better than $\frac{0.5\times 1+0.5\times(2-\epsilon)}{2-\epsilon} \leq 0.75+\epsilon$. □

Now we prove Theorem 2.9 that improves the above lower bound by showing that there is no algorithm for the prophet secretary problem with competitive ratio $\frac{11}{15}+\epsilon \approx 0.73+\epsilon$.

Proof of Theorem 2.9.

Suppose we have three distributions as follows. The first distribution always gives 1. The second distribution gives 2 with probability 0.5, and 2 with probability 0.5. The third distribution gives $\frac{1}{\epsilon}$ with probability ϵ and 0 with probability $1-\epsilon$.

We observe that with probability ϵ the maximum is $\frac{1}{\epsilon}$, with probability $\frac{(1-\epsilon)}{2}$ the maximum is 2 and with probability $\frac{(1-\epsilon)}{2}$ the maximum is 1. Thus, the expected outcome of the offline optimum algorithm is

$$\epsilon \times \frac{1}{\epsilon} + \frac{(1-\epsilon)}{2} \times 2 + \frac{(1-\epsilon)}{2} \times 1 = 2.5 - 1.5\epsilon.$$

Next we analyze the expected outcome of an online algorithm for the secretary problem. We have 6 permutations for three distributions 1, 2, and 3. Figure 2.1 shows the expected outcome of an online algorithm for the secretary problem for each permutation. Permutation ijk means that the draw from the i-th distribution appears the first, the draw from the j-th distribution appears after that, and draw from the k-th distribution appears

permutation	exp. outcome (expanded)	exp. outcome (closed)
123	$0.5 \times 2 + 0.5 \times \epsilon \times \frac{1}{\epsilon}$	1.5
132	$\epsilon \times \frac{1}{\epsilon} + (1-\epsilon) \times 0.5 \times 2$	$2 - \epsilon$
213	$0.5 \times 2 + 0.5 \times 1$	1.5
231	$0.5 \times 2 + 0.5 \times \epsilon \times \frac{1}{\epsilon} + 0.5 \times (1-\epsilon) \times 1$	$2 - \frac{\epsilon}{2}$
312	$\epsilon \times \frac{1}{\epsilon} + (1-\epsilon) \times 1$	$2 - \epsilon$
321	$\epsilon \times \frac{1}{\epsilon} + (1-\epsilon) \times 0.5 \times 2$	$2 - \epsilon$

Figure 2.1: Outcomes of an online algorithm for the secretary problem.

at the end. As we see in Figure 2.1, the expected outcome of any online algorithm for our example is at most $\frac{11+3.5\epsilon}{6}$. Thus, the competitive ratio of any online algorithm for the prophet secretary problem for $0 \leq \epsilon < \frac{2.5}{9}$ is bounded by

$$\frac{11+3.5\epsilon}{6 \times (2.5 - 1.5\epsilon)} = \frac{11+3.5\epsilon}{15-9\epsilon} = \frac{11-9\epsilon}{15-9\epsilon} + \frac{12.5\epsilon}{15-9\epsilon} < \frac{11}{15} + \epsilon.$$

2.2.8.1 Lower Bounds for Minimization Variants of Classical Stopping Theory Problems

In this section, we consider the minimization variant of the prophet inequality setting. First, we consider the simple case that numbers are drawn from identical and in-

dependent distributions. In particular, we prove Theorem 2.10 that shows, even for the simple case of identical and independent distributions, there is no constant competitive online algorithm for the minimization variant of the prophet inequality problem. Later, we give more power to the online algorithm and let it to change its decision once; we call this model *online algorithm with one exchange*. We show in Lemma 2.12 that there is no constant competitive online algorithm with one exchange for the minimization variants of the prophet inequality and prophet secretary problems.

Proof of Theorem 2.10.

Suppose we have n identical distributions, each gives 0 with probability $\frac{1}{3}$, 1 with probability $\frac{1}{3}$ and 2^n with probability $\frac{1}{3}$. One can see that with probability $(\frac{1}{3})^n$, all of the numbers are 2^n and thus the minimum number is 2^n. Also, with probability $(\frac{2}{3})^n - (\frac{1}{3})^n$, there is no 0 and there is at least one 1, and thus, the minimum is 1. In all the other cases, the minimum is 0. Therefore, the expected outcome of the offline optimum algorithm is $(\frac{1}{3})^n \times 2^n + (\frac{2}{3})^n - (\frac{1}{3})^n < \frac{2^{n+1}}{3^n}$.

For this example, without loss of generality we can assume that any online algorithm accept 0 as soon as it appears, and also it does not accept 2^n except for the last item. Assume $i+1$ is the first time that the algorithm is willing to accept 1. The probability that we arrive in this point is $(\frac{2}{3})^i$ and the probability that we see 1 at that point is $\frac{1}{3}$. On the other hand, the probability that such an algorithm does not accept anything up to the last number and sees that the last number is 2^n is at least $(\frac{2}{3})^i(\frac{1}{3})^{n-i}$. Therefore, the expected

outcome of the online algorithm is at least $(\frac{2}{3})^i\frac{1}{3} \times 1 + (\frac{2}{3})^i(\frac{1}{3})^{n-i} \times 2^n = \frac{2^i 3^{n-i-1}+3^i}{3^n}$. Thus, the competitive ratio is at least

$$\frac{2^i 3^{n-i-1} + 3^i}{2^{n+1}} = \frac{1}{6}(\frac{3^{n-i}}{2^{n-i}} + \frac{3^{i+1}}{2^n}).$$

If $i \leq 0.73n$, the left term is at least $\frac{(1.11)^n}{6}$; otherwise, the right term is at least $\frac{(1.11)^n}{6}$.

Theorem 2.12. *For any large positive number C, there is no C-competitive algorithm for minimization prophet inequality with one exchange.*

Proof. Suppose we have three distributions as follows. The first distribution always gives 1. The second distribution gives $\frac{1}{\epsilon}$ with probability ϵ and gives $\frac{\epsilon}{1-\epsilon}$ with probability $1-\epsilon$. The third distribution gives $\frac{1}{\epsilon}$ with probability ϵ and gives 0 with probability $1-\epsilon$. We set ϵ later.

We observe that the minimum number is 0 with probability $1-\epsilon$, is $\frac{\epsilon}{1-\epsilon}$ with probability $\epsilon(1-\epsilon)$ and is 1 with probability ϵ^2. Thus, the expected outcome of the optimum algorithm is

$$(1-\epsilon) \times 0 + \epsilon(1-\epsilon) \times \frac{\epsilon}{1-\epsilon} + \epsilon^2 \times 1 = 2\epsilon^2.$$

Now, we show that the outcome of any online algorithm with one exchange for this input is at least ϵ. In fact, this means that, the competitive ratio can not be less than $\frac{\epsilon}{2\epsilon^2} = \frac{1}{2\epsilon}$. If we set $\epsilon = \frac{1}{2C}$, this means that there is no C-competitive algorithm for the minimization prophet inequality with one exchange.

Recall that there is no uncertainty in the first number. If an algorithm withdraw the first number, it can take the outcome of either the second distribution or the third distribution. However, if we do this, with probability ϵ^2 the outcome is $\frac{1}{\epsilon}$. Thus, the expected outcome is at least ϵ as desired.

Now, we just need to show that if an algorithm does not select the first number, its expected outcome is at least ϵ. Observe that if this happens, then the algorithm has only the option of choosing either the second and the third distribution. We consider two cases. The first case is if the algorithm does not select the second number, then it must select the third number. Therefore, with probability ϵ, the outcome of the algorithm is 1 and thus, the expected outcome is at least ϵ. The second case occurs if the algorithm selects the second number and that is $\frac{\epsilon}{1-\epsilon}$. Therefore, with probability $1-\epsilon$, the outcome of the algorithm is $\frac{\epsilon}{1-\epsilon}$ and again the expected outcome is $\frac{\epsilon}{1-\epsilon} \times (1-\epsilon) = \epsilon$. □

Finally, we show that even for the minimization variant of the prophet secretary there is no hope to get a constant competitive ratio.

Corollary 2.3. *For any large number C, there is no C-competitive algorithm for minimization prophet secretary with one exchange.*

Proof. In Theorem 2.12, we have three distributions. Thus, in the prophet secretary model the worst case order happen with probability $\frac{1}{6}$. Thus, the competitive ratio can not be more than $\frac{\frac{1}{6}\cdot\epsilon}{2\epsilon^2} = \frac{1}{12\epsilon}$. If we set $\epsilon = \frac{1}{12C}$, this essentially means that there is no C-competitive algorithm for the minimization prophet secretary with one exchange. □

Corollary 2.4. *For any large number C there is no C-competitive algorithm for the minimization secretary problem with one exchange.*

Proof. Suppose for the sake of contradiction that there exists an algorithm Alg for the minimization secretary problem which is C-competitive. Consider all possible realizations of the example in Theorem 2.12. Algorithm Alg is C-competitive when each of the these realizations comes in a random order. Therefore, Algorithm Alg is C-competitive when the input is a distribution over these realizations. This says that Alg is a C-competitive algorithm for the minimization prophet secretary, which contradicts Corollary 2.3 and completes the proof. □

CHAPTER 3

Covering Problems

3.1 Node-weighted Network Design Problems

Steiner problems, where the goal is to find the minimum weight subgraph of a given (undirected) graph that satisfies a given set of connectivity requirements, form a fundamental class of optimization problems that have attracted substantial attention over the last few decades. The *Steiner tree* (ST) problem—which asks for the minimum weight subgraph connecting a given set of vertices called *terminals*—is perhaps the most representative problem in this class. This chapter deals with its well-studied generalization called the the *Steiner forest* (SF) problem where the connectivity requirement is represented by a set of vertex pairs called *terminal pairs* that need to be individually connected in the classical *online* model, i.e., the input graph is given offline but the terminal pairs arrive sequentially in online steps. The selected subgraph starts off as the empty subgraph, but has to be augmented to satisfy the new connectivity constraint in each online step.

We measure the performance of our algorithms using the classical notion of *competitive ratio*, i.e., the maximum ratio (over all input sequences) of the weight[1] of the algorithmic solution to that of the offline optimum.

Steiner problems have typically been studied in two weight models: the *edge-weighted* (EW) model and the *node-weighted* (NW) model, depending on whether the weight function is defined on the edges or the vertices respectively. The NW model is more general since an edge of weight w in the EW model can be replaced in the NW model by two edges connected by a node of weight w; therefore, algorithmic results in the NW model also apply to the EW model (see recent achievements on the offline topic in [BHL13] and the references therein).

The online ST problem was originally considered in the EW model, where Imase and Waxman [IW91] showed that the greedy strategy of adding the minimum cost subset of edges in each online step obtains a competitive ratio of $O(\log k)$, which is optimal up to constants. (Throughout this chapter, m and n will represent the number of edges and vertices in the input graph respectively, while k will represent the number of terminals.) This result was generalized to online EW SF by Awerbuch *et al* [AAB04], who showed a competitive ratio of $O(\log^2 n)$ for the greedy algorithm. This result was later improved by Berman and Coulston [BC97] who proposed an online algorithm based on the primal dual framework [AKR95, GW95] that achieves a competitive ratio of $O(\log n)$.

However, much less progress has been reported for NW versions of these problems.

[1] expected weight, if the algorithm is randomized

Unlike in the EW model, the NW ST problem generalizes the set cover problem. The first algorithm with a poly-logarithmic competitive ratio for the online set cover problem was proposed by Alon *et al* [AAA+09] who introduced an online adaptation of the classical LP relaxation technique to solve this problem. Recently, Naor *et al* [NPS11] used this technique in conjunction with structural properties of the NW ST problem to give an $O(\log n \log^2 k)$-competitive algorithm for the online NW ST problem. They also presented a poly-logarithmic competitive algorithm for the online NW SF problem, but the running time of the algorithm is quasi-polynomial. It is important to note that there is a qualitative difference between quasi-polynomial time algorithms and polynomial time algorithms for NW graphs. NW undirected graphs can be reduced to EW *directed* graphs, which allows for the application of known techniques for obtaining a poly-logarithmic approximation for the directed Steiner tree problem (see, e.g., Charikar *et al* [CCC+99]) in quasi-polynomial time. In fact, the algorithm of Naor *et al* for the online NW SF problem implicitly uses this transformation. In contrast, obtaining a poly-logarithmic approximation for the directed Steiner tree problem in polynomial time is a major open question. Therefore, polynomial-time algorithms for NW Steiner problems must develop novel techniques that successfully bypass this reduction to directed graphs. Naor *et al* developed such a set of techniques for the online NW ST problem, and posed the more general NW SF problem as their main open question. We resolve this question by introducing a new generic technique that we call *disk paintings* and using it in conjunction with a connection between the NW ST problem and the non-metric facility location problem previ-

ously observed by Naor *et al*. The competitive ratio of our algorithm is $O(\log n \log^2 k)$, which matches the competitive ratio for the NW ST problem [NPS11] and is optimal up to a logarithmic factor due to a lower bound of $\Omega(\log n \log k)$ [AAA+09, NPS11]. An interesting observation is that our algorithm yields a distinct algorithm for the online NW ST problem from that of Naor *et al* when applied to NW ST instances.

We show that in addition to solving the online NW SF problem on arbitrary graphs, disk paintings can be used to exploit combinatorial properties of graphs with an excluded constant-sized minor (such as *planar graphs* and more generally, graphs that can be embedded in surfaces of bounded genus) in online Steiner problems. We obtain an $O(\log n)$ competitive algorithm (optimal up to constants) for the online NW SF problem, which improves upon the online NW SF algorithm for general graphs described above. To the best of our knowledge, this is the first instance of an online network design problem where planarity (or more generally, exclusion of a fixed minor) has been successfully exploited to obtain an improved competitive ratio compared to the best known (in this case, also the best achievable) result for arbitrary graphs. We hope that disk paintings will be useful for other online network design problems in the future.

We also extend our results (both for general graphs and for graphs with an excluded fixed minor) to all $\{0, 1\}$-*proper functions* [GW95], which includes the SF problem as a special case but also includes other problems such as T-join, point-to-point connection problems, lower capacitated partitioning, and location-design/location-routing problems. We will formally describe a few examples of such problems later; for a detailed

description of problems that can be represented by proper functions, the reader is referred to [GW95].

Before describing our results and techniques in detail, we remark that offline NW Steiner problems have been studied previously. Klein and Ravi [KR95] introduced the notion of spider decompositions to give an $O(\log k)$-approximation for the NW ST problem, which is optimal up to constants. (The constant in the approximation ratio was improved later [GK99].) The Klein-Ravi result also extends to the SF problem, and more generally, to all proper functions. Recently, Demaine *et al* [DHK09] gave an $O(1)$-competitive algorithm for the NW SF problem in planar graphs, where the constant was further improved by Moldenhauer [Mol13] and the algorithm generalized to higher connectivity by Chekuri *et al* [CEV12a]. In fact, our online algorithm for planar graphs employs the disk painting technique in conjunction with structural ideas developed in [DHK09] for the corresponding offline problem. Several other offline NW network design problems have been considered in the literature (see, e.g., [GMNS99, MR07, Nut10]).

3.1.1 Problem Formulation

We are now ready to formally define the SF problem (from now on, the problems we refer to are online and NW, unless otherwise stated). Let $G = (V, E)$ be an undirected graph containing n vertices ($|V| = n$) and m edges ($|E| = m$), and let $w : V \to \mathbb{R}^{\geq 0}$ be the node weight function. The subgraph induced on G by a subset of vertices S is denoted by $G[S]$, and has weight $w(S) = \sum_{v \in S} w(v)$. The graph G and the weight function w are

given offline.

In the SF problem, we are given a new pair of terminals (also called a *demand*) $s_h, t_h \in V$ (s_h and t_h are called the endpoints of the demand) in online step h. The algorithm maintains a subset of vertices X such that $G[X]$ connects all pairs of terminals that have arrived thus far. In response to the arrival of a new terminal pair, the vertex set X can be augmented to ensure this property. The goal is to minimize the weight of X, i.e., $w(X)$. We note that the ST problem is a special case of the SF problem where one terminal in each terminal pair is a fixed vertex.

We also consider the more general class of Steiner problems that can be represented by $\{0,1\}$-proper functions [GW95]. We adapt an approach due to Goemans and Williamson [GW95] and Demaine, Hajiaghayi, and Klein [DHK09] to define a general family of node-weighted network-design problems. Let $G = (V, E)$ be a connected graph with a node-weight function w. Following Demaine et al. [DHK09], a $\{0,1\}$-function $f : 2^V \to \{0,1\}$ is *proper* for node-weighted problems if the following properties hold:

1. **Symmetry** $f(S) = f(V \backslash S)$ for every $S \subseteq V$.

2. **Disjointness** If S_1 and S_2 are disjoint, then $f(S_1) = f(S_2) = 0$ implies $f(S_1 \cup S_2) = 0$.

3. **Nullity** $f(\phi) = f(V) = 0$.

4. **Terminality** Every vertex v with $f(\{v\}) = 1$ has weight $w(v) = 0$.

5. **Efficiency** For any subset S, the value $f(S)$ is computable in polynomial time.

A proper function defines a family of cuts on the graph and the *network design* problem asks for a minimum-cost subgraph X which covers all the cuts, i.e., $G[X]$ must contain an edge of the cut $(S, V\backslash S)$ for every set S with $f(S) = 1$. For a subset of vertices $X \subseteq V$, we may use the set X and the subgraph $G[X]$ interchangeably, when it is clear from the context. A set X *satisfies* a proper function f, if X covers all the cuts defined by f. Equivalently, we say X is *feasible* if it satisfies f.

Following [QW11], we extend the notion of proper functions to the online setting where in each online step h, a proper function q_h on the vertices is presented. Let f_h denote the *cumulative function* of the step, i.e., $f_h(S) = \max_{j \leq h} q_j(S)$ for every $S \subseteq V$. The following shows that a cumulative function is proper too.

Proposition 3.1. *Let q_1 and q_2 be proper functions for a set of vertices V with node weights w. Let $f(S) := \max\{q_1(S), q_2(S)\}$ for every $S \subseteq V$. The function f is proper.*

Proof. For any $S \subseteq V$, $f(S) = 1$ if and only if either $q_1(S) = 1$ or $q_2(S) = 1$. Thus Terminality and Nullity of f follows. On the other hand by applying Symmetry for q_1 and q_2, if $f(S) = 1$ then either $q_1(V\backslash S) = 1$ or $q_2(V\backslash S) = 1$ which leads to $f(V\backslash S) = 1$. Finally to see the Disjointness of f, consider two arbitrary disjoint sets S_1 and S_2 such that $f(S_1) = f(S_2) = 0$. This implies $q_1(S_1) = q_1(S_2) = 0$ and $q_2(S_1) = q_2(S_2) = 0$. Thus $q_1(S_1 \cup S_2) = q_2(S_1 \cup S_2) = 0$ leading to $f(S_1 \cup S_2) = 0$. □

In the online variant of the problem, the solution produced at the end of online step h must satisfy f_h.

Note that the SF problem can be represented by the following proper function: in

online step h with terminal pair (s_h, t_h), for any subset $S \subseteq V$,

$$q_h(S) = \begin{cases} 1, & \text{if } |\{s_h, t_h\} \cap S| = 1 \\ 0, & \text{otherwise.} \end{cases}$$

These functions q_h satisfy properties 1-4 of proper functions by definition. For terminality, all terminals must have zero cost. We show later that this property can be ensured w.l.o.g.

As examples of other problems that can be represented by proper functions, let us now define the point-to-point connection problem [LMSL92] and the T-join problem [EJ01]. In the (offline) point-to-point connection problem, we are given a set of terminal pairs (s_h, t_h) (as in SF) but instead of pairing them, they are represented as a set of *sources* X and *sinks* Y such that $|X| = |Y|$. The goal is to find a minimum cost subgraph such that the number of sources in any connected component equals the number of sinks in the component. In the online version of the problem, every online step comprises a set of sources X_h and a set of sink Y_h such that $|X_h| = |Y_h|$. The corresponding proper function is given by: for any subset $S \subseteq V$,

$$q_h(S) = \begin{cases} 1, & \text{if } |X_h \cap S| \neq |Y_h \cap S| \\ 0, & \text{otherwise.} \end{cases}$$

Note that if $|X_h| = |Y_h| = 1$, then the problem is identical to online SF.

In the (offline) T-join problem, we are given an even set of terminals T and the goal is to find a minimum cost subgraph of the input graph that contains at least one edge for every cut that has an odd number of terminals on both sides. In the online version, each online step adds an even set of vertices T_h to T. The corresponding proper function is

given by: for any subset $S \subseteq V$,

$$q_h(S) = \begin{cases} 1, & \text{if } |T_h \cap S| \text{ is odd} \\ 0, & \text{otherwise.} \end{cases}$$

Finally, we need to define H-minor-free graphs. A graph H is a *minor* of graph G if H can be realized from G by the following set of operations: contracting an edge, deleting an edge, or deleting a vertex. As mentioned above, some of the results in this section will apply to classes of input graphs where a fixed graph H of constant size is not a minor of any graph in the class.

Assumptions. In the rest of the chapter, w.l.o.g., we assume the terminals are distinct and have weight 0. This can be ensured in the Steiner forest problem by attaching a proxy vertex of weight 0 to every vertex of the graph. In every online step, we interpret the corresponding proxy vertices as the terminal pair. Let T denote the set of vertices with weight 0, i.e., the possible terminals and let k denote the number of terminals that have arrived.

3.1.2 Our Results

We show the following result for proper functions.

Theorem 3.1. *There is a randomized online algorithm with competitive ratio $O(\log^2 k \log n)$ for network design problems characterized by proper functions.*

When applied to the SF problem, we obtain the following corollary.

Corollary 3.1. *There is a randomized polynomial-time algorithm for the online node-weighted Steiner forest problem that has a competitive ratio of* $O(\log n \log^2 k)$.

This result:

- improves upon the result of Naor *et al* [NPS11] for SF in two ways: their running time was quasi-polynomial and competitive ratio was $O(\log^3 n \log^7 k)$.
- matches the competitive ratio of Naor *et al* for ST up to constants.
- is optimal up to $O(\log n)$ since online set cover (which SF generalizes) has a randomized lower bound of $\Omega(\log n \log k)$ [AAA+09, Kor05].

Next, we obtain the following result for graphs with an excluded fixed minor.

Theorem 3.2. *There is a deterministic polynomial-time algorithm with a competitive ratio of* $O(\log k)$ *for* H*-minor-free node-weighted input graphs, where* H *is a fixed graph of constant size, for network design problems characterized by proper functions.*

When applied to the SF problem, we obtain the following corollary.

Corollary 3.2. *There is a deterministic polynomial-time algorithm for the online node-weighted Steiner forest problem that has a competitive ratio of* $O(\log k)$ *for* H*-minor-free input graphs, where* H *is a fixed graph of constant size.*

For H-minor-free graphs, this result:

- improves upon Theorem 3.1 for SF and upon [NPS11] for ST. (We note the lower bound of $O(\log n \log k)$ does not apply in the H-minor-free case.)
- is optimal up to a constant since there is an $\Omega(\log k)$ lower bound for the online EW ST problem. (This lower bound can be demonstrated by using a *diamond* graph, which is planar. We omit the details for brevity.)

3.1.3 Our Techniques: Disk Paintings

The principle of weak duality in minimization problems asserts that the optimal solution to a primal linear program (LP) is lower bounded by *any* feasible solution to the dual LP. This has inspired the classical (offline) primal-dual method [AKR95, GW95] where a progressively constructed dual solution guides the choices made by the algorithm in the primal solution. For Steiner problems, the dual solution is constructed by growing *moats* around every terminal. When two moats collide, they are merged by buying the path connecting the corresponding terminals and the merged moat continues growing. Thus, the sets with positive dual variables form a *laminar* family. The crux of the analysis is to charge the cost of purchased paths to the sum of the radii of the moats, and highly relies on the fact that when two growing moats collide, they have roughly the same radii.

However, in online settings, this property cannot be ensured since the terminals are identified sequentially in online steps. In fact, in any online step, there is only one moat that is growing, namely the one containing the new terminal. In spite of this difficulty, the primal-dual framework has recently been utilized for online algorithms in two distinct

lines of work: either a dual solution is used to guide the construction of a *fractional* solution to the primal LP, which is then been rounded online to produce an integer solution (see, e.g., the survey by Buchbinder and Naor [BN09a]) or the algorithm maintains a multi-layered collection of laminar families of moats (see, e.g., [QW11]).

While our technique of disk paintings also utilizes the broad framework of using a dual solution to guide the algorithmic choices, we deviate significantly from these previous approaches in the structure of the dual solution that we construct, which we describe below. A *disk painting* is simply a set of disjoint disks centered at terminals. Since we have NW graphs, disks intersect at vertices rather than edges. In fact, unlike in the EW case, more than two disks can cumulatively cause an intersection at a vertex. To visualize these disks, let us assign a distinct color to every disk and an area equal to its weight to every vertex. Then, a disk may color a vertex either wholly or partially. For example if x, y, z are three vertices of weight $2w$ each connected by edges (x, y) and (y, z), and we add a disk of radius $5w$ centered at x, then the disk colors y fully (i.e. its entire weight) but z only partially (half its weight). An intersection is caused at a vertex when the sum of weights colored by disks containing the vertex (either fully or partially) exceeds the total weight of the vertex.

To formally define disk paintings, we first need to define distances between vertices. For any pair of vertices u, v, let $d^w(u, v)$ denote the weight of the shortest path between u and v (including u, v) w.r.t. a node weight function w. For a set S of vertices, let $d^w(S, v) = \min_{u \in S} d^w(u, v)$. A *painting* is a function $p : V \times \Theta \to \mathbb{R}_{\geq 0}$, where Θ is a

set of colors. Let $p(v)$ denote the total colored area of vertex v, i.e., $p(v) = \sum_{\theta \in \Theta} p(v, \theta)$. A painting p is *feasible* if the colored area of a vertex does not exceed its weight, i.e., $p(v) \leq w(v)$ for every vertex v. The *union* of a set of paintings $p_1, p_2, \ldots$ is the painting $p = \sum_i p_i$ (i.e., $p(v, \theta) = \sum_i p_i(v, \theta)$ for every vertex v and color θ). Further, $p_1, p_2, \ldots$ are said to be *non-overlapping* if their union is feasible.

A *disk* of radius r centered at vertex v is a painting p in which the area within a radius r of v is colored by some unique color θ^v, i.e., for every vertex u,

$$p(u, \theta^v) = \begin{cases} w(u) & \text{if } d(v,u) \leq r \\ 0 & \text{if } d(v,u) - w(u) \geq r \\ r - (d(v,u) - w(u)) & \text{otherwise} \end{cases}$$

A painting that comprises a union of disks centered at terminals is called a *disk painting*. This is the only kind of painting that we will use in this section; hence, we will often simply call it a painting. A vertex u is *inside* the disk if $d(v, u)$ is *strictly* less than r, and on the *boundary* if it is not inside but has a neighbor that is inside. The *continent* of a disk is the set of vertices inside the disk.

Having described some of the key concepts of disk paintings, let us now give a high-level description of our algorithmic technique based on disk paintings. We will formally describe our algorithm later for all proper functions, but for the purpose of this informal discussion, let us focus on the SF problem. Our algorithms can be thought of as *augmented greedy* algorithms. In every online step, we initially perform a greedy augmentation of the primal solution that satisfies the new constraint, i.e., buy the cheapest

path in the SF problem between the terminal pair after reducing the cost of all nodes in the current solution to 0. To account for the resulting increase in the cost of the primal solution, we aim to add a disk of radius equal to (or a constant factor of) the increase in the primal cost. If we are able to place such a disk centered at either of the two terminals in the new pair without violating feasibility of the disk painting, then we terminate the online step. The more challenging scenario is when such disks cause infeasibility of the painting. In this case, the algorithm augments the primal solution with a graph element that depends on the problem. For example, in the case of SF in general graphs, the algorithm connects the two terminals to a vertex on which the infeasibility occurred. On the other hand, for SF in graphs with an excluded minor, the algorithm connects all the centers of the intersecting disks via a *spider*. A spider is a tree with at most one vertex of degree greater than two, which is called the *center* of the spider. The paths connecting the center to the leaves are called the *legs* of the spider. The crux of the analysis is to show that the total primal cost in these instances where we are unable to add a new disk to the disk painting can be amortized to the existing disks by charging the disks that caused the infeasibility.

3.2 Preliminaries

Before we describe the algorithms for the network design problem, we need to introduce a few notations. Consider a graph G with a vertex-weight function w and a proper function f defined over $2^{V(G)}$. Let T denote the set of vertices with zero weight.

A vertex t is a *terminal* of a proper function f, if $f(\{t\}) = 1$. Note that by Terminality, $t \in T$. For a set $S \subseteq V$, let $\delta(S) \subseteq V \backslash S$ denote the neighbors of S.

3.2.1 Properties of Proper Functions

The following is the direct result of the disjointness of a proper function.

Proposition 3.2. *If a set S does not contain a terminal, then $f(S) = 0$.*

Proof. Assume by contradiction that $f(S) = 1$. Consider the smallest subset $U \subseteq S$ whose $f(U) = 1$. Since S does not contain a terminal, $|U| \geq 2$. Let u be an arbitrary vertex in U. Both $\{u\}$ and $U \backslash \{u\}$ are strict subsets of U, thus $f(\{u\}) = f(U \backslash \{u\}) = 0$ which contradicts by Disjointness. □

A proper function is *trivial* if for every $S \subseteq V$, $f(S) = 0$. Observe that if for a set S, $f(S) = 1$, then $f(V \backslash S) = 1$. Thus by Proposition 3.2, both S and $V \backslash S$ contain at least one terminal. This leads to the next proposition.

Proposition 3.3. *Any non-trivial proper function has at least two terminals.*

For a subset $X \subseteq V$, let $CC(X)$ denote the collection of connected components of $G[X]$. The following lemma (formally proved in [DHK09]) gives a polynomial-time test for whether a set X is feasible. The lemma easily follows from applying the properties of proper functions on the connected components of $G[X]$.

Lemma 3.1 (Lemma 7 in [DHK09])**.** *A subset X is feasible if and only if X contains all the terminals and $f(C) = 0$ for every $C \in CC(X)$.*

Lemma 3.1, together with the Efficiency property, guarantee that given a subset $X \subseteq V$, in polynomial time we can either (i) find a set S such that $f(S) = 1$ and $\delta(S) \cap X = \phi$; or (ii) verify that X is a feasible solution. Indeed the following lemma provides a more refined structural property of a feasible solution.

Lemma 3.2. *Let $S \subseteq T \subseteq V$ and let X be a feasible solution. If $f(S) = 1$ and there are no terminals in $T \backslash S$, then $G[X]$ contains a path from a terminal $\tau \in S$ to a vertex $v \in \delta(T)$.*

Proof. First, we claim $f(T) = 1$. Since $f(S) = 1$, by Symmetry, $f(V \backslash S) = 1$. However, $f(T \backslash S) = 0$, thus by Disjointness, $f(V \backslash T)$ must be one. Now by Symmetry, $f(T) = 1$ too.

Consider the subgraph $G[X]$. Let Q denote the set of vertices reachable (in $G[X]$) from a *terminal* in S. We show that $Q \cap \delta(T) \neq \phi$. Assume by contradiction, that $Q \subseteq T$. The set Q is a collection of several connected components of $G[X]$. By Lemma 3.1 and Disjointness, $f(Q) = 0$. Since $f(T) = 1$, by Disjointness, $f(T \backslash Q) = 1$. However, there is no terminal in $T \backslash Q$ which contradicts with Proposition 3.2. □

Lemma 3.2 is particularly interesting when T is the continent of a disk centered at a terminal. If the other terminals are not inside the disk, then any feasible solution connects the center to a vertex on the boundary.

Given a graph G, *contracting* a connected set of vertices S denotes replacing S by a *super-vertex* adjacent to $\delta(S)$. A *contraction* of a graph G (denoted by $\overline{G}$) is obtained by contracting connected subsets of vertices in G. For a vertex v in $\overline{G}$, let $\gamma(v)$ denote the set

of vertices of G that have been contracted to form v. If a vertex v is not part of a contracted set, then it retains its label, i.e., $\gamma(v) = \{v\}$. We note that by definition, for every v in $V(\overline{G})$, $\gamma(v)$ is connected in G. We extend the notation by defining $\gamma(S) = \bigcup_{v \in S} \gamma(v)$ for any set $S \subseteq V(\overline{G})$. For simplicity, we consider G to be a contraction of itself. We refer to the original vertices in G as *simple vertices*.

Let $V(G)$ denote the set of vertices of G, and let w_G be a node weight function over $V(G)$. For a contraction $\overline{G}$, we derive a corresponding weight function by reducing the cost of super-vertices to zero, i.e., for every vertex $v \in V(\overline{G})$,

$$w_{\overline{G}}(v) = \begin{cases} w_G(v) & \text{if } \gamma(v) = \{v\} \\ 0 & \text{otherwise} \end{cases}$$

Let H be an induced subgraph of G. Let $\overline{G}$ be a contraction. The *contracted subgraph* $\overline{H}$ is obtained from H by contracting $V(H) \cap \gamma(v)$ to v for every $v \in \overline{G}$. Note that $\overline{H} \subseteq \overline{G}$. When there is no ambiguity, a contracted subgraph may retain the label of the original subgraph, i.e., we may refer to $\overline{H}$ by H as well.

Let f be a proper function w.r.t. the graph G. Given a contraction $\overline{G}$, a *contracted function* $f^{\overline{G}}$ is obtained by setting $f^{\overline{G}}(S) = f(\bigcup_{v \in S} \gamma(v))$ for every set $S \subseteq V(\overline{G})$. In other words, the cuts retain their f-value. Indeed, $f^{\overline{G}}$ is proper too.

Proposition 3.4. *The contracted function $f^{\overline{G}}$ is proper w.r.t. to $\overline{G}$ and $w_{\overline{G}}$.*

Proof. Let f be the original proper function for G. Note that the weight of a super-vertex is zero, thus if a super-vertex becomes a terminal, the Terminality property still holds. Efficiency and Nullity also carries over from f. To verify Symmetry, note that for every

set $S \subseteq V(\overline{G})$

$$
\begin{aligned}
f^{\overline{G}}(S) &= f(\bigcup_{v \in S} \gamma(v)) \\
&= f(V(G) \backslash \bigcup_{v \in S} \gamma(v)) && \textit{by Symmetry of } f \\
&= f\left(V(G) \cap (\bigcup_{v \in S} \gamma(v))^c\right) \\
&= f\left(V(G) \cap \bigcap_{v \in S} (\gamma(v))^c\right) && \textit{De Morgan's law} \\
&= f\left(u \in V(G) | \forall_{v \in S} u \notin \gamma(v)\right) \\
&= f\left(u \in V(G) | \exists_{v \notin S} u \in \gamma(v)\right) \\
&= f^{\overline{G}}(V(\overline{G}) \backslash S)
\end{aligned}
$$

Finally to verify Disjointness, let $S_1, S_2 \subseteq V(\overline{G})$ be disjoint sets.

$$
\begin{aligned}
f^{\overline{G}}(S_1 \cup S_2) &= f(\bigcup_{v \in \{S_1 \cup S_2\}} \gamma(v)) \\
&= f\left((\bigcup_{v \in S_1} \gamma(v)) \cup (\bigcup_{v \in S_2} \gamma(v))\right) \\
&\leq f(\bigcup_{v \in S_1} \gamma(v)) + f(\bigcup_{v \in S_2} \gamma(v)) && \textit{By Disjointness of } f \textit{ and } S_1 \cap S_2 = \phi \\
&= f^{\overline{G}}(S_1) + f^{\overline{G}}(S_2)
\end{aligned}
$$

□

3.2.2 Properties of Disk Paintings

The following propositions hold for a set of non-overlapping disks.

Proposition 3.5. *If u is on the boundary of a disk centered at v, then $p(u, \theta^v)$ is strictly positive.*

Proof. Let r be the radius of the disk. For a vertex u on the boundary, by definition $d(v, u) - w(u) < r$. On the other hand if u is not inside, then $p(u, \theta^v) \geq r - (d(v, u) - w(u)) > 0$. □

Proposition 3.6. *A vertex inside a disk cannot be on the boundary of another disk.*

Proof. Assume by contradiction, that a vertex u is inside a disk centered at v_1 while at the same time it is on the boundary of a disk centered at v_2. By definition, a vertex inside a disk is fully covered, i.e., $p(u, \theta^{v_1}) = w(u)$. On the other hand, by Proposition 3.5, $p(u, \theta^{v_2}) > 0$. Thus in the union of the two disks,

$$p(u) \geq p(u, \theta^{v_1}) + p(u, \theta^{v_2}) > w(u)$$

which is a contradiction. □

Fact 3.1. *Since the weight of a terminal is zero, the center of a disk is inside the disk.*

We emphasize that by Proposition 3.6, the disks may share a vertex *only* on their boundaries. This is indeed a crucial observation which ultimately leads to our algorithm for the network design problem. Fact 3.1 is implicitly used in our analysis since we

assume the center of a disk is inside the disk no matter how small is the radius. The next lemma shows the relationship between a painting and the optimal offline solution for SF. This lemma is also not used explicitly in our analysis; however, we exploit it to design our algorithms. This is the key property of disk paintings which might be of independent interest.

Lemma 3.3. *Let $\mathcal{L}$ be a painting comprising disks centered at a subset of terminals S such that for any terminal pair (s,t), s (resp., t) is not inside a disk centered at t (resp., s). If T be any subgraph of G connecting all terminal pairs with at least one terminal in S, then $w(T)$ is at least the sum of the radii of the disks.*

Proof. For a terminal $v \in S$, let r_v denote the radius of the disk centered at v. Consider an arbitrary terminal $v \in S$ at the center of a disk. It is sufficient to show that $\sum_{u \in V(T)} \mathcal{L}(u, \theta^v) \geq r_v$ because then one can argue

$$\sum_{v \in S} r_v \leq \sum_{v \in S} \sum_{u \in V(T)} \mathcal{L}(u, \theta^v) = \sum_{u \in V(T)} \sum_{v \in S} \mathcal{L}(u, \theta^v)$$

$$= \sum_{u \in V(T)} \mathcal{L}(u) \leq \sum_{u \in V(T)} w(u)$$

where the last inequality follows from the feasibility of $\mathcal{L}$. Now consider the painting induced to T. Every demand with an endpoint in S is satisfied. Thus for every $v \in S$, T enters the disk from outside of it and passes through its center v, hence staying for at least r_v units of distance in the disk.

We prove this more formally by induction. Let v' be the other endpoint of the demand concerning v. Since T satisfies the demand (v, v'), there is a path from v to

v' in T. Let u denote the first vertex in this path which is on the boundary of the disk centered at v. The vertex u exists since v' is not inside the disk. Let P be the part of the path between v and u. Note that all vertices except v' are inside the disk. Suppose $P = (v_0 = v, v_1, v_2, \ldots, v_k = u)$ for some $k \geq 1$. We use induction to show for every i, $0 \leq i \leq k-1$, $\sum_{j \leq i} \mathcal{L}(v_i, \theta^v) \geq d(v, v_i)$. The base of induction is trivial. To prove the inductive step for arbitrary $i > 0$, observe that $d(v, v_i) - w(v_i) \leq d(v, v_{i-1})$. On the other hand, since v_i is inside, $\mathcal{L}(v_i, \theta^v) = w(v_i)$. Thus by induction hypothesis

$$\begin{aligned}\sum_{j \leq i} \mathcal{L}(v_j, \theta^v) &\geq d(v, v_{i-1}) + \mathcal{L}(v_i, \theta^v) \\ &\geq (d(v, v_i) - w(v_i)) + w(v_i) \geq d(v, v_i)\end{aligned}$$

Finally since $\mathcal{L}(u, \theta^v) = r_v - (d(v, u) - w(u))$, we get

$$\begin{aligned}\sum_{0 \leq i \leq k} \mathcal{L}(v_i, \theta^v) &\geq d(v, v_{k-1}) + \mathcal{L}(u, \theta^v) \\ &= d(v, v_{k-1}) + r_v - (d(v, u) - w(u)) \geq r_v\end{aligned}$$

which completes the proof. □

3.3 Online Node-weighted Network Design

We start by describing a special variant of the well-studied facility location problem. The input comprises a set of facilities each with a setup cost, and a set of clients each with a connection cost to every facility, and a set of connectivity demands. The *group non-metric facility location* problem (GNFL) asks for a mapping of clients to facilities that

minimizes the sum of setup costs and connection costs while satisfying *demands* defined below.

Let Λ and Ψ denote the set of facilities and clients, respectively. For a facility $\lambda \in \Lambda$, let $\omega(\lambda) \in \mathbb{R}_{\geq 0}$ denote the setup cost of λ. For a client $\psi \in \Psi$ and a facility $\lambda \in \Lambda$, let $\overline{d}(\psi, \lambda) \in \mathbb{R}_{\geq 0}$ denote the cost of connecting ψ to λ.

A mapping $M : \Psi \to \Lambda$ assigns clients to facilities. A mapping is a partial function, i.e., some of the clients may not be connected to a facility. A facility λ is *open* if for some client ψ, $M(\psi) = \lambda$. The cost $c(M)$ of a mapping M is the sum of setup costs of open facilities and connection costs used in the mapping, i.e.,

$$c(M) = \sum_{\lambda | \exists \psi, M(\psi) = \lambda} \omega(\lambda) + \sum_{\psi, \lambda | M(\psi) = \lambda} \overline{d}(\psi, \lambda)$$

In the GNFL problem, the demands are in the form of groups of clients $\mathcal{D} = \langle g_1, g_2, \ldots \rangle$ where $g_i \subseteq \Psi$. A mapping M *satisfies* $\mathcal{D}$ if for every group $g_i \in D$ at least one client $\lambda \in g_i$ is mapped to a facility. Given a set of demands, the goal is to find a mapping of minimum cost that satisfies all demands. In the online variant of the problem, the set of facilities and setup costs are known in advance but the demands arrive online one at a time, revealing the connection costs of a client if it was not present in previous demands. Upon receiving a new demand, we need to augment the mapping to cover at least one client of the new demand.

Bounded Group Non-metric Facility Location. Given a graph $G = (V, E)$ with node weights w, the bounded group non-metric facility location problem w.r.t a real value r (r-BFL) is an instance of the GNFL problem as follows. Recall that $T \subseteq V$ is the set of

zero-weight vertices.

- For every vertex $v \in V$ we have a facility (with the same label);
- The setup cost function is identical to the node-weight function;
- For every zero-weight vertex $t \in T$ we have a client (with the same label); and
- For a client $t \in T$, consider a disk of radius r centered at t in G. For every vertex v on the *boundary* of the disk, the connection cost between t and v is $\overline{d}(t, v) = d^w(t, v) - w(v)$. For every other client-facility pair (t, v) the connection cost is infinity. Note that this includes the facilities that are too far ($d^w(t, v) - w(v) \geq r$) as well as those that are too close ($d^w(t, v) < r$).

In other words, a client can be mapped only to the facilities on the boundary of the disk, the cost of which is the distance for *touching* the facility in G.

For a pair of groups of vertices g_1 and g_2, let $d^w(g_1, g_2) = \min_{u \in g_1, v \in g_2} d^w(u, v)$ be the distance between the groups in G. Let $\mathcal{D}$ denote the set of demands. We restrict the input of r-BFL by adding the following assumption.

- Every pair of demands $g_1, g_2 \in \mathcal{D}$ should be at least $2r$ far from each other in G, i.e., $d^w(g_1, g_2) \geq 2r$.

At any time in the algorithm, we say a client is *active* if it has appeared in a demand so far. When the exact radius is not a concern, we may refer to r-BFL as BFL. Given a mapping M, the graph $H(M)$ is the subgraph of G induced by the vertices of shortest paths connecting t to v for every client t and facility v such that $M(t) = v$. Observe that $w\left(H(M)\right) \leq c(M)$.

We note that since the demands are disjoint in BFL, we may collapse every group to a single client and thus it reduces to the *non-metric facility location problem (NFL)* [Hoc82, AAA+06]. In other words, we replace clients in a demand g by a special client c_g such that for every facility v, $\overline{d}(c_g, v) = \min_{t \in g} \overline{d}(t, v)$. Let BFLALG be an online algorithm for the BFL problem with competitive ratio α_{BFL}. We use BFLALG as a black-box to show that the network design problem admits a competitive ratio of $O(\log(k) \cdot \alpha_{BFL})$. In fact, Alon et al. [AAA+06] give an online randomized algorithm for the NFL problem with competitive ratio $O(\log(k)\log(n))$, i.e., $\alpha_{BFL} = O(\log(k)\log(n))$. (Here n is the number of facilities and k is the number of active clients). Therefore our competitive ratio is $O(\log^2 k \log n)$.

Algorithm for Online Network Design. We are now ready to describe algorithm NDALG For every integer $i \in \mathbb{Z}$, the algorithm keeps a 2^i-BFL instance $\mathcal{L}_i$. We augment the mapping of the instances using BFLALG . Let $\mathcal{D}_i$ and M_i denote the demands and the current mapping for the instance $\mathcal{L}_i$, respectively. Our algorithm maintains a partial solution X guaranteeing that for every i, the solution for $\mathcal{L}_i$ is included in X, i.e., $H(M_i) \subseteq X$. Let I denote the number of BFL instances with at least one demand. Indeed using standard techniques one can modify the algorithm so that $I = O(\log(k))$, the details are presented at the end of the section.

The algorithm maintains the invariant that for any demand in $\mathcal{L}_i$ (corresponding to a group of clients g) the following *neighborhood clearance (NC)* holds at the time of arrival of the demand in $\mathcal{L}_i$.

Definition 3.1 (NC(g,i))**.** *The neighborhood of a group $g \subseteq T$ is* clear *in $\mathcal{L}_i$ if both the following conditions hold:*

- *The group g is at least $2 \cdot 2^i$ far in G from any previous demand in $\mathcal{D}_i$; and*
- *For every (currently) open facility v in M_i, the connection cost $\overline{d}(t, v)$ is infinite for every $t \in g$.*

If one of the conditions fails, $NC(g, i)$ does not hold and an active client of $\mathcal{L}_i$ or an open facility of $\mathcal{L}_i$ closest to g is the witness *of the failure.*

The algorithm starts by initializing X and M_i's to empty. At any time step h, let f_h be the cumulative function. Let $\mathcal{T}_h$ denote the set of terminals of f_h. We augment the solution X iteratively until it satisfies f_h. At each iteration, the following process is executed.

Let X be the current partial solution. Let $\overline{G}$ denote a contraction of G by contracting every connected component of $G[X]$. Let f denote the contracted function of f_h w.r.t. $\overline{G}$. Recall that by Proposition 3.3, f has at least two terminals. Let (τ_1, τ_2) denote the closest pair of terminals of f w.r.t. $w_{\overline{G}}$. Let D be the distance between them. We first buy the shortest path between τ_1 and τ_2 (thus incurring a cost of D). Note that this shortest path may contain super-vertices. Adding a super-vertex u to X implies setting X as the union of X and $\gamma(u)$. Recall that $\gamma(u)$ is the set of simple vertices contracted to u. Since $\gamma(u)$ denotes a connected component of X, adding a super-vertex u to X does not change X in this step of the algorithm.

Consider the integer i such that $4 \cdot 2^i \leq D < 4 \cdot 2^{i+1}$. Let $g(\tau_1) = \gamma(\tau_1) \cap \mathcal{T}_h$

(resp. $g(\tau_2) = \gamma(\tau_2) \cap \mathcal{T}_h$) be the group of terminals of f_h contracted to τ_1 (resp. τ_2). If the neighborhood of either $g(\tau_1)$ or $g(\tau_2)$ is clear, we give the corresponding group as a new demand to BFLALG for $\mathcal{L}_i$. We mimic the solution of BFLALG , i.e., if M_i is augmented by mapping a client $t \in T$ to a facility $v \in V$, we buy the shortest path in G connecting t to v. However, if none of the neighborhoods is clear, let z_1 and z_2 denote the witnesses of failure corresponding to $NC(g(\tau_1), i)$ and $NC(g(\tau_2), i)$. We then connect τ_1 to z_1 and τ_2 to z_2 by buying shortest paths w.r.t. w.

Algorithm 3 Online Network Design

Input: A node-weighted graph G and an online stream of proper functions $q_1, q_2, \ldots$.

Output: A set X such that $G[X]$ satisfies the connectivity demands.

Offline Process:

For every $i \in \mathbb{Z}$ initialize $\mathcal{L}_i$, a 2^i-BFL instance.

Initialize X to an empty set.

Online Scheme; assuming a proper function q_h is arrived:

Let f_h be the cumulative function and let $\mathcal{T}_h$ denote the set of terminals of f_h.

while X does not satisfy f_h **do**

  Form $\overline{G}$ from G by contracting $CC(X)$. Let f denote the contracted function.

  Let D denote the distance between the closest pair of terminals τ_1 and τ_2 in $\overline{G}$.

  Buy the shortest path between τ_1 and τ_2.

  Consider the integer i that $4 \cdot 2^i \leq D < 4 \cdot 2^{i+1}$.

  Let $g(\tau_1) = \gamma(\tau_1) \cap \mathcal{T}_h$ and $g(\tau_2) = \gamma(\tau_2) \cap \mathcal{T}_h$.

  if $NC(g(\tau_1), i)$ holds **then**

    Give $g(\tau_1)$ to BFLALG as a new demand for $\mathcal{L}_i$. Set $X = X \cup H(M_i)$.

  else if $NC(g(\tau_2), i)$ holds **then**

    Give $g(\tau_2)$ to BFLALG as a new demand for $\mathcal{L}_i$. Set $X = X \cup H(M_i)$.

  else

    Let z_1 and z_2 denote the witnesses w.r.t. $NC(g(\tau_1), i)$ and $NC(g(\tau_2), i)$.

    Buy the shortest paths in G connecting a terminal $t_1 \in \gamma(\tau_1)$ to z_1 and a terminal $t_2 \in \gamma(\tau_2)$ to z_2.

Analysis. We distinguish between two types of costs incurred by the algorithm. The *simulation cost* is the total weight of vertices being purchased for covering the augmentations of mappings in every iteration, i.e., the simulation cost is at most $\sum_i w(H(M_i))$. The cost due to the weight of other vertices in X is called the *connectivity cost*. Note that we may incur a connectivity cost in two places in the algorithm: (i) when buying the shortest path; or (ii) when buying the paths connecting terminals to the corresponding witnesses of failure.

Recall that I denotes the number of BFL instances with at least one demand. Let *OPT* denote the weight of an optimal (offline) solution of the network design problem. We show that both simulation and connectivity costs are at most $O(I \cdot \alpha_{BFL}) \cdot \mathit{OPT}$. For every $i \in \mathbb{Z}$, let OPT_i denote the cost of an optimal (offline) solution for $\mathcal{L}_i$. First we show that OPT_i is a lower bound for *OPT*. Intuitively, by applying Lemma 3.2 in every iteration, one can show that for every demand $g \in \mathcal{D}_i$, the optimal solution contains a path from a terminal $t \in g$ to a vertex at the boundary of a disk of radius 2^i centered at t. Indeed such a path connects t to a facility in $\mathcal{L}_i$. Thus we can get a feasible solution for $\mathcal{L}_i$ by opening a facility at the intersection of the optimal solution with the boundaries of the disks centered at the active terminals and then connecting every demand to the closest open facility (see Figure 3.1).

Lemma 3.4. *For every $i \in \mathbb{Z}$, $\mathit{OPT}_i \leq \mathit{OPT}$.*

Proof. Let X^* denote the optimal (offline) solution of the network design problem, i.e., $\mathit{OPT} = w(X^*)$. Let $\mathcal{D} = \langle g_1, \ldots, g_\kappa \rangle$ denote the set of demands for $\mathcal{L}_i$. Suppose for

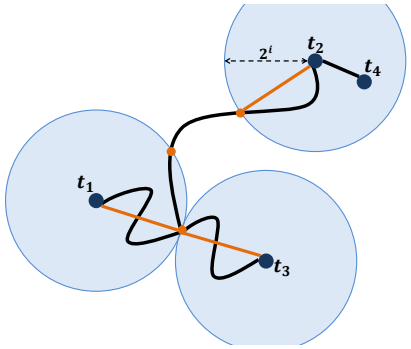


Figure 3.1: Depiction of an optimal solution

The black lines show the optimal solution for the network design problem. Suppose t_1, t_2, t_3 are the active clients. The orange lines represent a mapping for $\mathcal{L}_i$ based on the optimal solution.

$j < l \leq \kappa$, the demand g_j is given to $\mathcal{L}_i$ before g_l. Claim 3.1 below, shows that for every $l \in [\kappa]$, there exists a path p_l in X^* connecting a terminal $t \in g_l$ to a facility in the boundary of a disk of radius 2^i centered at t in G. Let e_l denote the facility at the endpoint of p_l.

Since we have neighborhood clearance for every new demand, the distance between every pair of demands is at least $2 \cdot 2^i$. Thus for two different demands g_j and g_l, the corresponding paths p_j and p_l may intersect only at the endpoints (i.e., e_j and e_l), otherwise g_j and g_l will be closer than $2 \cdot 2^i$. We can get a feasible solution for $\mathcal{L}_i$ by opening a facility at every e_l and connecting the closest terminal $t \in g_l$ to e_l. Since the paths may share only at the endpoints, the cost of such a mapping is a lower bound for $w(X^*)$, thus proving the lemma. Therefore proving the following claim completes the proof.

Claim 3.1. *For every $l \in [\kappa]$, X^* contains a path from a terminal $t \in g_l$ to a vertex on*

the boundary of a disk of radius 2^i centered at t in G.

Proof. Consider the iteration in which we have given the demand g_l to $\mathcal{L}_i$. Suppose we have received h online proper functions so far and let f_h be the cumulative function. Let $\overline{G}$ be the contraction obtained by contracting the connected components of the partial solution at that iteration. Let f denote the corresponding contracted function. Recall that for a terminal τ of f, $g_l \subseteq \gamma(\tau)$ (Line 7 of Algorithm 3). Furthermore, there should be a terminal τ' such that (τ, τ') is the closest pair of terminals. Let D be the distance between τ and τ' in $\overline{G}$. Let i be the index such that $4 \cdot 2^i \leq D < 4 \cdot 2^{i+1}$. Consider a disk of radius 2^i centered at τ in $\overline{G}$. At this iteration, τ and τ' are the closest pair of terminals and are at least $4 \cdot 2^i$ far from each other. Thus there is no terminal inside the disk except τ. Let Q be the continent of the disk.

We now define a slightly different contraction with its own set of contracted function and terminals. Let $\overline{G}'$ be the contraction obtained by contracting every connected component of the partial solution, *except* $\gamma(\tau)$. In other words, $\overline{G}$ can be obtained from $\overline{G}'$ by contracting $\gamma(\tau)$ to τ. Let f' denote the contracted function w.r.t. $\overline{G}'$. Let $Q' \subseteq V(\overline{G}')$ denote the same cut as Q, i.e., $Q' = (Q \backslash \{\tau\}) \cup \gamma(\tau)$.

Recall that in a contraction, the cuts retain their f-value. Since τ is a terminal of f, $f'(\gamma(\tau)) = 1$. Since $Q \backslash \{\tau\}$ has no terminal and $\overline{G}$ and $\overline{G}'$ differ only in $\gamma(\tau)$ and τ, there is no terminal of f' in $Q' \backslash \gamma(\tau)$ either. By Lemma 3.2, any feasible solution contains a path from a terminal $t \in \gamma(\tau)$ to a vertex in $\delta(Q)$. Therefore X^* has a path from a terminal $t \in g_l$ to a vertex at least 2^i far from t. Therefore such a path intersects with a

vertex on the boundary of a disk of radius 2^i centered at t in G. □

We are now ready to prove the bound on the cost of the algorithm.

Lemma 3.5. *The total cost incurred by the algorithm is within $O(I \cdot \alpha_{BFL})$ factor of OPT.*

Proof. Lemma 3.4 directly leads to the desired bound for the simulation cost:

$$\sum_i w(H(M_i)) \leq \sum_i c(M_i) \leq \alpha_{BFL} \sum_i \mathit{OPT}_i \leq \alpha_{BFL} I \cdot \mathit{OPT}$$

We show a similar upper bound for the connectivity cost. For every i, let $\mathcal{D}_i$ denote the set of demands so far for $\mathcal{L}_i$. First we claim that BFLALG incurs the cost of at least 2^i for satisfying each demand.

Claim 3.2. *For every i, $c(M_i) \geq |\mathcal{D}_i| \cdot 2^i$.*

Proof. We prove this by induction. Let $\mathcal{L}_i$ be one of the BFL instances. Suppose the claim holds just before the arrival of a new demand g. Note that neighborhood clearance holds for g. By the second condition of Definition 3.1, the distance for touching any open facility is at least 2^i, i.e., for every open facility v in M_i, $d(g, \{v\}) - w(v) \geq 2^i$. Such a facility cannot be on the boundary of a disk of radius 2^i centered at any terminal $t \in g$ (recall that a vertex v is on the boundary if $d(g, \{v\}) - w(v) < 2^i$). Thus the cost of assigning a terminal $t \in g$ to any facility that is already open is infinite. Hence, BFLALG has to open a new facility on the boundary of a disk of radius 2^i centered at a terminal $t \in g$. By definition, a vertex on the boundary is at least 2^i far from the center (including the cost of the endpoints). Thus BFLALG incurs the cost at least 2^i for satisfying a new demand. □

Consider an arbitrary iteration of the algorithm. Suppose h demands have arrived so far and let f_h denote the cumulative function. Let $\mathcal{T}_h \subseteq V(G)$ denote the terminals of f_h. Let X be the partial solution at the start of iteration and let $\overline{G}$ be the contraction obtained from G by contracting the connected components of X. Let f denote the corresponding contracted function. In the algorithm, we find the closest pair of terminals (τ_1, τ_2) that are D far from each other. We buy the shortest path between τ_1 and τ_2, thus incurring a connectivity cost D.

We partition the iterations into different classes. Class i comprises iterations for which i satisfies $4 \cdot 2^i \leq D < 4 \cdot 2^{i+1}$. We show at any time in the algorithm, the total connection cost incurred for Class i iterations is bounded by $O(1) \cdot c(M_i)$ which, together with Claim 3.2, completes the proof of the lemma. In the rest of proof, we only consider Class i iterations for an arbitrary $i \in [I]$.

We distinguish between two types of iterations.

- **Type I:** *At Line 7 of Algorithm 3, the neighborhood is clear for either τ_1 or τ_2.* In such iteration, the connectivity cost we incur is at most $D \leq 8 \cdot 2^i$. Let Λ denote the number of iterations of Type I. For every Type I iteration, we add a new demand to $\mathcal{L}_i$. Therefore $\Lambda \leq |\mathcal{D}_i|$. Hence the total connectivity cost of our algorithm in iterations of Type I is at most $\Lambda \cdot 8 \cdot 2^i < |\mathcal{D}_i| \cdot 8 \cdot 2^i$. By Claim 3.2, this is upper bounded by $8c(M_i)$.
- **Type II:** *None of the neighborhoods is clear at Line 7.* Let z_1 and z_2 denote the corresponding witnesses for $NC(g(\tau_1), i)$ and $NC(g(\tau_2), i)$. Since z_1 and z_2 are

failure witnesses, we have $d^w(\tau_1, \{z_1\}), d^w(\tau_2, \{z_2\}) < 2 \cdot 2^i$. Thus the connectivity cost we incur at a Type II iteration is less than $D + d^w(\tau_1, \{z_1\}) + d^w(\tau_2, \{z_2\}) < 12 \cdot 2^i$. On the other hand, z_1 and z_2 cannot belong to the same connected component of X. Otherwise $d^{w_{\overline{G}}}(z_1, z_2) = 0$ leading to

$$\begin{aligned} d^{w_{\overline{G}}}(\tau_1, \tau_2) &\leq d^{w_{\overline{G}}}(\tau_1, z_1) + d^{w_{\overline{G}}}(z_1, z_2) + d^{w_{\overline{G}}}(z_2, \tau_2) \\ &< 2 \cdot 2^i + 0 + 2 \cdot 2^i = 4 \cdot 2^i \leq D \\ &\Rightarrow d^{w_{\overline{G}}}(\tau_1, \tau_2) < D \qquad \textit{a contradiction.} \end{aligned}$$

which contradicts the fact that $d^{w_{\overline{G}}}(\tau_1, \tau_2) = D$. Now consider the connected component C of $G[X]$ that contains the witness z_1. The component C contains at least one active client of $\mathcal{L}_i$, because either z_1 is an active client, or if z_1 is an open facility, it is connected to an active client. On the other hand, observe that at Line 7, the vertices of a new demand are connected in $G[X]$. Thus, since C contains an active client, it also contains a demand of $\mathcal{L}_i$ containing that client. The same argument holds for the connected component that contains z_2. Note that after the iteration, z_1 and z_2 become connected in the partial solution through τ_1 and τ_2. Therefore the number of connected components of X having a demand of $\mathcal{L}_i$, decreases by at least one. Such iterations can happen at most $|\mathcal{D}_i| - 1$ times. Hence the total connectivity cost of Type II iterations is at most $|\mathcal{D}_i| \cdot 12 \cdot 2^i$. By Claim 3.2, this is upper bounded by $12c(M_i)$; which completes the proof.

□

3.3.1 Bounding the Number of BFL Instances

Let I denote the number of BFL instances with at least one demand. A guessing argument shows that we may modify the algorithm such that $I = O(\log k)$, losing at most an extra constant factor in the competitive ratio. For an instance σ of the online network design problem, let OPT_σ denote the cost of an (offline) optimal solution for σ.

First observe that one may assume the value of OPT_σ is known within a two factor, if one is willing to lose a constant factor in the competitive ratio. More formally, consider the *opt-aware variant* of the problem in which a guess $\Phi \in \mathbb{R}_{>0}$ is a part of the offline input. An algorithm is β-competitive for this problem, if the cost of output of the algorithm is at most $\beta \cdot OPT_\sigma$ for instances σ where $\Phi \leq OPT_\sigma < 2\Phi$.

Lemma 3.6. *Given a β-competitive algorithm ALG for the opt-aware variant, one can derive a (3β)-competitive algorithm ALG$'$ for the general variant of the problem.*

Proof. The algorithm ALG$'$ runs as follows. Let σ be an instance of the problem and suppose the minimum weight of a vertex is one. The algorithm guesses OPT_σ iteratively, doubling the guess anytime the cost exceeds a β factor of the guess. Starting from $i = 0$, in iteration i we set $\Phi = 2^i$. We initialize an opt-aware instance σ' by giving Φ as the guess, and giving the part of online input seen so far to ALG. We continue forwarding the arriving online input to ALG. In any step, if the cost of the (partial) solution of ALG was exceeding $\beta \cdot 2\Phi$, we terminate the iteration and we continue to iteration $i+1$ by doubling the guess.

Let i be the index of the last iteration. Since ALG is β-competitive, we have $\mathit{OPT}_\sigma \geq 2^i$; otherwise we would have finished processing the online input in previous iterations. Observe that in the last iteration, the cost of ALG is at most $\beta \cdot \mathit{OPT}_\sigma$. The cost we incur in an iteration $j < i$ is at most $\beta \cdot 2^{j+1}$. Thus the total cost of algorithm ALG$'$ is at most

$$\beta \cdot \mathit{OPT}_\sigma + \sum_{0 \leq j < i} \beta \cdot 2^{j+1} \leq \beta \cdot \mathit{OPT}_\sigma + \beta \cdot 2^{i+1} \leq 3\beta \cdot \mathit{OPT}_\sigma$$

□

By Lemma 3.6, we may assume that the cost of an optimal solution is known within a two factor. Let Φ denote the guess. Now consider Line 5 in Algorithm 3. By Lemma 3.2, any feasible solution contains a path of length at least $\frac{D}{2}$ attached to τ_1. Thus $D \leq 2\mathit{OPT} \leq 4\Phi$. Therefore

$$\textit{For any } i > \log_2(4\Phi)\textit{, we never give a demand to } \mathcal{L}_i. \qquad (\bar{I})$$

Consider a counter J initialized to zero at the beginning of Algorithm 3. We increment J anytime we enter the "while" loop at Line 3. Thus at the end of the algorithm, J shows the total number of iterations of the algorithm. We first claim that the final value of J is at most $k-1$. Recall that at Line 5, $\gamma(\tau_1)$ and $\gamma(\tau_2)$ are connected components of the partial solution X. By buying the path between τ_1 and τ_2, we reduce the number of connected components of the partial solution X by one. Every connected component has at least one terminal, thus the total number of iterations of Algorithm 3 is at most $k-1$.

We now modify the algorithm such that in every iteration, after Line 5, we stop the

iteration if $D \leq \frac{\Phi}{J}$. Such an iteration is said to be *clipped*. Thus we now add a demand to a BFL instance only in non-clipped iterations. Since the maximum possible value of J is $k-1$, we have

$$\textit{For any } i < \log_2(\frac{\Phi}{k}) - O(1), \textit{ we never give a demand to } \mathcal{L}_i. \qquad (\underline{I})$$

Finally, we have the ingredients for proving the first main theorem.

of Theorem 3.1. By Equations $\overline{I}$ and $\underline{I}$, in the modified algorithm, the number of BFL instances with at least one demand is at most $I \leq O(\log_2(4\Phi) - \log_2(\frac{\Phi}{k})) = O(\log k)$. By Lemma 3.5, the total cost of non-clipped iterations is at most $O(I)\alpha_{BFL} = O(\log k)\alpha_{BFL}$ where $\alpha_{BFL} = O(\log(k)\log(n))\mathit{OPT}$ ([AAA+06]). This leads to Theorem 3.1 since the total cost we incur in all clipped iterations is at most $\sum_{J=1}^{k-1} \frac{\Phi}{J} \leq O(\log k)\Phi \leq O(\log k)\mathit{OPT}$. □

3.4 Online Network Design in Graphs Excluding a Fixed Minor

Before we describe the deterministic algorithm for H-minor-free graphs, we need to introduce some notation. In the rest of the section, unless specified otherwise, all graphs are H-minor-free for a fixed graph H.

A painting is said to be r-*uniform* if it is a union of disks whose radii are r. When the exact radius is not important, we may refer to an r-uniform painting as a uniform painting.

Disk Fitting. We use a *disk fitting* process called DISKFIT (pseudocode in Process 1)

repeatedly to construct a painting. Recall that a vertex v is simple, if $v \in V(G)$. Let $\mathcal{L}$ be a painting of a contraction $\overline{G}$. Given a connected set S of simple vertices and a radius r, the DISKFIT process tries to contract S and add a disk of radius r centered at the resulting super-vertex in $\mathcal{L}$. We call this a trial. The trial might either be successful, or fail for multiple reasons that we describe below.

If for any reason the trial finishes unsuccessfully, $\mathcal{L}$ and $\overline{G}$ will remain unchanged and the process returns a vertex called a *witness*. Let us now describe a trial. If for a vertex $v \in S$, v is already contracted in $\overline{G}$ or $\mathcal{L}(v) > 0$, the trial fails and the corresponding witness is defined as the (super-)vertex $u \in V(\overline{G})$ for which $v \in \gamma(u)$. Otherwise, the trial contracts S to a vertex s. Let $\overline{G}'$ denote the resulting graph. Note that $\mathcal{L}$ is still a valid painting for $\overline{G}'$. Let p be a disk of radius r centered at s in $\overline{G}'$. If the union of p and $\mathcal{L}$ is feasible in $\overline{G}'$, we augment $\mathcal{L}$ to $\mathcal{L} + p$ (after changing the underlying graph to $\overline{G}'$) and the trial terminates successfully. Otherwise, p and $\mathcal{L}$ are said to *intersect* at the vertices where $\mathcal{L} + p$ is infeasible. The trial terminates unsuccessfully by reporting an infeasible vertex in $\mathcal{L} + p$ as the witness, breaking ties arbitrarily.

Process 1. [Disk Fitting]

Input: A painting $\mathcal{L}$ of a contraction $\overline{G}$, a set of simple vertices S connected in G, and a radius r.

1: **if** a vertex $v \in S$ is contracted in $\overline{G}$ or is (partially) painted in $\mathcal{L}$ **then**

2: Report the vertex $u \in V(\overline{G})$ whose $\gamma(u)$ contains v. Terminate unsuccessfully.

3: **else**

4: Contract S. Let p be a disk of radius r centered at the resulting super-vertex.

5: **if** $\mathcal{L} + p$ is feasible **then**

6: Set $\mathcal{L} = \mathcal{L} + p$. Terminate successfully.

7: **else**

8: Report a vertex u for which $\mathcal{L}(u) + p(u) > w(u)$. Terminate unsuccessfully.

Binding Spiders. Let $\mathcal{L}$ be the union of a set of disks on a contraction $\overline{G}$. Suppose v is a witness reported by the DISKFIT process in an unsuccessful trial for adding a disk over a set of simple (and connected) vertices S. Let L_v denote the centers of the disks whose boundary or continent contains v. (Note that if a vertex of S is already contracted in $\overline{G}$, then the witness may be inside a disk.) A binding spider w.r.t. to the witness v reported by DISKFIT process is a spider in $\overline{G}$ centered at v and connected with shortest paths w.r.t. $w_{\overline{G}}$ to (i) every vertex in L_v, and (ii) the vertex $u \in V(\overline{G})$ with $\gamma(u) \cap S \neq \phi$ that is closest to v. The following shows that the cost of buying a binding spider depends only on the degree of the center.

Lemma 3.7. *Let Υ be a binding spider w.r.t. an unsuccessful trial for putting a disk of radius r, centered at a set S, in an r-uniform painting. If the center of spider has degree d, then $w_{\overline{G}}(\Upsilon) \leq d \cdot r + w(S \cap \gamma(\Upsilon))$.*

Proof. Let v be the center of Υ. Recall that v is a witness of failure reported by Process 1. If v is in the continent of a disk, then v is in S and Υ is simply a path between v and the center of the disk. In this scenario d is one. Note that d cannot be zero since we assume the terminals are distinct. The distance between the center of a disk and a vertex in its continent is at most the radius of the disk. Thus in this scenario the weight of Υ is bounded

by r, as desired.

Now suppose L_v is the set of centers of disks whose boundary contains v. For a center $u \in L_v$, let m_u denote the closest neighbor of v to u (the distance is w.r.t. to the underlying contraction of the painting). Recall that m_u is inside the disk centered on u. In the rest of proof assume $v \notin S$; the argument when $v \in S$ is similar. Let p be a shortest path connecting v to a vertex $s \in S$. Let m_S denote the neighbor of v in p. The binding spider connects v to every $u \in L_v$ and to s through p. Let us denote the vertex weights of the underlying graph by w. The weight of the binding spider comprises of $w(v)$, $d(u, m_u)$ for every $u \in L_v$, and $d(s, m_S)$. Note that $d(u, m_u) \leq r$ for every $u \in L_v$ since m_u is inside the disk centered at u. On the other hand, after contracting S, putting a disk of radius r causes an infeasibility at v. Thus $d(s, m_S) \leq r + w(s)$.

The degree of the center of the spider is $d = |L_v| + 1$. Since v is on the boundary of the disks, for every $u \in L_v$, $\mathcal{L}(v, \theta^u) = r - (d(u, v) - w(v)) = r - d(u, m_u)$. Now suppose we contract S to s^*. Let D be a disk of radius r centered at s^*. The painting $D + \mathcal{L}$ is infeasible such that $w(v) < \mathcal{L}(v) + D(v)$. Thus, v is either inside the disk D or is on its boundary. Thus $D(v) \leq r - d(s^*, m_S) = r - d(s, m_S) + w(s)$. Therefore,

$$
\begin{aligned}
w(\Upsilon) &\leq w(v) + \sum_{u \in L_v} d(u, m_u) + d(s, m_S) \\
&\leq (\mathcal{L}(v) + D(v)) + \sum_{u \in L_v} d(u, m_u) + d(s, m_S) \\
&\leq \left(\sum_{u \in L_v} (r - d(u, m_u)) + (r - d(s, m_S) + w(s)) \right) + \sum_{u \in L_v} d(u, m_u) + d(s, m_S) \\
&= (|L_v| + 1)r + w(s) \leq d \cdot r + w(S \cap \Upsilon)
\end{aligned}
$$

□

In the algorithm, we only call the DISKFIT process for a set S if S is already in the output X. Therefore Lemma 3.7 implies that the cost of buying a binding spider is at most $w(\gamma(\Upsilon)\backslash X) \leq w_{\overline{G}}(\Upsilon) - w(S \cap \gamma(\Upsilon)) \leq d \cdot r$.

Recall that for a fixed graph H, the average degree of an H-minor-free graph is bounded by some constant c_H. (It is shown in [Mad67, Kos84] that $c_H = O(h\sqrt{\log h})$ where $h = |V(H)|$.) For simplicity of notation, we introduce two constants $\alpha = \max\{c_H, 3\}$ and $\mu = 2\alpha$. The following lemma together with Lemma 3.7 provides a means to charge the cost of spiders *with sufficiently large number of legs* to (the radii of) disks in our analysis. Note that if a binding spider has more than two leaves, its center has to be on the boundary of multiple disks.

Lemma 3.8. *Let $\mathcal{L}$ be the union of N disks on an H-minor-free graph. For a vertex v, let $\eta(v)$ denote the number of disks whose boundary contains v. Then,*

$$\sum_{v|\eta(v)\geq\alpha} \eta(v) \leq \alpha \cdot N.$$

Proof. For every disk, contract the continent of the disk to its center. Observe that if a vertex v is on the boundary of a disk, by Proposition 3.6, it is not contracted. The center of a disk is adjacent to a vertex v if and only if v is on the boundary of the original disk. Remove all the edges but those between a center of a disk and a vertex on the boundary. Now the degree of a vertex v which is not the center of a disk is exactly $\eta(v)$. Remove a vertex v if $\eta(v) < \alpha$ and it is not the center of a disk. The resulting graph is a bipartite

graph with centers of the disks at one side and the vertices at the boundaries on the other side. The number of edges of the resulting graph is the left hand side of the inequality, i.e., $\sum_{v|\eta(v)\geq\alpha}\eta(v)$. On the other hand, the graph is still an H-minor-free graph, thus the average degree of vertices is at most α. The minimum degree of a vertex which is not the center of a disk is α. Therefore the average degree of the centers of the disks is at most α, leading to $\sum_{v|\eta(v)\geq\alpha}\eta(v)\leq\alpha\cdot N$. □

3.4.1 Algorithm for H-Minor-Free Graphs.

We are now ready to describe algorithm MFSFALG . For every $i\in\mathbb{Z}$, the algorithm keeps a painting $\mathcal{L}_i$ on a contraction of G. Throughout the algorithm, the changes to the painting $\mathcal{L}_i$ are made only by adding disks of radius 2^i using the DISKFIT process. Therefore, $\mathcal{L}_i$ is a 2^i-uniform painting comprising non-overlapping disks of radius 2^i.

Initially, $\mathcal{L}_i$'s are empty paintings on G. At a time step h, let f_h denote the cumulative function. We iteratively augment the solution until f_h is satisfied. In every iteration, let X denote the current partial solution and let $\overline{G}$ denote a contraction obtained by contracting every connected component of $G[X]$. Let f be the corresponding contracted function. By Proposition 3.3, there are at least two terminals in $\overline{G}$. Let (τ_1,τ_2) denote the closest pair of terminals of f. Let $D=d^{w_{\overline{G}}}(\tau_1,\tau_2)$. We first buy the shortest path in $\overline{G}$ connecting τ_1 to τ_2. Now consider the integer i such that $\mu\cdot 2^i<D\leq\mu\cdot 2^{i+1}$. Using the DISKFIT process, we try putting a disk of radius 2^i centered at $\gamma(\tau_1)$ in $\mathcal{L}_i$. If the trial is unsuccessful, we do the same process for $\gamma(\tau_2)$. If both trials are unsuccessful, let c_1 and

c_2 denote the corresponding witnesses of failure. We buy the binding spiders w.r.t. c_1 and c_2. A pseudo-code for the algorithm fallows.

Algorithm 4 Online Network Design in Graphs with an Excluded Minor

Input: An H-minor-free graph G, a node-weight function w, and an online stream of proper functions $q_1, q_2, \ldots$.

Output: A set X such that $G[X]$ contains an edge of the cut $(S, V \setminus S)$ for every $S \subset V$ with $q_j(S) = 1$ for some j.

Offline Process:

1: For every integer i, initialize a painting $\mathcal{L}_i$ on G.

2: Initialize X to an empty set. Set $\mu = 2\alpha$.

Online Scheme; assuming a proper function q_h is arrived:

1: Let f_h be the cumulative function.

2: **while** X does not satisfy f_h **do**

3: Form $\overline{G}$ from G by contracting components of $CC(X)$. Let f denote the contracted function.

4: Let (τ_1, τ_2) denote the closest pair of terminals of f. Let $D = d^{w_{\overline{G}}}(\tau_1, \tau_2)$.

5: Buy the shortest path in $\overline{G}$ connecting τ_1 and τ_2.

6: Consider the integer i such that $\mu 2^i < D \leq \mu 2^{i+1}$.

7: Try putting a disk of size 2^i centered at $\gamma(\tau_1)$ in $\mathcal{L}_i$ using Process 1. If it was unsuccessful call Process 1 for a disk centered at $\gamma(\tau_2)$.

8: **if** both trials were unsuccessful **then**

9: Buy two binding spiders w.r.t. the corresponding witnesses of failure.

3.4.2 Analysis.

Let I denote the number of paintings with at least one disk. We will now show that the competitive ratio of MFSFALG is $O(I)$. Indeed, the same argument as that of Section 3.3, shows that the algorithm can be slightly modified such that $I = O(\log k)$, thereby proving Theorem 3.2. Let *OPT* denote the weight of an optimal solution to the network design problem. Let C_i denote the centers of disks in $\mathcal{L}_i$. First, we show the relationship between the paintings and *OPT*.

Lemma 3.9. *For every i, the total radii of disks in $\mathcal{L}_i$ is a lower bound for the optimal solution.*

Proof. Let C_i denote the centers of disks in $\mathcal{L}_i$. Observe that when a disk is placed in $\mathcal{L}_i$, the DISKFIT process never contracts the vertices inside or on the boundary of the disk in future. Let X^* denote the optimal solution for the network-design problem, i.e., $w(X^*) = \mathit{OPT}$. We prove the lemma by showing that for every $\tau \in C_i$, X^* contains a path from τ to a vertex on the boundary of the disk centered at τ. This completes the proof since for every $\tau \in C_i$, at least r units of the total weight of vertices of X^* will be colored by the color θ^τ. Thus $w(X^*)$ is at least the total radii of disks in $\mathcal{L}_i$.

Now consider an arbitrary $\tau \in C_i$. Consider the iteration in which the disk centered at τ is placed in $\mathcal{L}_i$. Let X and f_h respectively denote the partial solution and the cumulative function at the start of the iteration. Let $\overline{G}$ denote the contraction obtained by contracting connected components of $G[X]$. Let f be the contracted function. Since

in this iteration a disk is placed centered at τ in $\mathcal{L}_i$, then for a terminal τ' of f, τ and τ' are the closest pair of terminals of f (see Line 4 in Algorithm 4). Let D be the distance between τ and τ'. Recall that $\mu 2^i \leq D < \mu 2^{i+1}$.

Now consider a disk of radius 2^i centered at τ in $\overline{G}$. Let Q be the continent of the disk. At this iteration, there are no terminals in $Q \backslash \{\tau\}$ in $\overline{G}$ since the closest terminal to τ is the terminal τ' which is at least $\mu 2^i$ far from τ. Thus by Lemma 3.2, X^* contains a path from τ to a vertex in $\delta(Q)$, which completes the proof. □

We are now ready to prove the main lemma.

Lemma 3.10. *The total cost incurred by the algorithm is at most $O(I) \cdot$ OPT.*

Proof. Let X denote the partial solution at an iteration of the algorithm. For every $v \in V(G)$, let $\tilde{w}(v)$ denote the *cost of buying* a vertex v w.r.t. X, i.e., in $\tilde{w}(v) = w(v)$ if $v \notin X$ and let $\tilde{w}(v) = 0$ if $x \in V$. Let (τ_1, τ_2) denote the closest pair of terminals. An iteration is of Type i if $\mu 2^i < D \leq \mu 2^{i+1}$ where $D = d^{w_{\overline{G}}}(\tau_1, \tau_2)$ is the length of the shortest path between τ_1 and τ_2. We show that the cost of the vertices purchased in all iterations of Type i is bounded by $O(|C_i| \cdot 2^i) = O(\mathit{OPT})$. In the rest of proof, we only consider Type i iterations for an arbitrary i.

Recall that in every iteration, we buy a path connecting τ_1 and τ_2 and we may additionally buy two binding spiders. A binding spider Υ is said to be *expensive* if $\tilde{w}(\Upsilon)$, the cost of buying it, is strictly more than $\alpha \cdot 2^i$. [2] We classify iterations as follows. For

[2] Recall that the constant α is roughly the average degree of graphs that belong to the class of the H-minor free graphs.

each of the cases, we can show that the increase in the weight of X can be charged to the radii of the disks in $\mathcal{L}_i$.

1. Process DISKFIT successfully adds a disk of radius 2^i centered at either τ_1 or τ_2.

2. Process DISKFIT is unsuccessful and neither of the binding spiders is expensive.

3. Process DISKFIT is unsuccessful and at least one binding spider is expensive.

Case 1

The first case is the simplest since we explicitly add a disk of radius $2^i = \Omega(d^{\tilde{w}}(\tau_1, \tau_2))$ to $\mathcal{L}_i$. We charge the cost of the iteration, which is at most

$$d^{w_{\overline{G}}}(\tau_1, \tau_2) \leq \mu \cdot 2^{i+1},$$

to the new disk. A disk is charged in the first case only once at the time of creation. Thus the total cost of iterations of Case 1 is at most $2\mu \cdot |C_i| \cdot 2^i = O(\mathit{OPT})$.

Case 2

In the second case, the total cost of buying the shortest path and the two spiders is at most

$$\begin{aligned} & d^{w_{\overline{G}}}(\tau_1, \tau_2) + w(\gamma(\Upsilon_1)\backslash X) + w(\gamma(\Upsilon_2)\backslash X) \\ & \leq d^{w_{\overline{G}}}(\tau_1, \tau_2) + \alpha \cdot 2^i + \alpha \cdot 2^i \\ & \leq \mu \cdot 2^{i+1} + 2\alpha \cdot 2^i = O(1) \cdot 2^i, \end{aligned} \tag{3.1}$$

i.e., $O(1)$ times the radius of a single disk in $\mathcal{L}_i$. We will now show that by adding the two binding spiders, we decrease the number of connected components of $G[X]$ containing at least one terminal of C_i by at least one (see Figure 3.3). Then, we can conclude that there are at most $|C_i|$ iterations of Case 2. By Eqn. 3.1, the total cost of the iterations of Case 2 is then bounded by $O(|C_i|) \cdot 2^i = O(\mathit{OPT})$.

Let X_1 and X_2 respectively denote the set of nodes in the output set X before and after a single Case 2 iteration. Both Υ_1 and Υ_2 have at least one leaf which is the center of a disk; let c_1 and c_2 be such leaves. Since we buy the spiders and the path between τ_1 and τ_2, the two terminals c_1 and c_2 become connected in $G[X_2]$. So, we need to show that c_1 and c_2 are not connected in $G[X]$. Suppose not; then $d^{w_{\overline{G}}}(c_1, c_2) = 0$. Now consider the shortest path P_1 between τ_1 and c_1 w.r.t. $w_{\overline{G}}$. Clearly $w_{\overline{G}}(P_1) \leq w(\gamma(\Upsilon_1) \backslash X)$. Further, since Υ_1 is cheap, $w_{\overline{G}}(P_1) \leq \alpha \cdot 2^i$. We can define a similar (τ_2, c_2)-path P_2 with the same bound. Therefore we have a (τ_1, τ_2)-path of length

$$\begin{aligned} d^{w_{\overline{G}}}(\tau_1, \tau_2) &\leq d^{w_{\overline{G}}}(\tau_1, c_1) + d^{w_{\overline{G}}}(c_1, c_2) + d^{\tilde{w}}(c_2, \tau_2) \\ &= w_{\overline{G}}(P_1) + 0 + w_{\overline{G}}(P_2) \leq \mu \cdot 2^i, \end{aligned}$$

which contradicts the initial assumption $d^{w_{\overline{G}}}(\tau_1, \tau_2) > \mu \cdot 2^i$.

Case 3

The most complicated case is the third scenario involving expensive spiders. This case requires us to use the properties of H-minor-free graphs, in particular that the average

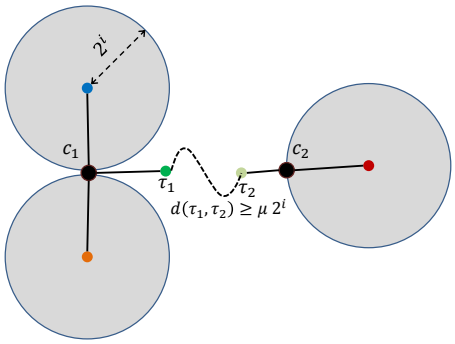


Figure 3.3: Binding Spiders

In this figure the disk fitting process is unsuccessful for both τ_1 and τ_2, reporting witnesses c_1 and c_2. The three straight lines to the left form the first binding spider Υ_1 centered at c_1. The two straight lines to the right form the second binding spider Υ_2 centered at c_2.

degree of vertices in G is α, to amortize the cost of the spiders to the radii of the disks[3]. Before giving a formal proof, let us sketch the main ideas involved. If a binding spider is expensive, Lemma 3.7 implies that the degree of the center of the spider is greater than the average degree of graph G. We charge the weight of the spider to the degree of its center. We will see that a vertex can be the center of at most one purchased expensive spider; thus the total charge is proportional to the total degree of the centers of spiders with degree at least α. By Lemma 3.8, the total degree of such vertices is bounded by the number of disks up to a constant factor. Therefore the cost of expensive spiders purchased at each painting is at most a constant factor of the total radii of the disks in the painting.

We now give a formal proof for the third case. Assume w.l.o.g. that the cost of Υ_1 is at least that of Υ_2. Let v be the center of Υ_1. Let L denote the centers c of disks such

[3]Caveat: If the average degree is less than 3, $\alpha = 3$

that (i) c is a leaf of Υ_1, and (ii) we have not purchased the shortest path between v and c in previous iterations. Note that Υ_1 is expensive, i.e., $w_{\overline{G}}(\Upsilon_1) > \alpha \cdot 2^i$. By Lemma 3.7, Υ_1 should have at least $\alpha + 1$ legs that have not been purchased yet. Since τ_1 is one of the leaves, it follows that $|L| \geq \alpha$. We say Υ_1 is the *primal spider* of this iteration. By Lemma 3.7, the cost of buying Υ_1 (and so that of Υ_2) is bounded by $(|L| + 1) \cdot 2^i$. The total cost of the iteration is

$$\begin{aligned} & d^{w_{\overline{G}}}(\tau_1, \tau_2) + w(\gamma(\Upsilon_1) \backslash X) + w(\gamma(\Upsilon_2) \backslash X) \\ & \leq \mu \cdot 2^{i+1} + 2(|L| + 1) \cdot 2^i \\ & \leq (2|L| + 2\mu + 2) \cdot 2^i \leq 7|L| \cdot 2^i, \end{aligned} \tag{3.2}$$

where the last inequality follows from $|L| \geq \alpha \geq 3$ (recall $\mu = 2\alpha$).

For a vertex v, let $\eta(v)$ denote the number of disks in $\mathcal{L}_i$ whose boundary contains v. Eqn. 3.2 shows that we can charge the cost of a Case 3 iteration to the number of *new* legs attached to v (by a factor of $7 \cdot 2^i$). In other words, the total cost of Case 3 iterations for which v is the center of the primal spider, is bounded by $7\eta(v) \cdot 2^i$. Hence, we charge the cost of such iterations to $\eta(v)$. Therefore the total cost of iterations of Case 3 is at most $7 \cdot 2^i \sum_{v|\eta(v) \geq \alpha} \eta(v)$. By Lemma 3.8, it is bounded by $7\alpha \cdot |C_i| \cdot 2^i = O(\mathit{OPT})$.

□

3.5 Online Prize Collecting Network Design

Prize collecting Steiner problems have been extensively considered in the offline model. In the EW variant, the current best approximation ratio for offline PCST is 1.9672 by Archer *et al* [ABHK11] improving upon a primal-dual $(2 - \frac{1}{n-1})$-approximation algorithm of Goemans and Williamson [GW95]. In the more generalized variant of the problem, prize-collecting Steiner forest (PCSF)[4], the best approximation ratio is 2.54 by Hajiaghayi and Jain [HJ06]. More generalized variants of the problem, including the higher connectivity demands, has been also studied (see e.g. [SSW07, HKKN12]). In the NW variant, Chekuri *et al* [CEV12b] give a general $O(\log(n))$-approximation algorithm for prize-collecting Steiner problems with higher connectivity demands. Various primal-dual techniques have been also developed for solving the more special cases, PCST and PCSF (see e.g. [GMNS99, MR07, BHL13, KSS13]).

The online Steiner tree (ST) problem was originally considered in the EW model, where Imase and Waxman [IW91] showed that a natural greedy algorithm has a competitive ratio of $O(\log n)$, which is optimal up to constants. This result was generalized to the online EW Steiner forest (SF) problem by Awerbuch *et al* [AAB04], who showed that the greedy algorithm has a competitive ratio of $O(\log^2 n)$. This result was later improved by Berman and Coulston [BC97] to $O(\log n)$. All the above results can be reproduced using dual-fitting techniques (though the original expositions relied on combinatorial ar-

[4] In prize-collecting Steiner forest problem, given a set of pair of vertices each with a penalty, for each pair one needs to either connect the pair or pay the corresponding penalty.

guments). This immediately shows that the competitive ratios hold against the respective fractional optimums as well. Very recently, Qian and Williamson [QW11] initiated the study of online PC Steiner problems by providing an $O(\log n)$-competitive algorithm for the online EW-PCST problem. The analysis of this algorithm is quite complicated and uses a dual moat growing approach that is typical in offline primal dual algorithms pioneered by Agrawal *et al* [AKR95] of Goemans and Williamson [GW95] and an amortized accounting scheme due to Berman and Coulston [BC97].

In contrast to EW problems, progress in online NW Steiner problems has been relatively slow. Note that edge weights can be represented by node weights but not vice-versa; so, NW problems are strictly more general. In fact, the NW-ST problem generalizes the set cover problem, for which the first online algorithm with a poly-logarithmic competitive ratio was obtained by Alon *et al* [AAA+09]. They introduced an online adaptation of the classical LP relaxation technique, which has since been used extensively in online optimization (see, e.g., the survey by Buchbinder and Naor [BN09a]). In particular, Naor *et al* [NPS11] used this technique in conjunction with structural properties of the NW-ST problem to give an $O(\log^3 n)$-competitive algorithm for the online NW-ST problem. They left the online NW-SF problem open, which was resolved very recently by Hajiaghayi *et al* [HLP13] who obtained an identical competitive ratio of $O(\log^3 n)$ for the SF problem. However, there is a crucial difference between these two results. Whereas Hajiaghayi *et al* use a dual-fitting approach, which shows that the competitive ratio holds for the fractional optimum as well, Naor *et al* use structural properties of *integral* Steiner trees. Therefore,

their results do not hold for the fractional optimum. As described below, this distinction turns out to be important in our work.

Techniques and Contributions.

Our first contribution is a simple but subtle reduction of online prize-collecting Steiner problems to their respective non-prize-collecting fractional variants losing a factor of $O(\log n)$ in the competitive ratio. This reduction is quite generic and can be applied for more general problems than ST and SF. Indeed, this approach can be applied to any problem which demands $\{0, 1\}$-connectivity on a family of cuts. This setting includes the T-join problem and group Steiner tree (GST)/ group Steiner forest (GSF) problems as special cases (see Theorem 3.5 for a formal description). All these problems are instances of covering problems. In our reduction, we run the algorithm of Buchbinder *et al* [BN09a] for solving covering problems *in parallel* with an algorithm for the non-prize-collecting variant (as a black box). At each online step, we first generate an online competitive fractional solution. Then we use the fractional solution to reveal a modified demand to the non-prize-collecting black box and finally output an integral solution for the prize-collecting problem. This reduction is oblivious of the cost model of the input graph, and hence can be applied to both EW and NW problems, thereby yielding online algorithms for various prize-collecting Steiner connectivity problems with poly-logarithmic competitive ratios. Indeed one main obstacle to solving the online prize-collecting variants of these problems is that the known rounding techniques do not seem to be effective for

rounding the standard linear relaxation of the problems. For example for the online NW prize-collecting ST, it is not known whether one can solve the fractional LP for PCST and then round it online. This may have been the reason that the only PC result known before this work, i.e, the Qian-Williamson algorithm [QW11] for online EW PCST, is quite sophisticated. A summary of results that follow from our reduction are shown in Table 3.1.

Next, we focus on the online EW-PCST and NW-PCST problems. For these problems, the generic reduction yields competitive ratios of $O(\log^2 n)$ and $O(\log^4 n)$ respectively. We improve both these competitive ratios by a logarithmic factor. For the EW problem, this matches the competitive ratio of the Qian-Williamson algorithm [QW11] and is optimal up to constants. Further, our analysis is extremely simple and uses a natural dual-fitting approach (see Section 3.5.4).

Theorem 3.3 (also in [QW11])**.** *The online edge-weighted prize-collecting Steiner tree problem admits an $O(\log(n))$-competitive algorithm.*

For the online NW-PCST problem, we use the generic reduction, but give an online algorithm for the NW-ST problem that has the optimal competitive ratio of $O(\log^2 n)$ *against the fractional objective*. While it is relatively straightforward to use a re-scaling argument for improving the integral competitive ratio of the algorithm of Naor *et al* [NPS11] by a logarithmic factor, a similar improvement for the (fractionally compet-

itive) algorithm of Hajiaghayi *et al* [HLP13] has fundamental difficulties. So, we first design a novel dual-fitting analysis of the algorithm of Naor *et al*, thereby proving that the competitive ratio of the algorithm now holds against the fractional optimum. However, using this new analysis, we can no longer use the re-scaling argument that improved a logarithmic factor for the integral algorithm. To overcome this difficulty, we introduce a new concept that we call *dual averaging*.

The celebrated moat-growing method of Agrawal, Klein, and Ravi [AKR95] and Goemans and Williamson [GW95] has been extensively studied for various Steiner connectivity problems. A general pattern in different variants of this method is as follows. We start by growing a moat over every terminal. When two moats collide, this signifies an opportunity for connecting the terminals at the center of the moats thus merging the moats. Since the moats have a dual vector interpretation, by weak duality one can charge the cost of an algorithm to the growth of the moats. When used in a dual-fitting argument, the crux of the analysis is to show that the dual moats do not intersect. In the NW setting, this turns out to be even more difficult since the next terminal that arrives might quickly collide with polynomially many disks at the same vertex thereby making merges of the moats infeasible. We circumvent this problem by introducing a *thinness* factor for the moats. For $\tau \in (0, 1]$, a τ-thin dual moat is obtained by scaling the dual variables corresponding to a moat by the factor τ. This allows us to have several *overlapping* moats; this is in contrast to standard dual-fitting methods in which the dual moats are disjoint. The strength of this natural modification is that one can exploit it to show that

certain *structural properties* may hold on *average*, although they may not hold for every dual moat independently. For example, consider the feasibility of a dual vector. A reader familiar with the standard dual program for ST may recall that every moat has a load on vertices inside or on the boundary of a moat. It often happens that for every vertex, a few moats have a high load on the vertex while the load of the rest of the moats is negligible. However, for different vertices, the moats with a high load might be different. By considering the proper thinness for the moats, one can balance the loads simultaneously for every vertex to ensure feasibility of the dual moats. We refer the reader to Section 3.5.3 for a formal discussion of this approach.

Theorem 3.4. *The online node-weighted Steiner tree problem admits an $O(\log^2(n))$-competitive algorithm. Moreover, the competitive ratio holds with respect to the optimal fractional solution.*

Note that the above algorithm is optimal in its competitive ratio since there is a known lower bound of $\Omega(\log^2 n)$ [AAA+09, Kor05] for the online set cover problem, which is a special case of the online NW-ST (and thus online NW-PCST) problem.

Applying the reduction in Theorem 3.5 to the algorithm in Theorem 3.4, we obtain an improved competitive ratio for the online NW-PCST problem. Furthermore, very recently, Hajiaghayi *et al* [HLP13] gave an algorithm with a tight competitive ratio of $O(\log(n))$ for the NW ST problem for planar graphs and more generally graphs excluding a fixed graph as a minor. Their results are based on a primal-dual technique and thus

the competitive ratio is w.r.t. the fractional optimum. Therefore we get the following results for the prize-collecting counterparts of these problems.

Corollary 3.3. *The online NW prize-collecting ST problem admits an $O(\log^3(n))$-competitive algorithm in general graphs. When restricted to graphs excluding a fixed graph as a minor, the problem admits an $O(\log^2(n))$-competitive algorithm.*

In the offline paradigm, algorithms for prize-collecting problems are used at the heart of algorithms for other connectivity problems. An important branch of such problems are *budgeted*[5] and *quota*[6] problems. The key to solving these problems is to design Lagrangian multiplier preserving (LMP) approximation algorithms for PCST. Indeed very recently, Konemann *et al.* [KSS13] gave an LMP $O(\log(n))$-approximation algorithm for offline NW PCST. Therefore, a natural question is whether one can hope for LMP *online* algorithms with poly-logarithmic competitive ratio. We show this is impossible even for the case of online edge-weighted Steiner tree. In Section 3.5.4.1, we show a lower bound of $\Omega(n)$ for the LMP competitive ratio of (randomized) algorithms for online EW-PCST.

3.5.1 Problem Formulation

Let $G = (V, E)$ be an undirected graph. For a set $S \subseteq V$, let $\delta(S) \subseteq V \backslash S$ denote the neighbors of S. Given a $\{0,1\}$-function $f : 2^V \to \{0,1\}$, a *demand system* with respect to f is defined as the following system of inequalities over a set of variables $\mathbf{x}(v)$

[5] Given an upper limit on the weight, the goal is to maximize the prize of the connected vertices.

[6] We want the minimum-weight subgraph that gathers at least a given amount of prize.

for every $v \in V$.

$$\sum_{v \in \delta(S)} \mathbf{x}(v) \geq f(S) \qquad \text{for every } S \subset V, S \neq \phi$$

$$\mathbf{x}(v) \in [0,1]$$

We say a $\{0,1\}$-function f is *efficient* if the following properties hold: (i) $f(\phi) = f(V) = 0$; (ii) for every $S \subseteq V$, $f(S)$ can be computed in polynomial time; and (iii) there exists a separation oracle such that for any vector $\mathbf{x} \in [0,1]^V$, it outputs a set S with $\sum_{v \in \delta(S)} \mathbf{x}(v) < f(S)$ iff there is a violated constraint. The oracle should run in polynomial time w.r.t. $|V|$.

As we will see, the last two properties are required so that our reduction runs in polynomial time. Although the first property is not necessary, it makes the demand system well defined even if we consider the constraints corresponding to $S = \phi$ and $S = V$ in the system. In the rest of the section, we use $\mathcal{F}$ to refer to a family of efficient functions. Furthermore, we use $\mathcal{G}$ to refer to a family of node-weighted graphs $G = (V, E, w)$ where for $v \in V$, $w_v \in \mathbb{R}_{\geq 0}$.

Problem.

A *network design problem* (ND) with respect to $\mathcal{F}$ and $\mathcal{G}$ is defined as follows. Let $G = (V, E, w)$ be a node-weighted graph in $\mathcal{G}$. Given a sequence of functions $f_1, \ldots, f_k \in \mathcal{F}$, the goal is to find a minimum-weight vector $\mathbf{x} \in \{0,1\}^V$ that simultaneously satisfies the demand systems for every function f_i. The ND problem can be formulated as the following integer program (IP). Throughout the section, for an integer k, let $[k]$ denote

$\{1, \ldots, k\}$.

$$\text{minimize} \sum_{v \in V} w_v \mathbf{x}(v) \tag{$\mathbb{ND}$}$$

$$\forall S \subseteq V, i \in [k] \sum_{v \in \delta(S)} \mathbf{x}(v) \geq f_i(S)$$

$$\mathbf{x}(v) \in \{0, 1\}$$

Given a feasible solution $\mathbf{x}$, we define *cost*$(\mathbf{x})$ as the total *weight* of $\mathbf{x}$, i.e, $\sum_{v \in V} w_v \mathbf{x}(v)$.

In a *prize-collecting network design problem* (PCND) w.r.t. $\mathcal{F}$ and $\mathcal{G}$, we are given $G = (V, E, w) \in \mathcal{G}$ and a sequence of *demands* $(f_1, \pi_1), \ldots, (f_k, \pi_k)$ where $f_i \in \mathcal{F}$ and $\pi_i \in \mathbb{R}_{\geq 0}$. For every demand (f_i, π_i), we need to either satisfy the demand system w.r.t. f_i or pay the *penalty* π_i. In other words, we need to find an optimal solution to the following IP.

$$\text{minimize} \sum_{v \in V} w_v \mathbf{x}(v) + \sum_{i \in [k]} \pi_i \mathbf{z}(i) \tag{$\mathbb{PCND}$}$$

$$\forall S \subseteq V, i \in [k] \sum_{v \in \delta(S)} \mathbf{x}(v) \geq f_i(S)(1 - \mathbf{z}(i))$$

$$\mathbf{x}(v), \mathbf{z}(i) \in \{0, 1\}$$

Given a feasible solution $(\mathbf{x}, \mathbf{z})$ to the IP, we define *cost*$(\mathbf{x}, \mathbf{z})$ as the weight of $\mathbf{x}$ plus the total penalty $\sum_{i \in [k]} \pi_i \mathbf{z}(i)$.

In what follows, we denote the cost of optimal solutions to the programs $\mathbb{ND}$ and $\mathbb{PCND}$ by OPT_{ND} and OPT_{PCND}. One can also relax the integrality constraints in both IPs to get corresponding linear relaxations. We denote the cost of the optimal fractional solutions of the corresponding linear programs (LP) by OPT^*_{ND} and OPT^*_{PCND}.

3.5.2 A Generic Algorithm for Online Prize-Collecting Problems

In the online variants of network design problems and their prize-collecting counterparts, demands arrive sequentially. However, we assume that the node-weighted graph $G = (V, E, w)$ is known in advance. More precisely, in an online prize-collecting network design problem (OPCND), at time $t \in [k]$, a new demand (f_t, π_t) arrives and we need to output a feasible solution $(\mathbf{x}^t, \mathbf{z}^t)$ for the integer program $\mathbb{PCND}$. The decisions are online in the sense that an online algorithm may only increase the values of the variables, i.e., for every $t' < t$, $\mathbf{x}^{t'}(v) \leq \mathbf{x}^t(v)$ and $\mathbf{z}^{t'}(i) \leq \mathbf{z}^{t'}(i)$, for every $v \in V$ and $i \in [k]$.

Consider an algorithm ALG for OPCND and a sequence of demands $\rho = (f_1, \pi_1), \ldots, (f_k, \pi_k)$. Let ALG($\rho$) denote the cost of the output of ALG on the online input ρ, i.e., ALG(ρ) $= cost(\mathbf{x}_k, \mathbf{z}_k)$. ALG is *$\alpha$-competitive* w.r.t. $\mathcal{G}$ and $\mathcal{F}$, if for every $G \in \mathcal{G}$ and every sequence of demands $\rho = (f_1, \pi_1), \ldots, (f_k, \pi_k)$ where $f_i \in \mathcal{F}$, we have ALG(ρ) $\leq \alpha OPT_{PCND}$. ALG is *strongly α-competitive*, if for every ρ, ALG(ρ) $\leq \alpha OPT^*_{PCND}$. One can define similar notations for online network design (OND) problems by dropping penalties and replacing PCND indices by ND. The main result of this section is the following reduction.

Theorem 3.5. *Let $\mathcal{G}$ and $\mathcal{F}$ respectively denote a family of graphs and a family of feasible functions. Given a strongly α-competitive*

algorithm for an online network design problem (OND) w.r.t. $\mathcal{G}$ and $\mathcal{F}$, one can derive a strongly competitive algorithm for the corresponding OPCND with a competitive ratio of $\alpha O(\log(|V|))$.

Before we prove Theorem 3.5, we need to recall the following theorem by Buchbinder *et al* [BN09b] (later improved by Gupta and Nagarajan [GN12][7]). Consider a minimization LP in the form that given a vector c and a matrix A, minimizes $c \cdot \mathbf{x}$ subject to $A\mathbf{x} \geq \mathbf{1}$.[8] A *covering LP* is a special case where all the entries of A are non-negative. In an *online covering problem*, the vector c is known in advance, however, the covering constraints arrive online. After the arrival of a new constraint, the online algorithm needs to output a (fractional) feasible solution without decreasing the previous values of variables.

Theorem 3.6 (Theorem 4.2 of [BN09b])**.** *Let n be the number of variables. There exists an algorithm for the online covering problem which finds a fractional solution with the cost within $O(\log(n))$ factor of the optimal fractional solution. Furthermore, the algorithm only increases a variable if it has a positive coefficient in the new constraint.*

Consider a non-trivial constraint[9] of the program $\mathbb{PCND}$ corresponding to a func-

[7] The competitive ratio in [GN12] is improved to $O(\log(k))$ where k is the maximum number of non-zero entries in a row.

[8] Let $\mathbf{1}$ and $\mathbf{0}$ denote the vectors where all the entries are one and zero, respectively.

[9] We say a constraint is trivial if $f_i(S) = 0$

tion f_i and a subset S. It can be re-written in the following standard format:

$$\mathbf{z}(i) + \sum_{v \in \delta(S)} \mathbf{x}(v) \geq 1$$

Thus all the constraints are covering constraints. Suppose we want to solve OPCND fractionally. We note that OPCND is not formally a special case of the online covering problem in two aspects. First, the objective function is not fully known, i.e., the prizes are revealed online. Second, in OPCND all constraints corresponding to f_i are revealed at the same time while in an online covering problem, we assume that the constraints are revealed one by one. The former is easy to handle since by Theorem 3.6, the algorithm of Buchbinder *et al* [BN09a] only changes a variable when it has a positive coefficient in a newly arrived constraint. Thus the variable $\mathbf{z}(i)$ changes only in step i after receiving the demand (f_i, π_i). The second discrepancy can be handled by using the efficiency of function f_i. At any step, let $(\mathbf{x}^*, \mathbf{z}^*)$ denote the current fractional solution. While the solution is not feasible, we find an infeasible constraint and reveal it to the online covering algorithm. This can be done in polynomial time since f_i is efficient. We continue this process until all the constraints are feasible.

Therefore the algorithm of Buchbinder and Naor can be applied to OPCND to obtain a fractional solution $(\mathbf{x}^*, \mathbf{z}^*)$ such that by Theorem 3.6, $\mathit{cost}(\mathbf{x}^*, \mathbf{z}^*) \leq \mathit{OPT}^*_{PCND} O(\log(|V|))$. Note that although the algorithm may increase the value of a variable beyond one, this can be ignored since feasibility is maintained even if we decrease such variables to one.

Algorithm.

Let ALG_{OND} be a strongly α-competitive algorithm for OND. The following algorithm for OPCND realizes Theorem 3.5. Let $\rho = (f_1, \pi_1), \ldots, (f_k, \pi_k)$ denote the online input. We run the online fractional algorithm of Buchbinder *et al* and an instance of ALG_{OND} in parallel. At any time step, let $(\mathbf{x}^*, \mathbf{z}^*)$ denote the (partial) output of the fractional algorithm and let $\mathbf{x}$ denote the (partial) output of ALG_{OND}, respectively. We also maintain an integral vector $\mathbf{z}$ which shows the integral decisions of our algorithm for paying the penalties of demands that have arrived.

At step i, we receive the new demand (f_i, π_i). We reveal the new demand to the online fraction algorithm which in return updates the values of $(\mathbf{x}^*, \mathbf{z}^*)$. In particular, it sets the first and final value of $\mathbf{z}^*(i)$. Now if $\mathbf{z}^*(i) \geq 1/2$, we pay the penalty of the new demand and set $\mathbf{z}(i) = 1$. Otherwise, we set $\mathbf{z}(i) = 0$, and we reveal the function f_i to the instance of ALG_{OND}. At the end of iteration, we report $(\mathbf{x}, \mathbf{z})$ as the output of our algorithm.

3.5.2.1 Proof of Theorem 3.5

We prove the theorem for the aforementioned algorithm. Observe that at the end of the algorithm, by Theorem 3.6, $\mathit{cost}(\mathbf{x}^*, \mathbf{z}^*) \leq \beta \cdot \mathit{OPT}^*_{PCND}(\rho)$, where $\beta = O(\log(|V|))$.

We pay the penalty of a demand i, only if $\mathbf{z}^*(i) \geq \frac{1}{2}$. Thus $\mathbf{z}(i) \leq 2\mathbf{z}^*(i)$ and we have

$$\sum_i \pi_i \mathbf{z}(i) \leq \sum_i \pi_i (2\mathbf{z}^*(i)) = 2 \sum_i \pi_i \mathbf{z}^*(i) \leq 2\mathit{cost}(\mathbf{x}^*, \mathbf{z}^*) \leq 2\beta \cdot \mathit{OPT}^*_{PCND}(\rho) \tag{3.3}$$

Recall that the total cost of our algorithm is $\mathit{cost}(\mathbf{x}, \mathbf{z}) = \sum_v w_v \mathbf{x}(v) + \sum_i \pi_i \mathbf{z}(i)$. Thus it only remains to bound the weight of $\mathbf{x}$. Consider the instance of the OND problem $\rho' = \langle f_i | \mathbf{z}(i) = 0 \rangle$. Observe that our algorithm indeed reveals only ρ' to ALG_{OND}. Let $\mathbf{x}'(v) = \min\{1, 2\mathbf{x}^*(v)\}$ for every vertex $v \in V$. We claim that $\mathbf{x}'$ is a feasible solution.

Claim 3.3. *The vector $\mathbf{x}'$ is feasible for the instance ρ'.*

Proof. Recall that $(\mathbf{x}^*, \mathbf{z}^*)$ is a feasible solution for the linear program corresponding to PCND. Now consider a function f_i in ρ' and an arbitrary set $S \subset V$. By definition, $\mathbf{z}(i) = 0$ and thus $\mathbf{z}^*(i) < \frac{1}{2}$. If for a $v \in \delta(S)$, $x'(v) = 1$, then the constraint for f_i and S is clearly satisfied. Otherwise, for every $v \in \delta(S)$, $x'(v) = 2x^*(v)$. We have

$$\begin{aligned}
\sum_{v \in \delta(S)} \mathbf{x}'(v) &= 2 \sum_{v \in \delta(S)} \mathbf{x}^*(v) \\
&\geq 2\left(f_i(S)(1 - \mathbf{z}^*(i))\right) && \text{by feasibility of } (\mathbf{x}^*, \mathbf{z}^*) \\
&\geq f_i(S) && \mathbf{z}^*(i) < \frac{1}{2}
\end{aligned}$$

which proves the claim. □

Now since $\mathbf{x}'$ is a feasible solution, the optimum fractional solution may only have

less weight, i.e.,

$$\mathit{OPT}^*_{ND}(\rho') \le \sum_v w_v \mathbf{x}'(v) \le 2 \sum_v w_v \mathbf{x}^*(v) \le 2\mathit{cost}(\mathbf{x}^*, \mathbf{z}^*) \le 2\beta \cdot \mathit{OPT}^*_{PCND}(\rho) \tag{3.4}$$

Since ALG_{OND} is strongly α-competitive, we get

$$\begin{aligned}
\mathit{cost}(\mathbf{x}, \mathbf{z}) &= \sum_v w_v \mathbf{x}(v) + \sum_i \pi_i \mathbf{z}(i) \\
&\le \sum_v w_v \mathbf{x}(v) + 2\beta \cdot \mathit{OPT}^*_{PCND}(\rho) && \text{by (3.3)} \\
&\le \alpha \mathit{OPT}^*_{ND}(\rho') + 2\beta \cdot \mathit{OPT}^*_{PCND}(\rho) && \text{by competitiveness of } \text{ALG}_{OND} \\
&\le \alpha \left(2\beta \cdot \mathit{OPT}^*_{PCND}(\rho)\right) + 2\beta \cdot \mathit{OPT}^*_{PCND}(\rho) && \text{by (3.4)} \\
&= 2\beta(\alpha + 1)\mathit{OPT}^*_{PCND} && = O(\log(|V|))\alpha \cdot \mathit{OPT}^*_{PCND}
\end{aligned}$$

which completes the proof.

3.5.3 An Asymptotically Optimal Algorithm for Online NW ST

The online node-weighted Steiner tree (NW-ST) problem is a fundamental OND problem: given a vertex root, every input function f_i characterizes the cuts that separate a vertex t_i from the root.

The Online Node-weighted Steiner Tree problem.

We are given an undirected connected graph $G = (V, E)$ where w_v is the weight of vertex $v \in V$. Let $n = |V|$. The online input comprises a sequence of vertices

$t_0, t_1, t_2, \ldots, t_k$ where $t_i \in V$. The output comprises a sequence $H_0, H_1, H_2, \ldots, H_k$, where (i) H_i is a connected subgraph of G; (ii) the terminals $\{t_0, t_1, t_2, \ldots, t_i\}$ are connected in H_i; and (iii) H_i is a subgraph of H_{i+1}. The objective is to minimize the total weight of vertices in H_k. Without loss of generality, we assume that the weight of every terminal is zero[10]. For simplicity, we will assume that the cost of the optimal solution is n and for every vertex v, $w_v \in (0, n]$. This is wlog up to a constant factor loss in the competitive ratio[11].

Naor *et al* [NPS11] solve the online NW ST for the first time via a reduction to an instance of a facility location problem[12]. They use an interesting combinatorial fact, namely the generalized spider decomposition, to show that the cost of the output of the algorithm is within $O(\log^3(n))$ factor of the optimal integral solution. Our algorithm follows the approach of Naor *et al* for solving the problem via an instance of a facility location problem. Although our algorithm is very similar to that of Naor *et al*, our analysis is quite different; while they use a combinatorial fact to prove the competitive ratio of the algorithm, we use the technique of dual averaging that we alluded to in the introduction.

[10]For every vertex v, attach a dummy vertex ρ_v with weight zero to v. Upon receiving a terminal t, we virtually assume that the terminal is ρ_t.

[11]By an online doubling strategy, we may assume that we know the objective value α of an optimal solution. We multiply all vertex weights by n/α so that the cost of the optimal solution is n. All vertices v with $w_v > n$ are discarded, while we set $x_v = 1$ for those satisfying $w_v \leq 1$.

[12]In a facility location problem, given a set of facilities with setup cost and an online sequence of clients with their connection costs to the facilities, the objective is to map the clients to facilities such that the connection costs used in the mapping plus the setup cost of the matched facilities is minimized.

This allows us to establish the tight competitive ratio of $O(\log^2(n))$ with respect to the fractional solution.

An Auxiliary Linear Program.

Let Γ denote the set of compound indices $V \cup (V \times [k])$, i.e., for every $v \in V$ and $i \in [k]$, both $v \in \Gamma$ and $(v, i) \in \Gamma$. A set $S \subseteq \Gamma$ is an *auxiliary cut* for a terminal t_i, if for every vertex v, S contains exactly one of v and (v, i). Let $\mathbf{S}_i$ denote the collection of all auxiliary cuts for t_i, i.e., for $i \in [k]$, $\mathbf{S}_i = \{S \subseteq \Gamma | \forall_{v \in V} \ |S \cap \{v, (v, i)\}| = 1\}$. For every $v \in V$ and $i \in [k]$, let $P_{(v,i)}$ denote a minimum-weight path between t_i and any previous terminal t_j (i.e. $0 \leq j < i$) which goes through v. For every (compound) index $\gamma \in \Gamma$ we define a weight $\mathbf{w}_\gamma$ as follows. For every $v \in V$, let $\mathbf{w}_v = w_v$. For every $v \in V$ and $i \in [k]$, $\mathbf{w}_{(v,i)}$ is the weight of $P_{(v,i)}$ minus the weight of v.

Consider the *auxiliary linear program* ALP (given below) with a variable $\mathbf{x}_\gamma$ for every index $\gamma \in \Gamma$. Let $\mathbf{x}$ be a feasible solution to the Program ALP. Observe that for every $v \in V$ and $i \in [k]$, we may assume $\mathbf{x}_{(v,i)} \leq \mathbf{x}_v$; otherwise by reducing $\mathbf{x}_{(v,i)}$ to $\mathbf{x}_v$ we can decrease the objective value while keeping the solution feasible[13]. Therefore in the rest of section, wlog, we assume that for every feasible solution, $\mathbf{x}_{(v,i)} \leq \mathbf{x}_v$.

[13]Recall that for every auxiliary cut S that contains (v, i), there exists a cut S' that replaces (v, i) with v. Thus if constraint P1 is feasible for S', by reducing $\mathbf{x}_{(v,i)}$ to $\mathbf{x}_v$, the constraint corresponding to S remains feasible.

$$\text{minimize} \quad \sum_{\gamma \in \Gamma} \mathbf{w}_\gamma \mathbf{x}_\gamma \qquad \text{(ALP)} \qquad\qquad \text{minimize} \quad \sum_{\gamma \in \Gamma} \tilde{\mathbf{w}}_\gamma \mathbf{x}_\gamma \qquad \text{(SALP)}$$

$$\forall\, i \in [k], S \in \mathbf{S}_i \quad \sum_{\gamma \in S} \mathbf{x}_\gamma \geq 1 \qquad \text{(P1)} \qquad \forall\, i \in [k], S \in \mathbf{S}_i \quad \sum_{\gamma \in S} \mathbf{x}_\gamma \geq 1 \qquad \text{(P2)}$$

$$\mathbf{x}_\gamma \in [0,1] \qquad\qquad \mathbf{x}_\gamma \in [0,1]$$

We claim that for every *integral* feasible solution $\mathbf{x}$ for Program ALP, there exists an integral solution for the Steiner tree instance having cost at most the same as the objective value of the program. We construct the subgraph $H \subseteq G$ as follows. We initialize the set of vertices V_H to the set of vertices v where $\mathbf{x}_v = 1$. Now for every $v \in V$ and $i \in [k]$ that $\mathbf{x}_{(v,i)} = 1$, we add the vertices of $P_{(v,i)}$ to V_H. Observe that adding the path $P_{(v,i)}$ may increase the weight of V_H by at most $\mathbf{w}_{(v,i)}$ since vertex v is already in V_H. Therefore the total weight of V_H is at most the objective value of Program ALP for $\mathbf{x}$. Furthermore, in the subgraph H induced by V_H, every terminal t_i for $i \in [k]$ is connected to a previous terminal t_j for a $j \in [0, \ldots, i-1]$, leading to that H is a solution for ST.

A main obstacle in solving the online ST problem is that the known rounding methods are not effective in rounding a fractional solution of the linear relaxation of standard programs for ST. Indeed an important property of the auxiliary LP is that a standard rounding method similar to the Set Cover problem can be used to round the solution by losing (roughly) a logarithmic factor. However, although one can obtain an integral solution for ST with the same cost as that for Program ALP, the converse does not hold. Naor *et al* [NPS11] use a combinatorial decomposition of an *integral* solution for ST to show that the converse holds if one is willing to incur a factor of $O(\log^2(n))$ in the cost.

This combinatorial fact does not have a fractional counterpart which is crucial to our reduction in solving PCST. Furthermore, using Program ALP leads to a competitive ratio of $O(\log^3(n))$ after applying the rounding method, which is off by a logarithmic factor from the known lower bound. We overcome both obstacles by using a dual averaging argument to show a similar relationship between *fractional* solution for ST and that for a *scaled auxiliary LP*.

We define a *scaled weight* $\tilde{\mathbf{w}}$ over the set of compound indices Γ as follows. For every $v \in V$ and $i \in [k]$, let $\tilde{\mathbf{w}}_{(v,i)} = \mathbf{w}_{(v,i)}$, while for every $v \in V$, let $\tilde{\mathbf{w}}_v = w_v \log(n)$.[14] The scaled auxiliary program SALP is given above. We split the objective function of this LP into two parts. For a feasible vector $\mathbf{x}$ for SALP, let the *facility cost*, $\mathit{FacCost}(\mathbf{x}) = \sum_{v \in V} \tilde{\mathbf{w}}_v \mathbf{x}_v$ and let the *connection cost*, $\mathit{ConCost}(\mathbf{x}) = \sum_{v \in V, i \in [k]} \tilde{\mathbf{w}}_{(v,i)} \mathbf{x}_{(v,i)}$. We may drop $\mathbf{x}$ from the notation when the vector is clear from the context. Observe that a feasible solution for SALP yields a feasible solution for NW ST with total cost at most $\frac{\mathit{FacCost}}{\log(n)} + \mathit{ConCost}$.

Algorithm.

We first find an online[15] fractional solution for SALP using the method of multiplicative updates. We then show the objective value of this fractional solution is within

[14] In the rest of the section, the base of all logarithmic terms is 2.

[15] In the online setting, at time step $i \in [k]$, the constraints for sets $S \in \mathbf{S}_i$ are revealed and the algorithm should output a feasible solution. However, the online algorithm may only increase the previous values of variables in each time step.

$O(\log^2(n))$ factor of the optimal fractional solution for ST. Note that the objective function of the program uses the scaled weights for vertices. A feasible solution for the *scaled program* can be rounded online with an additional loss of a *constant* factor using the standard rounding techniques. We will not describe this rounding procedure here; we refer the reader to Section 2 of [NPS11].

We now describe the process of finding a competitive fractional solution in more details. For every index $\gamma \in \Gamma$ that $\tilde{\mathbf{w}}_\gamma$ is zero, we initialize $\mathbf{x}_\gamma$ to one. For the rest of variables, we initialize the value of $\mathbf{x}_\gamma$ to $1/n^3$. When a new terminal t_i (together with the constraints corresponding to sets in $\mathbf{S}_i$) arrives online, we update the current solution in a sequence of multiplicative steps until the solution is feasible. In each multiplicative step, we identify an auxiliary cut S corresponding to an infeasible constraint and for each index $\gamma \in S$, we increase the value of $\mathbf{x}_\gamma$ to $\mathbf{x}_\gamma(1 + \frac{1}{\tilde{\mathbf{w}}_\gamma})$.

Analysis.

For a subset of vertices $S \subset V$, let $\delta(S) \subseteq V \backslash S$ denote the neighbors of S. Let $\mathcal{S}$ denote the collection of subsets of vertices that separate a subset of terminals from the terminal t_0, i.e, $S \in \mathcal{S}$ if and only if $S \cap \{t_1, \ldots t_k\} \neq \phi$ and $t_0 \notin S$. Consider the natural LP relaxation for the NW ST problem in which there is a flow of one going out of every

set in $\mathcal{S}$. The primal dual pair for this LP is as follows.

$$\begin{aligned} &minimize && \sum_{v \in V} w_v x_v \\ \forall S \in \mathcal{S} \quad & && \sum_{v \in \delta(S)} x_v \geq 1 \\ & && x_v \geq 0 \end{aligned}$$

$$\begin{aligned} &maximize && \sum_{S \in \mathcal{S}} y(S) \\ \forall v \quad & && \sum_{S \in \mathcal{S}: v \in \delta(S)} y(S) \leq w_v \qquad \text{(D1)} \\ & && y(S) \geq 0 \end{aligned}$$

For two vertices u and v, let $d(u, v)$ denote the weight of shortest path between u and v *excluding* the weight of the endpoints. We borrow the notation of Hajiaghayi *et al* [HLP13] for disks. A *painting* is a function $p : V \to \mathbb{R}_{\geq 0}$. A painting is *valid* if $p(v) \leq w_v$ for every vertex v. Intuitively, every vertex has an area equal to its weight and a painting is a (partial) coloring of these areas. The union of a set of paintings $p_1, \ldots, p_\ell$ is the painting p with $p(v) = \sum_{i \in [\ell]} p_i(v)$ for every $v \in V$.

Recall that the weight of a terminal is zero. A *disk* of radius r centered at terminal t, is a painting in which the area within a radius r of t is colored, i.e.,

$$p(v) = \begin{cases} w_v & \text{if } d(t,v) + w_v \leq r \\ r - d(t,v) & \text{if } d(t,v) + w_v > r \text{ but } d(t,v) \leq r \\ 0 & \text{if } d(t,v) \geq r \end{cases}$$

A disk p centered at a terminal t_i for $i \in [k]$ is *feasible* if $p(t_0) = 0$.

A *disk vector* corresponding to a disk p of radius r centered at a terminal t is a dual vector y generated by the following deterministic process:

1: Initialize $y = \mathbf{0}$ and $U = \{t\}$.

2: **while** the total dual objective $\sum_{S \in \mathcal{S}} y(S)$ is less than r **do**

3: Continuously increase $y(U)$ until $\sum_{S \in \mathcal{S}} y(S)$ reaches r, or for a vertex v the dual constraint $D1$ becomes tight.

4: If the latter happens, add v to U.

For a dual vector y, define the *load* of y on a vertex v as $\sum_{S \in \mathcal{S}: v \in \delta(S)} y(S)$. The construction of a disk vector directly yields the following lemma .

Lemma 3.11. *Let y be a disk vector corresponding to a disk p. For every $v \in V$, the load of y on v is exactly $p(v)$.*

Proof of Lemma 3.11. Consider the process of generating the disk vector y. We first use induction to show that a vertex v enters the set U in the process when the total growth of dual variables is exactly $d(t, v) + w_v$. The claim clearly holds at the start of the process. Now consider an arbitrary vertex v and let u be the closest neighbor of v to the center t, i.e., $d(t, v) = d(t, u) + w_u$. Observe that the load on v remains zero as long as none of its neighbors is in U. By the induction hypothesis, the first neighbor of v enters U when the the total growth is exactly $d(t, v)$. After that step, increasing the dual variables also increases the load on v until the constraint D1 becomes tight for v, at which point the vertex v is added to U. Therefore v enters U when the total growth is exactly $d(t, v) + w_v$; proving the claim.

The aforementioned claim shows that for every vertex v in U, the load on v is exactly w_v which is equal to $p(v)$ since $d(t,v) + w_v \leq r$. Now it only remains to show that the load on a vertex v outside U is also equal to $p(v)$. The argument is similar as before. Let u be the closest neighbor of v to the center t. The load on v is zero until the vertex u enters U, which happens when the total growth is exactly $d(t,u) + w_u = d(t,v)$. If $d(t,v) \geq r$, the load on v remains zero until the end. Otherwise, at the end the total load on v is exactly $r - d(t,v)$, which completes the proof. □

Corollary 3.4. *Let y be a disk vector corresponding to a disk of radius r centered at t. For every vertex $v \in V$ on which the load of y is strictly positive, we have $r \geq d(t,v)$. Furthermore, y is a feasible dual vector if and only if the disk is feasible, i.e., $r \leq d(t,t_0)$.*

For an arbitrary *thinness* factor $\tau \in (0,1]$, a τ-thin disk vector is a dual vector y' obtained by scaling the disk vector y by a factor of τ, i.e., $y'(S) = \tau y(S)$ for every $S \in \mathcal{S}$. When the thinness factor is clear from the context, we may refer to y' as simply a thin disk vector. We note that a τ-thin disk vector is feasible if and only if the corresponding (1-thin) disk vector is feasible. Observe that the total dual objective value of a τ-thin disk vector with radius r is exactly $r \times \tau$. Similar to paintings, the union of a set of disk-vectors $y_1, \ldots, y_\ell$ is the dual vector y where $y(S) = \sum_{i \in [\ell]} y_i(S)$ for every $S \subseteq \mathcal{S}$.

Multiplicative Updates Method.

Consider the cost of the fractional solution generated by the multiplicative steps. In what follows, let *opt* denote the cost of the optimal fractional solution for the NW-ST

problem. Recall that we assume $opt = n$. [16] In our algorithm, for every $\gamma \in \Gamma$, we initialize $\mathbf{x}_\gamma$ to $\frac{1}{n^3}$ (or to one if $\tilde{\mathbf{w}}_\gamma = 0$). Furthermore, for $v \in V$ and $i \in [k]$, $\tilde{\mathbf{w}}_{(v,i)} \leq n^2$ since we have assumed that the weight of a single vertex is at most n and a simple path may contain at most n vertices. SALP has at most $n^2 + n$ variables. Thus the objective cost of the initialization is at most $(n^2 + n)(n^2)\frac{1}{n^3} \leq 2n = 2opt$. Hence the cost of initialization adds a constant factor to the competitive ratio; we will ignore this factor in the remaining analysis.

The next two lemmas are standard in analyzing the multiplicative steps. The first lemma follows immediately from the fact that the sum of variables in each auxiliary cut that participates in a multiplicative update step is at most one.

Lemma 3.12. *The increase in the value of the objective function (also called the* cost*) of a single multiplicative update step is at most* 1.

Proof of Lemma 3.12. Consider an arbitrary multiplicative step. Let S be the selected auxiliary cut. For every variable $\gamma \in S$, we increase the value of $\mathbf{x}_\gamma$ to $\mathbf{x}_\gamma(1 + 1/\tilde{\mathbf{w}}_\gamma)$. Hence, the increase in the objective function is

$$\sum_{\gamma \in S} \tilde{\mathbf{w}}_\gamma \mathbf{x}_\gamma (\frac{1}{\tilde{\mathbf{w}}_\gamma}) = \sum_{\gamma \in S} \mathbf{x}_\gamma < 1$$

□

Lemma 3.12 directly implies that to bound the cost of the algorithm, it is sufficient

[16] Though in fact, in an actual implementation we only know that $n \leq opt \leq 2n$. However, as mentioned before, for simplicity we ignore this constant factor in the analysis.

to bound the number of multiplicative steps. The key for bounding the number of steps is the following observation.

Lemma 3.13. *An index $\gamma \in \Gamma$ can participate in at most $\tilde{\mathbf{w}}_\gamma \log(n^3)$ multiplicative steps.*

Proof of Lemma 3.13. The initial value of a variable is at least $\frac{1}{n^3}$. At every step we only choose an auxiliary cut if the total value of variables participating in the cut is less than one. Thus γ cannot participate in a multiplicative step when $\mathbf{x}_\gamma$ exceeds one. On the other hand, we increase the value of $\mathbf{x}_\gamma$ by a factor of $(1 + \frac{1}{\tilde{\mathbf{w}}_\gamma})$. Therefore if γ participates in m steps, at the beginning of the last step we have

$$\frac{1}{n^3}\left(1 + \frac{1}{\tilde{\mathbf{w}}_\gamma}\right)^{(m-1)} < 1 \Longrightarrow 2^{\frac{m-1}{\tilde{\mathbf{w}}_\gamma}} < n^3 \Longrightarrow m < \tilde{\mathbf{w}}_\gamma \log(n^3)$$

□

Using the notion of thin disks, we will now establish the competitive ratio of the algorithm. Therefore we use a union of a set of disks to account for the *number* of the multiplicative updates and thus by Lemma 3.12 bound the total cost of the algorithm.

Lemma 3.14. *Let FacCost and ConCost respectively denote the facility cost and the connection cost of the output of the algorithm. Then FacCost + ConCost $\leq O(\log^2(n))$opt.*

For $i \in [k]$, let s_i denote the number of multiplicative steps done for terminal t_i. We assume wlog that $s_i \geq 1$. Let $\tau = \frac{1}{4\log(n)}$. For every $i \in [k]$, consider a τ-thin disk vector y_i corresponding to a disk of radius $r_i = \frac{s_i}{5\log(n^3)}$ centered at terminal t_i. We first show for every $i \in [k]$, the thin disk vector y_i is indeed feasible. Note that for terminal t_i and

in every auxiliary cut we should have either t_0 or (t_0, i). However, the cost of a terminal, in particular that of t_0, is zero. Hence, $\mathbf{x}_{t_0}$ is initialized to one and thus only (t_0, i) can participate in a multiplicative step. Therefore by Lemma 3.13, $s_i \leq \tilde{\mathbf{w}}_{(t_0,i)} \log(n^3)$. Recall that $\tilde{\mathbf{w}}_{(t_0,i)}$ is the weight of shortest path between t_i and a terminal t_j for $j < i$, which passes through t_0. Hence by choosing $j = 0$, $\tilde{\mathbf{w}}_{(t_0,i)} \leq d(t_i, t_0)$. Thus the radius of the disk is at most $r_i = \frac{s_i}{5\log(n^3)} \leq \frac{d(t_0,t_i)}{5}$, which by Corollary 3.4 verifies that y_i is feasible.

Let y denote the union of y_i's for every $i \in [k]$. We prove Lemma 3.14 by showing that :

Lemma 3.15. *The dual vector y is feasible.*

Proof of Lemma 3.15. Let v be an arbitrary vertex. We show that the dual constraint D1 is feasible for v. Consider the sequence of indices of terminals $1 \leq a_1 < a_2 < \ldots < a_m \leq k$ such that for every $i \in [m]$, the load of y_{a_i} on v is strictly positive. Note that by Corollary 3.4, $r_{a_i} \geq d(t_{a_i}, v)$.

Consider the smallest index $q \in [m-1]$ that $r_{a_q} \leq 2r_{a_{(q+1)}}$. In the special case that no such index exists, let $q = m$. Observe that for every $i < q$, $r_{a_i} > 2r_{a_{(i+1)}}$, i.e., the radii drop by a ratio less than $1/2$. However, the minimum radius is $\frac{1}{5\log(n^3)}$ while the maximum radius of a feasible disk is n^2. Thus $q \leq 3\log(n)$ since the gap between the maximum and minimum radius is less than n^3.

For every $i \in [q+1, k]$ and $v \in V$, either v or (v, a_i) participates in every multiplicative step for t_{a_i}. Note that $\tilde{\mathbf{w}}_{(v,a_i)} \leq \min_{j\in[0..i-1]} d(t_{a_i}, v) + d(t_{a_j}, v) \leq d(t_{a_i}, v) + d(t_{a_{i-1}}, v)$. We first prove the following inequality.

Claim 3.4. $w_v \geq \frac{1}{\log(n)} \sum_{i=q+1}^{k} r_{a_i}$

Proof. Suppose, by contradiction, that $w_v < \frac{1}{\log(n)} \sum_{i=q+1}^{k} r_{a_i}$. Consider the multiplicative steps for terminals $t_{q+1}, \ldots, t_k$. The total number of such steps is at most the number of steps one can update $\mathbf{x}_v$ and $\mathbf{x}_{(v,a_i)}$ for every $i \in [q+1, k]$, since at least one of them is present in an auxiliary cut for a terminal t_{a_i}.

$$\begin{aligned}
\sum_{i=q+1}^{k} s_{a_i} &\leq \tilde{\mathbf{w}}_v \log^3(n) + \sum_{i=q+1}^{k} \tilde{\mathbf{w}}_{(v,a_i)} \log^3(n) && \text{by Lemma 3.13} \\
&\leq (w_v \log(n)) \log^3(n) \\
&\quad + \log^3(n) \sum_{i=q+1}^{k} (d(t_{a_i}, v) + d(t_{a_{i-1}}, v)) && \tilde{\mathbf{w}}_{(v,a_i)} \leq d(t_{a_i}, v) + d(t_{a_{i-1}}, v) \\
&< (\sum_{i=q+1}^{k} r_{a_i}) \log^3(n) \\
&\quad + \log^3(n) \sum_{i=q+1}^{k} (d(t_{a_i}, v) + d(t_{a_{i-1}}, v)) && w_v < \frac{1}{\log(n)} \sum_{i=q+1}^{k} r_{a_i}
\end{aligned}$$

$$\begin{aligned}
&\leq (\sum_{i=q+1}^{k} r_{a_i}) \log^3(n) + 2\log^3(n) \sum_{i=q+1}^{k} d(t_{a_i}, v) + \log^3(n) d(t_{a_q}, v) \\
&\leq (\sum_{i=q+1}^{k} r_{a_i}) \log^3(n) + 2\log^3(n) \sum_{i=q+1}^{k} r_{a_i} + \log^3(n) r_{a_q} && \forall i \in [m], d(t_{a_i}, v) \leq r_{a_i} \\
&\leq (\sum_{i=q+1}^{k} r_{a_i}) \log^3(n) + 2\log^3(n) \sum_{i=q+1}^{k} r_{a_i} + 2\log^3(n) r_{a_{q+1}} && r_{a_q} \leq 2 r_{a_{q+1}} \\
&\leq (\sum_{i=q+1}^{k} r_{a_i}) \log^3(n) + 4\log^3(n) \sum_{i=q+1}^{k} r_{a_i} \\
&= (5\log^3(n)) \sum_{i=q+1}^{k} r_{a_i}
\end{aligned}$$

However, the last inequality is a contradiction since by definition $s_{a_i} = (5\log^3(n))r_{a_i}$ and thus $\sum_{i=q+1}^{k} s_{a_i}$ cannot be strictly less than $(5\log^3(n))\sum_{i=q+1}^{k} r_{a_i}$. □

Now recall that a thin disk vector is feasible if and only if the corresponding 1-thin disk vector is feasible. Hence, for every $i \in [k]$, the constraint D1 is feasible even with thinness one:

$$\sum_{S\in\mathcal{S}:v\in\delta(S)} \frac{y_{a_i}(S)}{\tau} \leq w_v \Longrightarrow \sum_{S\in\mathcal{S}:v\in\delta(S)} y_{a_i}(S) \leq \tau w_v \tag{3.5}$$

Now we have all the ingredients to prove the Lemma. For every $i \leq q$, we bound the load of y_{a_i} on v using Equation (3.5). Since $q = O(\log^3(n))$, this adds up to only a constant fraction of w_v. On the other hand, by Claim 3.4, for all $i > q$, the total load of y_{a_i}'s is at most a constant fraction of w_v. Therefore the total load of y on v does not exceed w_v which completes the proof. More formally, we show the dual constraint D1 holds for v by adding all the dual variables together:

$$\sum_{S\in\mathcal{S}:v\in\delta(S)} y(S) = \sum_{i\in[m]} \sum_{S\in\mathcal{S}:v\in\delta(S)} y_{a_i}(S)$$

$$= \sum_{i=1}^{q} \sum_{S\in\mathcal{S}:v\in\delta(S)} y_{a_i}(S) + \sum_{i=q+1}^{k} \sum_{S\in\mathcal{S}:v\in\delta(S)} y_{a_i}(S)$$

$$\leq \sum_{i=1}^{q} \sum_{S\in\mathcal{S}:v\in\delta(S)} y_{a_i}(S) + \sum_{i=q+1}^{k} (\tau \times r_{a_i}) \qquad \sum_{S:v\in\delta(S)} y_{a_i}(S) \leq \sum_{S} y_{a_i}(S) \leq \tau \times r_{a_i}$$

$$\leq \sum_{i=1}^{q} \tau w_v + \sum_{i=q+1}^{k} (\tau \times r_{a_i}) \qquad \text{by constraint D1 for } y_i \text{ with thinness one}$$

$$\leq q(\tau \cdot w_v) + \tau \sum_{i=q+1}^{k} r_{a_i}$$

$$\leq w_v(\frac{3\log(n)}{4\log(n)}) + \tau(w_v \log(n)) \qquad q \leq 3\log(n) \text{ and Claim 3.4}$$

$$\leq w_v(3/4 + 1/4) = w_v$$

□

Now one can use Lemma 3.15 and the weak duality to prove Lemma 3.14.

Proof of Lemma 3.14. It follows from the weak duality and Lemma 3.15 that the total dual objective value of disks is upper bounded by *opt*. Therefore it follows from the definitions that

$$\textit{opt} \geq \sum_{i\in[k]}(r_i \times \tau) = \frac{1}{20\log(n^3)\log(n)} \sum_{i\in[k]} s_i \geq \frac{1}{O(\log^2(n))}(\textit{FacCost} + \textit{ConCost})$$

where the last inequality follows from Lemma 3.12. □

Finally as mentioned before, by using a standard rounding method, Theorem 3.4 follows.

Proof of Theorem 3.4. Let $\mathbf{x}'$ be a feasible solution for the (non-scaled) auxiliary program ALP. One can find an integral solution with total cost at most $O(\log(n)) \sum_{v\in V} w_v \mathbf{x}'_v + O(1) \sum_{v\in V, i\in[k]} \mathbf{w}_{(v,i)} \mathbf{x}'_{(v,i)}$ using the rounding method of Naor *et al* [NPS11]. On the other hand, both Program ALP and Program SALP have the same set of constraints. Let $\mathbf{x}$ be the fractional solution of our algorithm. The total cost our integral solution is at most

$$O(\log(n)) \sum_{v\in V} w_v \mathbf{x}_v + O(1) \sum_{v\in V, i\in[k]} \mathbf{w}_{(v,i)} \mathbf{x}_{(v,i)}$$

$$= O(1)\sum_{v\in V}\tilde{\mathbf{w}}_v\mathbf{x}_v + O(1)\sum_{v\in V, i\in[k]}\mathbf{w}_{(v,i)}\mathbf{x}_{(v,i)} \qquad \tilde{\mathbf{w}}_v = \log(n)w_v$$

$$= O(1)\sum_{v\in V}\tilde{\mathbf{w}}_v\mathbf{x}_v + O(1)\sum_{v\in V, i\in[k]}\tilde{\mathbf{w}}_{(v,i)}\mathbf{x}_{(v,i)} \qquad \tilde{\mathbf{w}}_{(v,i)} = \mathbf{w}_{(v,i)}$$

$$= O(1)(\textit{FacCost}(\mathbf{x}) + \textit{ConCost}(\mathbf{x}))$$

Therefore by Lemma 3.14, the cost of our algorithm is within $O(\log^2(n))$ factor of the cost of fractional optimal solution. □

3.5.4 An Asymptotically Optimal Algorithm for Online EW PCST

The online edge-weighted PCST problem can be considered a special instance of OPCND as follows:

- The problem is defined on the family of edge-weighted graphs.[17]
- The input contains a vertex $r \in V$ (also called root).
- For every $i \in [k]$, the function f_i is represented in the input by a single vertex t_i. For every $S \subseteq V$, $f_i(S) = 1$ if S contains t_i but does not contain the root; otherwise $f_i(S) = 0$.

[17] This is equivalent to the family of node-weighted graphs $\mathcal{G}$ where for every $G = (V, E, w) \in \mathcal{G}$, the weight of a vertex is positive only if its degree is two.

Preliminaries.

For simplicity, in this section we introduce a few notations specific for the edge-weighted PCST. We are given a graph $G = (V, E)$ with edge lengths ℓ_e and a root vertex r. We add edges between all pairs of vertices in the graph of length equal their respective distances. The online input comprises a sequence of terminals $T = t_1, t_2, \ldots, t_k$ and corresponding penalties $p_1, p_2, \ldots, p_k$. We round up all edge lengths and penalties to powers of 2. Observe that rounding up the edge lengths and penalties may only increase the competitive ratio by factor two. We ignore this factor in the rest of section.

We denote the output graph $H \subseteq G$. Initially, H only contains the root vertex r. On the arrival of t_i, we either pay the penalty p_i or augment H such that t_i is connected to r in H. The goal is to minimize the sum of the lengths of edges in H and the total penalty paid by the algorithm.

The primal-dual LP pair for this problem that we will use in the analysis is given in Figure 3.5. Here, for any subset of vertices $S \subseteq V \setminus \{r\}$,

$$x(S) = \sum_{e \in (S, \overline{S})} x_e \qquad \text{and} \qquad J(S) = \{j \in [k] : t_j \in S \cap T\}.$$

Given an integral solution (x, z) to the primal LP, one can obtain a solution to PCST as follows. We buy the edges e that $x_e = 1$. Furthermore, we pay the penalty of every terminal t such that for a set $J \subseteq [k]$ containing t, $z_J = 1$. On the other hand, let $H = (V_H, E_H)$ denote a solution for PCST. The corresponding LP solution (x, z) is as follows: for every $e \in E_H$, we set $x_e = 1$; and we set $z_{T \setminus V_H} = 1$.

Primal: Minimize $\sum_{e\in E} \ell_e x_e + \sum_{J:J\subseteq[k]} z_J \sum_{j\in J} \pi_j$ subject to

$$x(S) + \sum_{J\subseteq[k]:J(S)\subseteq J} z_J \geq 1, \quad \forall\, S \subseteq V \setminus \{r\} \tag{3.6}$$

$$x_e \geq 0, \quad \forall\, e \in E \tag{3.7}$$

$$z_J \geq 0, \quad \forall\, J \in [k] \tag{3.8}$$

Dual: Maximize $\sum_{S\subseteq V\setminus\{r\}} y_S$ subject to

$$\sum_{S:e\in(S,\overline{S})} y_S \leq \ell_e, \quad \forall\, e \in E \tag{3.9}$$

$$\sum_{S:J(S)\subseteq J} y_S \leq \sum_{j\in J} \pi_j, \quad \forall\, J \subseteq [k] \tag{3.10}$$

$$y_S \geq 0, \qquad \forall\, S \subseteq V \setminus \{r\} \tag{3.11}$$

Figure 3.5: The primal-dual LP pair used in the analysis.

Algorithm.

We maintain two sets of logarithmic numbers of duals that we call *penalty duals* $P_0, P_1, \ldots$ and *connection duals* $C_0, C_1, \ldots$.

Let π_i denote the *virtual penalty* of a terminal t_i. When t_i arrives, π_i is initialized to its (real) penalty p_i. Let ℓ_i denote the length of the shortest edge from t_i to r or a terminal connected to r in H. Now, we repeat the following:

1. Define $j = \lg \pi_i$ and $k = \lg \ell_i$.

2. If $\ell_i \leq \pi_i$, we add the edge of length ℓ_i connecting t_i to r in H. We also place a

disk(-vector)[18] of radius $\ell_i/2$ centered at t_i in the connection dual C_k.

3. If $\ell_i > \pi_i$, we try to place a disk of radius $\pi_i/2$ in the penalty dual P_j. If this disk does not intersect with any existing disk in P_j, we pay the penalty for t_i and add the dual disk to P_j. If this disk intersects with an existing disk centered at some terminal t_ℓ in P_j, then we remove the disk centered at t_ℓ, increase the virtual penalty of terminal t_i to $\pi_i = 2^{j+1}$ and repeat from Step 1.

Analysis.

We introduce a function ψ that represents a partition of the terminals. For any subset in the partition, there is a chosen terminal that specifies the *cluster head*, and for any terminal t_j, $\psi(t_j)$ denotes the cluster head of the subset in the partition that t_j belongs to. The cluster heads are exactly the terminals that have disks centered at them in the duals, including the new terminal. The new terminal is initially placed in a singleton subset in the partition, i.e. $\psi(t_i) = t_i$, and the partition is updated by the algorithm as follows. If Step 3 is executed by the algorithm and there is an intersection between disks centered at t_i and t_ℓ, then we merge the subsets represented by t_i and t_ℓ and choose t_i as the cluster head of the new subset, i.e. the new values of $\psi^{-1}(t_i)$ and $\psi^{-1}(t_\ell)$ are respectively $\psi^{-1}(t_i) \cup \psi^{-1}(t_\ell)$ and $\emptyset$. In all other steps of the above algorithm, the partition remains unchanged.

[18]For the EW problem, a painting can be thought of as coloring the edges instead of vertices. One can define a disk-vector similarly, mutatis mutandis. Thus for simplicity, in this section we denote a disk-vector as simply a disk.

We now describe the key invariants maintained by the algorithm. Suppose a dual disk of radius R is centered at a terminal t_j. We have three cases.

- **Case 1:** The dual disk centered at t_j is in the penalty duals.
- **Case 2:** The dual disk centered at t_j is in the connection duals.
- **Case 3:** t_j is the current terminal (i.e. $j = i$).

The invariants are:

1. All terminals in $\psi^{-1}(t_j)$ other than t_j itself paid their respective penalties. For t_j, there are three possibilities: t_j paid its penalty in Case 1, t_j got connected to r in H and did not pay its penalty in Case 2, and t_j has neither paid its penalty nor gotten connected to r yet in Case 3.

2. $R \geq \sum_{t_k \in \psi^{-1}(t_j)} p_k/2$ unless $\psi^{-1}(t_i) = \{t_i\}$ in Case 2. Further, in Case 2, $R \geq \ell_j/2$.

3. For any $R^* \leq 2R$, the sum of penalties of all terminals in $\psi^{-1}(t_j)$ that are at a distance of at most R^* from t_j is at least R^*.

   We now show that these invariants are maintained by the algorithm.

Lemma 3.16. *The algorithm above preserves the three invariants given above.*

Proof. The first invariant follows immediately from the definition of the algorithm.

For the second invariant, note that $R \geq \ell_j/2$ in Case 2 holds from the definition of the algorithm. For the first part of the second invariant, we will use induction on time.

Note that the only point where the invariant can be violated is in Step 3 if the disk centered at the new terminal t_i intersects with that centered at some terminal t_ℓ leading to merger of their respective subsets of the partition. Assume inductively that the invariant holds for the new terminal t_i and the center of the intersecting disk t_ℓ. Thus,

$$\begin{aligned} \sum_{t_k \in \psi^{-1}(t_i)} p_k/2 &\leq R \\ \sum_{t_k \in \psi^{-1}(t_\ell)} p_k/2 &\leq R. \end{aligned}$$

Thus,

$$\sum_{t_k \in \psi^{-1}(t_i) \cup \psi^{-1}(t_\ell)} p_k/2 \leq 2R,$$

and hence the property holds after merging the subsets $\psi^{-1}(t_i)$ and $\psi^{-1}(t_\ell)$ and doubling the radius of the disk centered at t_i.

Similar to the second invariant, we will prove the third invariant also by induction over time. Again, the only point where the invariant can be violated is in Step 3 if the disk centered at the new terminal t_i intersects with that centered at some terminal t_ℓ leading to merger of their respective subsets of the partition. By the inductive hypothesis for t_i, for any $R^* \leq 2R$, the sum of penalties of all terminals in $\psi^{-1}(t_i)$ (and therefore, also for all terminals in $\psi^{-1}(t_i) \cup \psi^{-1}(t_\ell)$) that are at a distance of at most R^* from t_i is at least R^*. Now, we consider any $R^* \in (2R, 4R]$. Recall that since disks of radius R centered at t_i and t_ℓ intersect, these two centers are at a distance of at most $2R$. Combining this with the inductive hypothesis for t_ℓ, for any $R^* \in (2R, 4R]$, the sum of penalties of all terminals in $\psi^{-1}(t_\ell)$ that are at a distance of at most R^* from t_i is at least $R^* - 2R$. Now, we can combine this with the inductive hypothesis for t_i at $R^* = 2R$ since $\psi^{-1}(t_i)$ is

disjoint from $\psi^{-1}(t_\ell)$, and conclude that for any $R^* \in (2R, 4R]$, the sum of penalties of all terminals in $\psi^{-1}(t_i) \cup \psi^{-1}(t_\ell)$ that are at a distance of at most R^* from t_i is at least R^*. Hence, the third invariant holds after merging $\psi^{-1}(t_i)$ and $\psi^{-1}(t_\ell)$. □

Next, we claim that the dual disks are non-intersecting.

Lemma 3.17. *The dual disks are non-intersecting.*

Proof. The lemma holds for penalty duals by definition of the algorithm. For connection duals, consider the first violation of the lemma if any. Let dual disks of radius R centered at the new terminal t_i and that centered at some previous terminal t_j intersect in a connection dual. Then, the distance between t_i and t_j is less than $2R$. By the first invariant, t_j must be connected to the root r in H. Thus, $\ell_i < 2R$, which contradicts the fact that we are trying to place a dual disk of radius R centered at t_i in the connection dual. □

Next, we obtain the key properties of the algorithm as corollaries of the two lemmas given above.

Corollary 3.5. *The total cost of the primal solution is at most twice the sum of radii of the disks in the duals.*

Proof. This follows immediately from the second invariant. □

Corollary 3.6. *Each individual connection dual and penalty dual is feasible.*

Proof. This follows immediately from the third invariant and Lemma 3.17. □

The above two corollaries immediately lead to Theorem 3.3 since there are a logarithmic number of duals.

3.5.4.1 Hardness of Designing Online LMP Competitive Algorithms

In this section, we show that generalizing the competitive results of the previous section to LMP results is not possible. Consider the EW prize-collecting Steiner tree problem. For an online algorithm ALG and a sequence of demands ρ, let $\text{ALG}^w(\rho)$ and $\text{ALG}^\pi(\rho)$ respectively denote the weight and the penalty induced by the final solution of the algorithm. An algorithm ALG is LMP α-competitive if for every sequence of demands ρ,

$$\text{ALG}^w(\rho) + \alpha \cdot \text{ALG}^\pi(\rho) \leq \alpha \cdot \textit{OPT}_{PCST}$$

We design an example for which no online algorithm can guarantee an LMP competitive ratio better than $\Omega(n)$. For an arbitrary integer $m > 1$, consider a star with $m+1$ leaves $t_0, t_1, \ldots, t_m$ (thus $n = m+2$). Let v denote the center of the star. All the edges have weight zero except the edge connecting t_0 to v; the weight of which is $m-1$. The leaf t_0 is the root.

At the online step i, the adversary asks for connecting t_i to the root t_0. If the online algorithm choose not to connect t_i, it will incur a penalty cost of 1. The adversary stops the input sequence as soon as the algorithm buys the expensive edge (t_0, v). If the algorithm never buys the expensive edge, the adversary stops after requesting the connectivity of t_m. We distinguish between three scenarios; we show that in all three, the LMP competitive ratio is $\Omega(m) = \Omega(n)$.

I) *The online algorithm never buys the expensive edge.* In this case, the optimum

solution is to connect all the leaves to the root, i.e., $OPT_{PCST} = m - 1$. However, the online algorithm pays the penalty of all terminals, i.e., $\text{ALG}^{\pi} = m$. Thus in this case the algorithm is not LMP competitive at all.

II) *The online algorithm buys the expensive edge at the end of step x for an $x \leq m - 1$.* Since there are at most $m - 1$ terminals in this case, the optimum solution is to pay the penalty of every terminal, i.e., $OPT_{PCST} = x$. On the other hand, the online algorithm pays the penalty of all the previous terminals and then connects the last terminal, thus if the algorithm is LMP α-competitive, we have

$$\begin{aligned} \text{ALG}^{w}(\rho) + \alpha \cdot \text{ALG}^{\pi}(\rho) \leq \alpha \cdot OPT_{PCST} \quad &\Longrightarrow \\ (m-1) + \alpha(x-1) \leq \alpha \cdot x \quad &\Longrightarrow \\ m - 1 \leq \alpha \end{aligned}$$

III) *The online algorithm buys the expensive edge at the end of step m.* In this case the optimum solution is to connect all the terminals, thus $OPT_{PCST} = m - 1$. Similar to Case II, we have

$$\begin{aligned} \text{ALG}^{w}(\rho) + \alpha \cdot \text{ALG}^{\pi}(\rho) \leq \alpha \cdot OPT_{PCST} \quad &\Longrightarrow \\ (m-1) + \alpha(m-1) \leq \alpha \cdot (m-1) \end{aligned}$$

Therefore in this case, the algorithm is not LMP competitive at all.

CHAPTER 4

Mixed Packing/Covering Problems

4.1 Online Degree-Bounded Steiner Connectivity

In the offline scenario, the results of Fürer, Raghavachari [FR90, FR94] and Agrawal, Klein, Ravi [AKR91] were the starting point of a very popular line of work on various degree-bounded network design problems [MRS+98, Goe06, Nut12, LS13, KKN13, EV14]. We refer the reader to the next sections for a brief summary. One particular variant that has been extensively studied is the *edge-weighted* DEGREE-BOUNDED SPANNING TREE. Initiated by Ravi *et al.* [MRS+98], in this version, we are given a weight function over the edges and a bound b on the maximum degree of a vertex. The goal is to find a minimum-weight spanning tree with maximum degree at most b. The groundbreaking results obtained by Goemans [Goe06] and Singh and Lau [SL07] settle the problem by giving an algorithm that computes a minimum-weight spanning tree with degree at most $b+1$. Slightly worse result are obtained by Singh and Lau [LS13] for

the Steiner tree variant. Unfortunately, in the online setting it is not possible to obtain a comparable result. We show that for any (randomized) algorithm $\mathcal{A}$ there exists a request sequence such that $\mathcal{A}$ outputs a sub-graph that either has weight $\Omega(n) \cdot \mathrm{OPT}_b$ or maximum degree $\Omega(n) \cdot b$.

4.1.1 Contributions

In the online variant of DEGREE-BOUNDED STEINER FOREST, we are given the graph G in advance, however, demands arrive in an online fashion. At online step i, a new demand (s_i, t_i) arrives. Starting from an empty subgraph, at each step the online algorithm should augment its solution so that the endpoints of the new demand s_i and t_i are connected. The goal is to minimize the maximum degree of the solution subgraph. In the *non-uniform* variant of the problem, a degree bound $b_v \in \mathbb{R}^+$ is given for every vertex v. For a subgraph H and a vertex v, let $\deg_H(v)$ denote the degree of v in H. The *load* of a vertex is defined as the ratio $\deg_H(v)/b_v$. In the non-uniform variant of ONLINE DEGREE-BOUNDED STEINER FOREST, the goal is to find a subgraph satisfying the demands while minimizing the maximum load of a vertex.

Our algorithm is a simple and intuitive greedy algorithm. Upon the arrival of a new demand (s_i, t_i), the greedy algorithm (GA) satisfies the demand by choosing an (s_i, t_i)-path P_i such that after augmenting the solution with P_i, the maximum load of a vertex in P_i is minimum. One of our main results is to prove that the maximum load of a vertex in the output of GA is within a logarithmic factor of *OPT*, the maximum load of a vertex in

an optimal offline solution which knows all the demands in advance.

Theorem 4.1. *The algorithm GA produces an output with maximum load at most* $O(\log n) \cdot$ *OPT.*

The crux of our analysis is establishing several structural properties of the solution subgraph. First we group the demands according to the maximum load of the bottleneck vertex at the time of arrival of the demand. We then show that for every threshold $r > 0$, vertices with load at least r at the end of the run of GA, form a cut set that well separates the group of demands with load at least r at their bottleneck vertex. Since the threshold value can be chosen arbitrarily, this leads to a series of cuts that form a chain. The greedy nature of the algorithm indicates that each cut highly disconnects the demands. Intuitively, a cut that highly disconnects the graph (or the demands) implies a lower bound on the number of branches of every feasible solution.

We use a natural dual-fitting argument to show that for every cut set, the ratio between the number of demands in the corresponding group, over the total degree bound of the cut, is a lower bound for *OPT*. Hence, the problem comes down to showing that one of the cuts in the series has ratio at least $1/O(\log n)$ fraction of the maximum load h of the output of GA. To this end, we first partition the range of $r \in (0, h]$ into $O(\log n)$ layers based on the total degree bound of the corresponding cut. We then show that the required cut can be found in an interval with maximum range of r. We analyze GA formally in Section 4.1.4.1.

We complement our first theorem by giving an example for a special case of DB ST in which no online (randomized) algorithm can achieve a competitive ratio better than $\Omega(\log n)$.

Theorem 4.2. *Any (randomized) online algorithm for the degree bounded online Steiner tree problem has (multiplicative) competitive ratio* $\Omega(\log n)$. *This already holds when* $b_v = 1$ *for every node.*

We further investigate the following extensions of the online degree bounded Steiner tree problem. First, we consider the edge-weighted variant of the degree-bounded Steiner tree problem. Second, we consider the group Steiner tree version in which each demand consists of a subset of vertices, and the goal is to find a tree that covers at least one vertex of each demand group. The following theorems show that one cannot obtain a non-trivial competitive ratio for these versions in their general form.[1]

Theorem 4.3. *Consider the edge weighted variant of Online Degree Bounded Steiner Tree. For any (randomized) online algorithm* $\mathcal{A}$*, there exists an instance and a request sequence such that either* $\mathrm{E}[\mathrm{maxdegree}(\mathcal{A})] \geq \Omega(n) \cdot b$ *or* $\mathrm{E}[\mathrm{weight}(\mathcal{A})] \geq \Omega(n) \cdot OPT_b$,

[1] Our lower bound results imply that one needs to restrict the input in order to achieve competitiveness. In particular for the edge-weighted variant, our proof does not rule out the existence of a competitive algorithm when the edge weights are polynomially bounded.

where OPT_b denotes the minimum weight of a Steiner tree with maximum degree b.

Theorem 4.4. *There is no deterministic algorithm with competitive ratio $o(n)$ for the* DEGREE-BOUNDED GROUP STEINER TREE *problem.*

4.1.2 Related Degree-Bounded Connectivity Problems

The classical family of degree-bounded network design problems have various applications in broadcasting information, package distribution, decentralized communication networks, etc. (see e.g. [GMK88, HGM03]). Ravi *et al.* ([MRS+98], J. Algorithms'98), first considered the general *edge-weighted* variant of the problem. They give a bi-criteria $(O(\log n), O(\log n) \cdot b)$-approximation algorithm, i.e., the degree of every node in the output tree is $O(\log n) \cdot b$ while its total weight is $O(\log n)$ times the optimal weight. A long line of work (see e.g. [KR00], STOC'00 and [KR05], SIAM J. C.) was focused on this problem until a groundbreaking breakthrough was obtained by Goemans ([Goe06], FOCS'06); his algorithm computes a minimum-weight spanning tree with degree at most $b + 2$. Later on, Singh and Lau ([SL07], STOC'07) improved the degree approximation factor by designing an algorithm that outputs a tree with optimal cost while the maximum degree is at most $b + 1$.

In the *degree-bounded survivable network design* problem, a number d_i is associ-

ated with each demand (s_i, t_i). The solution subgraph should contain at least d_i edge-disjoint paths between s_i and t_i. Indeed this problem has been shown to admit bi-criteria approximation algorithms with constant approximation factors (e.g. [LS13], STOC'08). We refer the reader to a recent survey in [LRS11]. This problem has been recently considered in the node-weighted variant too (see e.g. [Nut12, EV14]). The degree-bounded variant of several other problems such as k-MST and k-arborescence has also been considered in the offline setting for which we refer the reader to [KKN13, BKN09] and references therein.

4.1.3 Related Online Problems

Online network design problems have attracted substantial attention in the last decades. The online edge-weighted Steiner tree problem, in which the goal is to find a minimum-weight subgraph connecting the demand nodes, was first considered by Imase and Waxman ([IW91], SIAM J. D. M.'91). They showed that a natural greedy algorithm has a competitive ratio of $O(\log n)$, which is optimal up to constants. This result was generalized to the online edge-weighted Steiner forest problem by Awerbuch *et al.* ([AAB96], SODA'96) and Berman and Coulston ([BC97], STOC'97). Later on, Naor, Panigrahi, and Singh ([NPS11]), FOCS'11) and Hajiaghayi, Liaghat, and Panigrahi ([HLP13], FOCS'13), designed poly-logarithmic competitive algorithms for the more general *node-weighted* variant of Steiner connectivity problems. This line of work has been further investigated in the prize-collecting version of the problem, in which one can ig-

nore a demand by paying its given penalty. Qian and Williamson ([QW11], ICALP'11) and Hajiaghayi, Liaghat, and Panigrahi ([HLP14], ICALP'14) develop algorithms with a poly-logarithmic competitive algorithms for these variants.

The online b-matching problem is another related problem in which vertices have degree bounds but the objective is to maximize the size of the solution subgraph. In the worst case model, the celebrated result of Karp *et al.* ([KVV90], STOC'90) gives a $(1-1/e)$-competitive algorithm. Different variants of this problem have been extensively studied in the past decade, e.g., for the random arrival model see [FMMM09, KMT11, MY11, MSVV07], for the full information model see [MOGS12, MOGZ12], and for the prophet-inequality model see [AHL+11, AHL12, AHL13]. We also refer the reader to the comprehensive survey by Mehta [Meh12].

Many of the problems mentioned above can be described with an online packing or covering linear program. Initiated by work of Alon *et al.* [AAA+09] on the online set cover problem, Buchbinder and Naor developed a strong framework for solving packing/covering LPs fractionally online. For the applications of their general framework in solving numerous online problems, we refer the reader to the survey in [BN09a]. Azar *et al.* [ABFP13] generalize this method for the fractional *mixed* packing and covering LPs. In particular, they show an application of their method for integrally solving a generalization of capacitated set cover. Their result is a bi-criteria competitive algorithm that violates the capacities by at most an $O(\log^2 n)$ factor while the cost of the ouput is within $O(\log^2 n)$ factor of optimal. We note that although the fractional variant of our problem

is a special case of online mixed packing/covering LPs, we do not know of any online rounding method for Steiner connectivity problems.

4.1.4 Preliminaries

Let $G = (V, E)$ denote an undirected graph of size n ($|V| = n$). For every vertex $v \in V$, let $b_v \in \mathbb{R}^+$ denote the *degree bound* of v. In the DEGREE-BOUNDED STEINER FOREST problem, we are given a sequence of connectivity *demands*. The i^{th} demand is a pair of vertices (s_i, t_i) which we call the *endpoints* of the demand. An algorithm outputs a subgraph $H \subseteq G$ in which for every demand its endpoints are connected. The *load* of a vertex v w.r.t. H is defined as $\ell_H(v) = \deg_H(v)/b_v$. We may drop the subscript H when it is clear from the context. The goal is to find a subgraph H that minimizes the maximum load of a vertex. Observe that one can always find an optimal solution without a cycle, hence the name Steiner *forest*. Furthermore, without loss of generality[2], we assume that the endpoints of the demands are distinct vertices with degree one in G and degree bound infinity. We denote the maximum load of a vertex in an optimal subgraph by $\mathit{OPT} = \min_H \max_v \ell_H(v)$.

The following mixed packing/covering linear program ($\mathbb{P}$) is a relaxation for the natural integer program for DEGREE-BOUNDED STEINER FOREST. Let $\mathcal{S}$ denote the col-

[2]One can add a dummy vertex for every vertex $v \in V$ connected to v. We then interpret a demand between two vertices as a demand between the corresponding dummy vertices. The degree bound of a dummy vertex can be set to infinity. This transformation can increase the degree of any node by at most a factor of 2, which does not affect our asymptotic results.

lection of subsets of vertices that separate the endpoints of at least one demand. For a set of vertices S, let $\delta(S)$ denote the set of edges with exactly one endpoint in S. In $\mathbb{P}$, for an edge e, $\mathbf{x}(e) = 1$ indicates that we include e in the solution while $\mathbf{x}(e) = 0$ indicates otherwise. The variable α indicates an upper bound on the load of every vertex. The first set of constraints ensures that the endpoints of every demand is connected. The second set of constraints ensures that the load of a vertex is upper bounded by α. The program $\mathbb{D}$ is the dual of the LP relaxation $\mathbb{P}$.

$$\begin{aligned}
&\text{minimize} \quad \alpha && (\mathbb{P})\\
&\forall S \subseteq \mathcal{S} \quad \sum_{e \in \delta(S)} \mathbf{x}(e) \geq 1 && (\mathbf{y}(S))\\
&\forall v \in V \quad \sum_{e \in \delta(\{v\})} \mathbf{x}(e) \leq \alpha \cdot b_v && (\mathbf{z}(v))\\
&\qquad \mathbf{x}(e), \alpha \in \mathbb{R}_{\geq 0}
\end{aligned}$$

$$\begin{aligned}
&\text{maximize} \quad \sum_{S \in \mathcal{S}} \mathbf{y}(S) && (\mathbb{D})\\
&\forall e = (u, v) \quad \sum_{S : e \in \delta(S)} \mathbf{y}(S) \leq \mathbf{z}(u) + \mathbf{z}(v) && (\mathbf{x}(e))\\
&\qquad \sum_{v} \mathbf{z}(v) b_v \leq 1 && (\alpha)\\
&\qquad \mathbf{z}(v), \mathbf{y}(S) \in \mathbb{R}_{\geq 0}
\end{aligned}$$

In the online variant of the problem, G and the degree bounds are known in advance, however, the demands are given one by one. Upon the arrival of the i^{th} demand, the online algorithm needs to output a subgraph H_i that satisfies all the demands seen so far. The output subgraph can only be augmented, i.e., for every $j < i$, $H_j \subseteq H_i$. The *competitive ratio* of an online algorithm is then defined as the worst case ratio of $\max_v \ell_H(v)/\mathit{OPT}$ over all possible demand sequences where H is the final output of the algorithm.

4.1.4.1 Online Degree-Bounded Steiner Forest

Consider an arbitrary online step where a new demand (s, t) arrived. Let H denote the online output of the previous step. In order to augment H for connecting s and t, one can shortcut through the connected components of H. We say an edge $e = (u, v)$ is an *extension edge* w.r.t. H, if u and v are not connected in H. Let P denote an (s, t)-path in G. The extension part P^* of P is defined as the set of extension edges of P. Observe that augmenting H with P^* satisfies the demand (s, t). For a vertex v, we define $\ell_H^+(v) = \ell_H(v) + 2/b_v$ to be the *uptick load of* v. We slightly misuse the notation by defining $\ell_H^+(P^*) = \max_{v \in V(P^*)} \ell_H^+(v)$ as the uptick load of P^*, where $V(P^*)$ is the set of endpoints of edges in P^*.

The *greedy algorithm* (GA) starts with an empty subset H. Upon arrival of the i-th demand (s_i, t_i), we find a path P_i with smallest uptick load $\ell_H^+(P_i^*)$ where P_i^* is the extension part of P_i. We break ties in favor of the path with a smaller number of edges. Note that the path P_i can be found efficiently using Dijkstra-like algorithms. We define $\tau_i = \ell_H^+(P_i^*)$ to be the *arrival threshold* of the i-th demand. We satisfy the new demand by adding P_i^* to the edge set of H (see Algorithm 5).

4.1.5 Analysis

We now use a dual-fitting approach to show that GA has an asymptotically tight competitive ratio of $O(\log(n))$. In the following we use GA to also refer to the *final*

output of our greedy algorithm. We first show several structural properties of GA. We then use these combinatorial properties to construct a family of feasible dual vectors for $\mathbb{D}$. We finally show that there always exists a member of this family with an objective value of at least a $1/O(\log(n))$ fraction of the maximum load of a vertex in GA. This in turn proves the desired competitive ratio by using weak duality.

For a real value $r \geq 0$, let $\Gamma(r)$ denote the set of vertices with $\ell^{+}_{\mathsf{GA}}(v) \geq r$. Let $D(r)$ denote the indices of demands i for which the arrival threshold τ_i is at least r. For a subgraph X, let $\mathrm{CC}(X)$ denote the collection of connected components of X. For ease of notation, we may use the name of a subgraph to denote the set of vertices of that subgraph as well. Furthermore, for a graph X and a subgraph $Y \subseteq X$, let $X \setminus Y$ denote the graph obtained from X by removing the vertices of Y.

Recall that $\mathcal{S}$ is the collection of subsets that separate the endpoints of at least one demand. The following lemma, intuitively speaking, implies that $\Gamma(r)$ well-separates the endpoints of $D(r)$.

Lemma 4.1. *For any threshold $r > 0$, we have $|\mathrm{CC}(G \setminus \Gamma(r)) \cap \mathcal{S}| \geq |D(r)| + 1$.*

Proof. For a vertex $v \in G \setminus \Gamma(r)$ we use $\mathrm{CC}(v)$ to denote its connected component. Observe that, since the endpoints of demands are nodes with infinite degree bound, the endpoints are *not* contained in $\Gamma(r)$, and, hence, are in $G \setminus \Gamma(r)$.

We construct a graph F that has one node for every connected component in $G \setminus \Gamma(r)$ that contains an endpoint of a demand in $D(r)$. Edges in F are defined as follows. For every demand $i \in D(r)$ between s_i and t_i, we add an edge that connects the components

$\mathrm{CC}(s_i)$ and $\mathrm{CC}(t_i)$. In the following we argue that F does not contain cycles. This gives the lemma since in a forest $|D(r)| + 1 = |E_F| + 1 \leq |V_F| \leq |\mathrm{CC}(G \setminus \Gamma(r)) \cap \mathcal{S}|$.

Assume for contradiction that the sequence $(e_{i_0}, \ldots, e_{i_{k-1}})$, $k \geq 2$ forms a (minimal) cycle in F, where e_{i_j} corresponds to the demand between vertices s_{i_j} and t_{i_j} (see Figure 4.1). Assume wlog. that $e_{i_{k-1}}$ is the edge of the cycle for which the corresponding demand appears last, i.e., $i_{k-1} \geq i_j$ for every $j < k$. Let H denote the online solution at the time of arrival of the demand i_{k-1}. We can augment H to connect each t_{i_j} to $s_{i_{j+1 \bmod k}}$, $0 \leq j \leq k-1$ without using any node from $\Gamma(r)$, since these nodes are in the same component in $G \setminus \Gamma(r)$. But then we have a path P between $s_{i_{k-1}}$ and $t_{i_{k-1}}$ and the extension part P^* does not contain any vertex from $\Gamma(r)$. This is a contradiction since the arrival threshold for the demand i_{k-1} is at least r. □

Lemma 4.1 shows that cutting $\Gamma(r)$ highly disconnects the demands in $D(r)$. Indeed this implies a bound on the *toughness* of the graph. Toughness, first defined by Chvátal [Chv73] and later generalized by Agrawal *et al.* [AKR95], is a measure of how easy it is to disconnect a graph into many components by removing a few vertices. More formally, the toughness of a graph is defined as $\min_{X \subseteq V} \frac{|X|}{|\mathrm{CC}(G \setminus X)|}$. For the spanning tree variant of the problem, it is not hard to see that *OPT* is at least the reciprocal of toughness. Although it is more involved, we can still establish a similar relation for the non-uniform Steiner forest problem (see Lemma 4.3). However, first we need to show a lower bound on the number of demands separated by $\Gamma(r)$.

We establish a lower bound for $|D(r)|$ with respect to the load of vertices in $\Gamma(r)$.

For any $r, b > 0$ we define $\Gamma_b(r) := \{\ell^+_{\mathsf{GA}}(v) \geq r \wedge b_v \geq b\}$, as the set of nodes with degree bound at least b that have uptick load at least r in the final online solution. We further define

$$\text{excess}(r, b) = \sum_{v \in \Gamma_b(r)} \Big(\deg_{\mathsf{GA}}(v) - \lceil r b_v \rceil + 2 \Big) \ ,$$

which sums the (absolute) load that nodes in $\Gamma_b(r)$ receive *after* their *uptick load* became r or larger. The following lemma relates $|D(r)|$ to $\text{excess}(r, b)$.

Lemma 4.2. *For any $r, b > 0$, $\text{excess}(r, b) \leq 2|D(r)| + 3|\Gamma_b(v)|$.*

Proof. Consider some online step i. Let H denote the output of the previous step. Let P_i^* be the extension part of the path selected by GA and let $V(P_i^*)$ denote the endpoints of edges of P_i^*. Since in GA we break ties in favor of the path with the smaller number of edges, we can assume that the path does not go through a connected component of H more than once, i.e., for every $C \in \text{CC}(H)$, $|V(P_i^*) \cap C| \leq 2$.

Consider the variable quantity $\delta(r, b) := \sum_{v : \ell^+_H(v) \geq r \wedge b_v \geq b} (\deg_H(v) - \lceil r b_v \rceil + 2)$ throughout the steps of GA where H denotes the output of the algorithm at every step. Intuitively, at each step $\delta(r, b)$ denotes the total increment in degrees of those vertices in $\Gamma_b(r)$ that already reached uptick load r. In particular, at the end of the run of GA, $\delta(r, b) = \text{excess}(r, b)$.

Now suppose at step i adding the edges P_i^* to H increases $\delta(r, b)$ by q_i. There are two reasons for such an increase. On the one hand, if the demand i is from $D(r)$ it may simply increase $\deg_H(v)$ for some node with uptick load at least r. On the other hand also if the demand is not from $D(r)$ it may cause a node to increase its uptick load to r

in which case it could contribute to the above sum with at most 1 (in a step the degree may increase by 2; the first increase by 1 could get the uptick load to $\geq r$ and the second increase contributes to the sum).

Clearly increases of the second type can accumulate to at most $|\Gamma_b(r)|$. In order to derive a bound on the first type of increases recall that we assume that endpoints of demands are distinct vertices with degree one and thus s_i and t_i are outside any connected component of H. Since $V(P_i^*)$ contains at most two vertices of a connected component of H, we can assert that the path selected by GA is passing through at least $q_i/2$ components of H that contain some vertices of $\Gamma_b(r)$. Hence, after adding P_i^* to the solution, the number of connected components of the solution that contain some vertices of $\Gamma_b(r)$ decreases by at least $(q_i - 2)/2$. However, throughout all the steps, the total amount of such decrements cannot be more than the number of vertices in $\Gamma_b(r)$. Therefore

$$\begin{aligned}
\text{excess}(r, b) &= \sum_{i \in D(r)} q_i + \sum_{i \notin D(r)} q_i \\
&= 2|D(r)| + \sum_{i \in D(r)} (q_i - 2) + |\Gamma_b(r)| \\
&\leq 2|D(r)| + 3|\Gamma_b(r)| \ ,
\end{aligned}$$

and the lemma follows. □

Let $\Delta > 0$ denote the minimum degree bound of a vertex with non-zero degree in an optimal solution. Note that this definition implies that $OPT \geq 1/\Delta$. For a set of vertices Γ, let $b(\Gamma) = \sum_{v \in \Gamma} b_v$. We now describe the main dual-fitting argument for bounding the maximum load in GA.

Lemma 4.3. *For any $r > 0$, $|D(r)|/b(\Gamma_\Delta(r)) \leq$ OPT.*

Proof. Let G_Δ denote the graph obtained from G by removing vertices with degree bound below Δ. Similarly, let $\mathcal{S}_\Delta$ denote the collection of sets that separate a demand in G_Δ. Consider a slightly modified dual program $\mathbb{D}_\Delta$ defined on G_Δ and $\mathcal{S}_\Delta$, i.e., we obtain $\mathbb{D}_\Delta$ from the original dual program by removing all vertices with degree bound below Δ. We note that the objective value of a dual feasible solution for $\mathbb{D}_\Delta$ is still a lower bound for *OPT*. In the remainder of the proof, we may refer to $\mathbb{D}_\Delta$ as the dual program. Recall that in a feasible dual solution, we need to define a dual value $\mathbf{y}(S)$ for every cut $S \in \mathcal{S}_\Delta$ such that for every edge $e = (u, v)$ the total value of cuts that contain e does not exceed $\mathbf{z}(u) + \mathbf{z}(v)$. On the other hand, $\sum_v \mathbf{z}(v)b_v$ cannot be more than one.

For a real value $r > 0$, we construct a dual vector $(\mathbf{y}_r, \mathbf{z}_r)$ as follows. For every $v \in \Gamma_\Delta(r)$, set $\mathbf{z}_r(v) = 1/b(\Gamma_\Delta(r))$; for other vertices set $\mathbf{z}_r(v) = 0$. For every component $S \in \mathrm{CC}(G_\Delta \setminus \Gamma_\Delta(r)) \cap \mathcal{S}_\Delta$, set $\mathbf{y}_r(S) = 1/b(\Gamma_\Delta(r))$; for all other sets set $\mathbf{y}_r(S) = 0$. We prove the lemma by showing the feasibility of the dual vector $(\mathbf{y}_r, \mathbf{z}_r)$ for $\mathbb{D}_\Delta$.

Consider an arbitrary component $C \in \mathrm{CC}(G \setminus \Gamma(r))$. By definition, C separates at least one demand. Let i be such a demand and let $t_i \in C$ denote an endpoint of it. Removing vertices with degree bound below $1/\Delta$ from C may break C into multiple smaller components. However, an endpoint of a demand has degree bound infinity, and, hence, the component that contains t_i belongs to $\mathcal{S}_\Delta$. Therefore $|\mathrm{CC}(G \setminus \Gamma(r)) \cap \mathcal{S}| \leq |\mathrm{CC}(G_\Delta \setminus \Gamma_\Delta(r)) \cap \mathcal{S}_\Delta|$; which by Lemma 4.1 leads to $|\mathrm{CC}(G_\Delta \setminus \Gamma_\Delta(r)) \cap \mathcal{S}_\Delta| \geq |D(r)| + 1$. Therefore the dual objective for the vector $(\mathbf{y}_r, \mathbf{z}_r)$ is at least

$$\sum_{S \in \mathcal{S}_\Delta} \mathbf{y}_r(S) = \sum_{S \in \mathrm{CC}(G_\Delta \setminus \Gamma_\Delta(r)) \cap \mathcal{S}_\Delta} \frac{1}{b(\Gamma_\Delta(r))} \geq \frac{|D(r)|+1}{b(\Gamma_\Delta(r))} \ .$$

Thus we only need to show that $(\mathbf{y}_r, \mathbf{z}_r)$ is feasible for Program $\mathbb{D}_\Delta$. First, by construction we have

$$\sum_v \mathbf{z}_r(v) b_v = \sum_{v \in \Gamma_\Delta(r)} \frac{1}{b(\Gamma_\Delta(r))} \cdot b_v = 1 \ .$$

Now consider an arbitrary edge $e = (u, v)$. If e does not cross any of the components in $\mathrm{CC}(G_\Delta \setminus \Gamma_\Delta(r))$, then $\sum_{S: e \in \delta(S)} \mathbf{y}_r(S) = 0$ and we are done. Otherwise, $\sum_{S: e \in \delta(S)} \mathbf{y}_r(S) = 1/b(\Gamma_\Delta(r))$. However, exactly one endpoint of e is in $\Gamma_\Delta(r)$. Thus $\mathbf{z}_r(u) + \mathbf{z}_r(v) = 1/b(\Gamma_\Delta(r))$, which implies that the dual vector is feasible. □

We now have all the ingredients to prove the main theorem.

of Theorem 4.1. Let GA denote the final output of the greedy algorithm. Let h denote the maximum load of a vertex in GA, i.e., $h = \max_v \ell_{\mathsf{GA}}(v)$. Furthermore, let $h_\Delta = \max_{v: b_v \geq \Delta} \ell_{\mathsf{GA}}(v)$. In the following we first show that $h_\Delta \leq O(\log n) \cdot \mathit{OPT}$. For this we use the folowing claim.

Claim 4.1. *There exists an $r > 0$ such that*

$$\mathrm{excess}(r, \Delta) \geq \frac{h_\Delta - 1/\Delta}{4 \log_2 n + 6} \cdot b(\Gamma_\Delta(r)) - |\Gamma_\Delta(r)| \ .$$

Proof. Recall that $\Gamma_\Delta(r)$ is the set of vertices with uptick load at least r in GA and degree bound at least Δ. The function $b(\Gamma_\Delta(r))$ is non-increasing with r since for every $r_1 < r_2$, $\Gamma_\Delta(r_2) \subseteq \Gamma_\Delta(r_1)$. For $r = 1/\Delta$, $b(\Gamma_\Delta(r)) \leq n(n+1)\Delta$ as a node with degree bound

$b_v > (n+1)\Delta$ will have uptick load at most $(n+1)/b_v < 1/\Delta$, and, hence, will not be in $\Gamma_\Delta(r)$. Also, $r < h_\Delta$ implies that $b(\Gamma_\Delta(r)) \geq \Delta$ as there exists a node with load h_Δ in $\Gamma_\Delta(r)$ and this node has degree bound at least Δ.

We now partition the range of r (from $1/\Delta$ to h_Δ) into logarithmically many intervals. We define the q-th interval by

$$U(q) = \left\{ r \mid 1/\Delta \leq r < h_\Delta \ \wedge \ 2^q/\Delta \leq b(\Gamma_\Delta(r)) < 2^{q+1}/\Delta \right\} \ ,$$

for $q \in \{0, \ldots, \lceil \log_2((n+1)n) \rceil\}$. We further set $\overline{r}(q) = \max U(q)$ and $\underline{r}(q) = \min U(q)$, and call $\overline{r}(q) - \underline{r}(q)$ the length of the q-th interval. Since there are only $2\log_2 n + 3$ possible choices for q there must exist an interval of length at least $(h_\Delta - 1/\Delta)/(2\log_2 n + 3)$.

Consider a node v that is contained in $\Gamma_\Delta(\overline{r})$ and, hence, also in $\Gamma_\Delta(\underline{r})$. This node starts contributing to $\mathrm{excess}(\underline{r}, \Delta)$, once its uptick load reached $\underline{r}$ and contributes at least until its uptick load reaches $\overline{r}$. Hence, the total contribution is at least $\deg(\overline{r}) - \deg(\underline{r})$, where $\deg(\overline{r})$ and $\deg(\underline{r})$ denotes the degree of node v when reaching uptick load $\overline{r}$ and $\underline{r}$, respectively. We have,

$$\begin{aligned} \deg(\overline{r}) - \deg(\underline{r}) &= (\lceil \overline{r} b_v \rceil - 2) - (\lceil \underline{r} b_v \rceil - 2) \\ &\geq (\overline{r} - \underline{r}) b_v - 1 \ . \end{aligned}$$

Summing this over all nodes in $\Gamma_\Delta(\overline{r})$ gives

$$\begin{aligned} \mathrm{excess}(\underline{r}, \Delta) &\geq (\overline{r} - \underline{r}) \cdot b(\Gamma_\Delta(\overline{r})) - |\Gamma_\Delta(\overline{r})| \\ &\geq \frac{1}{2}(\overline{r} - \underline{r}) \cdot b(\Gamma_\Delta(\underline{r})) - |\Gamma_\Delta(\underline{r})| \\ &\geq \frac{h_\Delta - 1/\Delta}{4\log_2 n + 6} \cdot b(\Gamma_\Delta(\underline{r})) - |\Gamma_\Delta(\underline{r})| \ , \end{aligned}$$

where the second inequality uses the fact that $|\Gamma_\Delta(\overline{r})| \leq |\Gamma_\Delta(\underline{r})| \leq |\Gamma_\Delta(\overline{r})|/2$. □

Claim 4.2. $h_\Delta \leq (24 \log_2 n + 37)\mathit{OPT}$.

Proof. If we use the r from the previous claim and solve for h_Δ we obtain

$$\begin{aligned} h_\Delta &\leq (4 \log_2 n + 6) \cdot \frac{\text{excess}(r, \Delta) + \Gamma_\Delta(r)}{b(\Gamma_\Delta(r))} + \frac{1}{\Delta} \\ &\leq (4 \log_2 n + 6) \cdot \frac{2D(r) + 4\Gamma_\Delta(r)}{b(\Gamma_\Delta(r))} + \frac{1}{\Delta} \\ &\leq (4 \log_2 n + 6) \cdot \left(2\mathit{OPT} + 4\frac{1}{\Delta}\right) + \frac{1}{\Delta} \\ &\leq (24 \log_2 n + 37) \cdot \mathit{OPT} \ . \end{aligned}$$

Here the second inequality is due to Lemma 4.2, the third uses Lemma 4.3 and the fact that $b(\Gamma_\Delta(r)) \geq \Delta|\Gamma_\Delta(r)|$. The last inequality uses $\mathit{OPT} \geq 1/\Delta$. □

We now bound the maximum load h in terms of the restricted maximum load h_Δ. Consider a vertex v^* with maximum load $\ell_{\mathsf{GA}}(v^*) = h$. If $b_{v^*} \geq \Delta$ then $h_\Delta = h$ and we are done. Otherwise consider the last iteration i in which the degree of v^* is increased in the solution. Let H denote the output of the algorithm at the end of the previous iteration $i - 1$. The degree of v^* in the online solution is increased by at most two at iteration i. Hence

$$h \leq \ell_H^+(v^*) \leq \tau_i \ .$$

Recall that in our greedy algorithm, τ_i is the minimum uptick load of a path that satisfies the new demand. Let P denote a path that connects s_i and t_i in an optimal solution. Recall

that by the definition, $b_v \geq \Delta$ for every vertex v of P. For every vertex v in P we have

$$\begin{aligned}
\tau_i &\leq \ell_H^+(v) && \\
&\leq h_\Delta + \frac{2}{b_v} && \ell_H(v) \leq \ell_{\mathsf{GA}}(v) \leq h_\Delta \\
&\leq h_\Delta + \frac{2}{\Delta} && b_v \geq \Delta \\
&\leq h_\Delta + 2\mathit{OPT} && \mathit{OPT} \geq 1/\Delta \\
&\leq O(\log n) \cdot \mathit{OPT} && \text{by the above claim}
\end{aligned}$$

Therefore leading to $h \leq O(\log n) \cdot \mathit{OPT}$. □

4.1.5.1 An Asymptotically Tight Lower Bound

In the following we show a lower bound for ONLINE DEGREE-BOUNDED STEINER TREE. Consider a graph $G = (X \uplus Z, E)$, with $|Z| = 2^\ell$ and $|X| = \binom{2^\ell}{2}$. For every pair $\{z_1, z_2\}$ of nodes from Z there exists an edge $\{z_1, z_2\} \in E$ and a node $x \in X$ that is connected to z_1 and z_2. An arbitrary node from Z acts as root for the Steiner tree problem.

For a node $z \in Z$, an algorithm A, and a request sequence σ (consisting of nodes from X) we use $\deg_{A,\sigma}(z)$ to denote the number of neighbors of *z among all nodes from X* in the Steiner tree obtained when running algorithm A on request sequence σ. Similarly, we use $\deg'_{A,\sigma}(z)$ to denote the number of those neighbors of z that also appear in σ. Note that $\deg(\cdot)$ and $\deg'(\cdot)$ ignore edges between nodes from Z as an algorithm (online or offline) can simply connect all nodes in Z in a cycle which increases the degree of any

node by only 2.

Lemma 4.4. *Fix a possibly randomized online algorithm A. For any subset $S \subseteq Z$, $|S| = 2^s$, $0 \leq s \leq \ell$ there exists a request sequence σ_S consisting of terminals from X s.t.*

- *for a node $x \in \sigma_S$ both its neighbors in G are from set S;*
- *there exists a node $z^* \in S$ with $\mathrm{E}[\deg'_{A,\sigma}(z^*)] \geq s/2$;*
- *there exists an offline algorithm OFF for servicing requests in σ with $\max_{z \in S}\{\deg_{\mathrm{OFF},\sigma}(z)\} \leq 1$, and $\deg_{\mathrm{OFF},\sigma}(z) = 0$ for $z \in (Z \setminus S) \cup \{z^*\}$.*

Proof. We prove the lemma by induction over s. The base case $s = 0$ holds trivially when choosing the empty request sequence. For the induction step consider an arbitrary subset $S \subseteq Z$ with $|S| = 2^{s+1}$. Partition S into two disjoint subsets S_1 and S_2 of cardinality 2^s each.

Let σ_1 be the request sequence that exists due to induction hypothesis for set S_1. Hence, there is a node $z_1^* \in S_1$ with $\mathrm{E}[\deg'_{A,\sigma_1}(z_1^*)] \geq s/2$. Now, let $\tilde{A}$ behave like algorithm A *after* it already received a request sequence σ_1 (hence, it starts with all edges that are chosen when running A on σ_1; note, however, that $\deg'_{\tilde{A},\sigma_2}(\cdot)$ only takes into account edges incident to nodes from σ_2). Due to induction hypothesis for $\tilde{A}$ and set S_2 there exists a request sequence σ_2 such that $\mathrm{E}[\deg'_{\tilde{A},\sigma_2}(z_2^*)] \geq s/2$ for a node $z_2^* \in S_2$. Hence, the request sequence $\sigma = \sigma_1 \circ \sigma_2$ fulfills $\mathrm{E}[\deg'_{A,\sigma}(z_1^*)] \geq s/2$ and $\mathrm{E}[\deg'_{A,\sigma}(z_2^*)] \geq s/2$.

We extend the request sequence by appending the node x that is connected to z_1^* and z_2^* in G. After serving the request one of the edges $\{x, z_1\}$ or $\{x, z_2\}$ must be chosen with

probability at least 1/2 by A. Hence, afterwards either $\mathrm{E}[\deg'_{A,\sigma}(z_1^*)]$ or $\mathrm{E}[\deg' \delta_{A,\sigma}(z_2^*)]$ must be at least $(s+1)/2$.

It remains to argue that there exists a good offline algorithm. Combining the offline algorithms OFF_1 and OFF_2 for σ_1 and σ_2 gives an offline algorithm for $\sigma_1 \circ \sigma_2$ that has $\max_z\{\deg_{\mathrm{OFF},\sigma_1\circ\sigma_2}(z)\} \leq 1$ and $\deg_{\mathrm{OFF},\sigma_1\circ\sigma_2}(z)\} = 0$ for $z \notin S_1 \cup S_2$ and for $z \in \{z_1^*, z_2^*\}$. Now, when the node x connected to z_1^* and z_2^* is appended to the request sequence the offline algorithm can serve this request by either buying edge $\{x, z_1\}$ or $\{x, z_2\}$, and can therefore gurantee that z_* (either z_1 or z_2) has degree 0. □

of Theorem 4.2. Choosing $s = \log n$ in the above lemma gives our lower bound. □

4.1.5.2 Figures

4.1.6 Lower Bounds for Other Degree-Bounded Steiner Connectivity Problems

In this section we consider strong lower bounds for the general form of two variants of degree-bounded network design problems.

Algorithm 5 Online Degree-Bounded Steiner Forest

Input: A graph G, and an online stream of demands $(s_1, t_1), (s_2, t_2), \ldots$.

Output: A set H of edges such that every given demand (s_i, t_i) is connected via H.

Offline Process:

1: Initialize $H = \varnothing$.

Online Scheme; assuming a demand (s_i, t_i) is arrived:

1: Compute P_i, a (s_i, t_i)-path with the smallest uptick load $\ell_H^+(P_i^*)$ in its extension part.

- Shortcut the connected components of H by replacing the edges of a component with that of a clique.
- In the resulting graph, define the distance of a vertex v from s_i as the minimum uptick load of a (s_i, v)-path.
- Find P_i by evoking a Dijkstra-like algorithm according to this notion of distance.

2: Set $H = H \cup P_i^*$.

4.1.6.1 Online Degree-Bounded Edge-weighted Steiner Tree

Below we present a graph instance $G = (V, E)$ for online bounded-degree edge-weighted Steiner tree in which for any (randomized) online algorithm A there exists a demand sequence for which A either violates the degree bound by a large factor or gen-

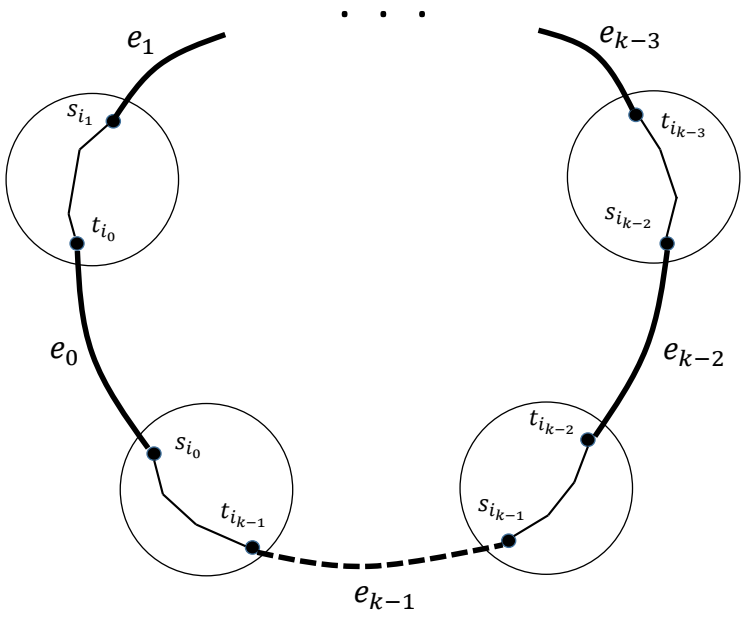
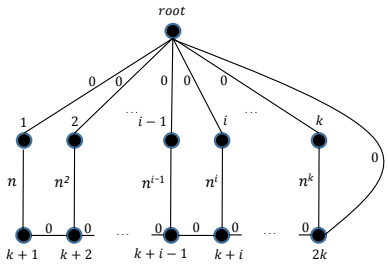


Figure 4.1: Existence of a cycle implies that of a path with low uptick load in its extension part.

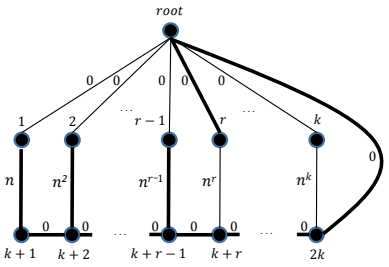
erates a much larger weight than the optimum.

Consider a graph G as shown in Figure 4.2a, which has $n = 2k + 1$ nodes and $3k$ edges. Every node i ($1 \leq i \leq k$) is connected to the root with a zero-weight edge and to node $k + i$ with weight n^i. In addition, there exist zero-weight edges connecting node i ($k + 1 \leq i < 2k$) to node $i + 1$, and there exists a zero-weight edge that connects node $2k$ to the root. We assume that all node weights b_v are equal to one and the degree bound b for *OPT* is equal to 3.

of Theorem 4.3. The adversary consecutively presents terminals starting from node 1. At each step i the algorithm A adds some edges to its current solution such that the i^{th}



(a) The graph G consists of $2k + 1$ nodes and $3k$ edges.



(b) The highlighted subtree represents an optimum solution OPT_3.

Figure 4.2: The hard example for edge-weighted Steiner tree

terminal $t_i = i$ gets connected to the $root$.

We use X_i to denote the subset of edges chosen by A after step i. We also use $0 \leq p_{ij} \leq 1$ and $0 \leq q_{ij} \leq 1$ to denote the probability that the edges $\{i, root\}$ and $\{i, k + i\}$, respectively, are in X_j . After each step j, all terminals t_i $(i \leq j)$ must be connected to the $root$ by at least one of $\{i, root\}$ or $\{i, k + i\}$. Therefore $p_{ij} + q_{ij} \geq 1$. In addition, for every $j_1 \leq j_2$ we have $q_{ij_1} \leq q_{ij_2}$ and $p_{ij_1} \leq p_{ij_2}$, because $X_{j_1} \subseteq X_{j_2}$. The adversary stops the sequence at the first step r for which $q_{rr} > 1/2$. If this never happens the sequence is stopped after requesting k nodes.

We use OPT_b to denote the weight of a minimum Steiner tree with maximum degree $b = 3$. In order to find an upper bound for OPT_b, consider the following tree T:

$$T = \{\{i, k + i\}|1 \leq i \leq r - 1\} \cup \{\{i, i + 1\}|k \leq i < 2k\} \cup \{\{2k, root\}, \{r, root\}\} \ .$$

As we can observe in Figure 4.2b, T meets the degree bounds and connects the first r

terminals to the root, so forms a valid solution. We have

$$w(\textit{OPT}) \leq w(T) = \sum_{i=1}^{r-1} n^i \in O(n^{r-1}) \ .$$

Back to algorithm A, the adversary causes one of the following two cases:

1. The process stops at step r with $q_{rr} > \frac{1}{2}$. Then

$$\mathrm{E}[w(X_r)] = \sum_{i=1}^{r} q_{ir} \cdot n^i \geq q_{rr} \cdot n^r \geq \frac{n^r}{2} \ .$$

   Hence, $\mathrm{E}[w(X)] \geq \Omega(n) \cdot w(\textit{OPT})$.

2. The process continues until step k, i.e., for every $i \leq k$, $q_{ii} \leq \frac{1}{2}$.

$$\mathrm{E}[\deg(\mathit{root})] \geq \sum_{i=1}^{k} p_{ik} \geq \sum_{i=1}^{k} p_{ii} \geq \sum_{i=1}^{k} (1 - q_{ii}) \geq \frac{k}{2} = \frac{n-1}{4} \ .$$

   Hence, $\mathrm{E}[\deg(\mathit{root})] \geq \Omega(n) \cdot b$.

This shows that for any online algorithm A there is a demand sequence on which A either generates a large weight or violates the degree bound by a large factor. □

4.1.6.2 Online Degree-Bounded Group Steiner Tree

In this section, presenting an adversary scenario, we show there is no deterministic algorithm for ONLINE DEGREE-BOUNDED GROUP STEINER TREE with competitive ratio $o(n)$ even if G is a star graph.

of Theorem 4.4. For any integer $n > 1$, we provide a graph instance G of size n and an online scenario, in which no deterministic algorithm can obtain a competitive ratio better

than $n-1$. Let G be a star with $n-1$ leaves v_2 to v_n, and v_1 be the internal node. For an algorithm $\mathcal{A}$, we describe the adversary scenario as follows.

Let v_1 be the root. Let the first demand group be the set of all leaves. Whenever $\mathcal{A}$ connects a node v_i to v_1 in H, adversary removes the selected nodes v_i from the next demand groups, until all leaves are connected to v_1 in H. In particular let C denote the set of all nodes connected to v_1 in H so far. Let the demand group be the set of all leaves in $\{v_2, v_3, \ldots, v_n\} \setminus C$. We do this until $\{v_2, v_3, \ldots, v_n\} \setminus C = \emptyset$, which means every leaf is connected to v_1 in H. G is a star, thus a leaf v_i is connected to v_1 in H iff H includes the edge between v_i and v_1. Thus after all demands $\mathcal{A}$ has added all edges in G to H. Hence $deg(v_1) = n-1$.

Now consider the optimal offline algorithm. Let g_i denote the i-th demand group. Assume we have k demand groups. By construction of the demand groups $g_k \subset g_{k-1} \subset \ldots \subset g_1$. Thus there is a single node that exists in all group demands. Hence the optimal offline algorithms only needs to connect that node to the root in H. Therefore, the degree of each node is at most 1 and the competitive ratio is $n-1$. □

Bibliography

[AAA+06] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms*, 2(4):640–660, 2006. 108, 120

[AAA+09] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM J. Comput.*, 39(2):361–370, 2009. 87, 88, 94, 135, 139, 178

[AAB96] Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized steiner problem. In *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 68–74, 1996. 177

[AAB04] Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized steiner problem. *Theor. Comput. Sci.*, 324(2-3):313–324, 2004. 86, 134

[ABFP13] Yossi Azar, Umang Bhaskar, Lisa Fleischer, and Debmalya Panigrahi. Online mixed packing and covering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 85–100. SIAM, 2013. 178

[ABHK11] Aaron Archer, MohammadHossein Bateni, Mohammad Taghi Hajiaghayi, and Howard J. Karloff. Improved approximation algorithms for prize-collecting steiner tree and tsp. In *SIAM J. Comput*, pages 309–332, 2011. 16, 134

[AGSc01] David Assaf, Larry Goldstein, and Ester Samuel-cahn. Ratio prophet inequalities when the mortal has several choices. *Ann. Appl. Prob*, 12:972–984, 2001. 53

[AHL+11] Saeed Alaei, Mohammad T. Hajiaghayi, Vahid Liaghat, Dan Pei, and Barna Saha. Adcell: ad allocation in cellular networks. In *Proceedings of the 19th European conference on Algorithms*, ESA'11, pages 311–322, 2011. 11, 20, 29, 178

[AHL12] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 18–35, 2012. 11, 20, 26, 29, 30, 53, 54, 178, 209

[AHL13] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. The online stochastic generalized assignment problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 11–25. 2013. 27, 178, 209

[AKR91] Ajit Agrawal, Philip Nathan Klein, and R Ravi. *How Tough is the Minimum-degree Steiner Tree?: A New Approximate Min-max Equality*. 1991. 17, 172

[AKR95] Ajit Agrawal, Philip N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995. 86, 95, 135, 138, 183

[Ala11] Saeed Alaei. Bayesian combinatorial auctions: Expanding single buyer mechanisms to many buyers. In *FOCS*, 2011. 24, 29, 30, 34, 37, 53

[AMW01] Miklos Ajtai, Nimrod Megiddo, and Orli Waarts. Improved algorithms and analysis for secretary problems and generalizations. *SIAM J. Discrete Math.*, 14(1):1–27, 2001. 52

[BC97] Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems (extended abstract). In *STOC*, pages 344–353, 1997. 86, 134, 135, 177

[BGK11] Anand Bhalgat, Ashish Goel, and Sanjeev Khanna. Improved approximation results for stochastic knapsack problems. In *SODA*, 2011. 24

[Bha12] Anand Bhalgat. A $(2+\epsilon)$-approximation algorithm for the stochastic knapsack problem. In *Unpublished Manuscript*, 2012. 24

[BHL13] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Improved approximation algorithms for (budgeted) node-weighted steiner problems. In *ICALP*, pages 81–92, 2013. 14, 15, 86, 134

[BHZ13] MohammadHossein Bateni, Mohammad Taghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. *ACM Transactions on Algorithms*, 9(4):32, 2013. 55

[BIK07] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA*, pages 434–443, 2007. 50, 55, 56

[BIKK07] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. In *APPROX*, pages 16–28, 2007. 50

[BIKK08] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exch.*, 7(2):1–11, 2008. 50, 54

[BJN07] Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, pages 253–264, 2007. 26

[BKN09] Nikhil Bansal, Rohit Khandekar, and Viswanath Nagarajan. Additive guarantees for degree-bounded directed network design. *SIAM J. Comput.*, 39(4):1413–1431, 2009. 177

[BN07] Niv Buchbinder and Seffi Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3):93–263, 2007. 5, 6

[BN09a] Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3):93–263, 2009. 96, 135, 136, 145, 178

[BN09b] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2):270–286, 2009. 6, 7, 8, 144

[CCC+99] Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed steiner problems. *J. Algorithms*, 33(1):73–91, 1999. 87

[CEV12a] Chandra Chekuri, Alina Ene, and Ali Vakilian. Node-weighted network design in planar and minor-closed families of graphs. In *ICALP*, pages 206–217, 2012. 89

[CEV12b] Chandra Chekuri, Alina Ene, and Ali Vakilian. Prize-collecting survivable network design in node-weighted graphs. In *APPROX-RANDOM*, pages 98–109, 2012. 134

[CG82] Paolo M Camerini and Giulia Galbiati. The bounded path tree problem. *SIAM Journal on Algebraic Discrete Methods*, 3(4):474–484, 1982. 17

[CGM80] Paolo M Camerini, Giulia Galbiati, and Francesco Maffioli. Complexity of spanning tree problems: Part i. *European Journal of Operational Research*, 5(5):346–352, 1980. 17

[CHMS10] Shuchi Chawla, Jason Hartline, David Malec, and Balasubramanian Sivan. Multi-parameter mechanism design and sequential posted pricing. 2010. 50, 60, 61

[Chv73] Vašek Chvátal. Tough graphs and hamiltonian circuits. *Discrete Mathematics*, 5(3):215–228, 1973. 17, 183

[CK00] Chandra Chekuri and Sanjeev Khanna. A ptas for the multiple knapsack problem. In *SODA*, 2000. 21, 25

[DGV04] Brian C. Dean, Michel X. Goemans, and Jan Vondrak. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *FOCS*, 2004. 23, 24, 30, 209

[DH09] Nikhil R. Devenur and Thomas P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM conference on Electronic commerce*, EC '09, pages 71–78, 2009. 11, 26, 209

[DHK09] Erik D. Demaine, MohammadTaghi Hajiaghayi, and Philip N. Klein. Node-weighted steiner tree and group steiner tree in planar graphs. In *ICALP (1)*, pages 328–340, 2009. 89, 90, 99

[DJSW11] Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *EC*, 2011. 26, 29, 209

[DSA12] Nikhil R. Devanur, Balasubramanian Sivan, and Yossi Azar. Asymptotically optimal algorithm for stochastic adwords. In *EC*, 2012. 26, 209

[Dyn63] E. B. Dynkin. The optimum choice of the instant for stopping a markov process. *Sov. Math. Dokl.*, 4:627–629, 1963. 52, 54

[EJ01] Jack Edmonds and Ellis L. Johnson. Matching: A well-solved class of integer linear programs. In *Combinatorial Optimization*, pages 27–30, 2001. 92

[EV14] Alina Ene and Ali Vakilian. Improved approximation algorithms for degree-bounded network design problems with node connectivity requirements. *STOC*, 2014. 172, 177

[FGMS06] Lisa Fleischer, Michel X. Goemans, Vahab S. Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *SODA*, 2006. 21, 25

[FHK+10] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Cliff Stein. Online stochastic packing applied to display ad allocation. In *ESA*, 2010. 26, 209

[FIMN08] Ureil Feige, Nicole Immorlica, Vahab Mirrokni, and Hamid Nazerzadeh. A combinatorial allocation mechanism with penalties for banner advertising. In *WWW*, 2008. 32

[FKM+09] Jon Feldman, Nitish Korula, Vahab Mirrokni, S. Muthukrishnan, and Martin Pál. Online ad assignment with free disposal. In *WINE*, 2009. 22, 23, 26, 209

[FMMM09] Jon Feldman, Aranyak Mehta, Vahab S. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating $1 - 1/e$. In *FOCS*, 2009. 10, 25, 178

[FNW78] M. Fisher, G. Nemhauser, and L. Wolsey. An analysis of the approximations for maximizing submodular set functions ii. In *Math. Prog. St.*, 1978. 25

[FR90] Martin Fürer and Balaji Raghavachari. An NC approximation algorithm for the minimum degree spanning tree problem. In *Allerton Conf. on Communication, Control and Computing*, pages 274–281, 1990. 17, 172

[FR94] Martin Fürer and Balaji Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *Journal of Algorithms*, 17(3):409–423, 1994. 18, 172

[Fre83] P. R. Freeman. The secretary problem and its extensions: a review. *Internat. Statist. Rev.*, 51(2):189–206, 1983. 55

[FV06] Uriel Feige and Jan Vondrak. Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e. In *FOCS*, 2006. 21, 25

[GHB83] Kenneth S. Glasser, Richard Holzsager, and Austin Barron. The d choice secretary problem. *Comm. Statist. C—Sequential Anal.*, 2(3):177–199, 1983. 52

[GI99] Ashish Goel and Piotr Indyk. Stochastic load balancing and related problems. In *FOCS*, 1999. 24

[GK99] Sudipto Guha and Samir Khuller. Improved methods for approximating node weighted steiner trees and connected dominating sets. *Inf. Comput.*, 150(1):57–74, 1999. 89

[GM66] John P. Gilbert and Frederick Mosteller. Recognizing the maximum of a sequence. *J. Amer. Statist. Assoc.*, 61:35–73, 1966. 55

[GM08] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '08, pages 982–991, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics. 10

[GMK88] Hector Garcia-Molina and Boris Kogan. An implementation of reliable broadcast using an unreliable multicast facility. In *Reliable Distributed Systems, 1988. Proceedings., Seventh Symposium on*, pages 101–111, 1988. 176

[GMNS99] Sudipto Guha, Anna Moss, Joseph Naor, and Baruch Schieber. Efficient recovery from power outage (extended abstract). In *STOC*, pages 574–582, 1999. 89, 134

[GN12] Anupam Gupta and Viswanath Nagarajan. Approximating sparse covering integer programs online. In *ICALP*, pages 436–448, 2012. 144

[Goe06] Michel X Goemans. Minimum bounded degree spanning trees. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 273–282, 2006. 172, 176

[GW95] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995. 16, 86, 88, 89, 90, 95, 134, 135, 138

[HGM03] Yongqiang Huang and Hector Garcia-Molina. Publish/subscribe tree construction in wireless ad-hoc networks. In *Mobile Data Management*, pages 122–140, 2003. 176

[HJ06] Mohammad Taghi Hajiaghayi and Kamal Jain. The prize-collecting generalized steiner tree problem via a new approach of primal-dual schema. In *SODA*, pages 631–640, 2006. 134

[HKKN12] Mohammad Taghi Hajiaghayi, Rohit Khandekar, Guy Kortsarz, and Zeev Nutov. Prize-collecting steiner network problems. *ACM Transactions on Algorithms*, 9(1):2, 2012. 134

[HKP04] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C. Parkes. Adaptive limited-supply online auctions. *EC*, 2004. 12, 50

[HKS07] Mohammad Taghi Hajiaghayi, Robert D. Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *AAAI*, 2007. 12, 24, 50, 51

[HLP13] MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Online node-weighted steiner forest and extensions via disk paintings. In *FOCS*, pages 558–567, 2013. 135, 138, 139, 154, 177

[HLP14] MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Near-optimal online algorithms for prize-collecting steiner problems. In *Automata, Languages, and Programming*, pages 576–587. 2014. 178

[HMZ11] Bernhard Haeupler, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching. In *WINE*, 2011. 26, 209

[Hoc82] Dorit S Hochbaum. Heuristics for the fixed cost median problem. *Mathematical programming*, 22(1):148–162, 1982. 108

[IKM06] Nicole Immorlica, Robert D. Kleinberg, and Mohammad Mahdian. Secretary problems with competing employers. In *WINE*, pages 389–400, 2006. 50

[IW91] Makoto Imase and Bernard M. Waxman. Dynamic steiner tree problem. *SIAM J. Discrete Math.*, 4(3):369–384, 1991. 86, 134, 177

[JMM+03] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM*, 2003. 29

[JMP00] David S. Johnson, Maria Minkoff, and Steven Phillips. The prize collecting steiner tree problem: theory and practice. In *SODA*, pages 760–769, 2000. 16

[Ken78] D. P. Kennedy. Prophet-type inequalities for multi-choice optimal stopping. In *In Stoch. Proc. Applic.* 1978. 51, 53

[KKN13] Rohit Khandekar, Guy Kortsarz, and Zeev Nutov. On some network design problems with degree constraints. *Journal of Computer and System Sciences*, 79(5):725–736, 2013. 172, 177

[Kle05] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA*, pages 630–631, 2005. 50, 55

[KMT11] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *STOC*, 2011. 25, 178

[Kor05] Simon Korman. On the use of randomization in the online set cover problem. *M.S. thesis, Weizmann Institute of Science*, 2005. 94, 139

[Kos84] Alexandr V. Kostochka. Lower bound of the hadwiger number of graphs by their average degree. *Combinatorica*, 4(4):307–316, 1984. 124

[KR95] Philip N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *J. Algorithms*, 19(1):104–115, 1995. 89

[KR00] Jochen Könemann and R Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 537–546, 2000. 176

[KR05] Jochen Könemann and R Ravi. Primal-dual meets local search: approximating msts with nonuniform degree bounds. *SIAM Journal on Computing*, 34(3):763–773, 2005. 176

[KRT97] Jon Kleinberg, Yuval Rabani, and Éva Tardos. Allocating bandwidth for bursty connections. In *STOC*, 1997. 24

[KS77] U. Krengel and L Sucheston. Semiamarts and finite values. In *Bull. Am. Math. Soc.* 1977. 24, 51, 53, 54

[KS78] U. Krengel and L Sucheston. On semiamarts, amarts, and processes with finite value. In *In Kuelbs, J., ed., Probability on Banach Spaces*. 1978. 24, 51, 53

[KSS13] Jochen Könemann, Sina Sadeghian Sadeghabad, and Laura Sanità. An LMP $O(\log n)$-approximation algorithm for node weighted prize collecting steiner tree. In *FOCS*, pages 568–577, 2013. 134, 140

[KVV90] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for online bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, STOC '90, pages 352–358, 1990. 10, 25, 178, 209

[KW12] Robert Kleinberg and S. Matthew Weinberg. Matroid prophet inequalities. In *STOC*, pages 123–136, 2012. 53, 61

[Lac14] O. Lachish. $o(\log \log \text{rank})$-competitive-ratio for the matroid secretary problem. In *FOCS*. 2014. 56

[Lah06] Sébastien Lahaie. An analysis of alternative slot auction designs for sponsored search. In *Proceedings of the 7th ACM conference on Electronic commerce*, EC '06, pages 218–227, New York, NY, USA, 2006. ACM. 10

[LLZ+13] Li Erran Li, Vahid Liaghat, Hongze Zhao, MohammadTaghi Hajiaghayi, Dan Li, Gordon T. Wilfong, Yang Richard Yang, and Chuanxiong Guo. PACE: policy-aware application cloud embedding. In *INFOCOM*, pages 638–646, 2013. 6

[LMSL92] Chung-Lun Li, S. Thomas McCormick, and David Simchi-Levi. The point-to-point delivery and connection problems: complexity and algorithms. *Discrete Applied Mathematics*, 36(3):267–292, 1992. 92

[LP86] L. Lovasz and M.D. Plummer. *Matching Theory*. North-Holland, Amsterdam-New York, 1986. 10

[LRS11] Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011. 177

[LS13] Lap Chi Lau and Mohit Singh. Additive approximation for bounded degree survivable network design. *SIAM Journal on Computing*, 42(6):2217–2242, 2013. 172, 177

[Mad67] Wolfgang Mader. Homomorphieeigenschaften und mittlere kantendichte von graphen. *Mathematische Annalen*, 174(4):265–268, 1967. 124

[MD79] R Garey Michael and S Johnson David. Computers and intractability: a guide to the theory of np-completeness. *WH Freeman & Co., San Francisco*, 1979. 17

[Meh12] Aranyak Mehta. Online matching and ad allocation. *Theoretical Computer Science*, 8(4):265–368, 2012. 178

[MOGS12] Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573, 2012. 178

[MOGZ12] Vahab S Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1690–1701, 2012. 178

[Mol13] Carsten Moldenhauer. Primal-dual approximation algorithms for node-weighted steiner forest on planar graphs. *Inf. Comput.*, 222:293–306, 2013. 89

[MOS11] Vahideh H. Manshadi, Shayan OveisGharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. In *SODA*, 2011. 10, 25

[MR07] Anna Moss and Yuval Rabani. Approximation algorithms for constrained node weighted steiner tree problems. *SIAM J. Comput.*, 37(2):460–481, 2007. 89, 134

[MRS+98] Madhav V Marathe, R Ravi, Ravi Sundaram, SS Ravi, Daniel J Rosenkrantz, and Harry B Hunt III. Bicriteria network design problems. *Journal of Algorithms*, 28(1):142–171, 1998. 172, 176

[MSVV07] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *J. ACM*, 2007. 11, 23, 26, 178, 209

[MY11] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: An approach based on strongly factor-revealing LPs. In *STOC*, 2011. 25, 178, 209

[Mye79] Roger B. Myerson. Incentive compatibility and the bargaining problem. In *Econometrica*, 1979. 61

[MZO12] Vahab S. Mirrokni, Morteza Zadimoghaddam, and Shayan OveisGharan. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *SODA*, 2012. 25, 209

[NPS11] Joseph Naor, Debmalya Panigrahi, and Mohit Singh. Online node-weighted steiner tree and related problems. In *FOCS*, pages 210–219, 2011. 87, 88, 94, 95, 135, 137, 149, 151, 153, 162, 177

[Nut10] Zeev Nutov. Approximating steiner networks with node-weights. *SIAM J. Comput.*, 39(7):3001–3022, 2010. 89

[Nut12] Zeev Nutov. Degree-constrained node-connectivity. In *LATIN 2012: Theoretical Informatics*, pages 582–593. 2012. 172, 177

[PY82] Christos H Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM*, 29(2):285–309, 1982. 17

[QW11] Jiawei Qian and David P. Williamson. An $o(\log n)$-competitive algorithm for online constrained forest problems. In *ICALP (1)*, pages 37–48, 2011. 16, 91, 96, 135, 137, 178, 210

[RT85] Prabhakar Raghavan and Clark D Thompson. Provably good routing in graphs: regular arrays. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 79–87, 1985. 18

[SL07] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 661–670, 2007. 172, 176

[SSW07] Yogeshwer Sharma, Chaitanya Swamy, and David P. Williamson. Approximation algorithms for prize collecting forest problems with submodular penalty functions. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1275–1284, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. 134

[ST93] David B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Math. Program.*, 1993. 25

[Van80] R. J. Vanderbei. The optimal choice of a subset of a population. *Math. Oper. Res.*, 5(4):481–486, 1980. 52

[Von08] Jan Vondrak. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, 2008. 25

[Wik12] Wikipedia. http://en.wikipedia.org/wiki/mana, 2012. 35

[Wil91] John G. Wilson. Optimal choice and assignment of the best m of n randomly arriving items. *Stochastic Process. Appl.*, 39(2):325–343, 1991. 52

	Worst Case	Random Arrival	Stochastic
Matching	$1 - 1/e$ [KVV90]	0.696 [MY11]	0.703 [MZO12]
Display Ad	FD: $1 - 1/e$ [FKM+09]	FD: $1 - 1/e$	UnD-LCD: $1 - O(\epsilon)$ [DJSW11] 0.66 [HMZ11] PI-LC: $1 - \epsilon$ [AHL12]
Adword	LC: $1 - 1/e$ [MSVV07]	LCD: $1 - O(\epsilon)$ [DH09]	PI-LC: $1 - \epsilon$ [AHL12] UnD-LC: $1 - \epsilon$ [DSA12]
Stoch. Knapsack			$\frac{1}{9.5}$ [DGV04]
GAP	FD: $1 - 1/e - \epsilon$ [FKM+09]	LCD: $1 - o(1)$ [FHK+10]	LC: $1 - \epsilon$ [AHL13]

Table 2.1: Summary of results for GAP and its special instances

Here "FD" stands for *Free Disposal*, "UnD" stands for *Unknown Distribution*, "PI" stands for *Prophet Inequality setting*, "LC" stands for *the Large-Capacity assumption where the ratio depends solely on* ϵ, and "LCD" stands for *the Large-Capacity assumption where the ratio also Depends on the number of bins and items.*

	PC ST	PC SF	PC SF in Planar graphs	PC GSF
EW	$O(\log(n))$ [QW11] Simplified here	$O(\log(n))$ [QW11]	$O(\log(n))$ [QW11]	$O(\log^7(n))$
NW	$O(\log^3(n))$	$O(\log^4(n))$	$O(\log^2(n))$	$O(\log^{11}(n))$ (quasi-polynomial)

Table 3.1: The competitive ratio for online prize-collecting (PC) Steiner problems

Note that the algorithm for NW Group Steiner Forest runs in quasi-polynomial time.