

## ABSTRACT

Title of dissertation: ONLINE DECISION MAKING  
VIA PROPHET SETTING

Soheil Ehsani  
Doctor of Philosophy, 2017

Dissertation directed by: Professor Mohammad Hajiaghayi  
Department of Computer Science

In the study of online problems, it is often assumed that there exists an adversary who acts against the algorithm and generates the most challenging input for it. This worst-case assumption in addition to the complete uncertainty about future events in the traditional online setting sometimes leads to worst-case scenarios with super-constant approximation impossibilities. In this dissertation, we go beyond this worst-case analysis of problems by taking advantage of stochastic modeling. Inspired by the prophet inequality problem, we introduce the prophet setting for online problems in which the probability distributions of the future inputs are available. This modeling not only considers the availability of statistical data in the design of mechanisms but also results in significantly more efficient algorithms.

To illustrate the improvements achieved by this setting, we study online problems within the contexts of auctions and networks. We begin our study with analyzing a fundamental online problem in optimal stopping theory, namely prophet inequality, in the special cases of iid and large markets, and general cases of ma-

troids and combinatorial auctions and discuss its applications in mechanism design. The stochastic model introduced by this problem has received a lot of attention recently in modeling other real-life scenarios, such as online advertisement, because of the growing ability to fit distributions for user demands. We apply this model to network design problems with a wide range of applications from social networks to power grids and communication networks. In this dissertation, we give efficient algorithms for fundamental network design problems in the prophet setting and present a general framework that demonstrates how to develop algorithms for other problems in this setting.

ONLINE DECISION MAKING  
VIA  
PROPHET SETTING

by

Soheil Ehsani

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  
2017

Advisory Committee:  
Professor Mohammad Hajiaghayi, Chair/Advisor  
Professor Peter Cramton  
Professor William Gasarch  
Professor David Mount  
Professor Aravind Srinivasan

© Copyright by  
Soheil Ehsani  
2017

*Dedicated to my family.*

## Acknowledgment

First and foremost, I would like to thank my advisor Prof. Mohammad Hajj-aghayy for his extraordinary support. He has been more than an advisor to me and has given me suggestions and ideas that I would not receive from anybody other than a close friend. I remember a lot of challenging situations in which Mohammad took my hand and supported me to get through them. I feel very fortunate to have had him as my Ph.D. advisor. Thank you Mohammad for everything and I look forward to our lifelong collaboration.

Many thanks to Prof. Peter Cramton, Prof. William Gasarch, Prof. David Mount, and Prof. Aravind Srinivasan for serving on my dissertation committee.

The graduate school provided me an opportunity to meet and collaborate with the smartest and humblest people. I would like to thank all my friends who I got to know during this time and learn a lot from them. I would like to thank my co-authors Prof. Harald Racke, Prof. Robert Kleinberg, Dr. Brendan Lucier, Prof. Thomas Kesselheim, Prof. Mohamad Ghodsi, Vahid Liaghat, Sina Dehghani, Saeed Seddighin, Hossein Esfandiari, Melika Abolhassani, Karthik Abinav Sankararaman, Brian Brubach, Sahil Singla, and Mehdi Borujeni for sharing their knowledge and experiences with me. Thank you all for helping me create this important chapter of my life and fill it with a lot of great memories.

I also gratefully acknowledge the support of the following grants during my Ph.D. studies: NSF CAREER award CCF-1053605, NSF BIGDATA grant IIS-1546108, NSF AF:Medium grant CCF-1161365, DARPA GRAPHS/AFOSR grant FA9550-12-1-0423, DARPA SIMPLEX grant, and World Quantitative and Science fellowship.

Leaving the most important for the last, I owe my deepest thanks to my family - my mother and father Yalda and Nemat, and my siblings Abouzar, Hosein, and Maryam for their infinite love and support! I don't know where I would be without you, and words cannot express the gratitude I owe you. I hope the dedication of this dissertation to you returns a little bit of your immense kindness.

# Table of Contents

Dedication	ii
Acknowledgement	iii
Table of Contents	v
List of Figures	viii
3 Overview	1
3.1 Introduction	1
3.1.1 Online vs Offline	2
3.1.2 Prophet Setting vs Online Setting	4
3.2 Outline	5
3.2.1 Online Network Design	6
3.2.2 Prophet Inequalities	9
3.2.3 Online Problems in Prophet Setting	11
4 Online Degree-Bounded Steiner Network Design	14
4.1 Introduction	14
4.1.1 Our Contributions	16
4.1.2 Related Degree-Bounded Connectivity Problems	21
4.1.3 Related Online Problems	22
4.1.4 Preliminaries	23
4.2 Online Degree-Bounded Steiner Forest	25
4.2.1 Analysis	26
4.3 An Asymptotically Tight Lower Bound	35
5 Online Weighted Degree-Bounded Steiner Networks	39
5.1 Introduction	39
5.1.1 Our Results and Techniques	42
5.1.1.1 Massaging the optimal solution	43
5.1.1.2 Bounded-frequency mixed packing/covering IPs	45
5.1.2 Preliminaries	48



5.1.3	Overview of the Chapter . . . . .	49
5.2	Finding the Right Integer Program . . . . .	50
5.3	Online Bounded Frequency Mixed Packing/Covering IPs . . . . .	55
5.4	Putting Everything Together . . . . .	60
6	Beating $1-1/e$ for Ordered Prophets . . . . .	62
6.1	Introduction . . . . .	62
6.1.1	Our Contribution . . . . .	64
6.1.2	Applications in Mechanism Design . . . . .	66
6.1.3	Other Related Work . . . . .	68
6.2	IID Distributions . . . . .	70
6.3	Non IID Distributions . . . . .	79
7	Prophet Secretary for Matroids and Combinatorial Auctions . . . . .	85
7.1	Introduction . . . . .	85
7.1.1	Our Techniques . . . . .	89
7.1.2	Related Work . . . . .	91
7.2	Our Approach using a Residual . . . . .	92
7.3	Prophet Secretary for Combinatorial Auctions . . . . .	96
7.3.1	Bipartite Matching . . . . .	97
7.3.2	XOS Combinatorial Auctions . . . . .	100
7.4	Prophet Secretary for Matroids . . . . .	103
7.5	Fixed Threshold Algorithms . . . . .	108
7.5.1	Single Item Prophet Secretary . . . . .	109
7.5.2	Impossibility for IID Prophet Inequalities . . . . .	112
8	Stochastic $k$ -Server Problem . . . . .	115
8.1	Introduction . . . . .	115
8.1.1	The Stochastic Model . . . . .	118
8.1.2	Our Results . . . . .	120
8.1.3	Further Related Work . . . . .	123
8.2	Preliminaries . . . . .	125
8.3	Structural Characterization . . . . .	128
8.4	Fractional Solutions . . . . .	132
8.4.1	Linear Program . . . . .	132
8.5	Reduction from Integral $k$ -server to Fractional $k$ -server . . . . .	135
8.5.1	Integrals Are as Strong as Fractionals On the Line . . . . .	135
8.5.2	Reduction for General Graphs . . . . .	137
9	Survivable Network Design and Prophets . . . . .	140
9.1	Introduction . . . . .	140
9.1.1	Our Results and Techniques . . . . .	145
9.1.2	Further Related Work . . . . .	152
9.2	Steiner Tree Packing . . . . .	154
9.2.1	Fractional Steiner Tree Packing . . . . .	154

9.2.2	Fractional Steiner Tree Packing of $k$ -connected Graphs . . . .	155
9.2.3	Steiner Forest Packing . . . . .	159
9.3	Uniform SNDP . . . . .	160
9.4	Non-Uniform SNDP . . . . .	171
9.5	From Oblivious I.I.D. to Prophet and Applications to Online Problems	173
9.6	Stochastic Survivable Network Design . . . . .	178
9.6.1	Algorithm . . . . .	180
9.6.2	Structural Lemma for $k$ -connected Graphs . . . . .	183
Appendices	. . . . .	187
A	Online Degree-Bounded Edge-weighted Steiner Tree . . . . .	187
B	Online Degree-Bounded Group Steiner Tree . . . . .	190
C	Omitted Proofs . . . . .	191
D	Missing Calculations in Example 7.1 . . . . .	201
E	Extension of FTA's to Bipartite Matchings . . . . .	201
F	Correlated Setting . . . . .	204
G	Program . . . . .	205
H	Experimental Results . . . . .	208
Bibliography		211

## List of Figures

4.1	Existence of a cycle implies that of a path with low uptick load in its extension part. . . . .	36
5.1	An example where every vertex has degree-bound 3 and every edge has weight 1. The first demand is $(v_2, v_5)$ and the second demand is $(v_3, v_6)$ . The optimal solution for $\text{SF\_IP}$ is a subgraph, say $H$ , with the set of all edges and vertices, i.e. $H = G$ . However an optimal solution for $\text{PC\_IP}$ is: Two subgraphs $H_1$ for the first request which has edges $\{e(v_1, v_2), e(v_1, v_4), e(v_4, v_5)\}$ and $H_2$ for the second request which has edges $\{e(v_2, v_3), e(v_4, v_5), e(v_4, v_6)\}$ . Note that $w(H) = 5$ and $w(H_1) + w(H_2) = 6$ , since we have edge $e(v_4, v_5)$ in both $H_1$ and $H_2$ . Moreover the number of edges incident with $v_4$ in the solution of $\text{PC\_IP}$ is 4, i.e. $\deg_{H_1}(v_4) + \deg_{H_2}(v_4) = 4$ . . . . .	52
9.1	An example graph illustrating that the greedy algorithm has $\Omega(n)$ -competitive ratio. . . . .	170
2	The graph $G$ consists of $2k + 1$ nodes and $3k$ edges. . . . .	187
3	The highlighted subtree represents an optimum solution $OPT_3$ . . . . .	188
4	The plot shows function $A(w)$ for values of $w$ from 0 to $\tan(a) \approx 3.7$ . . . .	196
5	Performance of our algorithm compared to the optimum. The dashed curve indicates two times the optimum. . . . .	210

## Chapter 3: Overview

### 3.1 Introduction

In the theoretical study of algorithms, it is often assumed that there exists an adversary that acts against the algorithm and generates the most challenging input for it. This is the basis of the worst-case analysis of algorithms. Despite its importance in the consideration of all aspects of algorithms for risk-averse applications, the worst-case analysis usually tightens our hands in designing more efficient algorithms for less adversarial but more likely instances. An evident example in which this issue occurs is the traditional online setting in which there are super-constant approximation impossibilities for many problems. This dissertation aims to go beyond the worst-case analysis of online algorithms by including stochastic data in the traditional online setting. Most importantly, the online stochastic setting that we introduce, namely the *prophet setting*, takes advantage of the developments of statistical and machine learning tools in demand prediction in order to model real-life problems. To illustrate the improvement achieved by this setting, we study online problems in the two contexts of network mechanism design and auction mechanism design.

### 3.1.1 Online vs Offline

Offline problems are perhaps the most basic form of problems in theoretical computer science. In the offline variant of a problem, we assume all the input is given at once, and the goal is to output a solution using a polynomial amount of memory and time. Examples of this class include minimum Steiner tree (graph theory), clustering a set of points (combinatorial optimization), finding an optimal assignment of goods to buyers (auction theory), and a plenty of other problems which follow the three-stage process of an offline problem, i.e. getting input, processing data, and outputting a solution.

Maybe one of the biggest issues with the offline variant of problems is that they lack the ability to model the real-life instances in which the input is not given (or is not available) all at once. To clarify this issue, let us discuss it in above examples. An application of the minimum Steiner tree problem is when a network provider wants to find cheap routings for connecting a set of devices in a network. The above issue arises when the set of devices is not known at the beginning, but the provider receives the connectivity requests from devices at different times and wants to connect each device to previous ones upon receiving its request. Similarly, in a clustering problem the points may arrive at different times, or in an auction the set of buyers and goods may be subject to change as people/goods appear at different times while our objective functions remain the same. Although one may suggest using offline algorithms to address the above issue, we argue that such algorithms may potentially result in entirely different solutions every time a part of the input

arrives. Of course, this would not be much appealing in many applications especially those with high cost of revoking decisions.

The goal of online variants of problems is to properly model the above scenario in which the input grows over time. In particular, in the online instance of a problem, we assume that an initial state/configuration of the problem is given, and then at each discrete time  $t$  a part of the input is revealed and the algorithm has to update its solution so that the recent part of the input is considered. As a clarifying example, in the above Steiner tree instance the network provider should route a new device through some connections while the connections between previous devices do not change. Similarly, in a clustering problem we should assign each new point to a cluster without changing the clusters of previous points, and in an online scenario of an auction we should serve new buyers without changing the way we served previous people.

An online problem can be formulated as follows. Let  $\mathcal{S}$  be the set of all initial states,  $\mathcal{I}$  be the collection of all inputs, and  $\mathcal{O}$  be the collection of all outputs for the problem. An online scenario goes in this way: first the algorithm receives an initial state  $S_0 \in \mathcal{S}$ , and then at every discrete time  $t$  it receives  $I_t \in \mathcal{I}$ , where  $I_{t-1} \subset I_t$ , and has to find a solution  $O_t$  to it where  $O_{t-1} \subseteq O_t \in \mathcal{O}$ . The goal of an online algorithm is to find such online solution that optimizes the objective function for every  $t$ . In contrast, in an offline problem the algorithm receives  $S_0 \in \mathcal{S}, I \in \mathcal{I}$  at once and is supposed to output a single  $O \in \mathcal{O}$  for the given input.

In this dissertation, we study the online variants of some of the fundamental problems in the contexts of network design and auction mechanism design. We begin

with introducing the online variant of degree-bounded Steiner forest problem and propose polylogarithmic competitive algorithms for them. We also outline some negative aspects of the online modeling of this problem by presenting hardness instances in which the online sequence of inputs makes the decision making difficult even for randomized algorithms.

### 3.1.2 Prophet Setting vs Online Setting

In general, the online setting assumes a high degree uncertainty about future demands. In particular, the algorithm has no information about  $I_t \setminus I_{t-1}$  at any time before  $t$ . This uncertainty allows the adversary to generate instances of the online input for which no algorithm can perform good enough. Our approach to deal with this hardness issue is to consider the online stochastic variant borrowed from the prophet inequality problem. In chapters 6 and 7 we discuss more about this problem, its special and general cases, and its application in mechanism design. Here, we emphasize that the main difference in the *prophet setting* and the traditional online setting is that the algorithm has partial information about the input received at each time  $t$ . More specifically, before the online scenario begins, in addition to the initial state  $S_0 \in \mathcal{S}$  the algorithm also gets probability distributions  $D_t$  for every  $\Delta I_t = I_t \setminus I_{t-1}$  which is to the part of input received at time  $t$ . Then in the online scenario and at each time  $t$ , the new input is chosen from a random process, i.e.  $\Delta I_t \sim D_t$ , and the algorithm has to update its solution  $O_t$  while optimizing the objective function.

There are two reasons why we study online problems in the prophet setting. First, in a lot of real-life applications of these problems, the input is not generated by an adversary but it is produced by natural parameters of the system. Therefore the adversarial hardness instances that limit the theoretical study of the online problems become unimportant. Second, in some applications there is a large amount of historical data available to the algorithm designer, which can be exploited using statistical tools to fit distributions for future inputs. Hence, the consideration of having partial information for future input in prophet setting properly captures this aspect and allows a more efficient design of algorithms.

## 3.2 Outline

In chapters 4 and 5 we consider network design problems in the traditional online setting. In chapters 6 and 7 we study the special cases and general cases of the prophet inequality problem, which is an important fundamental online decision making problem, and discuss their applications in auction mechanism design. In chapters 8 and 9 we borrow the setting of prophet inequality and apply it on two important network design problem, namely the  $k$ -server problems and the survivable network design problem, and show how this allows significant improvements upon previous results.



### 3.2.1 Online Network Design

One of the fundamental problems in the area of network design is the Steiner tree problem. In this problem we are given a graph  $G = (V, E)$  and a weight function  $w : E \rightarrow \mathcal{R}^+$  and a set of nodes  $S \subseteq V$ . The goal is to find a set of edges  $H \subseteq E$  with minimum  $w(H) = \sum_{e \in H} w(e)$  such that all vertices of  $S$  are connected in  $H$ . This problem was first introduced in the 70's in Garey and Johnson's *Black Book* of NP-Completeness [MD79]. As a special case when  $S = V$  the problem boils down to the minimum spanning tree problem which has extensively been studied and has near linear-time optimum algorithms. However, finding a minimum weight Steiner tree could be hard for some input instances. Hence, there is a considerable line of work for Steiner problems in approximation algorithms.

The goal of an approximation algorithm is to give a guarantee about the cost of its solution in comparison with the cost of the optimum solution. In other words, if  $ALG$  denotes the cost the solution found by an  $\alpha$ -approximation algorithm and  $OPT$  denotes the cost of the optimum solution, then the ratio  $ALG/OPT$  should never exceed  $\alpha$  for any input instance. We note that there are a few other slightly different definitions for the notion of approximation factor in the literature which we discuss in the next chapters.

In this dissertation, we study some of the classical variants of Steiner network design in the online setting and give approximation algorithms for them. As opposed to the classical (offline) Steiner tree problem, in the online setting, we do not have all the connectivity demands at the beginning, but we receive them over time and

have to immediately provide a connection for each demand upon its arrival. Recall that the problems in the online setting are usually harder to deal with because of the uncertainties of future demands.

In Chapter 4 we initiate the study of degree-bounded network design problems in the online setting. The degree-bounded Steiner tree problem – which asks for a subgraph with minimum degree that connects a given set of vertices – is perhaps one of the most representative problems in this class. We deal with its well-studied generalization called the degree-bounded Steiner forest problem where the connectivity demands are represented by vertex pairs that need to be individually connected.

In the classical online model, the input graph is given offline but the demand 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. The goal is to be competitive against an adversary that knows the input in advance.

The standard techniques for solving degree-bounded problems often fall in the category of *iterative* and *dependent* rounding techniques. Unfortunately, these rounding methods are inherently difficult to adapt to an online settings since the underlying fractional solution may change dramatically in between the rounding steps. Indeed, this might be the very reason that despite many advances in the online network design paradigm in the past two decades, the natural family of degree-bounded problems has remained widely open.

In our work we design an intuitive greedy-like algorithm that achieves a competitive ratio of  $O(\log n)$  where  $n$  is the number of vertices. We show that no (ran-

domized) algorithm can achieve a (multiplicative) competitive ratio  $o(\log n)$ ; thus our result is asymptotically tight. We further show strong hardness results for the group Steiner tree and the edge-weighted variants of degree-bounded connectivity problems.

We study the weighted generalization of online degree bounded Steiner forest in Chapter 5. In edge-weighted degree-bounded Steiner forest (EW-DB-SF) we are given an edge-weighted graph with a degree bound for every vertex. Given a root vertex in advance, we receive a sequence of terminal vertices in an online manner. Upon the arrival of a terminal, we need to augment our solution subgraph to connect the new terminal to the root. The goal is to minimize the total weight of the solution while respecting the degree bounds on the vertices.

In the offline setting, EW-DB-ST and its many variations have been extensively studied since early eighties. Unfortunately, the recent advancements in the online network design problems are inherently difficult to adapt for degree-bounded problems. In particular, it is not known whether the fractional solution obtained by standard primal-dual techniques for mixed packing/covering LPs can be rounded online.

In contrast, we obtain our result by using structural properties of the optimal solution, and reducing the EW-DB-SF problem to an exponential-size mixed packing/covering integer program in which every variable appears only once in covering constraints. We then design a generic *integral* algorithm for solving this restricted family of IPs.

As mentioned above, we demonstrate a new technique for solving mixed pack-

ing/covering integer programs. Define the *covering frequency*  $k$  of a program as the maximum number of covering constraints in which a variable can participate. Let  $m$  denote the number of packing constraints. We design an online *deterministic integral algorithm* with competitive ratio of  $O(k \log m)$  for the mixed packing/covering integer programs.

We prove the tightness of our result by providing a matching lower bound for any randomized algorithm. We note that our solution solely depends on  $m$  and  $k$ . Indeed, there can be exponentially many variables. Furthermore, our algorithm directly provides an integral solution, even if the integrality gap of the program is unbounded. We believe this technique can be used as an interesting alternative for the standard primal-dual techniques in solving online problems.

### 3.2.2 Prophet Inequalities

A simple but fundamental problem that models an online stochastic scenario is *prophet inequality*. In an instance of this problem, the input contains  $n$  probability distributions  $D_1, \dots, D_n$  with finite supports on  $R^+$ . Then the problem scenario continues in an online fashion such that at time  $i$ , random variable  $X_i$  is drawn from  $D_i$ . An algorithm can stop the scenario at anytime and claim a reward equal to the most recent observation. The goal is to find a policy  $\tau$  that maximizes

$$\frac{\mathbb{E}[ALG]}{\mathbb{E}[OPT]} = \frac{\mathbb{E}[X_\tau]}{\mathbb{E}[\max_{1 \leq i \leq n} X_i]} .$$

In Chapter 6 we study the prophet inequality problem with iid distributions. In this problem all values  $X_1, \dots, X_n$  come from a common distribution. Hill and

Kertz studied the prophet inequality on iid distributions [*The Annals of Probability* 1982]. They proved a theoretical bound of  $1 - \frac{1}{e}$  on the approximation factor of their algorithm. They conjectured that the best approximation factor for arbitrarily large  $n$  is  $\frac{1}{1+1/e} \simeq 0.731$ . This conjecture remained open for more than thirty years.

In this dissertation, we present a threshold-based algorithm for the prophet inequality with  $n$  iid distributions. Using a nontrivial and novel approach we show that our algorithm is a 0.738-approximation algorithm. By beating the bound of  $\frac{1}{1+1/e}$ , this refutes the conjecture of Hill and Kertz. Moreover, we generalize our results to non-iid distributions and discuss its applications in mechanism design. We note that prophet inequality and its generalizations such as matroids and matchings have received a considerable attention in the past decade due to their applications in mechanism design, auction design, ad allocation, etc. Therefore, we believe our work is a fuller treatment of the a major fundamental case of this important problem.

In Chapter 7 we study generalizations of the natural combination of prophet inequalities and secretary problems, which are central to the field of Stopping Theory. Recently, there has been a lot of work in generalizing these models to multiple items because of their applications in mechanism design. The most important of these generalizations are to matroids and to combinatorial auctions (extends bipartite matching). Kleinberg-Weinberg [KW12] and Feldman et al. [FGL15] show that for adversarial arrival order of random variables the optimal prophet inequalities give a 1/2-approximation. For many settings, however, it's conceivable that the arrival order is chosen uniformly at random, akin to the secretary problem. For such a random arrival model, we improve upon the 1/2-approximation and obtain

$(1 - 1/e)$ -approximation prophet inequalities for both matroids and combinatorial auctions. This also gives improvements to the results of Yan [Yan11] and Esfandiari et al. [EhLM15] who worked in the special cases where we can fully control the arrival order or when there is only a single item.

The techniques we use are threshold based. We convert our discrete problem into a continuous setting and then give a generic template on how to dynamically adjust these thresholds to lower bound the expected total welfare.

### 3.2.3 Online Problems in Prophet Setting

In Chapter 8 we study a stochastic variant of the celebrated  $k$ -server problem. In the  $k$ -server problem, we are required to minimize the total movement of  $k$  servers that are serving an online sequence of  $t$  requests in a metric. In the stochastic setting we are given  $t$  independent distributions  $\langle P_1, P_2, \dots, P_t \rangle$  in advance, and at every time step  $i$  a request is drawn from  $P_i$ .

Designing the optimal online algorithm in such setting is NP-hard, therefore the emphasis of our work is on designing an approximately optimal online algorithm. We first show a structural characterization for a certain class of *non-adaptive* online algorithms. We prove that in general metrics, the best of such algorithms has a cost of no worse than three times that of the optimal online algorithm. Next, we present an integer program that finds the optimal algorithm of this class for any arbitrary metric. Finally by rounding the solution of the linear relaxation of this program, we present an online algorithm for the stochastic  $k$ -server problem with

an approximation factor of 3 in the line and circle metrics and factor of  $O(\log n)$  in a general metric of size  $n$ . In this way, we achieve an approximation factor that is independent of  $k$ , the number of servers. Furthermore, we extend our results to the correlated setting where the probability of a request arriving at a certain point depends not only on the time step but also on the previously arrived requests.

In Chapter 9 we study the online and stochastic versions of *survivable network design* problem. In an instance of this problem we are given non-negative integers  $r_{uv}$  for each pair  $u, v \in V$ , the solution subgraph  $H$  should contain  $r_{uv}$  edge-disjoint paths for each pair  $u$  and  $v$ .

While this problem is known to admit good approximation algorithms in the offline case, the problem is much harder in the online setting. Gupta, Krishnaswamy, and Ravi [GKR12] (STOC'09) are the first to consider the online survivable network design problem. They demonstrate an algorithm with competitive ratio of  $O(k \log^3 n)$ , where  $k = \max_{u,v} r_{uv}$ . Note that the competitive ratio of the algorithm by Gupta *et al.* grows linearly in  $k$ . Hence, an important open problem in the online community [NPS11, GKR12] is whether the linear dependency on  $k$  can be reduced to a logarithmic dependency.

Consider an online greedy algorithm that connects every demand by adding a minimum cost set of edges to  $H$ . Surprisingly, we show that this greedy algorithm significantly improves the competitive ratio when a congestion of 2 is allowed on the edges or when the model is stochastic. While our algorithm is fairly simple, our analysis requires a deep understanding of  $k$ -connected graphs.

In particular, we prove that the greedy algorithm is  $O(\log^2 n \log k)$ -competitive

if one satisfies every demand between  $u$  and  $v$  by  $r_{uv}/2$  edge-disjoint paths. The spirit of our result is similar to the work of Chuzhoy and Li [CL] (FOCS'12), in which the authors give a polylogarithmic approximation algorithm for edge-disjoint paths with congestion 2.

Moreover, we study the greedy algorithm in the online stochastic setting. In particular, we consider the iid model, where each online demand is drawn from a single probability distribution, the unknown iid model, where every demand is drawn from a single but unknown probability distribution, and the *prophet* model in which online demands are drawn from (possibly) different probability distributions. Through a different analysis we prove that a similar greedy algorithm is constant competitive for the iid and the prophet models. Also, the greedy algorithm is  $O(\log n)$ -competitive for the unknown iid model, which is almost tight due to the lower bound of [GGLS08] for single connectivity.

Finally, we present a general framework which analyzes the algorithm for the prophet model through an oblivious algorithm for the iid model. By applying this framework to other problems we achieve constant competitive algorithms for vertex cover and facility location and a logarithmic competitive algorithm for set cover in the prophet model.



## Chapter 4: Online Degree-Bounded Steiner Network Design

### 4.1 Introduction

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 [Win89, 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]).

Shortly after the work of Agrawal *et al.*, 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.

In this thesis, 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.

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<sup>+</sup>98, Goe06, Nut12, LS13, KKN13, EV14]. We refer the reader to the next sections for a brief sum-

mary. One particular variant that has been extensively studied is the *edge-weighted* DEGREE-BOUNDED SPANNING TREE. Initiated by Marathe *et al.* ([MRS<sup>+</sup>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  $b + 1$ . A slightly worse result is 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  $\Omega(n) \cdot \text{OPT}_b$  or maximum degree  $\Omega(n) \cdot b$ .

#### 4.1.1 Our 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  $\text{deg}_H(v)$  denote the degree of  $v$  in  $H$ . The *load* of a vertex is defined as the ratio  $\text{deg}_H(v)/b_v$ .

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. A main result of our work 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 4.2.

We complement our first theorem by giving an example for a special case of ONLINE DEGREE-BOUNDED STEINER TREE in which no online (randomized) algorithm can achieve a (multiplicative) competitive ratio  $o(\log n)$ . This also implies that obtaining (non-trivial) additive competitiveness is not possible in the online setting.

**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.*

The previously known techniques. As discussed before, the majority of techniques used for solving the offline variants of degree-bounded problems involve rounding an optimal fractional solution of a relaxed linear program. Since one may need to buy a long path to connect the endpoints of a demand, independent rounding of a fractional solution is hardly efficient. Instead, dependent and iterative rounding methods are usually used for attacking degree-bounded problems. In the online

paradigm, one can maintain a competitive fractional solution for these problems, however, it is inherently difficult to apply the aforementioned rounding techniques in an online setting: the underlying online fractional solution changes in between the rounding steps, thus breaking the chain of dependencies.

In contrast to the works on the offline paradigm, in this thesis we propose a simple combinatorial algorithm with a dual-fitting analysis. We use the structural properties of the output of our algorithm to show the existence of a chain of cuts that well separates the demand endpoints. When restricted to the case of uniform bounds, these cuts imply an upper bound on the *toughness* of the graph. The toughness of a graph is defined as  $\min_{X \subseteq V} \frac{|X|}{|\text{CC}(G \setminus X)|}$ ; where for a graph  $H$ ,  $\text{CC}(H)$  denotes the collection of connected components of  $H$ . It can be shown that the reciprocal of the toughness gives a lower bound for  $OPT$ . Therefore we use a combinatorial argument to show that the minimum of this ratio over the cuts in our chain of cuts is within  $O(\log(n))$  approximation of the reciprocal of the maximum load of a vertex in our solution.

We would like to emphasize that although the concept of toughness is well-studied in the literature, this line of research is mainly focused on relating toughness conditions to the existence of cycle structures, see for example a comprehensive survey by Bauer et al. [BBS06]. The relation between the graph toughness and degree-bounded problems have been previously observed by Win [Win89] and Agrawal et al. [AKR91]. However as mentioned in the introduction, Agrawal et al. use a completely different argument for analyzing the problem when reduced to a congestion minimization problem. We hope that the structural properties introduced in this

thesis together with the dual interpretation of our analysis, paves the way for solving the classical problems in the family of degree-bounded problems.

Hardness under more general constraints. 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.<sup>1</sup>

**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  $\mathbb{E}[\max\text{degree}(\mathcal{A})] \geq \Omega(n) \cdot b$  or  $\mathbb{E}[\text{weight}(\mathcal{A})] \geq \Omega(n) \cdot \text{OPT}_b$ , where  $\text{OPT}_b$  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.*

---

<sup>1</sup>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.

## 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]). Marathe *et al.* ([MRS<sup>+</sup>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] and [KR05]) was focused on this problem until a groundbreaking breakthrough was obtained by Goemans ([Goe06]); his algorithm computes a minimum-weight spanning tree with degree at most  $b+2$ . Later on, Singh and Lau ([SL07]) 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 online setting, the follow-up work of Dehghani et al. [DEH<sup>+</sup>17a] proposes non-tight bicriteria algorithms for edge-weighted degree-bounded Steiner forest problem.

In the *degree-bounded survivable network design* problem, a number  $d_i$  is associated 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]). 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]). 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.* ([AAB04]) and Berman and Coulston ([BC97]). Later on, Naor, Panigrahi, and Singh ([NPS11]) and Hajiaghayi, Liaghat, and Panigrahi ([HLP13]), 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 ignore a demand by paying its given penalty. Qian and Williamson ([QW11]) and Hajiaghayi, Liaghat, and Panigrahi ([HLP14]) 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]) 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], for the full information model see [MOGS12, MGZ12], and for the prophet-inequality model see [AHL<sup>+</sup>11, AHL12, AHL13]. We also refer the reader to the comprehensive survey by Mehta [Meh12].

Many of the aforementioned problems can be characterized as an online packing or covering linear program. Initiated by work of Alon *et al.* [AAA<sup>+</sup>09] on online set cover, 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 [BN09]. 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 output is within  $O(\log^2 n)$  factor of optimum. We note that although the fractional variant of our problem is a special case of 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  $i^{\text{th}}$  demand is a pair of vertices  $(s_i, t_i)$  which we call the *end-*

points 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<sup>2</sup>, 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  $OPT = \min_H \max_v \ell_H(v)$ .

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 collection 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  $\alpha$  indicates an upper bound on the load of every vertex. The first set of constraints ensures that the endpoints of every demand are connected. The second set of constraints ensures that the load of a vertex is upper bounded by  $\alpha$ . The program  $\mathbb{D}$  is the dual of the LP relaxation  $\mathbb{P}$ .

---

<sup>2</sup>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.

$$\begin{array}{ll}
\text{minimize } \alpha & (\mathbb{P}) \qquad \text{maximize } \sum_{S \in \mathcal{S}} \mathbf{y}(S) \qquad (\mathbb{D}) \\
\forall S \subseteq \mathcal{S} \quad \sum_{e \in \delta(S)} \mathbf{x}(e) \geq 1 & (\mathbf{y}(S)) \qquad \forall e = (u, v) \quad \sum_{S: e \in \delta(S)} \mathbf{y}(S) \leq \mathbf{z}(u) + \mathbf{z}(v) \\
\forall v \in V \quad \sum_{e \in \delta(\{v\})} \mathbf{x}(e) \leq \alpha \cdot b_v & (\mathbf{x}(e)) \\
& (\mathbf{z}(v)) \qquad \sum_v \mathbf{z}(v) b_v \leq 1 \qquad (\alpha) \\
\mathbf{x}(e), \alpha \in \mathbb{R}_{\geq 0} & \mathbf{z}(v), \mathbf{y}(S) \in \mathbb{R}_{\geq 0}
\end{array}$$

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  $i^{\text{th}}$  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  $\max_v \ell_H(v) / OPT$  over all possible demand sequences where  $H$  is the final output of the algorithm.

## 4.2 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 abuse 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 1).

#### 4.2.1 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 **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  $\ell_{\mathbf{GA}}^+(v) \geq r$ . Let  $D(r)$  denote the indices of demands  $i$  for which the arrival threshold  $\tau_i$  is at least  $r$ . For a subgraph  $X$ , let  $\text{CC}(X)$  denote the collection of connected components

---

**Algorithm 1** Online Degree-Bounded Steiner Forest

---

**Input:** A graph  $G$ , and an online stream of demands  $(s_1, t_1), (s_2, t_2), \dots$

**Output:** A set  $H$  of edges such that every given demand  $(s_i, t_i)$  is connected via  $H$ .

**Offline Process:**

1: Initialize  $H = \emptyset$ .

**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  $H = H \cup P_i^*$ .

---

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  $|\text{CC}(G \setminus \Gamma(r)) \cap \mathcal{S}| \geq |D(r)| + 1$ .*

**Proof.** For a vertex  $v \in G \setminus \Gamma(r)$  we use  $\text{CC}(v)$  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  $\text{CC}(s_i)$  and  $\text{CC}(t_i)$ . In the following we argue that  $F$  does not contain cycles. This gives the lemma since in a forest  $|D(r)| + 1 = |E_F| + 1 \leq |V_F| \leq |\text{CC}(G \setminus \Gamma(r)) \cap \mathcal{S}|$ .

Assume for contradiction that the sequence  $(e_{i_0}, \dots, e_{i_{k-1}})$ ,  $k \geq 2$  forms a (minimal) cycle in  $F$ , where  $e_{i_j}$  corresponds to the demand between vertices  $s_{i_j}$  and  $t_{i_j}$ . 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  $r$ .  $\square$

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.* [AKR91], 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  $\min_{X \subseteq V} \frac{|X|}{|\text{CC}(G \setminus X)|}$ . 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  $\Gamma_b(r) := \{\ell_{\text{GA}}^+(v) \geq r \wedge b_v \geq b\}$ , 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

$$\text{excess}(r, b) = \sum_{v \in \Gamma_b(r)} \left( \deg_{\text{GA}}(v) - \lceil rb_v \rceil + 2 \right),$$

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  $r, b > 0$ ,  $\text{excess}(r, b) \leq 2|D(r)| + 3|\Gamma_b(r)|$ .*

**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  $\delta(r, b) := \sum_{v: \ell_H^+(v) \geq r \wedge b_v \geq b} (\deg_H(v) - \lceil rb_v \rceil + 2)$  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

$$\begin{aligned}
\text{excess}(r, b) &= \sum_{i \in D(r)} q_i + \sum_{i \notin D(r)} q_i \\
&= 2|D(r)| + \sum_{i \in D(r)} (q_i - 2) + |\Gamma_b(r)| \\
&\leq 2|D(r)| + 3|\Gamma_b(r)| \text{ ,}
\end{aligned}$$

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  $OPT \geq 1/\Delta$ . Indeed, replacing  $\Delta$  with one in the following would make the arguments slightly simpler. However, by doing so we incur an additive  $\log(n)$  term in the final result which can be arbitrary far from  $OPT$ . Therefore it is necessary to employ  $\Delta$  in the analysis. For a set of vertices  $\Gamma$ , 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  $G_\Delta$  denote the graph obtained from  $G$  by removing vertices with degree bound below  $\Delta$ . Similarly, let  $\mathcal{S}_\Delta$  denote the collection of sets that separate a demand in  $G_\Delta$ . Consider a slightly modified dual program  $\mathbb{D}_\Delta$  defined on  $G_\Delta$  and  $\mathcal{S}_\Delta$ , i.e., we obtain  $\mathbb{D}_\Delta$  from the original dual program by removing all vertices with degree bound below  $\Delta$ . 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 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

for cuts containing  $e$  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  $S \in \text{CC}(G_\Delta \setminus \Gamma_\Delta(r)) \cap \mathcal{S}_\Delta$ , 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  $C \in \text{CC}(G \setminus \Gamma(r))$ . 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  $|\text{CC}(G \setminus \Gamma(r)) \cap \mathcal{S}| \leq |\text{CC}(G_\Delta \setminus \Gamma_\Delta(r)) \cap \mathcal{S}_\Delta|$ ; which by Lemma 4.1 leads to  $|\text{CC}(G_\Delta \setminus \Gamma_\Delta(r)) \cap \mathcal{S}_\Delta| \geq |D(r)| + 1$ . Therefore the dual objective for  $(\mathbf{y}_r, \mathbf{z}_r)$  is at least

$$\sum_{S \in \mathcal{S}_\Delta} \mathbf{y}_r(S) = \sum_{S \in \text{CC}(G_\Delta \setminus \Gamma_\Delta(r)) \cap \mathcal{S}_\Delta} \frac{1}{b(\Gamma_\Delta(r))} \geq \frac{|D(r)| + 1}{b(\Gamma_\Delta(r))}$$

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  $\text{CC}(G_\Delta \setminus \Gamma_\Delta(r))$ , 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.  $\square$

We now have all the ingredients to prove the main theorem.

**Proof of Theorem 4.1:** Let  $\mathbf{GA}$  denote the final output of the greedy algorithm. Let  $h$  denote the maximum load of a vertex in  $\mathbf{GA}$ , i.e.,  $h = \max_v \ell_{\mathbf{GA}}(v)$ . Furthermore, let  $h_\Delta = \max_{v: b_v \geq \Delta} \ell_{\mathbf{GA}}(v)$ . In the following we first show that  $h_\Delta \leq O(\log n) \cdot \mathit{OPT}$ . For this we use the following claim.

**Claim 4.1.** *There exists an  $r > 0$  such that  $\mathit{excess}(r, \Delta) \geq \frac{h_\Delta - 1/\Delta}{4 \log_2 n + 6} \cdot b(\Gamma_\Delta(r)) - |\Gamma_\Delta(r)|$ .*

**Proof.** Recall that  $\Gamma_\Delta(r)$  is the set of vertices with uptick load at least  $r$  in  $\mathbf{GA}$  and degree bound at least  $\Delta$ . 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  $b(\Gamma_\Delta(r)) \geq \Delta$  as there exists a node with load  $h_\Delta$  in  $\Gamma_\Delta(r)$  and this node has degree bound at least  $\Delta$ .

We now partition the range of  $r$  (from  $1/\Delta$  to  $h_\Delta$ ) into logarithmically many intervals. We define the  $q$ -th interval by

$$U(q) = \{r \mid 1/\Delta \leq r < h_\Delta \wedge 2^q/\Delta \leq b(\Gamma_\Delta(r)) < 2^{q+1}/\Delta\},$$

for  $q \in \{0, \dots, \lceil \log_2((n+1)n) \rceil\}$ . We further set  $\bar{r}(q) = \max U(q)$  and  $\underline{r}(q) = \min U(q)$ , and call  $\bar{r}(q) - \underline{r}(q)$  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  $\Gamma_\Delta(\bar{r})$  and, hence, also in  $\Gamma_\Delta(\underline{r})$ . This node starts contributing to  $\text{excess}(\underline{r}, \Delta)$ , once its uptick load reached  $\underline{r}$  and contributes at least until its uptick load reaches  $\bar{r}$ . Hence, the total contribution is at least  $\text{deg}(\bar{r}) - \text{deg}(\underline{r})$ , where  $\text{deg}(\bar{r})$  and  $\text{deg}(\underline{r})$  denotes the degree of node  $v$  when reaching uptick load  $\bar{r}$  and  $\underline{r}$ , respectively. We have

$$\text{deg}(\bar{r}) - \text{deg}(\underline{r}) = (\lceil \bar{r} b_v \rceil - 2) - (\lceil \underline{r} b_v \rceil - 2) \geq (\bar{r} - \underline{r}) b_v - 1.$$

Summing this over all nodes in  $\Gamma_\Delta(\bar{r})$  gives

$$\begin{aligned} \text{excess}(\underline{r}, \Delta) &\geq (\bar{r} - \underline{r}) \cdot b(\Gamma_\Delta(\bar{r})) - |\Gamma_\Delta(\bar{r})| \\ &\geq \frac{1}{2}(\bar{r} - \underline{r}) \cdot b(\Gamma_\Delta(\underline{r})) - |\Gamma_\Delta(\underline{r})| \\ &\geq \frac{h_\Delta - 1/\Delta}{4 \log_2 n + 6} \cdot b(\Gamma_\Delta(\underline{r})) - |\Gamma_\Delta(\underline{r})| , \end{aligned}$$

where the second inequality uses the fact that  $|\Gamma_\Delta(\bar{r})| \leq |\Gamma_\Delta(\underline{r})| \leq |\Gamma_\Delta(\bar{r})|/2$ .  $\square$

**Claim 4.2.**  $h_\Delta \leq (24 \log_2 n + 37) \text{OPT}$ .

**Proof.** If we use the  $r$  from the previous claim and solve for  $h_\Delta$  we obtain

$$\begin{aligned} h_\Delta &\leq (4 \log_2 n + 6) \cdot \frac{\text{excess}(r, \Delta) + \Gamma_\Delta(r)}{b(\Gamma_\Delta(r))} + \frac{1}{\Delta} \\ &\leq (4 \log_2 n + 6) \cdot \frac{2D(r) + 4\Gamma_\Delta(r)}{b(\Gamma_\Delta(r))} + \frac{1}{\Delta} \end{aligned}$$

*Lemma 4.2*

$$\leq (4 \log_2 n + 6) \cdot \left( 2\text{OPT} + 4\frac{1}{\Delta} \right) + \frac{1}{\Delta}$$

*Lemma 4.3 and  $b(\Gamma_\Delta(r)) \geq \Delta |\Gamma_\Delta(r)|$*

$$\leq (24 \log_2 n + 37) \cdot \text{OPT}$$

$$\text{OPT} \geq 1/\Delta .$$

□

We now bound the maximum load  $h$  in terms of the restricted maximum load  $h_\Delta$ . Consider a vertex  $v^*$  with maximum load  $\ell_{\text{GA}}(v^*) = h$ . 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,  $\tau_i$  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

$$\begin{aligned}
 \tau_i &\leq \ell_H^+(v) \leq h_\Delta + \frac{2}{b_v} & \ell_H(v) &\leq \ell_{\text{GA}}(v) \leq h_\Delta \\
 &\leq h_\Delta + \frac{2}{\Delta} & & b_v &\geq \Delta \\
 &\leq h_\Delta + 2OPT & & OPT &\geq 1/\Delta \\
 &\leq O(\log n) \cdot OPT & & & \text{by the above claim}
 \end{aligned}$$

Therefore leading to  $h \leq O(\log n) \cdot OPT$ .

□

### 4.3 An Asymptotically Tight Lower Bound

In the following we show a lower bound for ONLINE DEGREE-BOUNDED STEINER TREE. Consider a graph  $G = (X \uplus Z, E)$ ,

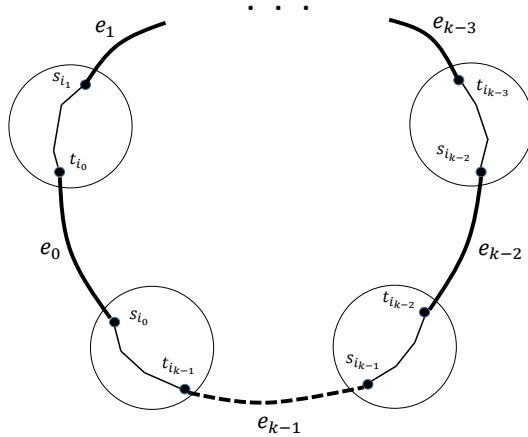


Figure 4.1: Existence of a cycle implies that of a path with low uptick load in its extension part.

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  $\sigma$  (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  $\sigma$ . Similarly, we use  $\deg'_{A,\sigma}(z)$  to denote the number of those neighbors of  $z$  that also appear in  $\sigma$ . 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  $\sigma_S$  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  $\mathbb{E}[\deg'_{A,\sigma}(z^*)] \geq s/2$ ;
- there exists an offline algorithm OFF for servicing requests in  $\sigma$  with  $\max_{z \in S} \{\deg_{\text{OFF},\sigma}(z)\} \leq 1$ , and  $\deg_{\text{OFF},\sigma}(z) = 0$  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  $\sigma_1$  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  $\mathbb{E}[\deg'_{A,\sigma_1}(z_1^*)] \geq s/2$ . Now, let  $\tilde{A}$  behave like algorithm  $A$  *after* it already received a request sequence  $\sigma_1$  (hence, it starts with all edges that are chosen when running  $A$  on  $\sigma_1$ ; note, however, that  $\deg'_{\tilde{A},\sigma_2}(\cdot)$  only takes into account edges incident to nodes from  $\sigma_2$ ). Due to induction hypothesis for  $\tilde{A}$  and set  $S_2$  there exists a request sequence  $\sigma_2$  such that  $\mathbb{E}[\deg'_{\tilde{A},\sigma_2}(z_2^*)] \geq s/2$  for a node  $z_2^* \in S_2$ . Hence, the request sequence  $\sigma = \sigma_1 \circ \sigma_2$  fulfills  $\mathbb{E}[\deg'_{A,\sigma}(z_1^*)] \geq s/2$  and  $\mathbb{E}[\deg'_{A,\sigma}(z_2^*)] \geq s/2$ .

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  $\mathbb{E}[\deg'_{A,\sigma}(z_1^*)]$  or  $\mathbb{E}[\deg'_{A,\sigma}(z_2^*)]$  must be at least  $(s + 1)/2$ .

It remains to argue that there exists a good offline algorithm. Combining the offline algorithms OFF<sub>1</sub> and OFF<sub>2</sub> for  $\sigma_1$  and  $\sigma_2$  gives an offline algorithm for  $\sigma_1 \circ \sigma_2$



that has  $\max_z \{\deg_{\text{OFF}, \sigma_1 \circ \sigma_2}(z)\} \leq 1$  and  $\deg_{\text{OFF}, \sigma_1 \circ \sigma_2}(z) = 0$  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 the request by either buying edge  $\{x, z_1\}$  or  $\{x, z_2\}$ , and can therefore guarantee that  $z^*$  ( $z_1$  or  $z_2$ ) has degree 0.  $\square$

**Proof of Theorem 4.2:** Choosing  $s = \log n$  in the above lemma gives our lower bound.  $\square$

## Chapter 5: Online Weighted Degree-Bounded Steiner Networks

### 5.1 Introduction

*Degree-bounded* network design problems comprise an important family of network design problems since the eighties. Aside from various real-world applications such as vehicle routing and communication networks [BV95, OP05, VoB92], the family of degree-bounded problems has been a testbed for developing new ideas and techniques. 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]. In this problem, we are required to find a spanning tree of a given graph with the goal of minimizing the maximum degree of the vertices in the tree. Let  $b^*$  denote the maximum degree in the optimal spanning tree. Fürer and Raghavachari give a parallel approximation algorithm which produces a spanning tree of degree at most  $O(\log(n)b^*)$ . This result was later generalized by Agrawal, Klein, and Ravi [AKR91] to the case of degree-bounded Steiner tree (DEGREE-BOUNDED STEINER TREE) and degree bounded Steiner forest (DEGREE-BOUNDED STEINER FOREST) problem. In DEGREE-BOUNDED STEINER TREE, given a set of terminal vertices, we need to find a subgraph of minimum maximum degree that connects the terminals. In

the more generalized DEGREE-BOUNDED STEINER FOREST problem, we are given pairs of terminals and the output subgraph should contain a path connecting each pair. 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  $b^* + 1$ .

The study of DEGREE-BOUNDED STEINER TREE and DEGREE-BOUNDED STEINER FOREST was the starting point of a very popular line of work on various degree-bounded network design problems; e.g. [MRS<sup>+</sup>98, Nut12, LS13, KKN13, EV14] and more recently [FR12, EV14]. One particular variant that has been extensively studied was initiated by Marathe *et al.* [MRS<sup>+</sup>98]: In the *edge-weighted degree-bounded spanning tree* problem, given a weight function over the edges and a degree bound  $b$ , the goal is to find a minimum-weight spanning tree with maximum degree at most  $b$ . The initial results for the problem generated much interest in obtaining approximation algorithms for the edge-weighted degree-bounded spanning tree problem [CRRT09, CRRT06, Goe06, KKRR04, KR00, KR05, LNSS09, Rag96, RMR<sup>+</sup>01, RS06]. 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  $b + 1$ . Singh and Lau [LS13] generalize their result for the *edge-weighted Steiner tree* (EW-DB-ST) and *edge-weighted Steiner forest* (EW-DB-SF) variants. They design an algorithm that finds a Steiner forest with cost at most twice the cost of the optimal solution while violating the degree constraints by at most three.

Despite these achievements in the offline setting, it was not known whether degree-bounded problems are tractable in the *online setting*. The online counterparts of the aforementioned Steiner problems can be defined as follows. The underlying graph and degree bounds are known in advance. The demands arrive one by one in an online manner. At the arrival of a demand, we need to augment the solution subgraph such that the new demand is satisfied. The goal is to be competitive against an offline optimum that knows the demands in advance.

Recently, Dehghani et al. [DEHL16] explore the tractability of the Online DEGREE-BOUNDED STEINER FOREST problem by showing that a natural greedy algorithm produces a solution in which the degree bounds are violated by at most a factor of  $O(\log n)$ , which is asymptotically *tight*. They analyze their algorithm using a dual fitting approach based on the combinatorial structures of the graph such as the toughness<sup>1</sup> factor. Unfortunately, they can also show that greedy methods are not competitive for the edge-weighted variant of the problem. Hence, it seems unlikely that the approach of [DEHL16] can be generalized to EW-DB-SF.

The *online edge-weighted Steiner connectivity* problems (with no bound on the degrees) have been extensively studied in the last decades. Imase and Waxman [IW91] use a dual-fitting argument to show that the greedy algorithm has a competitive ratio of  $O(\log n)$ , which is also asymptotically tight. Later the result was generalized to the EW SF variant by Awerbuch *et al.* [AAB04] and Berman and Coulston [BC97]. In the past few years, various primal-dual

---

<sup>1</sup>The toughness of a graph is defined as  $\min_{X \subseteq V} \frac{|X|}{|\text{CC}(G \setminus X)|}$ ; where for a graph  $H$ ,  $\text{CC}(H)$  denotes the collection of connected components of  $H$ .

techniques have been developed to solve the more general node-weighted variants [AAA<sup>+</sup>09,NPS11,HLP13], prize-collecting variants [QW11,HLP14], and multi-commodity buy-at-bulk [CEKP15]. These results are obtained by developing various primal-dual techniques [AAA<sup>+</sup>09,HLP13] while generalizing the application of combinatorial properties to the online setting [NPS11,HLP14,CEKP15]. In this thesis however, we develop a primal approach for solving *bounded-frequency mixed packing/covering integer programs*. We believe this framework would be proven useful in attacking other online packing and covering problems.

### 5.1.1 Our Results and Techniques

In this thesis, we consider the online Steiner tree and Steiner forest problems at the presence of both edge weights and degree bounds. In the Online EW-DB-SF problem, we are given a graph  $G = (V, E)$  with  $n$  vertices, edge-weight function  $w$ , degree bound  $b_v$  for every  $v \in V$ , and an online sequence of connectivity demands  $(s_i, t_i)$ . Let  $w_{\text{opt}}$  denote the minimum weight subgraph which satisfies the degree bounds and connects all demands. Let  $\rho = \frac{\max_e w(e)}{\min_{e:w(e)>0} w(e)}$ .

**Theorem 5.1.** *There exists an online deterministic algorithm which finds a subgraph with total weight at most  $O(\log^3 n)w_{\text{opt}}$  while the degree bound of a vertex is violated by at most a factor of  $O(\log^3(n) \log(n\rho))$ .*

*If one favors the degree bounds over total weight, one can find a subgraph with degree-bound violation  $O(\log^3(n) \frac{\log(n\rho)}{\log \log(n\rho)})$  and total cost  $O(\log^3(n) \frac{\log(n\rho)}{\log \log(n\rho)})w_{\text{opt}}$ .*

We note that the logarithmic dependency on  $\rho$  is indeed necessary. It follows

from the result of [DEHL16] that the competitive ratio of any algorithm is either  $\Omega(n)$  or  $\Omega(\log \rho)$ .

Our technical contribution for solving the EW-DB-SF problem is twofold. First by exploiting a structural result and massaging the optimal solution, we show a formulation of the problem that falls in the restricted family of *bounded-frequency mixed packing/cover IPs*, while losing only logarithmic factors in the competitive ratio. We then design a generic online algorithm with a logarithmic competitive ratio that can solve any instance of the bounded-frequency packing/covering IPs. In what follows, we describe our results in detail.

#### 5.1.1.1 Massaging the optimal solution

Initiated by work of Alon *et al.* [AAA<sup>+</sup>09] on online set cover, 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 [BN09]. Azar *et al.* [ABFP13] generalize this method for the fractional *mixed* packing and covering LPs. The natural linear program relaxation for EW-DB-SF, commonly used in the literature, is a special case of mixed packing/covering LPs: one needs to select an edge from every cut that separates the endpoints of a demand (covering constraints), while for a vertex we cannot choose more than a specific number of its adjacent edges (packing constraints). Indeed, one can use the result of Azar *et al.* [ABFP13] to find an online *fractional* solution with polylogarithmic competitive ratio. However, doing

the rounding in an online manner seems very hard.

Offline techniques for solving degree-bounded problems often fall in the category of iterative and dependent rounding methods. Unfortunately, these methods are inherently difficult to adapt for an online settings since the underlying fractional solution may change dramatically in between the rounding steps. Indeed, this might be the very reason that despite many advances in the online network design paradigm in the past two decades, the natural family of degree-bounded problems has remained widely open. In this thesis, we circumvent this by reducing EW-DB-ST to a novel formulation beyond the scope of standard online packing/covering techniques and solving it using a new online integral approach.

The crux of our IP formulation is the following structural property: Let  $(s_i, t_i)$  denote the  $i^{\text{th}}$  demand. We need to augment the solution  $Q_{i-1}$  of previous steps by buying a subgraph that makes  $s_i$  and  $t_i$  connected. Let  $G_i$  denote the graph obtained by contracting the pairs of vertices  $s_j$  and  $t_j$  for every  $j < i$ . Note that any  $(s_i - t_i)$ -path in  $G_i$  corresponds to a feasible augmentation for  $Q_{i-1}$ . Some edges in  $G_i$  might be already in  $Q_{i-1}$  and therefore by using them again we can save both on the total weight and the vertex degrees. However, in Section 5.2 we prove that there always exists a path in  $G_i$  such that even without sharing on any of the edges in  $G_i$  and therefore paying completely for the increase in the weight and degrees, we can approximate the optimal solution up to a logarithmic factor. This in fact, enables us to have a formulation in which the covering constraints for different demands are *disentangled*. Indeed, we only have one covering constraint for each demand. Unfortunately, this implies that we have exponentially many variables, one for each

possible path in  $G_i$ . This may look hopeless since the competitive factors obtained by standard fractional packing/covering methods introduced by Buchbinder and Naor [BN09] and Azar *et al.* [ABFP13], depend on the logarithm of the number of variables. Therefore we come up with a new approach for solving this class of mixed packing/covering integer programs (IP).

### 5.1.1.2 Bounded-frequency mixed packing/covering IPs

We derive our result for EW-DB-ST by demonstrating a new technique for solving mixed packing/covering *integer* programs. We believe this approach could be applicable to a broader range of online problems. The integer program [IIP1](#) describes a general mixed packing/covering IP with the set of integer variables  $\mathbf{x} \in \mathbb{Z}_{\geq 0}^n$  and  $\alpha$ . The packing constraints are described by a  $m \times n$  non-negative matrix  $P$ . Similarly, the  $q \times n$  matrix  $C$  describes the covering constraints. The *covering frequency* of a variable  $x_i$  is defined as the number of covering constraints in which  $x_i$  has a positive coefficient. The covering frequency of a mixed packing/covering program is defined as the maximum covering frequency of its variables.

$$\begin{aligned}
 & \text{minimize} && \alpha , && && (\text{IIP1}) \\
 & && && && \\
 & \text{s.t.} && P\mathbf{x} \leq \alpha . && && \\
 & && && && \\
 & && C\mathbf{x} \geq 1 . && && \\
 & && && && \\
 & && \mathbf{x} \in \mathbb{Z}_{\geq 0}, \alpha \in \mathbb{R}_{>0} . && &&
 \end{aligned}$$

In the online variant of mixed packing and covering IP, we are given the packing



constraints in advance. However the covering constraints arrive in an online manner. At the arrival of each covering constraint, we should *increase* the solution  $\mathbf{x}$  such that it satisfies the new covering constraint. We provide a deterministic algorithm for solving online mixed packing/covering IPs.

**Theorem 5.2.** *Given an instance of the online mixed packing/covering IP, there exists a deterministic integral algorithm with competitive ratio  $O(k \log m)$ , where  $m$  is the number of packing constraints and  $k$  is the covering frequency of the IP.*

We note that the competitive ratio of our algorithm is independent of the number of variables or the number of covering constraints. Indeed, there can be exponentially many variables.

Our result can be thought of as a generalization of the work of Aspnes et al. [AAF<sup>+</sup>97] on virtual circuit routing. Although not explicit, their result can be massaged to solve mixed packing/covering IPs in which all the coefficients are zero or one, and the covering frequency is one. They show that such IPs admit a  $O(\log(m))$ -competitive algorithms. Theorem 5.2 generalizes their result to the case with arbitrary non-negative coefficients and any bounded covering frequency.

We complement our result by proving a matching lower bound for the competitive ratio of any *randomized* algorithm. This lower bound holds even if the algorithm is allowed to return fractional solutions.

**Theorem 5.3.** *Any randomized online algorithm  $A$  for integral mixed packing and covering is  $\Omega(k \log m)$ -competitive, where  $m$  denotes the number of packing constraints, and  $k$  denotes the covering frequency of the IP. This even holds if  $A$  is*

allowed to return a fractional solution.

As mentioned before, Azar *et al.* [ABFP13] provide a fractional algorithm for mixed packing/covering LPs with competitive ratio of  $O(\log m \log d)$  where  $d$  is the maximum number of variables in a single constraint. They show an almost matching lower bound for deterministic algorithms. We distinguish two advantages of our approach compared to that of Azar *et al.*:

- The algorithm in [ABFP13] outputs a *fractional* competitive solution which then needs to be rounded online. For various problems such as Steiner connectivity problems, rounding a solution online is very challenging, even if offline rounding techniques are known. Moreover, the situation becomes hopeless if the integrality gap is unbounded. However, for bounded-frequency IPs, our algorithm directly produces an integral competitive solution. Thus it does not depend on rounding methods, and is applicable to problems with large integrality gap, or the problems for which it is shown that rounding methods do not preserve any approximation guarantee, and as such, the traditional approach fails.
- Azar *et al.* find the best competitive ratio with respect to the number of packing constraints and the size of constraints. Although these parameters are shown to be bounded in several problems, in many problems such as connectivity problems and flow problems, formulations with exponentially many variables are very natural. Our techniques provide an alternative solution with a tight competitive ratio, for formulations with bounded covering frequency.

## 5.1.2 Preliminaries

Let  $G = (V, E)$  be an undirected graph of size  $n$  ( $|V| = n$ ). Let  $w : E \rightarrow \mathbb{Z}_{>0}$  be a function denoting the edge weights. For a subgraph  $H \subseteq G$ , we define  $w(H) := \sum_{e \in E(H)} w(e)$ . For every vertex  $v \in V$ , let  $b_v \in \mathbb{Z}_{>0}$  denote the degree bound of  $v$ . Let  $\deg_H(v)$  denote the degree of vertex  $v$  in subgraph  $H$ . We define the load  $l_H(v)$  of vertex  $v$  w.r.t.  $H$  as  $\deg_H(v)/b_v$ . In DEGREE-BOUNDED STEINER FOREST we are given graph  $G$ , degree bounds, and  $k$  connectivity demands. Let  $\sigma_i$  denote the  $i$ -th demand. The  $i$ -th demand is a pair of vertices  $\sigma_i = (s_i, t_i)$ , where  $s_i, t_i \in V$ . In DEGREE-BOUNDED STEINER FOREST the goal is to find a subgraph  $H \subseteq G$  such that for each demand  $\sigma_i$ ,  $s_i$  is connected to  $t_i$  in  $H$ , for every vertex  $v \in V$ ,  $l_H(v) \leq 1$ , and  $w(H)$  is minimized. In this thesis without loss of generality we assume the demand endpoints are distinct vertices with degree one in  $G$  and degree bound infinity.

In the online variant of the problem, we are given graph  $G$  and degree bounds in advance. However the sequence of demands are given one by one. At arrival of demand  $\sigma_i$ , we are asked to provide a subgraph  $H_i$ , such that  $H_{i-1} \subseteq H_i$  and  $s_i$  is connected to  $t_i$  in  $H_i$ .

The following integer program is a natural mixed packing and covering integer program for EW-DB-SF. Let  $\mathcal{S}$  denote the collection 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 SF-IP, for an edge  $e$ ,  $x_e = 1$  indicates that we include  $e$  in the solution while  $x_e = 0$  indicates otherwise.

The variable  $\alpha$  indicates an upper bound on the violation of the load of every vertex and an upper bound on the violation of the weight. The first set of constraints ensures that the load of a vertex is upper bounded by  $\alpha$ . The second constraint ensures that the violation for the weight is upper bounded by  $\alpha$ . The third set of constraints ensures that the endpoints of every demand are connected. Here we assume  $w_{\text{opt}}$  is known to the algorithm, although this can be waived by standard doubling techniques.

$$\text{minimize } \alpha . \tag{SF-IP}$$

$$\forall v \in V \quad \frac{1}{b_v} \sum_{e \in \delta(\{v\})} x_e \leq \alpha . \tag{5.1}$$

$$\frac{1}{w_{\text{opt}}} \sum_{e \in E} w(e)x_e \leq \alpha . \tag{5.2}$$

$$\forall S \subseteq \mathcal{S} \quad \sum_{e \in \delta(S)} x_e \geq 1 . \tag{5.3}$$

$$x_e \in \{0, 1\}, \alpha \in \mathbb{Z}_{>0} .$$

### 5.1.3 Overview of the Chapter

We begin Section 5.2 by providing a bounded frequency IP for EW-DB-SF. The IP is not a proper formulation of the problem, however, we can show that one can map feasible solutions of EW-DB-SF to feasible solutions of the IP without increasing the cost too much. In Section 5.3 we provide a deterministic algorithm for online bounded frequency mixed packing/covering IPs. We also provide a matching lower bound for the competitive ratio of any randomized algorithm. Finally, in Section 5.4 we merge our techniques to obtain online polylogarithmic-competitive

algorithms for EW-DB-SF.

## 5.2 Finding the Right Integer Program

In this section we design an online mixed packing and covering integer program for EW-DB-SF. We show this formulation is near optimal, i.e. any  $f$ -approximation for this formulation, implies an  $O(f \log^2 n)$ -approximation for EW-DB-SF. In Section 5.4 we show there exists an online algorithm that finds an  $O(\log n)$ -approximation solution for this IP and violates degree bounds by  $O(\log^3 n \log w_{\text{opt}})$ , where  $w_{\text{opt}}$  denotes the optimal weight.

First we define some notations. For a sequence of demands  $\sigma = \langle (s_1, t_1), \dots, (s_k, t_k) \rangle$ , we define  $R_\sigma(i)$  to be a set of  $i$  edges, connecting the endpoints of the first  $i$  demands. In particular  $R_\sigma(i) := \bigcup_{j=1}^i e(s_j, t_j)$ , where  $e(s_j, t_j)$  denotes a direct edge from  $s_j$  to  $t_j$ . Moreover, we say subgraph  $H_i$  satisfies the connectivity of demand  $\sigma_i = (s_i, t_i)$ , if  $s_i$  and  $t_i$  are connected in graph  $H_i \cup R_\sigma(i-1)$ . Let  $\mathcal{H}_i$  denote the set of all subgraphs that satisfy the connectivity of demand  $\sigma_i$ . In [PC\\_IP](#) variable  $\alpha$  denotes the violation in the packing constraints. Furthermore for every subgraph  $H \subseteq G$  and demand  $\sigma_i$ , there exists a variable  $x_H^i \in \{0, 1\}$ .  $x_H^i = 1$  indicates we add the edges of  $H$  to the existing solution, at arrival of demand  $\sigma_i$ . The first set of constraints ensure the degree-bounds are not violated more than  $\alpha$ . The second constraint ensures the weight is not violated by more than  $\alpha$ . The third set of constraints ensure the endpoints of every demand are connected.

$$\text{minimize } \alpha . \tag{PC_IP}$$

$$\forall v \in V \quad \frac{1}{b_v} \sum_{i=1}^k \sum_{H \subseteq G} \deg_H(v) x_H^i \leq \alpha . \tag{5.4}$$

$$\frac{1}{w_{\text{opt}}} \sum_{i=1}^k \sum_{H \subseteq G} w(H) x_H^i \leq \alpha . \tag{5.5}$$

$$\forall \sigma_i \quad \sum_{H \in \mathcal{H}_i} x_H^i \geq 1 . \tag{5.6}$$

$$\forall H \subseteq G, 1 \leq i \leq k \quad \mathbf{x}_H^i \in \{0, 1\} .$$

$$\alpha > 0 .$$

We are considering the online variant of the mixed packing and covering program. We are given the packing Constraints (5.4) and (5.5) in advance. At arrival of demand  $\sigma_i$ , the corresponding covering Constraint (5.6) is added to the program. We are looking for an online solution which is feasible at every online stage. Moreover the variables  $x_H$  should be monotonic, i.e. once an algorithm sets  $x_H = 1$  for some  $H$ , the value of  $x_H$  is 1 during the rest of the algorithm. Figure (5.1) illustrates an example which indicates the difference between the solutions of [PC\\_IP](#) and [SF\\_IP](#).

Let  $\text{popt}$  and  $\text{lopt}$  denote the optimal solutions for [PC\\_IP](#) and [SF\\_IP](#), respectively. Lemma 5.1 shows that given an online solution for [PC\\_IP](#) we can provide a feasible online solution for [SF\\_IP](#) of cost  $\text{popt}$ .

**Lemma 5.1.** *Given a feasible solution  $\{\mathbf{x}, \alpha\}$  for [PC\\_IP](#), there exists a feasible solution  $\{\mathbf{x}', \alpha\}$  for [SF\\_IP](#).*

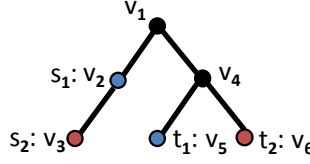


Figure 5.1: An example where every vertex has degree-bound 3 and every edge has weight 1. The first demand is  $(v_2, v_5)$  and the second demand is  $(v_3, v_6)$ . The optimal solution for [SF-IP](#) is a subgraph, say  $H$ , with the set of all edges and vertices, i.e.  $H = G$ . However an optimal solution for [PC-IP](#) is: Two subgraphs  $H_1$  for the first request which has edges  $\{e(v_1, v_2), e(v_1, v_4), e(v_4, v_5)\}$  and  $H_2$  for the second request which has edges  $\{e(v_2, v_3), e(v_4, v_5), e(v_4, v_6)\}$ . Note that  $w(H) = 5$  and  $w(H_1) + w(H_2) = 6$ , since we have edge  $e(v_4, v_5)$  in both  $H_1$  and  $H_2$ . Moreover the number of edges incident with  $v_4$  in the solution of [PC-IP](#) is 4, i.e.  $\deg_{H_1}(v_4) + \deg_{H_2}(v_4) = 4$ .

In the rest of this section, we show that we do not lose much by changing [SF-IP](#) to [PC-IP](#). In particular we show  $\text{popt} \leq O(\log^2 n)\text{lopt}$ . To this end, we first define the *connective* list of subgraphs for a graph  $G$ , a forest  $F$ , and a list of demands  $\sigma$ . We then prove an existential lemma for such a list of subgraphs with a desirable property for any  $\langle G, F, \sigma \rangle$ . With that in hand, we prove  $\text{popt} \leq O(\log^2 n)\text{lopt}$ .

Given  $G$ , a list of demands  $\sigma = \langle (s_1, t_1), \dots, (s_k, t_k) \rangle$ , and a forest  $F \subseteq G$ :

**Definition 5.1.** Let  $Q = \langle Q_1, Q_2, Q_3, \dots, Q_k \rangle$  be a list of  $k$  subgraphs of  $F$ . We say  $Q$  is a *connective list of subgraphs* for  $\langle G, F, \sigma \rangle$  iff for every  $1 \leq i \leq k$  there exists no cut disjoint from  $Q_i$  that separates  $s_i$  from  $t_i$ , but does not separate any  $s_j$  from  $t_j$  for  $j < i$ .

The intuition behind the definition of connective subgraphs is the following: If  $Q$  is a connective list of subgraphs for an instance  $\langle G, F, \sigma \rangle$  then for every  $i$  we are guaranteed that the union of all subgraphs  $\cup_{j=1}^i Q_j$  connects  $s_i$  to  $t_i$ . In Lemma 5.2 we show for every  $\langle G, F, \sigma \rangle$ , there exists a connective list of subgraphs for  $\langle G, F, \sigma \rangle$ , such that each edge of  $F$  appears in at most  $O(\log^2 n)$  subgraphs of  $Q$ .

**Lemma 5.2.** *Let  $G$  be a graph and  $F$  be a forest in  $G$ . If  $\sigma$  is a collection of  $k$  demands  $\langle (s_1, t_1), \dots, (s_k, t_k) \rangle$ , then there exists a connective list of subgraphs  $Q = \langle Q_1, Q_2, \dots, Q_k \rangle$  for  $\langle G, F, \sigma \rangle$  such that every edge of  $F$  appears in at most  $3 \log^2 |V(F)|$  number of  $Q_i$ 's.*

**Proof.** Here we give a sketch of the proof of lemma; we refer the reader to the full version for detailed proofs. We first prove a cost-minimization variant of the lemma. Consider an arbitrary weight vector  $\hat{w} : F \rightarrow \mathbb{R}^{\geq 0}$ . We argue that there is a connective list  $Q$ , such that  $\sum_i \hat{w}(Q_i) \leq O(\log^2 n) \hat{w}(F)$ . Let  $\hat{H}_i = (V, F \cup R_\sigma(i), \hat{w}_i)$  denote a weighted graph for which  $\hat{w}_i(e) = \hat{w}(e)$  for  $e \in F$ , and  $\hat{w}_i(e) = 0$  for  $e \in R_\sigma(i)$ . Now we note that there is no cost-sharing among  $Q_i$ 's in the goal  $\sum_i \hat{w}(Q_i)$ . Therefore the optimal choice for  $Q_i$  corresponds to the minimum-weight  $(s_i, t_i)$ -path in  $\hat{H}_{i-1}$ . Hence, we need to analyze the cost of these greedy choices.

Awerbuch et al. [AAB96] showed that the greedy algorithm is indeed  $O(\log^2 n)$ -competitive for the edge-weighted Steiner forest problem. The standard greedy algorithm is slightly different from the greedy process we discussed above. In the greedy algorithm of Awerbuch et al., at time step  $i$  we choose a minimum-cost  $(s_i, t_i)$ -path in a graph in which there is a zero-cost edge between *any* pair of vertices in



the same connected component of the current solution; not just the  $(s_j, t_j)$  pairs of the previous demands. However, in their analysis they only use the zero-cost edges among the terminals of a previous demand. This is indeed not surprising since we hardly have any control on the greedy choices other than the fact that they satisfy the demands. Therefore the following claim follows from the result of Awebuch et al.<sup>2</sup>:

**Claim 5.1** (implicitly proven in Theorem 2.1 of [AAB96]). *For any weight function  $\hat{w}$  defined over  $F$ , there exists a connective list  $Q$  for which*

$$\sum_i \hat{w}(Q_i) \leq O(\log^2 n) \hat{w}(F)$$

.

However, Claim 5.1 is not enough for us. We need a solution in which every edge is used at most  $O(\log^2 n)$  times, not just in an amortized sense. Indeed we can show that since there is a solution for *every* weight function, we can have a *fractional* connective list  $Q$  in which every edge is used (fractionally) at most  $O(\log^2 n)$  times. This implies that we have a fractional connective list. Finally, we provide a rounding argument which obtains an integral connective list by losing only a constant factor; which completes the proof of lemma.

□

Finally, we can leverage Lemma 5.2 to show  $\text{popt} \leq O(\log^2 n) \text{lopt}$ . This shows

---

<sup>2</sup>There is also a lower bound of  $\Omega(\log n)$  for the competitive ratio of the greedy algorithm. Closing the gap between this lower bound and the upper bound of  $O(\log^2 n)$  for EW Steiner forest is an important open problem.

we can use [PC-IP](#) as an online mixed packing/covering IP to obtain an online solution for ONLINE EDGE-WEIGHTED DEGREE-BOUNDED STEINER FOREST losing a factor of  $O(\log^2 n)$ .

In Section [5.4](#) we show this formulation is an online bounded frequency mixed packing/covering IP, thus we leverage our technique for such IPs to obtain a polylogarithmic-competitive algorithm for online EW-DB-SF.

### 5.3 Online Bounded Frequency Mixed Packing/Covering IPs

In this section we consider bounded frequency online mixed packing and covering integer programs. For every online mixed packing and covering IP with covering frequency  $k$ , we provide an online algorithm that violates each packing constraint by at most a factor of  $O(k \log m)$ , where  $m$  is the number of packing constraints. We note that this bound is independent of the number of variables, the number of covering constraints, and the coefficients of the mixed packing and covering program. Moreover the algorithm is for integer programs, which implies obtaining an integer solution does not rely on (online) rounding.

In particular we prove there exists an online  $O(k \log m)$ -competitive algorithm for any mixed packing and covering IP such that every variable has covering frequency at most  $k$ , where the covering frequency of a variable  $x_r$  is the number of covering constraints with a non-zero coefficient for  $x_r$ .

We assume that all variables are binary. One can see this is without loss of generality as long as we know every variable  $x_r \in \{1, 2, 3, \dots, 2^l\}$ . Since we can

replace  $x_r$  by  $l$  variables  $y_r^1, \dots, y_r^l$  denoting the digits of  $x_r$  and adjust coefficients accordingly. Furthermore, for now we assume that the optimal solution for the given mixed packing and covering program is 1. In Theorem 5.4 we prove that we can use a doubling technique to provide an  $O(k \log m)$ -competitive solution for online bounded frequency mixed packing and covering programs with any optimal solution. The algorithm is as follows. We maintain a family of subsets  $\mathcal{S}$ . Initially  $\mathcal{S} = \emptyset$ . Let  $\mathcal{S}(j)$  denote  $\mathcal{S}$  at arrival of  $C_{j+1}$ . For each covering constraint  $C_{j+1}$ , we find a subset of variables  $S_{j+1}$  and add  $S_{j+1}$  to  $\mathcal{S}$ . We find  $S_{j+1}$  in the following way. For each set of variables  $S$ , we define a cost function  $\tau_S(\mathcal{S}(j))$  according to our current  $\mathcal{S}$  at arrival of  $C_{j+1}$ . We find a set  $S_{j+1}$  that satisfies  $C_{j+1}$  and minimizes  $\tau_S(\mathcal{S}(j))$ . More precisely we say a set of variables  $S$  satisfies  $C_{j+1}$  if

- $\sum_{x_r \in S} C_{j+1,r} x_r \geq 1$ , where  $C_{j+1,r}$  denotes the coefficient of  $C_{j+1}$  for  $x_r$ .
- For each packing constraint  $P_i$ ,  $\sum_{x_r \in S} \frac{1}{k} P_{ir} \leq 1$ .

Now we add  $S_{j+1}$  to  $\mathcal{S}$  and for every  $x_r \in S_{j+1}$ , we set  $x_r = 1$ . We note that there always exists a set  $S$  that satisfies  $C_{j+1}$ , since we assume there exists an optimal solution with value 1. Setting  $S$  to be the set of all variables with value one in an optimal solution which have non-zero coefficient in  $C_{j+1}$ , satisfies  $C_{j+1}$ . It only remains to define  $\tau_S(\mathcal{S}(j))$ . But before that we need to define  $\Delta_i(S)$  and  $F_i(\mathcal{S}(j))$ . For packing constraint  $P_i$  and subset of variables  $S$ , we define  $\Delta_i(S)$  as  $\Delta_i(S) := \sum_{x_r \in S} \frac{1}{k} P_{ir}$ . For packing constraint  $P_i$  and  $\mathcal{S}(j)$ , let

$$F_i(\mathcal{S}(j)) := \sum_{S \in \mathcal{S}(j)} \Delta_i(S) . \tag{5.7}$$

Now let  $\tau_S(\mathcal{S}(j)) = \sum_{i=1}^m \rho^{F_i(\mathcal{S}(j)) + \Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))}$ , where  $\rho > 1$  is a constant to be defined later.

---

**Algorithm 2**

---

**Input:** Packing constraints  $P$ , and an online stream of covering constraints  $C_1, C_2, \dots$

**Output:** A feasible solution for online bounded frequency mixed packing/covering.

**Offline Process:**

1: Initialize  $\mathcal{S} \leftarrow \emptyset$ .

**Online Scheme; assuming a covering constraint  $C_{j+1}$  is arrived:**

1:  $S_{j+1} \leftarrow \arg \min_S \{\tau_S(\mathcal{S}(j)) \mid S \text{ satisfies } C_{j+1}\}$ .

2: **for all**  $x_r \in S_{j+1}$  **do**

3:  $x_r \leftarrow 1$ .

---

Let  $\mathbf{x}^*$  be an optimal solution, and  $\mathbf{x}^*(j)$  denote its values at online stage  $j$ .

We define  $G_i(j)$  as

$$G_i(j) := \sum_{l=1}^j \sum_{r: C_{lr} > 0} \frac{1}{k} x_r^* P_{lr} . \quad (5.8)$$

Now we define a potential function  $\Phi_j$  for online stage  $j$ .

$$\Phi_j = \sum_{i=1}^m \rho^{F_i(\mathcal{S}(j))} (\gamma - G_i(j)) , \quad (5.9)$$

where  $\rho, \gamma > 1$  are constants to be defined later.

**Lemma 5.3.** *There exist constants  $\rho$  and  $\gamma$ , such that  $\Phi_j$  is non-increasing.*

**Proof.** We find  $\rho$  and  $\gamma$  such that  $\Phi_{j+1} - \Phi_j \leq 0$ . By the definition of  $\Phi_j$ ,

$$\Phi_{j+1} - \Phi_j = \sum_{i=1}^m \rho^{F_i(\mathcal{S}(j+1))} (\gamma - G_i(j+1)) - \rho^{F_i(\mathcal{S}(j))} (\gamma - G_i(j)) . \quad (5.10)$$

By Equation (5.7),  $\rho^{F_i(\mathcal{S}(j+1))} - \rho^{F_i(\mathcal{S}(j))} = \rho^{F_i(\mathcal{S}(j))+\Delta_i(\mathcal{S})} - \rho^{F_i(\mathcal{S}(j))}$ . Moreover by Equation (5.8),  $(\gamma - G_i(j+1)) - (\gamma - G_i(j)) = -\sum_{r:C_{j+1,r}>0} \frac{1}{k} x_r^* P_{ir}$ . For simplicity of notation we define  $B_i(j+1) := \sum_{r:C_{j+1,r}>0} \frac{1}{k} x_r^* P_{ir}$ . Thus we can write Equation (5.10) as:

$$\begin{aligned} \Phi_{j+1} - \Phi_j &= \sum_{i=1}^m \rho^{F_i(\mathcal{S}(j+1))} (\gamma - G_i(j) - B_i(j+1)) - \rho^{F_i(\mathcal{S}(j))} (\gamma - G_i(j)) \quad (5.11) \\ &= \sum_{i=1}^m (\gamma - G_i(j)) (\rho^{F_i(\mathcal{S}(j))+\Delta_i(\mathcal{S})} - \rho^{F_i(\mathcal{S}(j))}) - \rho^{F_i(\mathcal{S}(j+1))} B_i(j+1) \quad \text{Since } G_i(j) \geq 0 \\ &\leq \sum_{i=1}^m \gamma (\rho^{F_i(\mathcal{S}(j))+\Delta_i(\mathcal{S})} - \rho^{F_i(\mathcal{S}(j))}) - \rho^{F_i(\mathcal{S}(j+1))} B_i(j+1) \quad F_i(\mathcal{S}(j+1)) \geq F_i(\mathcal{S}(j)) \\ &\leq \sum_{i=1}^m \gamma (\rho^{F_i(\mathcal{S}(j))+\Delta_i(\mathcal{S})} - \rho^{F_i(\mathcal{S}(j))}) - \rho^{F_i(\mathcal{S}(j))} B_i(j+1) . \end{aligned}$$

Now according to the algorithm for each subset of variables  $S'$  such that  $\sum_{x_r \in S'} C_{j+1}(x_r) \geq 1$ , either  $\tau_{\mathcal{S}}(\mathcal{S}(j)) \leq \tau_{S'}(\mathcal{S}(j))$  or there exists a packing constraint  $P_i$  such that  $\Delta_i(S') > 1$ . In  $B_i(j+1)$ , we are considering variables  $x_r$  such that  $x_r^* = 1$ , thus for every  $P_i$ ,  $B_i(j+1) \leq 1$ . Therefore setting  $S'$  to be the set of variables  $x_r$  such that  $x_r^* = 1$  and  $C_{j+1,r} > 0$ , we have  $\tau_{\mathcal{S}}(\mathcal{S}(j)) \leq \tau_{S'}(\mathcal{S}(j))$ . Thus  $\sum_{i=1}^m \rho^{F_i(\mathcal{S}(j))+\Delta_i(\mathcal{S})} - \rho^{F_i(\mathcal{S}(j))} \leq \sum_{i=1}^m \rho^{F_i(\mathcal{S}(j))+B_i(j+1)} - \rho^{F_i(\mathcal{S}(j))}$ . Therefore we can rewrite Inequality (5.11) as

$$\begin{aligned} \Phi_{j+1} - \Phi_j &\leq \sum_{i=1}^m \gamma (\rho^{F_i(\mathcal{S}(j))+B_i(j+1)} - \rho^{F_i(\mathcal{S}(j))}) - \rho^{F_i(\mathcal{S}(j))} B_i(j+1) \quad (5.12) \\ &= \sum_{i=1}^m \rho^{F_i(\mathcal{S}(j))} (\gamma \rho^{B_i(j+1)} - \gamma - B_i(j+1)) . \end{aligned}$$

We would like to find  $\rho$  and  $\gamma$  such that  $\Phi_j$  is non-increasing. We find  $\rho$  and  $\gamma$  such that for each packing constraint  $P_i$ ,  $\gamma\rho^{B_i(j+1)} - \gamma - B_i(j+1) \leq 0$ . Thus

$$\gamma\rho^{B_i(j+1)} - \gamma \leq B_i(j+1) \quad \text{Since } 0 \leq B_i(j+1) \leq 1 \quad (5.13)$$

$$\gamma\rho B_i(j+1) - \gamma \leq B_i(j+1) \quad \text{By simplifying} \quad (5.14)$$

$$\rho \leq 1 + 1/\gamma . \quad (5.15)$$

Thus if we set  $\rho \leq 1 + 1/\gamma$ ,  $\Phi_j$  is non-increasing, as desired.  $\square$

Now we prove Algorithm 2 obtains a solution of at most  $O(k \log m)$ .

**Lemma 5.4.** *Given an online bounded frequency mixed packing covering IP with optimal value 1, there exists a deterministic integral algorithm with competitive ratio  $O(k \log m)$ , where  $m$  is the number of packing constraints and  $k$  is the covering frequency of the IP.*

**Proof.** By Lemma 5.3 for each stage  $j$ ,  $\Phi_{j+1} \leq \Phi_j$ . Therefore  $\Phi_j \leq \Phi_0 = \gamma m$ .

Thus for each packing constraint  $P_i$ ,

$$\rho^{F_i(\mathcal{S}(j))}(\gamma - G_i(j)) \leq \gamma m . \quad (5.16)$$

Thus,

$$\rho^{F_i(\mathcal{S}(j))} \leq \frac{\gamma m}{(\gamma - G_i(j))} \leq \frac{\gamma m}{\gamma - 1} . \quad \text{Since } G_i(j) \leq 1 \quad (5.17)$$

Thus we can conclude

$$F_i(\mathcal{S}(j)) \in O(\log m) . \quad (5.18)$$

By definition of  $F_i(\mathcal{S}(j))$ ,  $F_i(\mathcal{S}(j)) = \sum_{S \in \mathcal{S}(j)} \Delta_i(S) = \sum_{S \in \mathcal{S}(j)} \sum_{x_r \in S} \frac{1}{k} P_{ir}$ . Since each variable  $x_r$  is present in at most  $k$  sets,  $\frac{1}{k} P_i \cdot \mathbf{x}(j) \leq F_i(\mathcal{S}(j))$ . Thus by Inequality (5.18)  $P_i \mathbf{x}(j) \in O(k \log m)$ , which completes the proof.  $\square$

Finally we prove there exists an online  $O(k \log m)$ -competitive algorithm for bounded frequency online mixed packing and covering integer programs with any optimal value.

**Theorem 5.4.** *Given an instance of the online mixed packing/covering IP, there exists a deterministic integral algorithm with competitive ratio  $O(k \log m)$ , where  $m$  is the number of packing constraints and  $k$  is the covering frequency of the IP.*

## 5.4 Putting Everything Together

In this section we consider the online mixed packing/covering formulation discussed in Section 5.2 for ONLINE EDGE-WEIGHTED DEGREE-BOUNDED STEINER FOREST [PC\\_IP](#). In this section we show this formulation is an online bounded frequency mixed packing/covering IP. Therefore we use our techniques discussed in Section 5.3 to obtain a polylogarithmic-competitive algorithm for ONLINE EDGE-WEIGHTED DEGREE-BOUNDED STEINER FOREST.

First we assume we are given the optimal weight  $w_{\text{opt}}$  as well as degree bounds. We can obtain the following theorem.

**Theorem 5.5.** *Given the optimal weight  $w_{\text{opt}}$ , there exists an online deterministic algorithm which finds a subgraph with total weight at most  $O(\log^3 n)w_{\text{opt}}$  while the degree bound of a vertex is violated by at most a factor of  $O(\log^3 n)$ .*

**Proof.** By Lemma 5.1, given a feasible online solution for [PC\\_IP](#) with violation  $\alpha$ , we can provide an online solution for [SF\\_IP](#) with violation  $\alpha$ . Moreover, in

Section 5.2 we show that  $\text{popt} \leq O(\log^2 n)\text{lopt}$ . Thus given an online solution for [PC\\_IP](#) with competitive ratio  $f$ , there exists an  $O(f \log n)$ -competitive algorithm for ONLINE DEGREE-BOUNDED STEINER FOREST. We note that in [PC\\_IP](#) we know the packing constraints in advance. In addition every variable  $x_H^i$  has non-zero coefficient only in the covering constraint corresponding to connectivity of the  $i$ -th demand endpoints, i.e. the covering frequency of every variable is 1. Therefore by Theorem 5.4 there exists an online  $O(\log m)$ -competitive solution for [PC\\_IP](#), where  $m$  is the number of packing constraints, which is  $n + 1$ . Thus there exists an online  $O(\log^3 n)$ -competitive algorithm for ONLINE DEGREE-BOUNDED STEINER FOREST. This means the violation for both degree bounds and weight is of  $O(\log^3 n)$ .  $\square$

Finally if  $w_{\text{opt}}$  is not given, we show that by applying standard doubling techniques one can prove Theorem 5.1 using the result shown above.



## Chapter 6: Beating $1-1/e$ for Ordered Prophets

### 6.1 Introduction

Online auctions play a major role in modern markets. In online markets, information about customers and goods is revealed over time. Irrevocable decisions are made at certain discrete times, such as when a customer arrives to the market. One of the fundamental and basic tools to model this scenario is the *prophet inequality* and its variants.

In a prophet inequality instance we are given a sequence of distributions. Iteratively, we draw a value from one of the distributions, based on a predefined order. In each step we face two choices, either we accept the value and stop, or we reject the value and move to the next distribution. The goal in this problem is to maximize the expected value of the item selected. We say an algorithm for a prophet inequality instance is an  $\alpha$ -approximation, for  $\alpha \leq 1$ , if the expectation of the value picked by the algorithm is at least  $\alpha$  times that of an optimum solution which knows all of the values in advance.

Prophet inequalities were first studied in the 1970's by Krengel and Sucheston [Ken87, KS77, KS78]. Hajiaghayi, Kleinberg and Sandholm [HKS07] studied the relation between online auctions and prophet inequalities. In particular they

showed that algorithms used in the derivation of prophet inequalities can be reinterpreted as truthful mechanisms for online auctions. Later Chawla, Hartline, Malec, and Sivan [CHMS10a] used prophet inequalities to design sequential posted price mechanisms whose revenue approximates that of the Bayesian optimal mechanism.

In the classical definition of the prophet inequality, the values can be drawn from their distributions in an arbitrary (a.k.a. *adversarial* or *worst*) order. Assuming an adversarial order, the problem has a 0.5 approximation algorithm which is tight. Recently, Yan [Yan11] considered a relaxed version of this problem in which the algorithm designer is allowed to pick the order of distributions (a.k.a. *best order*), and provided a  $1 - \frac{1}{e}$  approximation algorithm for this problem. Later, Esfandiari, Hajiaghayi, Liaghat and Monemizadeh [EHLM15] showed that there exists a  $1 - \frac{1}{e}$  approximation algorithm even when the distributions arrive in a *random order*. Both results provided by Yan and Esfandiari et al. are not tight.

In this work we consider prophet inequalities in both best order and random order settings and take steps towards providing tight approximation algorithms for these problems. Particularly, we consider this problem assuming a large market assumption (i.e. we have several copies of each distribution). Indeed, the large market assumption is well-motivated in this context [BJN07, DH09, EKM15, MSVV07, MGZ12].

### 6.1.1 Our Contribution

First we consider the prophet inequality on a set of *identical and independent distributions* (*iid*). The prophet inequality on iid distributions has been previously studied by Hill and Kertz [HK82] in the 1980's. Hill and Kertz provided an algorithm based on complicated recursive functions. They proved a theoretical bound of  $1 - \frac{1}{e}$  on the approximation factor of their algorithm, and used a computer program to show that their algorithm is a 0.745-approximation when the number of distributions is  $n = 10000$ . They conjectured that the best approximation factor for arbitrarily large  $n$  is  $\frac{1}{1+1/e} \simeq 0.731$ . This conjecture remained open for more than three decades.

In this thesis we present a simple threshold-based algorithm for the prophet inequality with  $n$  iid distributions. Using a nontrivial and novel approach we show that our algorithm is a 0.738-approximation algorithm for large enough  $n$ , beating the bound of  $\frac{1}{1+1/e}$  conjectured by Hill and Kertz. This is the first algorithm which is theoretically proved to have an approximation factor better than  $1 - \frac{1}{e}$  for this problem. Indeed, beating the  $1 - \frac{1}{e}$  barrier is a substantial work in this area [FV06, FMMM09]. The following theorem states our claim formally.

**Theorem 6.1.** *There exists a constant number  $n_0$  such that for every  $n \geq n_0$ , there exists a 0.738-approximation algorithm for any prophet inequality instance with  $n$  iid distributions.*

Next, we extend our results to support different distributions. However, we assume that we have several copies of each distribution. This can be reinterpreted as a large market assumption. We say a multiset of independent distributions

$\{F_1, \dots, F_n\}$  is *m-frequent* if for each distribution  $F_i$  in this multiset there are at least  $m$  copies of this distribution in the multiset. We show that by allowing the algorithm to pick the order of the distributions, there exists a 0.738-approximation algorithm for any prophet inequality instance on a set of  $m$ -frequent distributions, for large enough  $m$ . The following theorem states this fact formally.

**Theorem 6.2.** *There exists a constant number  $m_0$  such that there exists a 0.738-approximation best order algorithm for any prophet inequality instance on a set of  $m_0$ -frequent distributions.*

Our next theorem shows that even in the random order setting one can achieve a 0.738-approximation algorithm on  $m$ -frequent distributions, for large enough  $m$ .

**Theorem 6.3.** *There exists a constant number  $c_0$  such that there exists a 0.738-approximation random order algorithm for any prophet inequality instance on a set of  $(c_0 \log(n))$ -frequent distributions.*

To conclude the presentation of our results we show that it is not possible to extend our results to the worst order setting. The following theorem states this fact formally.

**Theorem 6.4.** *For any arbitrary  $m$ , there is a prophet inequality instance on a set of  $m$ -frequent distributions such that the instance does not admit any  $0.5 + \epsilon$ -approximation worst order algorithm.*

## 6.1.2 Applications in Mechanism Design

The prophet inequality has numerous applications in mechanism design and optimal search theory, so our improved prophet inequality for  $m$ -frequent distributions has applications in those areas as well. By way of illustration, we present here an application to optimal search theory. In Weitzman’s [Wei79] “box problem”, there are  $n$  boxes containing independent random prizes,  $v_1, \dots, v_n$ , whose distributions are not necessarily identical. The cost of opening box  $i$  is  $c_i \geq 0$ . A decision maker is allowed to open any number of boxes, after which she is allowed to claim the largest prize among the open boxes. The costs of the boxes, and the distributions of the prizes inside, are initially known to the decision maker, but the value  $v_i$  itself is only revealed when box  $i$  is opened. A search policy is a (potentially adaptive) rule for deciding which box to open next—or whether to stop—given the set of boxes that have already been opened and the values of the prizes inside. Weitzman [Wei79] derived the structure of the optimal search policy, which turns out to be wonderfully simple: one computes an “option value”  $\sigma_i$  for each box  $i$ , satisfying the equation  $E[\max\{0, v_i - \sigma_i\}] = c_i$ . Boxes are opened in order of decreasing  $\sigma_i$  until there is some open box  $i$  such that  $v_i > \sigma_j$  for every remaining closed box  $j$ , then the policy stops. Kleinberg, Waggoner, and Weyl [KWW16] presented an alternative proof of this result which works by relating any instance of the box problem to a modified instance in which opening boxes is cost-free, but the prize in box  $i$  is  $\min\{v_i, \sigma_i\}$  rather than  $v_i$ . The proof shows that when we run any policy on the modified instance, its net value (prize minus combined cost) weakly improves, and that the net

value is preserved if the policy is *non-exposed*, meaning that whenever it opens a box with  $v_i > \sigma_i$ , it always claims the prize inside.

An interesting variant of the box problem arises if one constrains the decision maker, upon stopping, to choose the prize in the most recently opened box, rather than the maximum prize observed thus far. In other words, upon opening box  $i$  the decision maker must irrevocably decide whether to end the search and claim prize  $v_i$ , or continue the search and relinquish  $v_i$ . Let us call this variant the *impatient box problem*. It could be interpreted as modeling, for example, the decision problem that an employer faces when scheduling a sequence of costly job interviews in a labor market where hiring decisions must be made immediately after the interview. The factor  $1 - \frac{1}{e}$  prophet inequality of Yan and Esfandiari et al. implies that if the decision maker is allowed to choose the order in which to inspect boxes (or even if a random order is used), the net value of the optimal impatient box problem policy is at least  $1 - \frac{1}{e}$  times the net value of the optimal policy for the corresponding instance of the original (non-impatient) box problem; for the proof of this implication, see Corollary 3 and Remark 1 in [KWW16]. A consequence of Theorem 6.2 above is that this ratio improves to 0.738 if the instance of the impatient box problem contains sufficiently many copies of each type of box.

Our results also have applications to a recent line of work that employs prophet inequalities to design posted-price mechanisms. In the standard posted-price setup, a seller has a collection of resources to distribute among  $n$  buyers. The buyers' values are drawn independently from distributions that are known in advance to the seller. The seller can use this distributional knowledge to set a (possibly adaptive) price on

the goods for sale. Buyers then arrive sequentially and make utility-maximizing purchases. Hajiaghayi et al. [HKS07] noted the close connection between this problem and the prophet inequality, with the price corresponding to a choice of threshold. This has immediate implications for designing prices for welfare maximization, and one can additionally obtain bounds for revenue by applying the prophet inequality to virtual welfare [CHK07, CHMS10a]. There has subsequently been a significant line of work extending this connection to derive posted-price mechanisms for broader classes of allocation problems, such as matroid constraints [KW12], multi-item auctions [Ala11, CHMS10a] and combinatorial auctions [FGL15]. The result of Yan and Esfandiari et al. [EHL15] implies that for the original case of a single item for sale, if the seller is allowed to choose the order in which the buyers arrive (or if they can be assumed to arrive in random order), then a posted-price mechanism can obtain expected welfare that is at least  $1 - \frac{1}{e}$  times the expected welfare of the optimal assignment. Theorem 6.2 implies that this ratio improves to 0.738 if the pool of buyers contains sufficiently many individuals whose values are drawn from the same distribution.

### 6.1.3 Other Related Work

The first generalization of the prophet inequality is the *multiple-choice prophet inequality* [Ken87, Ken85, Ker86]. In the multiple-choice prophet inequality we are allowed to pick  $k$  values, and the goal is to maximize the total sum of picked values. Alaei [Ala11] gives an almost tight  $(1 - \frac{1}{\sqrt{k+3}})$ -approximation algorithm for the  $k$ -

choice prophet inequality (the lower bound is proved in Hajiaghayi, Kleinberg, and Sandholm [HKS07]).

Prophet inequalities have been studied under complicated combinatorial structures such as matroid, polymatroid, and matching. Kleinberg and Weinberg [KW12] consider matroid prophet inequalities, in which the set of selected values should be an independent set of a predefined matroid. They give a tight 0.5-approximation worst order algorithm for this problem. Later, Dütting and Kleinberg extended this result to polymatroids [DK15]. More recently, Ehsani, Hajiaghayi, Kesselheim, and Singla [EHKS17] present a  $1 - 1/e$ -approximation algorithm for prophet secretary in matroid and combinatorial auction settings.

Alaei, Hajiaghayi, and Liaghat study matching prophet inequalities [AHL<sup>+</sup>11, AHL13, AHL12]. They extend the multiple-choice prophet inequality and give an almost tight  $(1 - \frac{1}{\sqrt{k+3}})$ -approximation worst order algorithm for any matching prophet inequality instance, where  $k$  is the minimum capacity of a vertex.

Rubinfeld considers the prophet inequalities restricted to an arbitrary downward-closed set system [Rub16]. He provides an  $O(\log n \log r)$ -approximation algorithm for this problem, where  $n$  is the number of distributions and  $r$  is the size of the largest feasible set. Babaioff, Immorlica and Kleinberg show a lower bound of  $\Omega(\frac{\log n}{\log \log n})$  for this problem [BIK07]. Prophet inequalities has also been studied restricted to independent set in graphs [GHK<sup>+</sup>14].



## 6.2 IID Distributions

In this section we give a 0.738-approximation algorithm for prophet inequality with iid items. Let us begin with some definitions. Assume that  $X_1, \dots, X_n$  are iid random variables with common distribution function  $F$ . For simplicity, assume that  $F$  is continuous and strictly increasing on a subinterval of  $\mathcal{R}^{\geq 0}$ . An algorithm based on a sequence of thresholds  $\theta_1, \dots, \theta_n$  is the one that selects the first item  $k$  such that  $X_k \geq \theta_k$ .

Let  $\tau$  denote the stopping time of this algorithm, where  $\tau$  is  $n + 1$  when the algorithm selects no item. For simplicity suppose  $X_{n+1}$  is a zero random variable. The approximation factor of an algorithm based on  $\theta_1, \dots, \theta_n$  is defined as  $E[X_\tau]/E[\max X_i]$ . This factor captures the ratio between what a player achieves in expectation by acting based on these thresholds and what a prophet achieves in expectation by knowing all  $X_i$ 's in advance and taking the maximum of them.

In Algorithm 3 we presents a simple oblivious algorithm for every  $n$  and distribution function  $F$ . Theorem 6.5 proves that this algorithm is at least 0.738-approximation for large enough number of items.

---

**Algorithm 3**

---

**Input:**  $n$  iid items with distribution function  $F$ .

---

- 1: Set  $a$  to 1.306 (root of  $\cos(a) - \sin(a)/a - 1$ ).
  - 2: Set  $\theta_i = F^{-1}(\cos(ai/n)/\cos(a(i-1)/n))$ .
  - 3: Pick the first item  $i$  for which  $X_i \geq \theta_i$ .
-

**Theorem 6.5.** *For every  $\epsilon > 0$  there exists a number  $n_\epsilon$  (a function of  $\epsilon$  and independent of  $n$ ) such that for every  $n \geq n_\epsilon$  Algorithm 3 for  $n$  items is at least  $(1 - \epsilon)\alpha$ -approximation where  $\alpha = 1 - \cos(a) \approx 0.7388$ .*

In the following we walk you through the steps of the design of Algorithm 3 and provide a proof for Theorem 6.5. For a given sequence of thresholds let  $q_0, q_1, \dots, q_n$  denote the probability of the algorithm not choosing any of the first items. More specifically, let  $q_i = \Pr[\tau > i]$  for every  $0 \leq i \leq n$ . Knowing the thresholds  $\theta_1, \dots, \theta_n$  one can find this sequence by starting from  $q_0 = 1$  and computing the rest using  $q_i = q_{i-1}F(\theta_i)$ . Inversely, one can simply find the thresholds from  $q_1, \dots, q_n$  using  $\theta_i = F^{-1}(q_i/q_{i-1})$ . Hence, we focus the design of our algorithm on finding the sequence  $q_1, \dots, q_n$ . To this end, we aim to find a continuous function  $h : [0, 1] \rightarrow [0, 1]$  with  $h(0) = 1$  such that by setting  $q_i = h(i/n)$  we can achieve our desired set of thresholds.

Note that such a function  $h$  has to meet certain requirements. For instance, it has to be strictly decreasing, because at every step the algorithm picks an item with some positive probability, therefore  $h(i/n) = q_i = \Pr[\tau > i]$  is smaller for larger  $i$ . In the following we define a class of functions which has two additional properties. We prove that these properties can be useful in designing a useful threshold algorithm.

**Definition 6.1.** *A continuous and strictly decreasing function  $h : [0, 1] \rightarrow [0, 1]$  with  $h(0) = 1$  is a threshold function if it has the following two properties:*

*i.  $h$  is a strictly concave function.*

*ii. For every  $\epsilon > 0$  there exists some  $\delta_0 \leq \epsilon$  such that for every  $\delta \leq \delta_0$ ,  $\epsilon + \delta \leq$*

$s \leq 1$ :

$$h'(s - \delta)/h(s - \delta) \leq (1 - \epsilon)h'(s)/h(s) .$$

As shown in the following lemma, the first property leads to a decreasing sequence of thresholds. Also, we exploit the second property to show that the approximation factor of  $h$  improves by increasing the number of items.

**Lemma 6.1.** *If  $h$  is a threshold function, then the sequence of thresholds  $\theta_1, \dots, \theta_n$  achieved from  $h$  is decreasing.*

**Proof.** For every  $1 \leq i \leq n$  we have  $\theta_i = F^{-1}(q_i/q_{i-1})$ . Since every  $q_i = h(i/n)$ , we have  $\theta_i = F^{-1}(\frac{h(i/n)}{h((i-1)/n)})$ . Note that  $F$  is a strictly increasing function, therefore having  $\theta_i > \theta_{i+1}$  requires  $\frac{h(i/n)}{h((i-1)/n)} > \frac{h((i+1)/n)}{h(i/n)}$ . For simplicity let  $x = i/n$  and  $\delta = 1/n$ . From the first property of threshold functions we have:

$$h(x) > \frac{h(x + \delta) + h(x - \delta)}{2} .$$

By raising both sides to the power of 2, and subtracting  $(h(x + \delta)/2 - h(x - \delta)/2)^2$  from each side we have:

$$\begin{aligned} h(x)^2 - \left( \frac{h(x + \delta) - h(x - \delta)}{2} \right)^2 &> \left( \frac{h(x + \delta) + h(x - \delta)}{2} \right)^2 - \left( \frac{h(x + \delta) - h(x - \delta)}{2} \right)^2 \\ &= h(x + \delta)h(x - \delta) . \end{aligned}$$

Therefore  $h(x)^2 > h(x + \delta)h(x - \delta)$ , which means  $h(x)/h(x - \delta) > h(x + \delta)/h(x)$  and the proof is complete.  $\square$

Next, we define a class of functions and prove for every function of this class that its approximation factor approaches  $\alpha$  for a large enough  $n$ . This enables us to narrow down our search for a useful function  $h$ .

**Definition 6.2.** A threshold function  $h$  is  $\alpha$ -strong if it has the following properties:

i.  $h(1) \leq 1 - \alpha$ .

ii.  $\int_0^1 h(r)dr \geq \alpha$ .

iii.  $\forall 0 \leq s \leq 1 : 1 - h(s) - h'(s)/h(s) \int_s^1 h(r)dr \geq \alpha(1 - \exp(h'(s)/h(s)))$  .

The following theorem formally states the mentioned claim for  $\alpha$ -strong functions.

**Theorem 6.6.** If  $h$  is an  $\alpha$ -strong function, then for every  $\epsilon > 0$  there exists an  $n_\epsilon$  such that for every  $n \geq n_\epsilon$  the threshold algorithm that acts based on  $h$  is at least  $(1 - \epsilon)\alpha$ -approximation on  $n$  iid items.

**Proof.** Let  $OPT$  be a random variable that denotes the optimum solution and  $ALG$  be a random variable that denotes the value picked by the algorithm. We can write the expectation of  $OPT$  as

$$\mathbb{E}[OPT] = \int_0^\infty Pr[\max X_i \geq x]dx \quad (6.1)$$

Similarly the expectation of  $ALG$  is

$$\mathbb{E}[ALG] = \int_0^\infty Pr[X_\tau \geq x]dx \quad (6.2)$$

The main idea behind the proof is to show for  $\alpha$ -strong functions that the integrand in (6.2) is an approximation of the integrand in (6.1) for every non-negative value of  $x$ . In particular, for every  $\epsilon$  there exists some  $n_\epsilon$  such that for every  $n \geq n_\epsilon$  the second integrand is at least  $(1 - \epsilon)\alpha$  times the first integrand and this proves the theorem.

Let us begin with finding an upper bound for the integrand in (6.1). Let  $G(x) = 1 - F(x)$  for every  $x \in \mathcal{R}^{\geq 0}$ . The following lemma gives an upper bound for  $Pr[\max X_i \geq x]$  based on  $G(x)$  and  $n$ .

**Lemma 6.2.** *For every  $\epsilon > 0$  there exists an  $n_\epsilon$  such that for every  $n \geq n_\epsilon$  the following inequality holds :*

$$Pr[\max X_i \geq x] \leq \frac{1 - \exp(-nG(x))}{1 - \epsilon} .$$

Lemma 6.2 gives us an upper bound on  $Pr[\max X_i \geq x]$ . Now we aim to find a lower bound for  $Pr[X_\tau \geq x]$ . Through these two bounds we are able to find a lower bound on the approximation factor of the algorithm.

In Lemma 6.1 we showed that the thresholds are decreasing. Hence for an  $x \in \mathcal{R}^{\geq 0}$ , if  $x < \theta_n$  then  $Pr[X_\tau \geq x]$  is equal to  $Pr[X_\tau \geq \theta_n]$  because the algorithm never selects an item below that value. Moreover,  $Pr[X_\tau \geq \theta_n]$  is equal to  $Pr[\tau \leq n]$  which is equal to  $1 - Pr[\tau > n] = 1 - q_n = 1 - h(1)$ . The first property of  $\alpha$ -strong functions ensures that this number is at least  $\alpha$ . Since  $Pr[\max X_i \geq x]$  is no more than 1, therefore, for every  $x < \theta_n$  we have  $Pr[X_\tau \geq x] \geq \alpha Pr[\max X_i \geq x]$ .

Now suppose  $x \in \mathcal{R}^{\geq 0}$  and  $x \geq \theta_n$ . For  $Pr[X_\tau \geq x]$  we have,

$$\begin{aligned} Pr[X_\tau \geq x] &= \sum_{i=1}^n Pr[X_\tau \geq x | \tau = i] Pr[\tau = i] \\ &= \sum_{i=1}^n q_{i-1} (1 - F(\max\{\theta_i, x\})) . \end{aligned} \tag{6.3}$$

Since the thresholds are decreasing, there exists a unique index  $j(x)$  for which  $\theta_{j(x)} > x \geq \theta_{j(x)+1}$ . For the sake of simplicity we assume there is an imaginary item  $X_0$  for which  $\theta_0 = \infty$ . In this way  $j(x)$  is an integer number from 0 to  $n - 1$ . By

expanding (6.3) we have:

$$\begin{aligned}
Pr[X_\tau \geq x] &= \sum_{i=1}^n q_{i-1}(1 - F(\max\{\theta_i, x\})) \\
&= \sum_{i=1}^n q_{i-1}G(\max\{\theta_i, x\}) \\
&= \sum_{i=1}^{j(x)} q_{i-1}G(\theta_i) + \sum_{i=j(x)+1}^n q_{i-1}G(x) . \tag{6.4}
\end{aligned}$$

The first sum in (6.4) is indeed the probability of selecting one of the first  $j(x)$  items, therefore we can rewrite it as  $1 - q_{j(x)}$ . Hence,

$$\begin{aligned}
Pr[X_\tau \geq x] &= 1 - q_{j(x)} + \sum_{i=j(x)+1}^n q_{i-1}G(x) \\
&= 1 - q_{j(x)} + nG(x) \sum_{i=j(x)+1}^n q_{i-1} \frac{1}{n} \\
&= 1 - q_{j(x)} + nG(x) \sum_{i=j(x)+1}^n h((i-1)/n) \frac{1}{n} \\
&\geq 1 - q_{j(x)} + nG(x) \int_{j(x)/n}^1 h(r)dr . \tag{6.5}
\end{aligned}$$

The integral in (6.5) comes from the fact that  $h$  is a decreasing function and for such functions the Riemann sum of an interval is an upper bound of the integral of the function in that interval. For simplicity let  $s(x) = j(x)/n$ . Inequality (6.5) can be written as follows:

$$Pr[X_\tau \geq x] \geq 1 - h(s(x)) + nG(x) \int_{s(x)}^1 h(r)dr . \tag{6.6}$$

In order to complete the proof of the theorem, we need to show that the right hand side of Inequality (6.6) is an approximation of  $Pr[\max X_i \geq x]$ . To this end, we use the following lemma.

**Lemma 6.3.** *For every  $\epsilon > 0$  there exists an  $n_\epsilon$  such that for every integer  $n \geq n_\epsilon$  the following inequality holds for every  $x \geq \theta_n$ :*

$$1 - h(s(x)) + nG(x) \int_{s(x)}^1 h(r)dr \geq (1 - \epsilon)\alpha(1 - \exp(-nG(x))) .$$

To wrap up the proof of the theorem we combine the results of the previous lemmas. Suppose  $n_1$  and  $n_2$  are the lower bounds of Lemma 6.2 and Lemma 6.3 for  $n$ , respectively, such that their inequalities hold for  $\epsilon/2$ . For every  $n \geq n_\epsilon = \max\{n_1, n_2\}$  we have:

$$Pr[X_\tau \geq x] \geq 1 - h(s(x)) + nG(x) \int_{s(x)}^1 h(r)dr \quad \text{Inequality (6.6)} \quad (6.7)$$

$$\geq (1 - \epsilon/2)\alpha(1 - \exp(-nG(x))) \quad \text{Lemma 6.3} \quad (6.8)$$

$$\geq (1 - \epsilon/2)^2\alpha Pr[\max X_i \geq x] \quad \text{Lemma 6.2} \quad (6.9)$$

$$\geq (1 - \epsilon)\alpha Pr[\max X_i \geq x] .$$

This shows that for every non-negative value of  $x$  the chance of the algorithm in selecting an item with value at least  $x$  is an approximation of the corresponding probability for the optimum solution. More specifically, we showed that for every  $n \geq n_\epsilon$  and for every  $x \geq 0$  the integrand of (6.2) is a  $(1 - \epsilon)\alpha$ -approximation of the integrand of (6.1), hence the theorem is proved.  $\square$

Now we have all the materials needed to prove Theorem 6.5. In order to prove the theorem, we show that the function  $h(s) = \cos(as)$  is an  $\alpha$ -strong function, where  $a \approx 1.306$  is a root of  $\cos(a) + \sin(a)/a - 1$  and  $\alpha = 1 - \cos(a) \approx 0.7388$ . To this end, we first need to show that this function is a threshold function:

- i. To show the concavity of  $h$  it suffices to show that its second derivative is

negative for every  $0 < s \leq 1$ . Note that  $h'(s) = -a \sin(as)$  and  $h''(s) = -a^2 \cos(as)$ .

- ii. The ratio of  $h'(s)/h(s)$  for every  $s$  is equal to  $-a \tan(as)$ . For every  $\epsilon$  we need to show that there exists some  $\delta_0 \leq \epsilon$  such that for every  $\delta \leq \delta_0$  and  $\epsilon + \delta \leq s \leq 1$  the following holds:

$$-a \tan(a(s - \delta)) \leq -(1 - \epsilon)a \tan(as)$$

or equivalently, by dividing both sides to  $-a$  and changing the direction of the inequality we want to have:

$$\tan(as - a\delta) \geq (1 - \epsilon) \tan(as) .$$

Note that  $\tan(as)$  is a convex function because  $\tan''(as) = 2 \tan(as) \sec^2(as) \geq 0$  for  $0 \leq s \leq 1$ . For every  $0 \leq \delta \leq s$  in such functions we have:

$$\frac{\tan(as) - \tan(as - a\delta)}{a\delta} \leq \tan'(as) = \sec^2(as) \leq \sec^2(a) .$$

Therefore,

$$\tan(as) \leq \tan(as - a\delta) + a\delta \sec^2(a) .$$

By multiplying both sides by  $(1 - \epsilon)$  and assuming that  $\delta \leq \delta_0 = \frac{\epsilon \tan(a\epsilon)}{a(1 - \epsilon) \sec^2(a)}$  we have:

$$\begin{aligned} (1 - \epsilon) \tan(as) &\leq (1 - \epsilon)(\tan(as - a\delta) + a\delta \sec^2(a)) \\ &\leq \tan(as - a\delta) - \epsilon \tan(as - a\delta) + (1 - \epsilon)a\delta \sec^2(a) \\ &\leq \tan(as - a\delta) - \epsilon \tan(as - a\delta) + \epsilon \tan(a\epsilon) \\ &= \tan(as - a\delta) - \epsilon(\tan(a(s - \delta)) - \tan(a\epsilon)) \end{aligned} \quad (6.10)$$



Note that  $\tan(x)$  is an increasing function, therefore for every  $s \geq \epsilon + \delta$  Inequality (6.10) is less than or equal to  $\tan(as - a\delta)$ , thus the second property holds as well.

We showed that  $h(s) = \cos(as)$  is a threshold function. Now we prove that this threshold function is also an  $\alpha$ -strong function. Due to definition  $\alpha = 1 - \cos(a) = 1 - h(1)$ , thus the first property holds. Moreover,  $\int_0^1 h(r)dr = \sin(a)/a$ . Again, due to definition  $a$  is a root of  $\cos(a) + \sin(a)/a - 1$ , and thus  $\sin(a)/a = 1 - \cos(a) = \alpha$ . Now we only need to show that the third property of  $\alpha$ -strong functions holds. To do so, we need to show that:

$$1 - \cos(as) + a \tan(as)[\sin(a)/a - \sin(as)/a] \geq \alpha(1 - \exp(-a \tan(as))) \quad . \quad (6.11)$$

By subtracting  $\alpha(1 - \exp(-a \tan(as)))$  from both sides and multiplying them by  $\cos(as)$  we have:

$$\cos(as) - \cos(as)^2 + \sin(a)\sin(as) - \sin(as)^2 - \alpha \cos(as) + \alpha \cos(as) \exp(-a \tan(as)) \geq 0 \quad .$$

Note that  $\cos^2(as) + \sin^2(as) = 1$ , therefore the above inequality is equivalent to:

$$(1 - \alpha) \cos(as) + \sin(a) \sin(as) + \alpha \cos(as) \exp(-a \tan(as)) \geq 1 \quad .$$

Since  $\sin(a)/a = 1 - \cos(a) = \alpha$  we can replace  $\sin(a)$  with  $\alpha a$ . Also, from the relation between trigonometric functions we have  $\cos(x) = 1/\sqrt{1 + \tan^2(x)}$  and  $\sin(x) = \tan(x)/\sqrt{1 + \tan^2(x)}$ . By considering these equalities and assuming that  $w = \tan(as)$  the above inequality becomes simplified as follows:

$$\frac{1 - \alpha}{\sqrt{1 + w^2}} + \frac{\alpha a w}{\sqrt{1 + w^2}} + \frac{\alpha \exp(-aw)}{\sqrt{1 + w^2}} \geq 1 \quad .$$

By multiplying both sides by  $\sqrt{1+w^2}$  and raising them to the power of two, and subtracting  $1+w^2$  from both sides we have:

$$(1 - \alpha + \alpha aw + \alpha \exp(-aw))^2 - 1 - w^2 \geq 0 .$$

Now we use the following lemma to finish the proof.

**Lemma 6.4.** *Suppose  $A(w) = (1 - \alpha + \alpha aw + \alpha \exp(-aw))^2 - 1 - w^2$  where  $a \approx 1.306$  is a root of  $\cos(a) + \sin(a)/a - 1$  and  $\alpha = 1 - \cos(a) \approx 0.7388$ . Then for every  $0 \leq w \leq \tan(a)$  we have  $A(w) \geq 0$ .*

Lemma 6.4 shows that this inequality holds for every  $0 \leq w \leq \tan(a)$ . Consequently, Inequality (6.11) holds for every  $0 \leq s \leq 1$ . This completes the proof that  $h(s) = \cos(as)$  is an  $\alpha$ -strong function for  $\alpha \approx 0.7388$ , since it has all the three properties.

### 6.3 Non IID Distributions

In this section we study more generalized cases of the prophet inequalities problem. Suppose  $X_1, \dots, X_n$  are random variables from distribution functions  $F_1, \dots, F_n$ . Similar to Section 6.2 we assume, for the sake of simplicity, that all distribution functions are continuous and strictly increasing on a subinterval of  $\mathcal{R}^+$ . The goal of this section is to show improving results for the best order and a random order of *large market* instances. We use the term large market as a general term to refer to instances with repeated distributions. The following definition formally captures this concept.

**Definition 6.3.** *A set of  $n$  items with distribution functions  $F_1, \dots, F_n$  is  $m$ -frequent if for every item in this set there are at least  $m - 1$  other items with the same distribution function.*

In the remainder of this section we show for the best order and a random order of a large market instance that one can find a sequence of thresholds which in expectation performs as good as our algorithm for iid items. Roughly speaking, we design algorithms that are  $\alpha$ -approximation for large enough  $m$ -frequent instances, where  $\alpha \approx 0.7388$ . The following two theorems formally state our results for the best order and a random order, respectively.

**Theorem 6.7.** *For every  $\epsilon > 0$  and set  $\mathbb{X}$  of  $n$  items, there exists a number  $m_\epsilon$  (a function of  $\epsilon$  and independent of  $n$ ) such that if  $\mathbb{X}$  is  $m$ -frequent for  $m \geq m_\epsilon$  then there exists an algorithm which is  $(1 - \epsilon)\alpha$ -approximation on a permutation of  $\mathbb{X}$ .*

**Theorem 6.8.** *For every  $\epsilon > 0$  and set  $\mathbb{X}$  of  $n$  items there exists a number  $c_\epsilon$  (a function of  $\epsilon$  and independent of  $n$ ) such that if  $\mathbb{X}$  is  $m$ -frequent for  $m \geq c_\epsilon \log(n)$  then there exists an algorithm which in expectation is  $(1 - \epsilon)\alpha$ -approximation on a random permutation of  $\mathbb{X}$ .*

To prove the theorems we first provide an algorithm for a specific class of large market instances, namely partitioned sequences. Lemma 6.5 states that this algorithm is  $\alpha$ -approximation when the number of partitions is large. We later show how to apply this algorithm on the best order and a random order of large market instances to achieve a similar approximation factor. Following is a formal definition of partitioned sequences.

**Definition 6.4.** A sequence of items with distribution functions  $F_1, \dots, F_n$  is  $m$ -partitioned if  $n = mk$  and the sequence of functions  $F_{ik+1}, \dots, F_{ik+k}$  is a permutation of  $F_1, \dots, F_k$  for every  $0 \leq i < m$ .

The following algorithm exploits Algorithm 3 for iid items in order to find thresholds for a partitioned large market instance.

---

**Algorithm 4**

---

**Input:** An  $m$ -partitioned sequence of items with distribution functions  $F_1, \dots, F_n$ .

- 1: Let  $k = n/m$ .
  - 2: Let  $F(x) = \prod_{i=1}^k F_i(x)$ .
  - 3: Let  $\theta_1, \dots, \theta_m$  be the thresholds by Algorithm 3 for  $m$  iid items with distribution function  $F$ .
  - 4: Pick the first item  $i$  if  $X_i \geq \theta_{\lceil i/k \rceil}$ .
- 

**Lemma 6.5.** For every  $\epsilon > 0$  there exists a number  $m_\epsilon$  (a function of  $\epsilon$  and independent of the number of items) such that for every  $m \geq m_\epsilon$  Algorithm 4 is  $(1 - \epsilon)\alpha$ -approximation on an  $m$ -partitioned input.

Now we are ready to prove Theorem 6.7 and Theorem 6.8.

**Proof of Theorem 6.7:** Let  $s$  be the lower bound on the number of partitions in Lemma 6.5 for  $\epsilon/2$ , and let  $m_\epsilon = 2(s - 1)/\epsilon$ . The outline of the proof is as follows. Let  $\mathbb{X}$  be an  $m$ -frequent set of items for  $m \geq m_\epsilon$ . We uniformly group the items into  $s$  parts with  $\lfloor m/s \rfloor$  items of each type in every group. Let  $\mathbb{Y}$  denote the set of partitioned items. In order to make all parts similar, we may need to

discard some of the items, however, we show this does not hurt the approximation factor significantly. Finally, by applying Algorithm 4 to  $\mathbb{Y}$  we achieve the desired approximation factor.

The following lemma shows that discarding a fraction of items influences the approximation factor proportionally.

**Lemma 6.6.** *Let  $\{X_1, \dots, X_n\}$  be a  $k$ -frequent set of items. Suppose for some  $S \subseteq \{1, \dots, n\}$  that the set  $\{X_{S_1}, \dots, X_{S_r}\}$  is  $p$ -frequent and contains every  $X_i$  for  $1 \leq i \leq n$ . Then we have*

$$\mathbb{E}[\max_{i \in S} X_i] \geq \frac{p}{k} \mathbb{E}[\max_{1 \leq i \leq n} X_i] .$$

Note that in partitioning  $\mathbb{X}$  to  $s$  groups there might be at most  $s - 1$  items of each type being discarded in  $\mathbb{Y}$ , therefore  $\mathbb{Y}$  is  $(m - s + 1)$ -frequent. Let  $ALG$  be a random variable that denotes the value of the item picked by our algorithm. We have:

$$\begin{aligned} \mathbb{E}[ALG] &\geq (1 - \epsilon/2)\alpha \mathbb{E}[\max_{Y \in \mathbb{Y}} Y] && \text{Lemma 6.5} \\ &\geq (1 - \epsilon/2)\alpha \frac{m - s + 1}{m} \mathbb{E}[\max_{X \in \mathbb{X}} X] && \text{Lemma 6.6} \\ &\geq (1 - \epsilon/2)^2 \alpha \mathbb{E}[\max_{X \in \mathbb{X}} X] \\ &\geq (1 - \epsilon)\alpha \mathbb{E}[\max_{X \in \mathbb{X}} X] . \end{aligned}$$

Therefore, for every  $m$ -frequent set  $\mathbb{X}$  there exists an ordering of its items on which our algorithm is  $(1 - \epsilon)\alpha$ -approximation.  $\square$

**Proof of Theorem 6.8:** Let  $\pi$  be a random permutation of the items. Consider  $s$  different partitions for the items, i.e. one from  $X_{\pi_1}$  to  $X_{\pi_{n/s}}$ , one from

$X_{\pi_{n/s+1}}$  to  $X_{\pi_{2n/s}}$ , so on so forth. We show that when the number of similar items is large enough then a random permutation is very likely to uniformly distribute similar items into these parts. Therefore, by discarding a small fraction of the items  $X_{\pi_1}, \dots, X_{\pi_n}$  can be assumed as an  $s$ -partitioned sequence, hence Algorithm 4 can be applied to it.

Note that  $\mathbb{X}$  is  $m$ -frequent, which means that for every item  $i$  there are at least  $m - 1$  other items with the same distribution functions as  $F_i$ . We refer to a set of similar items as a type. Therefore, there are at least  $m$  items of every type in  $\mathbb{X}$ . We use the following lemma to show for every type that with a high probability the number of items of that type in every partition is almost  $m/s$ .

**Lemma 6.7** ([PS97]). *Let  $x_1, \dots, x_m$  be a sequence of negatively correlated boolean (i.e. 0 or 1) random variables, and let  $X = \sum_{i=1}^m x_i$ . We have:*

$$\Pr[|X - \mathbb{E}[X]| \geq \delta \mathbb{E}[X]] \leq 3 \exp(-\delta^2 \mathbb{E}[X]/3) .$$

Since  $\pi$  is a random permutation, the expected number of these items in a fixed partition is  $m/s$ . Using Lemma 6.7, with probability at most  $3 \exp(\frac{-\delta^2 m}{3s})$  there are less than  $(1 - \delta)m/s$  of these items in a fixed partition. Using Union Bound on all the  $s$  partitions and all types of items (note that there at at most  $n/m$  types), with probability at most  $3s \frac{n}{m} \exp(\frac{-\delta^2 m}{3s})$  there is a type of item which has less than  $(1 - \delta)m/s$  items in a partitions. If we choose  $\delta = \epsilon/3$  then for every  $m \geq \frac{3s}{\delta^2} (\log(n) + \log(\frac{9}{\epsilon}))$  this probability becomes less than  $\epsilon/3$ .

Now we are ready to wrap up the proof. If we choose  $s = m_{\epsilon/3}$  using Lemma 6.5,  $\delta = \epsilon/3$ , and  $c_\epsilon = \frac{3s \log(9/\epsilon)}{\delta^2}$  then for every  $m \geq c_\epsilon \log(n)$  with probability at

least  $(1 - \epsilon/3)$  there are at least  $(1 - \epsilon/3)m/s$  items of each type in every partitions. In such cases by discarding at most  $\epsilon/3$  fraction of the items of each type we have exactly  $(1 - \epsilon/3)m/s$  of them in each partition. Lemma 6.6 states that removing this fraction of items changes the approximation factor by at most  $(1 - \epsilon/3)$ . This means that for a random permutation of the items, with probability at least  $(1 - \epsilon/3)$  we can loose on the approximation factor by no worse than  $(1 - \epsilon/3)$  and have an  $s$ -partitioned sequence. Due to Lemma 6.5, Algorithm 4 is  $(1 - \epsilon/3)\alpha$ -approximation on this number of partitions. Therefore, the approximation factor of our method is  $(1 - \epsilon/3)^3\alpha$  which is more than  $(1 - \epsilon)\alpha$ . □

## Chapter 7: Prophet Secretary for Matroids and Combinatorial Auctions

### 7.1 Introduction

Suppose there is a sequence of  $n$  buyers arriving with different *values* to your *single* item. On arrival a buyer offers a take-it-or-leave-it value for your item. How should you decide which buyer to assign the item to in order to maximize the value. There are two popular models in the field of Stopping Theory to study this problem: the *secretary* and the *prophet inequality* models. In the secretary model we assume no prior knowledge about the buyer values but the buyers arrive in a uniformly random order [Dyn63]. Meanwhile, in the prophet inequality model we assume stochastic knowledge about the buyer values but the arrival order of the buyers is chosen by an adversary [KS78,KS77]. Since the two models complement each other, both have been widely studied in the fields of mechanism design and combinatorial optimization (see related work).

These models assume that either the buyer values or the buyer arrival order is chosen by an adversary. In practice, however, it is often conceivable that there is no adversary acting against you. Can we design better strategies in such settings?



The *prophet secretary* model introduced in [EHL15] is a natural way to consider such a process where we assume both a stochastic knowledge about buyer values and that the buyers arrive in a uniformly random order. The goal is to design a strategy that maximizes expected accepted value, where the expectation is over the random arrival order, the stochastic buyer values, and also any internal randomness of the strategy.

In this thesis, we consider generalizations of the above problem to combinatorial settings. Suppose the buyers correspond to elements of a matroid<sup>1</sup> and we are allowed to accept any independent set in this matroid rather than only a single buyer. The buyers again arrive and offer take-it-or-leave-it value for being accepted. In the prophet inequality model, a surprising result of Kleinberg-Weinberg [KW12] gives a  $1/2$ -approximation strategy to this problem, i.e., the value of their strategy, in expectation, is at least half of the value of the expected offline optimum that selects the best set of buyers in *hindsight*. Simple examples show that for adversarial arrival one cannot improve this factor. On the other hand, if we are also allowed to control the arrival order of the buyers, Yan [Yan11] gives a  $1 - 1/e \approx 0.63$ -approximation strategy. But what if the arrival order is neither adversarial and nor in your control. In particular, can we beat the  $1/2$ -approximation for a uniformly random arrival order?

---

<sup>1</sup>A matroid  $\mathcal{M}$  consists of a ground set  $[n] = \{1, 2, \dots, n\}$  and a non-empty downward-closed set system  $\mathcal{I} \subseteq 2^{[n]}$  satisfying the matroid exchange axiom: for all pairs of sets  $I, J \in \mathcal{I}$  such that  $|I| < |J|$ , there exists an element  $x \in J$  such that  $I \cup \{x\} \in \mathcal{I}$ . Elements of  $\mathcal{I}$  are called independent sets.

**Matroid Prophet Secretary Problem (MPS):** *Given a matroid  $\mathcal{M} = ([n], \mathcal{I})$  on  $n$  buyers (elements) and independent probability distributions on their values, suppose the outcome buyer values are revealed in a uniformly random order. Whenever a buyer value is revealed, the problem is to immediately and irrevocably decide whether to select the buyer. The goal is to maximize the sum of values of the selected buyers, while ensuring that they are always feasible in  $\mathcal{I}$ .*

Besides being a natural problem that relates two important Stopping Theory models, MPS is also interesting because of its applications in mechanism design. Often while designing mechanisms, we have to balance between maximizing revenue/welfare and the simplicity of the mechanism. While there exist optimal mechanisms such as VCG or Myerson’s mechanism, they are impractical in real markets [AM06, Rot07]. On the other hand, simple *Sequentially Posted Pricing mechanisms*, where we offer take-it-or-leave-it prices to buyers, are known to give good approximations to optimal mechanisms, since this reduces the problem to designing a prophet inequality [CHMS10b, Yan11, Ala11, KW12, FGL15].

Esfandiari et al. [EHL15] study MPS in the special case of a rank 1 matroid and give a  $(1 - 1/e)$ -approximation algorithm. For general matroids, as in the original models of [CHMS10b, Yan11, KW12], it was unclear prior to our work whether beating the factor of  $1/2$  is possible. In Section 7.4 we prove the following result.

**Theorem 7.1.** *MPS There exists a  $(1 - 1/e)$ -approximation algorithm to MPS.*

Note that the approximation in this theorem as well as the following ones compare to the expected *optimal offline solution* for the particular outcomes of the distributions.

That is, in the case of matroids, we have  $\mathbb{E}[\text{GA}] \geq (1 - 1/e) \cdot \mathbb{E}[\max_{I \in \mathcal{I}} \sum_{i \in I} v_i]$ , where  $v_i$  is the value of buyer  $i$ .

Next, let us consider a combinatorial auctions setting. Suppose there are  $n$  buyers that take combinatorial valuations (say, submodular) for  $m$  indivisible items from  $n$  independent probability distributions. The problem is to decide how to allocate the items to the buyers, while trying to maximize the *welfare*—the sum of valuations of all the buyers. Feldman et al. [FGL15] show that for XOS<sup>2</sup> (a generalization of submodular) valuations there exist *static prices* for items that gets a 1/2-approximation for buyers arriving in an adversarial order. Since this factor cannot be improved for adversarial arrival, this leaves an important open question if we can design better algorithms when the arrival order can be controlled. Or ideally, we want to beat 1/2 even when the arrival order cannot be controlled but is chosen uniformly at random.

**Combinatorial Auctions Prophet Secretary Problem (CAPS):** *Suppose  $n$  buyers take XOS valuations for  $m$  items from  $n$  independent probability distributions. The outcome buyer valuations are revealed in a uniformly random order. Whenever a buyer valuations is revealed, the problem is to immediately and irrevocably assign a subset of the remaining items to the buyer. The goal is maximize the sum of the valuations of all the buyers for their assigned subset of items.*

In Section 7.3.2 we use *dynamic prices* to improve the result of [FGL15] for

---

<sup>2</sup>A function  $v: 2^M \rightarrow \mathcal{R}$  is an XOS function if there exists a collection of additive functions  $A_1, \dots, A_k$  such that for every  $S \subseteq M$  we have  $v(S) = \max_{1 \leq i \leq k} A_i(S)$ .

random order.

**Theorem 7.2.** *CAPS* *There exists a  $(1 - 1/e)$ -approximation algorithm to CAPS.*

Given access to demand and XOS oracles for stochastic utilities of different buyers, the algorithm in Theorem 7.2 can be made efficient. This is interesting because it matches the best possible  $(1 - 1/e)$ -approximation for XOS-welfare maximization in the offline setting [DNS10, Fei06].

A desirable property in the design of an economically viable mechanism is *incentive-compatibility*. In particular, a buyer is more likely to make decisions about their allocations based on their own personal incentives rather than to accept a given allocation that might optimize the social welfare but not the individuals' profit. For the important case of unit-demand buyers (aka bipartite matching), in Section 7.3.1 we extend Theorem 7.2 to additionally obtain this property.

**Theorem 7.3.** *bipMatching* *For bipartite matchings, when buyers arrive in a uniformly random order, there exists an incentive-compatible mechanism that gives a  $(1 - 1/e)$ -approximation to the optimal welfare.*

Finally, in Section 7.5 we conclude by showing that for the single-item case one can obtain a  $(1 - 1/e)$ -approximation even by using static prices, and that nothing better is possible.

### 7.1.1 Our Techniques

In this section we discuss our three main ideas for a combinatorial auction. In this setting, our algorithm is threshold based, which means that we set *dynamic*

*prices* to the items and allow a buyer to purchase a set of items only if her value is more than the price of that set. This allows us to view total value as the sum of *utility* of the buyers and the total generated *revenue*. Although powerful, dynamic prices often lead to involved calculations and become difficult to analyze beyond a single item setting [EHL15, AEE<sup>+</sup>17]. To overcome this issue, our *first idea* is to convert our discrete problem into a continuous setting. We achieve this by noticing that a random permutation of buyers can be viewed as each buyer arriving at a time chosen uniformly at random between 0 and 1. The benefit of such a transformation is that it lets us talk about continuous functions such as the expected revenue/utility obtained till time  $t \in [0, 1]$  and to use powerful tools from integral calculus.

Our algorithm for combinatorial auctions sets a *base price*  $b_j$  for every item  $j$  based on its contribution to the expected offline optimum  $\mathbb{E}[OPT]$ . Our *second idea* is to define two time varying continuous functions: *discount* and *residual*. The discount function  $\alpha(t): [0, 1] \rightarrow [0, 1]$  is chosen such that the price of an unsold item  $j$  at time  $t$  is exactly  $\alpha(t) \cdot b_j$ . We define a *residual* function  $r(t): [0, 1] \rightarrow \mathbb{R}_{\geq 0}$  that intuitively denotes the expected value remaining in the instance at time  $t$ . Hence,  $r(0) = \mathbb{E}[OPT]$  and  $r(1) = 0$ . Computing  $r(t)$  is difficult for a combinatorial auction since it depends on several random variables. However, assuming that we know  $r(t)$ , we use application specific techniques to compute lower bounds on both the expected revenue and the expected utility in terms of the functions  $r(t)$  and  $\alpha(t)$ .

Finally, although we do not know  $r(t)$ , our *third idea* is to show that if the residual function satisfies some “nice” properties (see Definition 7.1) then we can choose the function  $\alpha(t)$  in a way that allows us to simplify the sum of expected

revenue and utility, without ever computing  $r(t)$  explicitly. This step exploits properties of the exponential function for integration (see Lemma 7.1).

### 7.1.2 Related Work

Starting with the works of Krengel-Sucheston [KS78, KS77] and Dynkin [Dyn63], there has been a long line of research on both prophet inequalities and secretary problems. One of the first generalizations is the *multiple-choice prophet inequalities* [Ken87, Ken85, Ker86] in which we are allowed to pick  $k$  items and the goal is to maximize their sum. Alaei [Ala11] gives an almost tight  $(1 - 1/\sqrt{k+3})$ -approximation algorithm for this problem (the lower bound is due to [HKS07]). Similarly, the *multiple-choice secretary* problem was first studied by Hajiaghayi et al. [HKP04], and Kleinberg [Kle05] gives a  $(1 - O(\sqrt{1/k}))$ -approximation algorithm.

The research investigating the relation between prophet inequalities and online auctions is initiated in [HKS07, CHMS10b]. This led to several interesting follow up works for matroids [Yan11, KW12] and matchings [AHL12]. Meanwhile, the connection between secretary problems and online auctions is first explored in Hajiaghayi et al. [HKP04]. Its generalization to matroids is considered in [BIK07, Lac14, FSZ15] and to matchings in [GM08, KP09, MY11, KMT11, KRTV13, GS17].

Secretary problems and prophet inequalities have also been studied beyond a matroid/matching. For the intersection of  $p$  matroids, Kleinberg and Weinberg [KW12] give an  $O(p)$ -approximation prophet inequality. Dütting and Klein-

berg [DK15] extend this result to polymatroids. Feldman et al. [FGL15] study the generalizations to combinatorial auctions. Later, Dütting et al. [DFKL16] give a general framework to prove such prophet inequalities. Submodular variants of the secretary problem have been considered in [BHZ13, GRST10, FZ15, KMZ15]. Prophet and secretary problems have also been studied for many classical combinatorial problems (see e.g., [Mey01, GGLS08, GHK<sup>+</sup>14, DEH<sup>+</sup>15, DEH<sup>+</sup>17b]). Rubinstein [Rub16] and Rubinstein-Singla [RS17] consider these problems for arbitrary downward-closed constraints.

In the prophet secretary model, Esfandiari et al. [EHL15] give a  $(1 - 1/e)$ -approximation in the special case of a single item. Going beyond  $1 - 1/e$  has been challenging. Only recently, Abolhasani et al. [AEE<sup>+</sup>17] and Correa et al. [CFH<sup>+</sup>17] improve this factor for the single item i.i.d. setting. Extending this result to non-identical items or to matroids are interesting open problems.

## 7.2 Our Approach using a Residual

In this section, we define a residual and discuss how it can be used to design an approximation algorithm for a prophet secretary problem. Suppose there are  $n$  requests that arrive at times  $(T_i)_{i \in [n]}$  drawn i.i.d. from the uniform distribution in  $[0, 1]$ . These requests correspond to buyers of a combinatorial auction or to elements of a matroid.

Whenever a request arrives, we have to decide if and how to serve it. Depending on how we serve request  $i$ , say  $x_i$ , we gain a certain value  $v_i(x_i)$ . Our task

is to maximize the sum of values over all requests  $\sum_{i=1}^n v_i(x_i)$ . Our algorithm Alg includes a time-dependent payment component. The payment that request  $i$  has to make is the product of a time-dependent *discount* function  $\alpha(t)$  and a *base price*  $b(x_i)$ , where the base price depends on how much this choice limits other allocations but is time-independent. If request  $i$  has to pay  $p_i(x_i, T_i) = \alpha(t)b(x_i)$  for our decision  $x_i$ , then its *utility* is given by  $u_i = v_i(x_i) - p_i(x_i, T_i)$ . We write  $\text{Utility} = \sum_{i=1}^n u_i$  for the sum of utilities and  $\text{Revenue} = \sum_{i=1}^n p_i(x_i, T_i)$  for the sum of payments. The value achieved by Alg equals  $\text{Utility} + \text{Revenue}$ .

Next we define a residual function that has the interpretation of “expected remaining value in the instance at time  $t$ ”. In Lemma 7.1 we show that the existence of a residual function for Alg suffices to give a  $(1 - 1/e)$ -approximation prophet secretary.

**Definition 7.1** (Residual). *Consider a prophet secretary problem with expected offline value  $\mathbb{E}[\text{OPT}]$ . For any algorithm Alg based on a continuous differentiable discount function  $\alpha(t): [0, 1] \rightarrow [0, 1]$ , a continuous differentiable function  $r(t): [0, 1] \rightarrow \mathbb{R}_{\geq 0}$  is called a residual if it satisfies the following three conditions for every choice of  $\alpha$ .*

$$r(0) = \mathbb{E}[\text{OPT}] \tag{7.1a}$$

$$\mathbb{E}[\text{Revenue}] \geq - \int_{t=0}^1 \alpha(t) \cdot r'(t) \cdot dt \tag{7.1b}$$

$$\mathbb{E}[\text{Utility}] \geq \int_{t=0}^1 (1 - \alpha(t)) \cdot r(t) \cdot dt. \tag{7.1c}$$

As an illustration of Definition 7.1, consider the case of a single item. That is, we are presented a sequence of  $n$  real numbers and may select only up to one of



them (previously studied in [EHL15]).

**Example 7.1** (Single Item). *Suppose buyer  $i \in [n]$  arrives with random value  $v_i$  at time  $T_i$  chosen uniformly at random between 0 and 1. Define  $b = \mathbb{E}[\max_i v_i]$  as the base price of the single item. A buyer arriving at time  $t$  is offered the item at price  $\alpha(t) \cdot b$ , and she accepts the offer if and only if  $v_i \geq \alpha(t) \cdot b$ . We show that  $r(t) = \Pr[\text{item not sold before } t] \cdot b$  is a residual function.*

*By definition, (7.1a) holds trivially. To see that (7.1b) holds, observe that the increase in revenue from time  $t$  to time  $t + \epsilon$  is approximately  $\alpha(t) \cdot b$  if the item is allocated during this time, and is 0 otherwise. That is, the expected increase in revenue is approximately  $\alpha(t)(r(t) - r(t + \epsilon))$ . Taking the limit for  $\epsilon \rightarrow 0$  then implies (7.1b), i.e.,  $\mathbb{E}[\text{Revenue}] = - \int_{t=0}^1 \alpha(t)r'(t)dt$ .*

*For (7.1c), we consider the expected utility of a buyer  $i$  conditioning on her arriving at time  $t$*

$$\mathbb{E}[u_i \mid T_i = t] = \mathbb{E}[\mathbf{1}_{\text{item not sold before } t} \cdot (v_i - \alpha(t) \cdot b)^+ \mid T_i = t].$$

*Although there are non-trivial correlations involved, one can show (see Appendix D) that*

$$\mathbb{E}[u_i \mid T_i = t] \geq \Pr[\text{item not sold before } t] \cdot \mathbb{E}[(v_i - \alpha(t) \cdot b)^+].$$

*Next, we take the sum over all buyers  $i$  and use that  $\sum_{i=1}^n (v_i - \alpha(t) \cdot b)^+ \geq \max_i (v_i - \alpha(t) \cdot b) = (1 - \alpha(t)) \cdot b$  to get*

$$\sum_{i=1}^n \mathbb{E}[u_i \mid T_i = t] \geq \Pr[\text{item not sold before } t] \cdot (1 - \alpha(t)) \cdot b = (1 - \alpha(t)) \cdot r(t).$$

This implies

$$\mathbb{E}[Utility] = \sum_{i=1}^n \int_{t=0}^1 \mathbb{E}[u_i | T_i = t] dt = \int_{t=0}^1 \sum_{i=1}^n \mathbb{E}[u_i | T_i = t] dt \geq \int_{t=0}^1 (1 - \alpha(t)) \cdot r(t) \cdot dt.$$

We now use the properties of a residual function to design a  $(1 - 1/e)$ -approximation algorithm. To this end, we choose  $\alpha(t)$  in a manner that makes the sum of the expected revenue and buyers' utilities independent of  $r(t)$ . This allows us to compute expected welfare, even though we cannot compute  $r(t)$  directly.

**Lemma 7.1.** *For a prophet secretary problem, if there exists a residual function  $r(t)$  for algorithm Alg as defined in Definition 7.1, then setting  $\alpha(t) = 1 - e^{t-1}$  gives a  $(1 - 1/e)$ -approximation.*

**Proof.** To further simplify Eq. (7.1b), we observe that applying integration by parts gives

$$\int r'(t) \cdot \alpha(t) \cdot dt = r(t) \cdot \alpha(t) - \int r(t) \alpha'(t) \cdot dt.$$

So in combination

$$\mathbb{E}[\text{Revenue}] = - \left( [r(t) \cdot \alpha(t)]_{t=0}^1 - \int_{t=0}^1 r(t) \cdot \alpha'(t) \cdot dt \right). \quad (7.2)$$

Now adding (7.2) and (7.1c) gives,

$$\begin{aligned} \mathbb{E}[\text{Alg}] &= \mathbb{E}[\text{Utility}] + \mathbb{E}[\text{Revenue}] \\ &\geq \int_{t=0}^1 r(t) \cdot (1 - \alpha(t)) \cdot dt - [r(t)\alpha(t)]_{t=0}^1 + \int_{t=0}^1 r(t)\alpha'(t) \cdot dt \\ &= \int_{t=0}^1 r(t) \cdot (1 - \alpha(t) + \alpha'(t)) \cdot dt - [r(t)\alpha(t)]_{t=0}^1. \end{aligned}$$

Although we do not know  $r(t)$  and computing  $\int_{t=0}^1 r(t) \cdot (1 - \alpha(t) + \alpha'(t)) \cdot dt$  seems difficult, we have the liberty of selecting the function  $\alpha(t)$ . By choosing  $\alpha(t)$  satis-

ying  $1 - \alpha(t) + \alpha'(t) = 0$  for all  $t$ , this integral becomes independent of  $r(t)$  and simplifies to 0. In particular, let  $\alpha(t) = 1 - e^{t-1}$ . This gives,

$$\mathbb{E}[Alg] \geq -[r(t) \cdot \alpha(t)]_{t=0}^1 = \left(1 - \frac{1}{e}\right) r(0) = \left(1 - \frac{1}{e}\right) \mathbb{E}[OPT].$$

□

### 7.3 Prophet Secretary for Combinatorial Auctions

Let  $N$  denote a set of  $n$  buyers and  $M$  denote the set of  $m$  indivisible items. Suppose buyer  $i$  arrives at a time  $T_i$  chosen uniformly at random between 0 and 1. Let  $v_i: 2^M \rightarrow \mathbb{R}_{\geq 0}$  (similarly  $\hat{v}_i$ ) denote the random combinatorial valuation function of buyer  $i$ . We assume that the distribution of  $v_i$  has a polynomial support  $\{v_i^1, v_i^2, \dots\}$ , where  $\sum_k \Pr[v_i = v_i^k] = 1$ . Note that this assumption only simplifies notation. If we only have sample access to the distributions, then we can replace  $\{v_i^1, v_i^2, \dots\}$  by an appropriate number of samples.

By  $\mathbf{T}$  and  $\mathbf{v}$  (similarly  $\hat{\mathbf{v}}$ ) we denote the vector of all the buyer arrival times and valuations, respectively. Also, let  $\mathbf{v}_{-i}$  (similarly  $\hat{\mathbf{v}}_{-i}$ ) denote valuations of all buyers except buyer  $i$ . For the special case of single items, we let  $v_{ij}$  denote  $v_i(\{j\})$ . Let  $q_j(t)$  denote the probability that item  $j$  has not been sold before time  $t$ , where the probability is over valuations  $\mathbf{v}$ , arrival times  $\mathbf{T}$ , and any randomness of the algorithm.

### 7.3.1 Bipartite Matching

In the bipartite matching setting all buyers are unit-demand, i.e.  $v_i(S) = \max_{j \in S} v_{ij}$ . We can therefore assume that no buyer buys more than one item. We restate our result.

To define prices of items, let *base price*  $b_j$  denote the expected value of the buyer that buys item  $j$  in the offline welfare maximizing allocation (maximum weight matching). Now consider an algorithm that prices item  $j$  at  $\alpha(t) \cdot b_j$  at time  $t$  and allows the incoming buyer to pick any of the unsold items; here  $\alpha(t)$  is a continuous differentiable discount function.

Consider the function  $r(t) = \sum_j q_j(t) \cdot b_j$ . Clearly,  $r(0) = \mathbb{E}[OPT]$ . Using the following Lemma 7.2 and Claim 7.1, we prove that  $r$  is a residual function for our algorithm. Since the algorithm is clearly incentive-compatible, Lemma 7.1 implies Theorem 7.3.

**Lemma 7.2.** *We can lower bound the total expected utility by*

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[Utility] \geq \sum_j \int_{t=0}^1 q_j(t) \cdot (1 - \alpha(t)) \cdot b_j \cdot dt. \quad (7.3)$$

**Proof.** Since buyer  $i$  arriving at time  $t$  can pick any of the unsold items, we have

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] = \mathbb{E}_{\mathbf{v}} \left[ \max_j \mathbf{1}_{j \text{ not sold before } t} \cdot (v_{i,j} - \alpha(t) \cdot b_j)^+ \mid T_i = t \right].$$

One particular choice of buyer  $i$  is to choose item  $OPT_i(v_i, \hat{\mathbf{v}}_{-i})$  if it is still available,

and no item otherwise. This gives us a lower bound of

$$\begin{aligned}\mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] &\geq \mathbb{E}_{\mathbf{v}} \left[ \mathbf{1}_{OPT_i(v_i, \hat{\mathbf{v}}_{-i}) \text{ not sold before } t} \cdot (v_{i, OPT_i(v_i, \hat{\mathbf{v}}_{-i})} - \alpha(t) \cdot b_j)^+ \mid T_i = t \right] \\ &= \sum_j \mathbb{E}_{\mathbf{v}, \hat{\mathbf{v}}} \left[ \mathbf{1}_{j \text{ not sold before } t} \cdot \mathbf{1}_{j=OPT_i(v_i, \hat{\mathbf{v}}_{-i})} \cdot (v_{i,j} - \alpha(t) \cdot b_j)^+ \mid T_i = t \right].\end{aligned}$$

Note that in the product, the fact whether  $j$  is sold before  $t$  only depends on  $\mathbf{v}_{-i}$  and the arrival times of the other buyers. It does not depend on  $v_i$  or  $\hat{\mathbf{v}}$ . The remaining terms, in contrast, only depend on  $v_i$  and  $\hat{\mathbf{v}}_{-i}$ . Therefore, we can use independence to split up the expectation and get

$$\begin{aligned}\mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] &\geq \sum_j \Pr[j \text{ not sold before } t \mid T_i = t] \cdot \mathbb{E}_{v_i, \hat{\mathbf{v}}_{-i}} \left[ \mathbf{1}_{j=OPT_i(v_i, \hat{\mathbf{v}}_{-i})} \cdot (v_{i,j} - \alpha(t) \cdot b_j)^+ \mid T_i = t \right].\end{aligned}$$

Next, we use that  $\Pr[j \text{ not sold before } t \mid T_i = t] \geq q_j(t)$  by Lemma 7.3 and that  $v_i$  and  $\hat{v}_i$  are identically distributed. Therefore, we can swap their roles inside the expectation. Overall, this gives us

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] \geq \sum_j q_j(t) \cdot \mathbb{E}_{\hat{\mathbf{v}}} \left[ \mathbf{1}_{j=OPT_i(\hat{\mathbf{v}})} \cdot (\hat{v}_{i,j} - \alpha(t) \cdot b_j) \right]. \quad (7.4)$$

Next, observe that  $\mathbb{E}_{\hat{\mathbf{v}}}[\sum_i \mathbf{1}_{j=OPT_i(\hat{\mathbf{v}})} \cdot \hat{v}_{i,j}] = b_j$  by the definition of  $b_j$ . Therefore, using linearity of expectation, summing up (7.4) over all buyers  $i$  gives us

$$\sum_i \mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] \geq q_j(t) \cdot (1 - \alpha(t)) \cdot b_j.$$

Now, taking the expectation over  $t$ , we get

$$\begin{aligned}\mathbb{E}_{\mathbf{v}, \mathbf{T}} \left[ \sum_i u_i \right] &= \sum_i \int_{t=0}^1 \mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] \cdot dt = \int_{t=0}^1 \sum_i \mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] \cdot dt \\ &\geq \int_{t=0}^1 \sum_j q_j(t) \cdot (1 - \alpha(t)) \cdot b_j \cdot dt = \sum_j \int_{t=0}^1 q_j(t) \cdot (1 - \alpha(t)) \cdot b_j \cdot dt.\end{aligned}$$

□

We next give a bound on the revenue generated by our algorithm.

**Claim 7.1.** *We can bound the total expected revenue by*

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[\text{Revenue}] = - \sum_j \int_{t=0}^1 q'_j(t) \alpha(t) \cdot b_j \cdot dt. \quad (7.5)$$

**Proof.** Since  $-q'_j(t)dt$  is the probability that item  $j$  is bought between  $t$  and  $t + dt$  (note  $q_j(t)$  is decreasing in  $t$ ), we have

$$\mathbb{E}[\text{Revenue}] = - \sum_j \int_{t=0}^1 q'_j(t) \alpha(t) \cdot b_j \cdot dt.$$

□

Finally, we prove the missing lemma that removes the conditioning on the arrival time.

**Lemma 7.3.** *We have*

$$\mathbb{E}_{\mathbf{v}_{-i}} \left[ \Pr_{\mathbf{T}}[j \text{ not sold before } t \mid T_i = t] \right] \geq q_j(t).$$

**Proof.** Consider the execution of our algorithm on two sequences that only differ in the arrival time of buyer  $i$ . To this end, let  $\mathbf{v}$  be arbitrary values and  $\mathbf{T}$  be arbitrary arrival times. Let  $A_{t'}$  be the set of items that are sold before time  $t'$  on the sequence defined by  $\mathbf{v}$  and  $\mathbf{T}$ . Furthermore, let  $B_{t'}$  be the set of items sold before time  $t'$  if we replace  $T_i$  by  $t$ . Ties are broken in the same way in both sequences.

We claim that  $B_{t'} \subseteq A_{t'}$  for all  $t' \leq t$ .

To this end, we observe that by definition  $B_{t'} = A_{t'}$  for  $t' \leq \min\{T_i, t\}$  because the two sequences are identical before  $\min\{T_i, t\}$ . This already shows the claim for

$T_i \geq t$ . Otherwise, assume that there is some  $t' \leq t$  for which  $B_{t'} \not\subseteq A_{t'}$ . Let  $t_{\text{inf}}$  be the infimum among these  $t'$ . It has to hold that some buyer  $i'$  arrives at time  $t_{\text{inf}}$  and buys item  $j_A \notin A_{t_{\text{inf}}}$  in the original sequence and  $j_B \notin B_{t_{\text{inf}}}$  in the modified sequence. Furthermore, we now have to have  $B_{t_{\text{inf}}} \not\subseteq A_{t_{\text{inf}}}$  because  $t_{\text{inf}}$  was defined to be the infimum of all  $t'$  for which  $B_{t'} \subseteq A_{t'}$  is not fulfilled. Therefore,  $j_A \notin B_{t_{\text{inf}}}$ . Additionally,  $j_B \notin A_{t_{\text{inf}}}$ . The reason is that for any  $t' < t_{\text{inf}}$  before the next arrival  $B_{t'} = B_{t_{\text{inf}}} \cup \{j_B\}$ .

Overall this means that in both sequences at time  $t_{\text{inf}}$  buyer  $i'$  has the choice between  $j_B$  and  $j_A$ . As his values are identical and ties are broken the same way, it has to hold that  $j_B = j_A$ , which then contradicts that  $B_{t_{\text{inf}}} \not\subseteq A_{t_{\text{inf}}}$ .

Taking the expectation over both  $\mathbf{v}$  and  $\mathbf{T}$ , we get

$$\Pr_{\mathbf{T}, \mathbf{v}}[j \notin A_t] \leq \Pr_{\mathbf{T}, \mathbf{v}}[j \notin B_t].$$

This implies the Lemma 7.3 because

$$\Pr_{\mathbf{T}, \mathbf{v}}[j \text{ not sold before } t] = \Pr_{\mathbf{T}, \mathbf{v}}[j \notin A_t]$$

$$\Pr_{\mathbf{T}, \mathbf{v}}[j \text{ not sold before } t \mid T_i = t] = \Pr_{\mathbf{T}, \mathbf{v}}[j \notin B_t].$$

□

### 7.3.2 XOS Combinatorial Auctions

In this section we prove our main result (restated below) for combinatorial auctions.

Recollect that the random valuation  $v_i$  of every buyer  $i$  has a polynomial support. We can therefore write the following expectation-version of the configuration LP, which gives us an upper bound on the expected offline social welfare.

$$\begin{aligned}
& \max \sum_i \sum_k \sum_S v_i^k(S) \cdot x_{i,S}^k \\
& \text{s.t.} \quad \sum_i \sum_k \sum_{S:j \in S} x_{i,S}^k = 1 && \text{for all } j \in M \\
& \quad \sum_S x_{i,S}^k = \Pr[v_i = v_i^k] && \text{for all } i, k
\end{aligned}$$

The above configuration LP can be solved with a polynomial number of calls to demand oracles of buyer valuations (see [DNS10]). Since all functions  $v_i^k$  are XOS, there exist additive supporting valuations; that is, there exist numbers  $v_{i,j}^{k,S} \geq 0$  s.t.  $v_{i,j}^{k,S} = 0$  for  $j \notin S$  and  $\sum_{j \in S} v_{i,j}^{k,S} = v_i^k(S)$ . Before describing our algorithm, we define a *base price* for every item.

**Definition 7.2.** *The base price  $b_j$  of every item  $j \in M$  is  $\sum_{i,k} \sum_{S:j \in S} v_{i,j}^{k,S} x_{i,S}^k$ .*

Since  $\sum_S x_{i,S}^k = \Pr[v_i = v_i^k]$ , consider an algorithm that on arrival of buyer  $i$  with valuation  $v_i^k$  draws an independent random set  $S$  with probability  $x_{i,S}^k / \Pr[v_i = v_i^k]$ . Let  $S_i^*$  denote this drawn set. This distribution also satisfies that for every item  $j$ ,

$$\sum_i \mathbb{E}_{v_i, S_i^*} \left[ \mathbf{1}_{j \in S_i^*} \cdot v_{i,j}^{k, S_i^*} \right] = \sum_{i,k} \Pr[v_i = v_i^k] \cdot \sum_{S:j \in S} \frac{x_{i,S}^k}{\Pr[v_i = v_i^k]} \cdot v_{i,j}^{k, S_i^*} = b_j. \tag{7.6}$$

Now consider the supporting additive valuation for  $S_i^*$  in the XOS valuation function  $v_i^k$  of buyer  $i$ . This can be found using the XOS oracle for  $v_i^k$  [DNS10]. Our algorithm



assigns her every item  $j$  for which  $v_{i,j}^{k,S_i^*} \geq \alpha(t) \cdot b_j$ , where  $\alpha(t)$  is a continuous differentiable function of  $t$ . Note that since we do not allow buyer  $i$  to choose items outside set  $S_i^*$ , the mechanism defined by this algorithm need not be incentive compatible.

Consider the function  $r(t) = \sum_j q_j(t) \cdot b_j$ , where again  $q_j(t)$  denotes the probability that item  $j$  has not been sold before time  $t$ . Clearly,  $r(0) = OPT$ . Using the following Lemma 7.4 and Claim 7.2, we prove that  $r$  is a residual function for our algorithm. Hence, Lemma 7.1 implies Theorem 7.2.

**Lemma 7.4.** *The expected utility of the above algorithm is lower bounded by*

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[Utility] \geq \sum_j \int_{t=0}^1 q_j(t) \cdot (1 - \alpha(t)) \cdot b_j \cdot dt. \quad (7.7)$$

**Proof.** Given that buyer  $i$  arrives at  $t$  and only buys item  $j$  if  $v_{i,j}^{k,S_i^*} \geq \alpha(t) \cdot b_j$ , her utility is

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] = \sum_j \mathbb{E}_{\mathbf{v}, \mathbf{T}, S_i^*} \left[ \mathbf{1}_{j \text{ not sold by } t} \cdot \mathbf{1}_{j \in S_i^*} \cdot \left( v_{i,j}^{k,S_i^*} - \alpha(t) \cdot b_j \right)^+ \mid T_i = t \right]$$

Using the fact that whether  $j$  is sold before  $t$  only depends on  $\mathbf{v}_{-i}$  and  $\mathbf{T}$ , and not on  $v_i$  or  $S_i^*$ ,

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] = \sum_j \Pr_{\mathbf{v}_{-i}, \mathbf{T}}[j \text{ not sold by } t \mid T_i = t] \cdot \mathbb{E}_{v_i, S_i^*} \left[ \mathbf{1}_{j \in S_i^*} \cdot \left( v_{i,j}^{k,S_i^*} - \alpha(t) \cdot b_j \right)^+ \right].$$

Now, observe that in our algorithm every buyer  $i$  independently decides which set of items  $S_i^*$  it will attempt to buy. Crucially, the probability of an item  $j$  being sold by time  $t$  can only increase if more buyers arrive before  $t$ . Therefore,

$$\Pr_{\mathbf{v}_{-i}, \mathbf{T}}[j \text{ not sold by } t \mid T_i = t] \geq \Pr_{\mathbf{v}, \mathbf{T}}[j \text{ not sold by } t] = q_j(t).$$

Thus, we get

$$\begin{aligned}\mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i | T_i = t] &\geq \sum_j q_j(t) \cdot \mathbb{E}_{v_i, S_i^*} \left[ \mathbf{1}_{j \in S_i^*} \cdot \left( v_{i,j}^{k, S_i^*} - \alpha(t) \cdot b_j \right)^+ \right] \\ &\geq \sum_j q_j(t) \cdot \mathbb{E}_{v_i, S_i^*} \left[ \mathbf{1}_{j \in S_i^*} \cdot \left( v_{i,j}^{k, S_i^*} - \alpha(t) \cdot b_j \right) \right].\end{aligned}$$

Finally, recollect from Eq. (7.6) that  $\sum_i \mathbb{E}_{v_i, S_i^*} \left[ \mathbf{1}_{j \in S_i^*} \cdot v_{i,j}^{k, S_i^*} \right] = b_j$ . Moreover,

$$\sum_i \mathbb{E}_{v_i, S_i^*} \left[ \mathbf{1}_{j \in S_i^*} \right] = \sum_{i,k} \Pr[v_i = v_i^k] \cdot \sum_{S: j \in S} \frac{x_{i,S}^k}{\Pr[v_i = v_i^k]} = 1.$$

Hence, by linearity of expectation

$$\sum_i \mathbb{E}[u_i | T_i = t] \geq \sum_j q_j(t) \cdot (1 - \alpha(t)) \cdot b_j.$$

□

We next give a bound on the revenue generated by our algorithm.

**Claim 7.2.** *We can bound the total expected revenue by*

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[\text{Revenue}] = - \sum_j \int_{t=0}^1 q_j'(t) \alpha(t) \cdot b_j \cdot dt. \quad (7.8)$$

**Proof.** Since  $-q_j'(t)dt$  is the probability that item  $j$  is bought between  $t$  and  $t + dt$  (note  $q_j(t)$  is decreasing in  $t$ ), we have

$$\mathbb{E}[\text{Revenue}] = - \sum_j \int_{t=0}^1 q_j'(t) \alpha(t) \cdot b_j \cdot dt.$$

□

## 7.4 Prophet Secretary for Matroids

Let  $v_i$  denote the random value of the  $i$ 'th buyer (element) and let  $\hat{v}_i$  denote another independent draw from the value distribution of the  $i$ 'th buyer. The problem

is to select a subset  $I$  of the buyers that form a feasible set in matroid  $\mathcal{M}$ , while trying to maximize  $\sum_{i \in I} v_i$ . We restate our main result for the matroid setting.

We need the following notation to describe our algorithm.

**Definition 7.3.** For a given vector of values  $\hat{\mathbf{v}}$ , we define the following terms:

- Let  $Opt(\hat{\mathbf{v}} \mid A)$  denote the optimal solution set in the contracted matroid  $\mathcal{M}/A$ .
- Let  $R(A, \hat{\mathbf{v}}) := \sum_{i \in Opt(\hat{\mathbf{v}} \mid A)} \hat{\mathbf{v}}_i$  denote the remaining value after selecting set  $A$ .

We next define a *base price* of for every buyer  $i$ .

**Definition 7.4.** Let  $A$  denote the independent set of buyers that have been accepted till now.

- Let  $b_i(A, \hat{\mathbf{v}}) := R(A, \hat{\mathbf{v}}) - R(A \cup \{i\}, \hat{\mathbf{v}})$  denote a threshold for buyer  $i$ .
- Let  $b_i(A) := \mathbb{E}_{\hat{\mathbf{v}}} [b_i(A, \hat{\mathbf{v}})]$  denote the base price for buyer  $i$ .

Starting with  $A_0 = \emptyset$ , let  $A_t^{\mathbf{T}}$  denote the set of accepted buyers *before* time  $t$  when buyers take values  $\mathbf{v}$  and arrive in order  $\mathbf{T}$ . When  $\mathbf{T}$  is clear from context, we abuse notation and use  $A_t$  instead of  $A_t^{\mathbf{T}}$ . Suppose a buyer  $i$  arrives at time  $t$ , then our algorithm selects  $i$  iff both  $v_i > \alpha(t) \cdot b_i(A_t)$  and selecting  $i$  is feasible in  $\mathcal{M}$ .

Consider the function  $r(t) := \mathbb{E}_{\mathbf{v}, \hat{\mathbf{v}}, \mathbf{T}} [R(A_t, \hat{\mathbf{v}})]$ , where  $A_t$  is a function of  $\mathbf{v}$ . Clearly,  $r(0) = \mathbb{E}[OPT]$ . Using the following Lemma 7.5 and Claim 7.3, we prove that  $r$  is a residual function. Hence, Lemma 7.1 implies Theorem 7.1.

**Claim 7.3.**

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}} [Revenue] = - \int_{t=0}^1 \alpha(t) \cdot r'(t) dt.$$

**Proof.** Consider the time from  $t$  to  $t + \epsilon$  for some  $t \in [0, 1]$ ,  $\epsilon > 0$ . Let us fix the arrival times  $\mathbf{T}$  and values  $\mathbf{v}$  of all elements. This also fixes the sets  $(A_t)_{t \in [0, 1]}$ . Let  $i_1, \dots, i_k$  be the arrivals between  $t$  and  $t + \epsilon$  that get accepted in this order. Note that it is also possible that  $k = 0$ . The revenue obtained between  $t$  and  $t + \epsilon$  is now given as

$$\begin{aligned} \text{Revenue}_{\leq t+\epsilon} - \text{Revenue}_{\leq t} &= \sum_{j=1}^k \alpha(t_{i_j}) b_{i_j}(A_{t_{i_j}}) \\ &= \sum_{j=1}^k \alpha(t_{i_j}) \mathbb{E}_{\hat{\mathbf{v}}} [R(A_t \cup \{i_1, \dots, i_{j-1}\}, \hat{\mathbf{v}}) - R(A_t \cup \{i_1, \dots, i_j\}, \hat{\mathbf{v}})] \\ &\geq \alpha(t + \epsilon) \mathbb{E}_{\hat{\mathbf{v}}} [R(A_t, \hat{\mathbf{v}}) - R(A_{t+\epsilon}, \hat{\mathbf{v}})]. \end{aligned}$$

Taking the expectation over  $\mathbf{v}$  and  $\mathbf{T}$ , we get by linearity of expectation

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[\text{Revenue}_{\leq t+\epsilon}] - \mathbb{E}_{\mathbf{v}, \mathbf{T}}[\text{Revenue}_{\leq t}] \geq \alpha(t + \epsilon)(r(t) - r(t + \epsilon)).$$

By the same argument, we also have

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[\text{Revenue}_{\leq t+\epsilon}] - \mathbb{E}_{\mathbf{v}, \mathbf{T}}[\text{Revenue}_{\leq t}] \leq \alpha(t)(r(t) - r(t + \epsilon)).$$

In combination, we get that

$$\frac{d}{dt} \mathbb{E}_{\mathbf{v}, \mathbf{T}}[\text{Revenue}_{\leq t}] = -\alpha(t)r'(t),$$

which implies the claim. □

**Lemma 7.5.**

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[\text{Utility}] \geq \int_{t=0}^1 (1 - \alpha(t)) \cdot r(t) dt.$$

**Proof.** The utility of buyer  $i$  arriving at time  $t$  is given by

$$\begin{aligned}\mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] &= \mathbb{E}_{\mathbf{v}, \mathbf{T}_{-i}} \left[ (v_i - \alpha(t) \cdot b_i(A_t))^+ \cdot \mathbf{1}_{i \notin \text{Span}(A_t)} \mid T_i = t \right] \\ &= \mathbb{E}_{\mathbf{v}_{-i}, \mathbf{T}_{-i}} \left[ \mathbb{E}_{v_i} \left[ (v_i - \alpha(t) \cdot b_i(A_t))^+ \right] \cdot \mathbf{1}_{i \notin \text{Span}(A_t)} \mid T_i = t \right]\end{aligned}$$

because  $A_t$  is independent of  $v_i$ . Since  $v_i$  and  $\hat{v}_i$  are identically distributed, we can also write

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] = \mathbb{E}_{\mathbf{v}, \hat{\mathbf{v}}, \mathbf{T}_{-i}} \left[ (\hat{v}_i - \alpha(t) \cdot b_i(A_t))^+ \cdot \mathbf{1}_{i \notin \text{Span}(A_t)} \mid T_i = t \right]. \quad (7.9)$$

Now observe that buyer  $i$  can belong to  $\text{Opt}(\hat{\mathbf{v}} \mid A_t)$  only if it's not already in  $\text{Span}(A_t)$ , which implies  $\mathbf{1}_{i \notin \text{Span}(A_t)} \geq \mathbf{1}_{i \in \text{Opt}(\hat{\mathbf{v}} \mid A_t)}$ . Using this and removing non-negativity, we get

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] \geq \mathbb{E}_{\mathbf{v}, \hat{\mathbf{v}}, \mathbf{T}_{-i}} \left[ (\hat{v}_i - \alpha(t) \cdot b_i(A_t)) \cdot \mathbf{1}_{i \in \text{Opt}(\hat{\mathbf{v}} \mid A_t)} \mid T_i = t \right].$$

Now we use Lemma 7.6 to remove the conditioning on buyer  $i$  arriving at time  $t$  as this gives a valid lower bound on expected utility,

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] \geq \mathbb{E}_{\mathbf{v}, \hat{\mathbf{v}}, \mathbf{T}} \left[ (\hat{v}_i - \alpha(t) \cdot b_i(A_t)) \cdot \mathbf{1}_{i \in \text{Opt}(\hat{\mathbf{v}} \mid A_t)} \right]. \quad (7.10)$$

We can now lower bound sum of buyers' utilities using Eq. (7.10) to get

$$\begin{aligned}\mathbb{E}_{\mathbf{v}, \mathbf{T}}[\text{Utility}] &= \sum_i \int_{t=0}^1 \mathbb{E}_{\mathbf{v}, \mathbf{T}}[u_i \mid T_i = t] \cdot dt \\ &\geq \sum_i \int_{t=0}^1 \mathbb{E}_{\mathbf{v}, \hat{\mathbf{v}}, \mathbf{T}} \left[ (\hat{v}_i - \alpha(t) \cdot b_i(A_t)) \cdot \mathbf{1}_{i \in \text{Opt}(\hat{\mathbf{v}} \mid A_t)} \right] \cdot dt.\end{aligned}$$

By moving the sum over buyers inside the integrals, we get

$$\begin{aligned}\mathbb{E}_{\mathbf{v}, \mathbf{T}}[\text{Utility}] &\geq \int_{t=0}^1 \mathbb{E}_{\mathbf{v}, \hat{\mathbf{v}}, \mathbf{T}} \left[ \sum_i (\hat{v}_i - \alpha(t) \cdot b_i(A_t)) \cdot \mathbf{1}_{i \in \text{Opt}(\hat{\mathbf{v}} \mid A_t)} \right] \cdot dt \\ &= \int_{t=0}^1 \mathbb{E}_{\mathbf{v}, \hat{\mathbf{v}}, \mathbf{T}} \left[ R(A_t, \hat{\mathbf{v}}) - \alpha(t) \cdot \sum_{i \in \text{Opt}(\hat{\mathbf{v}} \mid A_t)} b_i(A_t) \right] \cdot dt.\end{aligned}$$

Finally, using Lemma 7.7 for  $V = \text{Opt}(\hat{\mathbf{v}} \mid A_t)$ , we get

$$\mathbb{E}_{\mathbf{v}, \mathbf{T}}[\text{Utility}] \geq \int_{t=0}^1 \mathbb{E}_{\mathbf{v}, \hat{\mathbf{v}}, \mathbf{T}} [(1 - \alpha(t)) \cdot R(A_t, \hat{\mathbf{v}})] \cdot dt.$$

□

Finally, we prove the missing lemma that removes the conditioning on item  $i$  arriving at  $t$ .

**Lemma 7.6.** *For any  $i$ , any time  $t$ , and any fixed  $\mathbf{v}, \hat{\mathbf{v}}$ , we have*

$$\mathbb{E}_{\mathbf{T}_{-i}} [(\hat{v}_i - \alpha(t)) \cdot b_i(A_t) \cdot \mathbf{1}_{i \in \text{Opt}(\hat{\mathbf{v}} \mid A_t)} \mid T_i = t] \geq \mathbb{E}_{\mathbf{T}} [(\hat{v}_i - \alpha(t)) \cdot b_i(A_t) \cdot \mathbf{1}_{i \in \text{Opt}(\hat{\mathbf{v}} \mid A_t)}].$$

**Proof.** We prove the lemma for any fixed  $\mathbf{T}_{-i}$ . Suppose we draw a uniformly random  $T_i \in [0, 1]$ . Observe that if  $T_i \geq t$  then we have equality in the above equation because set  $A_t$  is the same both with and without  $i$ . This is also the case when  $T_i < t$  but  $i$  is not selected into  $A_t$ . Finally, when  $T_i < t$  and  $i \in A_t$  we have  $\mathbf{1}_{i \in \text{Opt}(\hat{\mathbf{v}} \mid A_t)} = 0$  in the presence of item  $i$  (i.e., RHS of lemma), making the inequality trivially true. □

**Lemma 7.7.** *For any fixed  $\mathbf{v}, \mathbf{T}$ , time  $t$ , and set of elements  $V$  that is independent in the matroid  $\mathcal{M}/A_t$ , we have*

$$\sum_{i \in V} b_i(A_t) \leq \mathbb{E}_{\hat{\mathbf{v}}} [R(A_t, \hat{\mathbf{v}})].$$

**Proof.** By definition

$$\sum_{i \in V} b_i(A_t) = \mathbb{E}_{\hat{\mathbf{v}}} \left[ \sum_{i \in V} (R(A_t, \hat{\mathbf{v}}) - R(A_t \cup \{i\}, \hat{\mathbf{v}})) \right].$$

Fix the values  $\hat{\mathbf{v}}$  arbitrarily, we also have

$$\sum_{i \in V} (R(A_t, \hat{\mathbf{v}}) - R(A_t \cup \{i\}, \hat{\mathbf{v}})) \leq R(A_t, \hat{\mathbf{v}}).$$

This follows from the fact that  $R(A_t, \hat{\mathbf{v}}) - R(A_t \cup \{i\}, \hat{\mathbf{v}})$  are the respective critical values of the greedy algorithm on  $\mathcal{M}/A_t$  with values  $\hat{\mathbf{v}}$ . Therefore, the bound follows from Lemma 3.2 in [LB10]. An alternative proof is given as Proposition 2 in [KW12] while in our case the first inequality can be skipped and the remaining steps can be followed replacing  $A$  by  $A_t$ .

Taking the expectation over  $\hat{\mathbf{v}}$ , the claim follows. □

## 7.5 Fixed Threshold Algorithms

In this section we discuss the powers and limitations of *Fixed-Threshold Algorithms* (FTAs) for single item prophet secretary. In an FTA we set a fixed threshold for the item at the beginning of the process and then assign it to the first buyer whose valuation exceeds the threshold. The motivation to study FTAs comes from their simplicity, transparency, and fairness in the design of a posted price mechanism (see e.g., [FGL15]).

In Section 7.5.1 we give a  $(1 - 1/e)$ -approximation FTA for single-item prophet secretary. Next, in Section 7.5.2 we present an upper bound for FTAs. In particular, we show that there is no FTA, even for identical distributions, with an approximation factor better than  $1 - 1/e$ . This indicates the tightness of our algorithm for prophet secretary. Furthermore, in Appendix E we generalize these single item ideas to present an alternate  $(1 - 1/e)$ -approximation algorithm for bipartite matching

prophet secretary.

### 7.5.1 Single Item Prophet Secretary

Our analysis in this section is based on discrete arrival times for the buyers. In particular, we assume that the buyers arrive based on an initially unknown permutation  $\pi$ , and at every time  $i$  the values of  $\pi(i)$  and  $v_{\pi(i)}$  are revealed in an online fashion. We prove the following result.

**Theorem 7.4.** *There exists a  $(1 - 1/e)$ -approximation FTA for prophet secretary.*

**Proof.** Without loss of generality, we assume that all distributions have a finite expectation and a continuous CDF<sup>3</sup>. As two extreme selections for the threshold, if we set  $\tau$  to zero then the FTA selects the first item, and if we set it to infinity then no item will be selected. Therefore, the assumption for the continuity of the distribution function allows us to select a threshold  $\tau$  such that the FTA reaches the end of the sequence with an exact probability of  $1/e$ . This means all of drawn values are below  $\tau$  with probability  $1/e$ . In the remainder, we show that the FTA based on this choice of  $\tau$  lead to a  $(1 - 1/e)$ -approximation algorithm.

Let  $OPT$  denote the maximum of all  $v_i$ s and  $Alg$  be a random variable that indicates the value selected by the algorithm, or is zero if no item is selected. The goal is to show

$$\mathbb{E}[Alg] \geq \left(1 - \frac{1}{e}\right) \mathbb{E}[OPT] .$$

---

<sup>3</sup>This assumption is without loss because the actual CDF can be approximated with arbitrary precision by a continuous function. This approximation corresponds to a randomized tie-breaking in case of pointmasses.



We have  $\mathbb{E}[\text{Alg}] = \mathbb{E}[\text{Revenue}] + \mathbb{E}[\text{Utility}]$ . Due to the definition of  $\tau$  the algorithm makes a selection with probability  $1 - 1/e$ , therefore  $\mathbb{E}[\text{Revenue}] = (1 - \frac{1}{e}) \tau$ . In the remainder, we complete the proof by showing a lower bound on  $\mathbb{E}[\text{Utility}]$  based on the expectation of  $OPT$  when it is greater than or equal to  $\tau$ .

Let us first define some notations. We use  $q(j)$  to denote the probability that the algorithm does not pick any of the first  $j$  item, i.e.  $q(j) := \Pr[\max_{1 \leq k \leq j} \{v_{\pi(k)}\} < \tau]$ . We use  $q_{-i}(j)$  as the probability of the algorithm not picking any of the first  $j - 1$  items conditioned on the event that item  $i$  appears at position  $j$ . We use the following lemma to lower bound  $q_{-i}(j)$  with  $q(j)$ .

**Lemma 7.8.** *For any item  $1 \leq i \leq n$  and any position  $1 \leq j \leq n$  we have  $q_{-i}(j) \geq q(j)$ .*

**Proof.** Note that  $q(j)$  has two sources of randomness, one for the choices of  $\pi$  and one for the valuations of  $v_i$ 's. The lemma can be proven by carefully analyzing the former, i.e. by considering whether item  $i$  appears among the first  $j - 1$  items or not. More precisely,

$$q(j) = \Pr[\pi^{-1}(i) < j] \Pr \left[ \max_{1 \leq k \leq j} \{v_{\pi(k)}\} < \tau \mid \pi^{-1}(i) < j \right] \quad (7.11)$$

$$+ \Pr[\pi^{-1}(i) \geq j] \Pr \left[ \max_{1 \leq k \leq j} \{v_{\pi(k)}\} < \tau \mid \pi^{-1}(i) \geq j \right] . \quad (7.12)$$

For  $\Pr [\max_{1 \leq k \leq j} \{v_{\pi(k)}\} < \tau \mid \pi^{-1}(i) < j]$  in (7.11) we have choices of  $j - 1$  items other than  $i$  and require all of their values to be below  $\tau$ . Therefore this probability is no more than  $q_{-i}(j)$ . Similarly for  $\Pr [\max_{1 \leq k \leq j} \{v_{\pi(k)}\} < \tau \mid \pi^{-1}(i) \geq j]$  in (7.12) we have choices of at least  $j - 1$  items other than  $i$  and again require all of their

values to be below  $\tau$ . Hence, this term is at most  $q_{-i}(j)$  too, and we have

$$q(j) \leq \Pr[\pi^{-1}(i) < j]q_{-i}(j) + \Pr[\pi^{-1}(i) \geq j]q_{-i}(j) = q_{-i}(j) ,$$

which completes the proof.  $\square$

Now for the utility we have

$$\mathbb{E}[\text{Utility}] = \sum_{i=1}^n \mathbb{E}[u_i] = \sum_{i=1}^n \sum_{j=1}^n \Pr[\pi(j) = i] q_{-i}(j) \mathbb{E}[v_i \cdot \mathbf{1}_{v_i \geq \tau}] = \sum_{i=1}^n \sum_{j=1}^n q_{-i}(j) \mathbb{E}[v_i \cdot \mathbf{1}_{v_i > \tau}] \frac{1}{n}.$$

By applying Lemma 7.8 we get

$$\begin{aligned} \mathbb{E}[\text{Utility}] &\geq \sum_{i=1}^n \sum_{j=1}^n q(j) \mathbb{E}[v_i \cdot \mathbf{1}_{v_i \geq \tau}] \frac{1}{n} && \text{Lemma 7.8} \\ &= \sum_{i=1}^n \mathbb{E}[v_i \cdot \mathbf{1}_{v_i \geq \tau}] \sum_{j=1}^n q(j) \frac{1}{n} && (7.13) \end{aligned}$$

$$\begin{aligned} &= \mathbb{E}\left[\sum_{i=1}^n v_i \cdot \mathbf{1}_{v_i \geq \tau}\right] \sum_{j=1}^n q(j) \frac{1}{n} \\ &\geq \mathbb{E}[OPT \cdot \mathbf{1}_{OPT \geq \tau}] \sum_{j=1}^n q(j) \frac{1}{n}. && (7.14) \end{aligned}$$

To complete the proof of the theorem we need to show that the sum in Inequality (7.14) is at least  $1 - 1/e$ . Define  $p(i) := \Pr[v_i < \tau]$ . For every  $j$  we have

$$\begin{aligned} q(j) &= \Pr\left[\max_{1 \leq k \leq j} \{v_{\pi(k)}\} < \tau\right] = \mathbb{E}_{\pi} \left[ \prod_{k=1}^j p(\pi(k)) \right] = \mathbb{E}_{\pi} \left[ \exp\left(\sum_{k=1}^j \ln p(\pi(k))\right) \right] \\ &\stackrel{\text{exp is convex}}{\geq} \exp\left(\mathbb{E}_{\pi} \left[ \sum_{k=1}^j \ln p(\pi(k)) \right]\right) = \exp\left(\frac{j}{n} \sum_{k=1}^n \ln p(k)\right) = \exp\left(-\frac{j}{n}\right) . \end{aligned}$$

The last inequality holds because our choice of  $\tau$  results in  $q(n) = \prod_{i=1}^n p(i) = 1/e$ .

Therefore we have

$$\sum_{j=1}^n q(j) \frac{1}{n} \geq \sum_{j=1}^n \exp\left(-\frac{j}{n}\right) \cdot \frac{1}{n} \geq \int_0^1 \exp(-x) dx = 1 - \frac{1}{e} .$$

This gives us  $\mathbb{E}[\text{Utility}] \geq (1 - 1/e) \mathbb{E}[\text{OPT} \cdot \mathbf{1}_{\text{OPT} \geq \tau}]$ . Now we are ready to wrap up the proof. We have

$$\begin{aligned}
\mathbb{E}[\text{Alg}] &= \mathbb{E}[\text{Revenue}] + \mathbb{E}[\text{Utility}] \\
&\geq \left(1 - \frac{1}{e}\right) \tau + \left(1 - \frac{1}{e}\right) \mathbb{E}[\text{OPT} \cdot \mathbf{1}_{\text{OPT} \geq \tau}] \\
&\geq \left(1 - \frac{1}{e}\right) (\mathbb{E}[\text{OPT} \cdot \mathbf{1}_{\text{OPT} < \tau}] + \mathbb{E}[\text{OPT} \cdot \mathbf{1}_{\text{OPT} \geq \tau}]) \\
&= \left(1 - \frac{1}{e}\right) \mathbb{E}[\text{OPT}] ,
\end{aligned}$$

which completes the proof.  $\square$

## 7.5.2 Impossibility for IID Prophet Inequalities

In the following we prove an impossibility result for FTAs for single item prophet secretary. We show this impossibility even for the special case of iid items. For every  $n$ , we give a common distribution  $D$  for every item such that no FTA can achieve an approximation factor better than  $1 - 1/e$ . This also implies the tightness of the algorithm discussed in Section 7.5.1.

**Theorem 7.5.** *Any FTA for iid prophet inequality is at most  $(1 - \frac{1}{e} + O(\frac{1}{n}))$ -approximation.*

**Proof.** We prove the theorem by giving a hard input instance for every  $n$  as follows: every  $v_i$  is  $n/(e - 1)$  with probability  $1/n^2$  and is  $(e - 2)/(e - 1)$  otherwise. The expected maximum value of these  $n$  items is

$$\mathbb{E}[\text{OPT}] = \left(1 - \frac{1}{n^2}\right)^n \cdot \frac{e - 2}{e - 1} + \left(1 - \left(1 - \frac{1}{n^2}\right)^n\right) \frac{n}{e - 1} = 1 - O\left(\frac{1}{n}\right) .$$

In this instance, if  $\tau < (e-2)/(e-1)$  then the algorithm selects the first item, and if  $(e-2)/(e-1) < \tau \leq n/(e-1)$  then the algorithm can only select  $n/(e-1)$ . In these cases the approximation factor can be at most  $(e-2)/(e-1) \approx 0.58$ .

Now, note that the CDF of this input distribution is not continuous. Reshaping a discrete distribution function into a continuous one, however, does not change the approximation factor because for example in the above instance we only need a very slight change at the point  $(e-2)/(e-1)$  of the CDF. This change gives us a randomness when  $\tau = (e-2)/(e-1)$ , which is equivalent to flipping a random coin and skipping every item with some probability  $p \leq 1 - 1/n^2$  if the drawn value is  $(e-2)/(e-1)$ . With this assumption we have

$$\begin{aligned} \mathbb{E}[\text{Alg}] &= \sum_{i=1}^n p^i \mathbb{E}[v_i \cdot \mathbf{1}_{v_i \geq \tau}] = \frac{1-p^n}{1-p} \mathbb{E}[v_i \cdot \mathbf{1}_{v_i \geq \tau}] \\ &= \frac{1-p^n}{1-p} \left( \left(1 - \frac{1}{n^2} - p\right) \frac{e-2}{e-1} + \frac{1}{n^2} \frac{n}{e-1} \right) \\ &< \frac{1-p^n}{e-1} \left( e-2 + \frac{1}{n(1-p)} \right). \end{aligned} \tag{7.15}$$

To complete the proof, it suffices to show that the right hand side of Inequality (7.15) is at most  $1 - 1/e + O(1/n)$ . To this end, we try to maximize this term based on parameter  $c$  where  $p = 1 - c/n$ . We can rewrite the right hand side of the inequality as

$$\frac{1 - (1 - \frac{c}{n})^n}{e-1} \left( e-2 + \frac{1}{c} \right).$$

If  $c \in \Theta(n)$  then this term is at most  $(e-2 + \Theta(1/n))/(e-1) \approx 0.41 + O(1/n)$  which is below  $1 - 1/e$  for sufficiently large  $n$ . Otherwise  $c/n \ll 1$  and we can approximate  $(1 - c/n)^n$  as  $e^{-c} + O(1/n)$ . This upper bounds Inequality (7.15) by

$(1 - e^{-c})(e - 2 + 1/c)/(e - 1) + O(1/n)$ , where the first term is independent of  $n$  and is at most  $1 - 1/e$  for different constants  $c$ ; thereby completing the proof.  $\square$

We would like to note that the continuity of the CDF of the input distributions is a useful and natural property that can be used by an FTA. This is because making this assumption allows us to design a  $(1 - 1/e)$ -approximation algorithm, as shown by Theorem 7.4, but not assuming this puts a barrier of  $1/2$  for any FTA, which is shown by [EHL15]<sup>4</sup>. For example, in the above instance the approximation factor without assuming continuity would be at most  $(e - 2)/(e - 1) \approx 0.58$ , which is below the  $1 - 1/e \approx 0.63$  claim of Theorem 7.4. This contradiction is because without this assumption on the input distribution the algorithm could not set  $\tau$  in a way that the probability of selecting an item became exactly  $1 - 1/e$ .

---

<sup>4</sup>Their hardness instance contains different distributions, hence does not necessarily apply to iid prophet inequality.

## Chapter 8: Stochastic $k$ -Server Problem

### 8.1 Introduction

The  $k$ -server problem is one of the most fundamental problems in online computation that has been extensively studied in the past decades. In the  $k$ -server problem we have  $k$  mobile servers on a metric space  $\mathcal{M}$ . We receive an online sequence of  $t$  requests where the  $i^{\text{th}}$  request is a point  $r_i \in \mathcal{M}$ . Upon the arrival of  $r_i$ , we need to move a server to  $r_i$ , at a cost equal to the distance from the current position of the server to  $r_i$ . The goal is to minimize the total cost of serving all requests.

Manasse, McGeoch, and Sleator [MMS90] introduced the  $k$ -server problem as a natural generalization of several online problems, and a building block for other problems such as the metrical task systems. They considered the adversarial model, in which the online algorithm has no knowledge of the future requests. Following the proposition of Sleator and Tarjan [ST85], they evaluate the performance of an online algorithm using competitive analysis. In this model, an online algorithm ALG is compared to an *offline* optimum algorithm  $OPT$  which is aware of the entire input in advance. For a sequence of requests  $\rho$ , let  $|\text{ALG}(\rho)|$  and  $|\text{OPT}(\rho)|$  denote the total cost of ALG and  $OPT$  for serving  $\rho$ . An algorithm is  $c$ -competitive if for every

$\rho$ ,  $|\text{ALG}(\rho)| \leq c|\text{OPT}(\rho)| + c_0$  where  $c_0$  is independent of  $\rho$ .

Manasse *et al.* [MMS90] showed a lower bound of  $k$  for the competitive ratio of any deterministic algorithm in any metric space with at least  $k + 1$  points. The celebrated *k-server conjecture* states that this bound is tight for general metrics. For several years the known upper bounds were all exponential in  $k$ , until a major breakthrough was achieved by Koutsoupias and Papadimitriou [KP95], who showed that the so-called *work function algorithm* is  $(2k - 1)$ -competitive. Proving the tight competitive ratio has been the “holy grail” of the field in the past two decades. This challenge has led to the study of the problem in special spaces such as the uniform metric (also known as the paging problem), line, circle, and trees metrics (see [CKPV91, CL91] and references therein). We also refer the reader to Section 8.1.3 for a short survey of randomized algorithms, particularly the recent result of Bansal, Buchbinder, Madry, and Naor [BBMN11] which achieves the competitive ratio of  $O(\log^3 n \log^2 k)$  for discrete metrics that comprise  $n$  points.

The line metric (or Euclidean 1-dimensional metric space) is of particular interest for developing new ideas. Chrobak, Karloof, Payne, and Vishwnathan [CKPV91] were the first to settle the conjecture in the line by designing an elegant  $k$ -competitive algorithm. Chrobak and Larmore [CL91] generalized this approach to tree metrics. Later, Bartal and Koutsoupias [BK04] proved that the work function algorithm is also  $k$ -competitive in line. Focusing on the special case of  $k = 2$  in line, Bartal *et al.* [BCL00] show that, using randomized algorithms, one can break the barrier of lower bound  $k$  by giving a 1.98-competitive algorithm for the case where we only have two servers.

Despite the strong lower bounds for the  $k$ -server problem, there are heuristics algorithms that are *constant* competitive in practice. For example, for the paging problem- the special case of uniform metric- the least recently used (LRU) strategy is shown to be experimentally constant competitive (see Section 8.1.3). In this chapter we propose an algorithm and run it on real world data to measure its performance. In particular we use the distribution of car accidents obtained from road safety data. Our experiments illustrate our algorithm is performing even better in practice. The idea of comparing the performance of an online algorithm (with zero-knowledge of the future) to the request-aware offline optimum has led to crisp and clean solutions. However, that is not without its downsides. The results in the online model are often very pessimistic leading to theoretical guarantees that are hardly comparable to experimental results. Indeed, one way to tighten this gap is to use stochastic information about the input data as we describe in this chapter.

We should also point out that the competitive analysis is not the only possible or necessarily the most suitable approach for this problem. Since the distributions from which the input is generated are known, one can use dynamic programming (or enumeration of future events) to derive the optimal movement of servers. Unfortunately, finding such an optimal online solution using the distributions is an NP-hard problem <sup>1</sup>, thus the dynamic programming or any other approach takes exponential time. This raises the question that how well one can perform in comparison to the

---

<sup>1</sup>Reduction from  $k$ -median to Stochastic  $k$ -server: to find the  $k$  median of set  $S$  of vertices, one can construct an instance of stochastic  $k$ -server with  $t = 1$  and  $P_1(v) = 1/|S|$  for every  $v \in S$ . The best initialization of the servers gives the optimum solution to  $k$ -median of  $S$ .



best online solution. In the rest of the chapter we formally define the model and address this question.

### 8.1.1 The Stochastic Model

In this chapter, we study the *stochastic  $k$ -server problem* where the input is not chosen adversarially, but consists of draws from given probability distributions. This problem has applications such as equipment replacement in data centers. The current mega data centers contain hundreds of thousands of servers and switches with limited life-span. For example servers usually retire after a few years. The only efficient way to scale up the maintenance in data centers is by automation, and robots are designed to handle maintenance tasks such as repairs or manual operations on servers. The replacement process can be modeled as requests that should be satisfied by robots, and robots can be modeled as servers. This problem also has applications in physical networks. As an example, suppose we model a shopping service (e.g. Google Express) as a  $k$ -server problem in which we receive an online sequence of shopping requests for different stores. We have  $k$  shopping cars (i.e., servers) that can serve the requests by traveling to the stores. It is quite natural to assume that on a certain time of the week/day, the requests arrive from a distribution that can be discovered by analyzing the history. We formalize this stochastic information as follows.

For every  $i \in [1 \cdots t]$ , a discrete probability distribution  $P_i$  is given in advance from which request  $r_i$  will be drawn at time step  $i$ . The distributions are chosen by

the adversary and are assumed to be independent but not necessarily identical. This model is inspired by the well-studied model of *prophet inequalities*<sup>2</sup> [KS77,HKS07]. As mentioned before, the case of line metric has proven to be a very interesting restricted case for studying the  $k$ -server problem. In this thesis, we focus mainly on the class of line metric though our results carry over to circle metric and general metrics as well.

In the adversarial model, the competitive ratio seems to be the only well-defined notion for analyzing the performance of online algorithms. However, in the presence of stochastic information, one can derive a much better benchmark that allows us to make fine-grained distinctions between the online algorithms. We recall that in the offline setting, for a class of algorithms  $\mathcal{C}$ , the natural notion to measure the performance of an algorithm  $\text{ALG} \in \mathcal{C}$  is the *approximation ratio* defined as the worse case ratio of  $|\text{ALG}|$  to  $|\text{OPT}(\mathcal{C})|$  where  $\text{OPT}(\mathcal{C})$  is the optimal algorithm in the class. In this thesis, we also measure the performance of an online algorithm by its approximation ratio— compared to the *optimal online solution*. We note that given distributions  $P_1, \dots, P_t$ , one can iteratively compute the optimal online solution by solving the following exponential-size dynamic program: for every  $i \in [0 \dots t]$  and every possible placement  $A$  of  $k$  servers (called a *configuration*) on the metric, let  $\tau(i, A)$  denote the minimum expected cost of an online algorithm for serving the

---

<sup>2</sup>In the prophet inequality setting, given (not necessarily identical) distributions  $P_1, \dots, P_t$ , an online sequence of values  $x_1, \dots, x_n$  where  $x_i$  is drawn from  $P_i$ , an onlooker has to choose one item from the succession of the values, where  $x_i$  is revealed at step  $i$ . The onlooker can choose a value only at the time of arrival. The goal is to maximize the chosen value.

first  $i$  requests and then moving the servers to configuration  $A$ . Note that  $\tau(i, A)$  can inductively be computed via the following recursive formula

$$\tau(i, A) = \min_B \tau(i-1, B) + \mathbb{E}_{r_i \sim P_i} [\text{min. distance from } B \text{ to } A \text{ subject to serving } r_i] ,$$

where  $\tau(0, A)$  is initially zero for every  $A$ .

### 8.1.2 Our Results

Our first main result is designing a constant approximation algorithm in the line metric when the distributions for different time steps are not necessarily identical.

**Theorem 8.1.** *There exists a 3-approximation online algorithm for the stochastic  $k$ -server problem in the line metric. The running time is polynomial in  $k$  and the size of the support of the input distributions. The same guarantee holds for the circle metric.*

For the general metric, we present an algorithm with a logarithmic approximation guarantee.

**Theorem 8.2.** *There exists a  $O(\log n)$ -approximation online algorithm for the stochastic  $k$ -server problem in a general metric of size  $n$ .*

We prove the theorems using two important structural results. The first key ingredient is a general reduction from class of online algorithms to a restricted class of *non-adaptive algorithms* while losing only a constant factor in the approximation ratio. Recall that a configuration is a placement of  $k$ -servers on the metric. We

say an algorithm ALG is *non-adaptive* if it follows the following procedure: ALG pre-computes a sequence of configurations  $A_0, A_1, \dots, A_t$ . We start by placing the  $k$ -servers on  $A_0$ . Upon the arrival of  $r_i$ , (i) we move the servers to configuration  $A_i$ ; next (ii) we move the closest server  $s$  to  $r_i$ ; and finally (iii) we return  $s$  to its original position in  $A_i$ . We first prove the following structural result.

**Theorem 8.3.** *For the stochastic  $k$ -server problem in the general metric, the optimal non-adaptive online algorithm is within 3-approximation of the optimal online algorithm.*

Using the aforementioned reduction, we focus on designing the optimal non-adaptive algorithm. We begin by formulating the problem as an integer program. The second ingredient is to use the relaxation of this program to formalize a natural *fractional variant* of the problem. In this variant, a configuration is a fractional assignment of *server mass* to the points of the metric such that the total mass is  $k$ . To serve a request at point  $r_i$ , we need to move some of the mass to have at least one amount of server mass on  $r_i$ . The cost of moving the server mass is naturally defined as the integral of the movement of infinitesimal pieces of the server mass. By solving the linear relaxation of the integer program, we achieve the optimal fractional non-adaptive algorithm. We finally prove Theorems 8.1 and 8.2 by leveraging the following rounding techniques. The rounding method in line has been also observed by Türkoglu [Tür05]. We provide the proof for the case of line in Section 8.5 for the sake of completeness. The rounding method for general metrics is via the well-known embedding of a metric into a distribution of well-separated

trees while losing a logarithmic factor in the distortion. Bansal *et al.* [BBMN11] use a natural rounding method similar to that of Blum, Burch, and Kalai [BBK99] to show that any fractional  $k$ -server movement on well-separated trees can be rounded to an integral counterpart by losing only a constant factor.

**Theorem 8.4** (first proven in [Tür05]). *Let  $\text{ALG}_f$  denote a fractional  $k$ -server algorithm in the line, or circle. One can use  $\text{ALG}_f$  to derive a randomized integral algorithm  $\text{ALG}$  such that for every request sequence  $\sigma$ ,  $\mathbb{E}[|\text{ALG}(\sigma)|] = |\text{ALG}_f(\sigma)|$ . The expectation is over the internal randomness of  $\text{ALG}$ . Furthermore, in the stochastic model  $\text{ALG}$  can be derandomized.*

**Theorem 8.5** (proven in [BBMN11]). *Let  $\text{ALG}_f$  denote a fractional  $k$ -server algorithm in any metric. One can use  $\text{ALG}_f$  to derive a randomized integral algorithm  $\text{ALG}$  such that for every request sequence  $\sigma$ ,  $\mathbb{E}[|\text{ALG}(\sigma)|] \leq O(\log n) |\text{ALG}_f(\sigma)|$ .*

We further show that in the stochastic setting, if the number of possible input scenarios is  $m$ , even if the distributions are correlated, one can compute the best *fractional* online competitive algorithm in time polynomial in  $m$  and  $n$ . Note that since the number of placements of  $k$  servers on  $n$  points is exponential, it is not possible to enumerate all the possible choices of an online algorithm. We solve this problem by presenting a non-trivial LP relaxation of the problem with size polynomial in  $n$  and  $m$ ; therefore obtaining the following result. We present the formal model and analysis in Appendix F.

**Theorem 8.6.** *The optimal online algorithm of the stochastic  $k$ -server problem with correlated setting in line and circle can be computed in polynomial time w.r.t.*

the number of possible scenarios. In general metrics, an  $O(\log n)$ -approximation algorithm can be obtained.

### 8.1.3 Further Related Work

The randomized algorithms often perform much better in the online paradigm. For the  $k$ -server problem, a lower bound of  $\Omega(\log k)$  is shown by [KRR94] for the competitive ratio of randomized algorithms in most common metrics. Despite the exponential gap, compared to the lower bound of deterministic algorithms, very little is known about the competitiveness of randomized algorithms. In fact, the only known algorithms with competitive ratios below  $k$ , work either in the uniform metric (also known as the paging problem [FKL+91, MS91, ACN00, BBN12]), a metric comprising  $k + 1$  points [FM03], and two servers on the line [BCL00]. Two decades after the introduction of the  $k$ -server problem, a major breakthrough was achieved by Bansal *et al.* [BBMN11] in discrete metrics with sub-exponential size. If  $\mathcal{M}$  comprise  $n$  points, their randomized algorithm achieves a competitive ratio of  $O(\log^3 n \log^2 k)$ .

The case of uniform metric has been extensively studied under various stochastic models motivated by the applications in computer caching. Koutsoupias and Papadimitriou [KP95] consider two refinements of the competitive analysis for server problems. First, they consider the *diffuse adversary* model. In this model, at every step  $i$  the adversary chooses a distribution  $D_i$  over the uniform metric of the paging problem. Then the  $i^{\text{th}}$  request is drawn from  $D_i$  which needs to be served. The

distribution  $D_i$  is not known to the online algorithm and it may depend on the previous requests. However, in their chapter, they consider the case wherein it is guaranteed that for every point  $p$ ,  $D_i(p) \leq \epsilon$  for a small enough  $\epsilon$ ; i.e., the next request is not predictable with absolute certainty for the adversary. The results of Koutsoupias and Papadimitriou and later Young [You98] shows that the optimum competitive ratio in this setting is close to  $1 + \Theta(k\epsilon)$ .

The second refinement introduced in [KP95] restricts the optimal solution to having lookahead at most  $\ell$ . Hence, one can define a *comparative ratio* which indicates the worst-case ratio of the cost of the best online solution to the best solution with lookahead  $\ell$ . They show that for the  $k$ -server problem, and more generally the metrical task system problem, there are online algorithms that admit a comparative ratio of  $2\ell + 1$ ; for some instances this ratio is tight.

Later, Panagiotou and Souza [PS06] considered the paging problem when the adversary is restricted to certain local constraints on the request. These constraints are motivated by the locality of reference in a memory cache and typical memory-access patterns. They show that under these constraints the LRU algorithm is constant competitive, which indeed gives a theoretical explanation to why LRU works pretty well in practice.

Various other models of restricting the adversary (access graph model [BIRS95, IKP96, FM97], fault rate model [KPR00, AFG02, Den83], etc) have also been considered for the paging problem (see [PS06, Bec04] and references therein for a further survey of these results). Unfortunately, many of the stochastic settings considered for the paging problem do not seem to have a natural generalization beyond the uni-

form metric setting. For example, in the diffuse adversary model, most of the studied distributions do not weaken the adversary in the general metric. In this chapter, we look for polynomial-time *approximation* algorithms in the class of online algorithms that have access to the distributions.

We would like to mention that various online problems have been previously considered under prophet inequality model or i.i.d. model (where all distributions are identical). Motivated by ad auctions, the maximum matching problem has been extensively studied in these models, achieving near one competitive ratios [AHL12, AHL<sup>+</sup>11, AHL13]. In the graph connectivity problems, Garg, Gupta, Leonardi, and Sankowski [GGLS08] consider the online variants of Steiner tree and several related problems under the i.i.d. stochastic model. In the adversarial model, there exists an  $\Omega(\log n)$  lower bound on the competitive ratio of any online algorithm, where  $n$  is the number of demands. However, Garg *et al.* show that under the i.i.d. assumption, these problems admit online algorithms with constant or  $O(\log \log n)$  competitive ratios. We refer the reader to the excellent book by Borodin and El-Yaniv [BEY05] for further study of online problems.

## 8.2 Preliminaries

In this section we formally define the stochastic  $k$ -server problem. The classical  $k$ -server problem is defined on a metric  $\mathcal{M}$  which consists of points that could be infinitely many. For every two points  $x$  and  $y$  in metric  $\mathcal{M}$ , let  $d(x, y)$  denote the distance of  $x$  from  $y$  which is a symmetric function and satisfies the triangle



inequality. More precisely for every three points  $x$ ,  $y$ , and  $z$  we have

$$d(x, x) = 0 \tag{8.1}$$

$$d(x, y) = d(y, x) \tag{8.2}$$

$$d(x, y) + d(y, z) \geq d(x, z). \tag{8.3}$$

In the  $k$ -server problem the goal is to place  $k$  servers on  $k$  points of the metric, and move these servers to satisfy the requests. We refer to every placement of the servers on the metric points by a *configuration*. Let  $\rho = \langle r_1, r_2, \dots, r_t \rangle$  be a sequence of requests, the goal of the  $k$ -server problem is to find configurations  $\langle A_0, A_1, A_2, \dots, A_t \rangle$  such that for every  $i$  there exists a server on point  $r_i$  in configuration  $A_i$ . We say such a list of configurations is *valid* for the given list of requests. A valid sequence of configurations is optimal if  $\sum d(A_{i-1}, A_i)$  is minimized where  $d(X, Y)$  stands for the minimum cost of moving servers from configuration  $X$  to configuration  $Y$ . An optimal sequence  $\langle A_0, A_1, \dots, A_t \rangle$  of configurations is called an optimal offline solution of OFKS( $\mathcal{M}, \rho$ ) when  $\rho$  is known in advance. We refer to the optimal cost of such movements with  $|\text{OFKS}(\mathcal{M}, \rho)| = \sum d(A_{i-1}, A_i)$ .

We also define the notion of *fractional configuration* as an assignment of the metric points to non-negative real numbers. More precisely, each number specifies a mass of fractional server on a point. Every fractional solution adheres to the following condition: The total sum of the values assigned to all points is exactly equal to  $k$ . Analogously, a fractional configuration serves a request  $r_i$  if there is a mass of size at least 1 of server assigned to point  $r_i$ . An offline fractional solution of the  $k$ -server problem for a given sequence of requests  $\rho$  is defined as a sequence

of fractional configurations  $\langle A_0, A_1, \dots, A_t \rangle$  such that  $A_i$  serves  $r_i$ .

In the online  $k$ -server problem, however, we're not given the whole sequence of requests in the beginning, but we will be informed of every request once it is realized. An algorithm  $\mathbb{A}$  is an online algorithm for the  $k$ -server problem if it reports a configuration  $A_0$  as an initial configuration and upon realization of every request  $r_i$  it returns a configuration  $A_i$  such that  $\langle A_0, A_1, A_2, \dots, A_i \rangle$  is valid for  $\langle r_1, r_2, \dots, r_i \rangle$ . If  $\mathbb{A}$  is deterministic, it generates a unique sequence of configurations for every sequence of requests. Let  $\mathbb{A}(\mathcal{M}, \rho)$  be the sequence that  $\mathbb{A}$  generates for requests in  $\rho$  and  $|\mathbb{A}(\mathcal{M}, \rho)|$  denote its cost.

In the online stochastic  $k$ -server problem, in addition to metric  $\mathcal{M}$ , we are also given  $t$  independent probability distributions  $\langle P_1, P_2, \dots, P_t \rangle$  which show the probability that every request  $r_i$  is realized on a point of the metric at each time. An algorithm  $\mathbb{A}$  is an online algorithm for such a setting, if it generates a configuration for every request  $r_i$  not solely based on  $\langle r_1, r_2, \dots, r_i \rangle$  and  $\langle A_0, A_1, \dots, A_{i-1} \rangle$  but also with respect to the probability distributions. Similarly, we define the cost of an online algorithm  $\mathbb{A}$  for a given sequence of requests  $\rho$  with  $|\mathbb{A}(\mathcal{M}, \rho, \langle P_1, P_2, \dots, P_t \rangle)|$ . We define the expected cost of an algorithm  $\mathbb{A}$  on metric  $\mathcal{M}$  and with probability distributions  $\langle P_1, P_2, \dots, P_t \rangle$  by

$$|\mathbb{A}(\mathcal{M}, \langle P_1, P_2, \dots, P_t \rangle)| = \mathbb{E}_{\forall i, r_i \sim P_i} |\mathbb{A}(\mathcal{M}, \rho, \langle P_1, P_2, \dots, P_t \rangle)|.$$

For every metric  $\mathcal{M}$  and probability distributions  $\langle P_1, P_2, \dots, P_t \rangle$  we refer to the online algorithm with the minimum expected cost by  $OPT_{\mathcal{M}, \langle P_1, P_2, \dots, P_t \rangle}$ .

An alternative way to represent a solution of the  $k$ -server problem is as a

vector of configurations  $\langle B_0, B_1, \dots, B_t \rangle$  such that  $B_i$  does not necessarily serve request  $r_i$ . The cost of such solution is equal to  $\sum d(B_{i-1}, B_i) + \sum 2d(B_i, r_i)$  where  $d(B_i, r_i)$  is the minimum distance of a server in configuration  $B_i$  to request  $r_i$ . The additional cost of  $2d(B_i, r_i)$  can be thought of as moving a server from  $B_i$  to serve  $r_i$  and returning it back to its original position. Thus, every such representation of a solution can be transformed to the other representation. Similarly,  $d(B_i, r_i)$  for a fractional configuration  $B_i$  is the minimum cost which is incurred by placing a mass 1 of server at point  $r_i$ . We use letter  $B$  for the configurations of such solutions throughout this chapter.

In this chapter the emphasis is on the stochastic  $k$ -server problem on the line metric. We define the line metric  $\mathcal{L}$  as a metric of points from  $-\infty$  to  $+\infty$  such that the distance of two points  $x$  and  $y$  is always equal to  $|x - y|$ . Moreover, we show that deterministic algorithms are as powerful as randomized algorithms in this setting, therefore we only focus on deterministic algorithms here. Thus, from here on, we omit the term deterministic and every time we use the word algorithm we mean a deterministic algorithm unless otherwise is explicitly mentioned.

### 8.3 Structural Characterization

In this section we define a class of online algorithms for the stochastic  $k$ -server problem and show an important structural property for this class. Later, we leverage this property to provide a polynomial time algorithm for the problem. Although we give the algorithm for the line metric, the structural characterization holds for

general graphs and is of independent interest.

Recall that an online algorithm  $\mathbb{A}$  has to fulfill the task of reporting a configuration  $A_i$  upon arrival of request  $r_i$  based on  $\langle A_0, A_1, \dots, A_{i-1} \rangle$ ,  $\langle r_1, r_2, \dots, r_i \rangle$ , and  $\langle P_1, P_2, \dots, P_t \rangle$ . We say an algorithm  $\mathbb{B}$  is *request oblivious*, if it reports configuration  $B_i$  regardless of request  $r_i$ . As such,  $\mathbb{B}$  generates configurations  $\langle B_0, B_1, \dots, B_t \rangle$  for a sequence of requests  $\langle r_1, r_2, \dots, r_t \rangle$  and the cost of such configuration is  $\sum d(B_{i-1}, B_i) + \sum 2d(B_i, r_i)$ . More precisely, no matter what request  $r_i$  is,  $\mathbb{B}$  will generate the same configuration for a given list of past configurations  $\langle B_0, B_1, \dots, B_{i-1} \rangle$ , a given sequence of past requests  $\langle r_1, r_2, \dots, r_{i-1} \rangle$ , and the sequence of probability distributions  $\langle P_1, P_2, \dots, P_t \rangle$ . In the following we show that every online algorithm  $\mathbb{A}$  can turn into a request oblivious algorithm  $\mathbb{B}_{\mathbb{A}}$  that has a cost of at most  $3|\mathbb{A}(\mathcal{M}, \rho, \langle P_1, P_2, \dots, P_t \rangle)|$  for a given sequence of requests  $\rho$ .

**Lemma 8.1.** *Let  $\mathbb{A}$  be an online algorithm for the stochastic  $k$ -server problem. For any metric  $\mathcal{M}$ , there exists a request oblivious algorithm  $\mathbb{B}_{\mathbb{A}}$  such that*

$$|\mathbb{B}_{\mathbb{A}}(\mathcal{M}, \langle P_1, P_2, \dots, P_t \rangle)| \leq 3|\mathbb{A}(\mathcal{M}, \langle P_1, P_2, \dots, P_t \rangle)|.$$

**Proof.** Let  $\rho$  be a sequence of requests. We define online algorithm  $\mathbb{B}_{\mathbb{A}}$  as follows: The configuration that  $\mathbb{B}_{\mathbb{A}}$  reports for a given list of input arguments  $\langle B_0, B_1, \dots, B_i \rangle$ ,  $\langle r_1, r_2, \dots, r_i \rangle$ , and  $\langle P_1, P_2, \dots, P_t \rangle$  is the output of algorithm  $\mathbb{A}$  on inputs  $\langle B_0, B_1, \dots, B_i \rangle$ ,  $\langle r_1, r_2, \dots, r_{i-1} \rangle$ , and  $\langle P_1, P_2, \dots, P_t \rangle$  (The same input except that  $r_i$  is dropped from the sequence of requests). We show the cost of such algorithm for input  $\rho$  is at most 3 times the cost of  $\mathbb{A}$  for the same input.

Let  $\langle A_0, A_1, \dots, A_t \rangle$  be the sequence of configurations that  $\mathbb{A}$  generates for

requests  $\rho$  and  $\langle B_0, B_1, \dots, B_t \rangle$  be the output of algorithm  $\mathbb{B}_A$ . According to the construction of  $\mathbb{B}_A$ ,  $B_0 = A_0$  and  $B_i = A_{i-1}$  for all  $1 \leq i \leq t$ . Note that for algorithm  $\mathbb{A}$ , we assume every  $A_i$  serves request  $r_i$ . By definition, the cost of solution  $\langle B_0, B_1, B_2, \dots, B_t \rangle$  is equal to  $\sum d(B_{i-1}, B_i) + 2 \sum d(B_i, r_i)$ . Since  $B_0 = B_1 = A_0$  and  $B_i = A_{i-1}$ ,

$$\sum_{i=1}^t d(B_{i-1}, B_i) = \sum_{i=1}^{t-1} d(A_{i-1}, A_i) \leq \sum_{i=1}^t d(A_{i-1}, A_i) = |\mathbb{A}(\mathcal{M}, \rho, \langle P_1, P_2, \dots, P_t \rangle)|. \quad (8.4)$$

Moreover, since every  $A_i$  servers request  $r_i$ ,  $d(B_i, r_i) \leq d(B_i, A_i) = d(A_{i-1}, A_i)$ .

Hence,

$$2 \sum_{i=1}^t d(B_i, r_i) \leq 2 \sum_{i=1}^t d(B_i, A_i) = 2 \sum_{i=1}^t d(A_{i-1}, A_i) = 2|\mathbb{A}(\mathcal{M}, \rho, \langle P_1, P_2, \dots, P_t \rangle)|. \quad (8.5)$$

Inequality (8.4) along with Equation (8.5) implies

$$|\mathbb{B}_A(\mathcal{M}, \rho, \langle P_1, P_2, \dots, P_t \rangle)| \leq 3|\mathbb{A}(\mathcal{M}, \rho, \langle P_1, P_2, \dots, P_t \rangle)|.$$

Since this holds for all requests  $\rho \sim \langle P_1, P_2, \dots, P_t \rangle$ , we have

$$|\mathbb{B}_A(\mathcal{M}, \langle P_1, P_2, \dots, P_t \rangle)| \leq 3|\mathbb{A}(\mathcal{M}, \langle P_1, P_2, \dots, P_t \rangle)|$$

and the proof is complete.  $\square$

An immediate corollary of Lemma 8.1 is that the optimal request oblivious algorithm has a cost of at most  $3\text{OPT}_{\mathcal{M}, \langle P_1, P_2, \dots, P_t \rangle}(\mathcal{M}, \langle P_1, P_2, \dots, P_t \rangle)$ . Therefore, if we only focus on the request oblivious algorithms, we only lose a factor of 3 in comparison to the optimal online algorithm. The following lemma states a key structural lemma for an optimal request oblivious algorithm.

**Lemma 8.2.** *For every request oblivious algorithm  $\mathbb{B}$ , there exists a randomized request oblivious algorithm  $\mathbb{B}'$  with the same expected cost which is not only oblivious to the last request, but also oblivious to all requests that have come prior to this.*

**Proof.** For any given request oblivious online algorithm  $\mathbb{B}$ , we construct an online algorithm  $\mathbb{B}'$  which is oblivious to all of the requests as follows: For an input  $\langle B_1, B_2, \dots, B_{i-1} \rangle$  of configurations and probability distributions  $\langle P_1, P_2, \dots, P_t \rangle$ , draw a sequence of requests  $\langle r_1, r_2, \dots, r_i \rangle$  from  $\langle P_1, P_2, \dots, P_t \rangle$  conditioned on the constraint that  $\mathbb{B}$  would generate configurations  $\langle B_1, B_2, \dots, B_{i-1} \rangle$  for requests  $\langle r_1, r_2, \dots, r_{i-1} \rangle$ . Now, report the output of  $\mathbb{B}$  for inputs  $\langle B_1, B_2, \dots, B_{i-1} \rangle$ ,  $\langle r_1, r_2, \dots, r_i \rangle$ , and  $\langle P_1, P_2, \dots, P_t \rangle$ .

We define the cost of step  $i$  of Algorithm  $B'$  as  $d(B_{i-1}, B_i) + 2d(B_i, r_i)$ . Due to the construction of algorithm  $\mathbb{B}'$ , the expected cost of this algorithm at every step  $i$  for a random sequence of requests is equal to the expected cost of algorithm  $\mathbb{B}$  for a random sequence of requests drawn from  $\langle P_1, P_2, \dots, P_t \rangle$ . Therefore, the expected cost of both algorithms for a random sequence of requests are equal and thus  $|\mathbb{B}(\mathcal{M}, \langle P_1, P_2, \dots, P_t \rangle)| = |\mathbb{B}'(\mathcal{M}, \langle P_1, P_2, \dots, P_t \rangle)|$ .  $\square$

Lemma 9.1 states that there always exists an optimal randomized request oblivious online algorithm that returns the configurations regardless of the requests. We call such an algorithm *non-adaptive*. Since a non-adaptive algorithm is indifferent to the sequence of the requests, we can assume it always generates a sequence of configurations just based on the distributions. For an optimal of such algorithms, all such sequence of configurations should be optimal as well. Therefore, there al-

ways exists an optimal non-adaptive online algorithm which is deterministic. By Lemma 8.1 not only do we know the optimal request oblivious algorithm is at most 3-approximation, but also the same holds for the optimal non-adaptive algorithm.

**Theorem 8.7.** *There exists a sequence of configurations  $\langle B_0, B_1, \dots, B_t \rangle$  such that an online algorithm which starts with  $B_0$  and always returns configuration  $B_i$  upon arrival of request  $r_i$  has an approximation factor of at most 3.*

## 8.4 Fractional Solutions

In this section we provide a fractional online algorithm for the  $k$ -server problem that can be implemented in polynomial time. Note that by Theorem 8.7 we know that there exist configurations  $\langle \mathbb{B}_1, \mathbb{B}_2, \dots, \mathbb{B}_t \rangle$  such that the expected cost of a non-adaptive algorithm that always returns these configurations is at most 3. Therefore, we write an integer program to find such configurations with the least expected cost. Next, we provide a relaxed LP of the integer program and show that every feasible solution of such LP corresponds to a fractional online algorithm for the stochastic  $k$ -server problem. Hence, solving such a linear program, that can be done in polynomial time, gives us a fractional online algorithm for the problem.

### 8.4.1 Linear Program

Recall that given  $t$  independent distributions  $\langle P_1, \dots, P_t \rangle$  for online stochastic  $k$ -server, an adaptive algorithm can be represented by  $t + 1$  configurations  $\langle B_0, \dots, B_t \rangle$ . Upon the arrival of each request  $r_i$ , we move the servers from config-

uration  $B_{i-1}$  to  $B_i$  and then one server serves  $r_i$  and goes back to its position in  $B_i$ . The objective is to find the configurations such that the cost of moving to new configurations in addition to the expected cost of serving the requests is minimized. Therefore the problem can be formulated in an offline manner. First we provide an integer program in order to find a vector of configurations with the least cost.

The decision variables of the program represent the configurations, the movement of servers from one configuration to another, and the way that each possible request is served. In particular, at each time step  $\tau$ :

- For each node  $v$  there is a variable  $b_{\tau,v} \in N$  denoting the number of servers on node  $v$ .
- For each pair of nodes  $u$  and  $v$ , there is a movement variable  $f_{\tau,u,v} \in N$  denoting the number of servers going from  $u$  to  $v$  for the next round.
- For each node  $v$  and possible request node  $r$ , there is a variable  $x_{\tau,v,r} \in \{0, 1\}$  denoting whether  $r$  is served by  $v$  or not.

In the following integer program, the first set of constraints ensures the number of servers on nodes at each time is updated correctly according to the movement variables. The second set of constraints ensures that each possible request is served by at least one server. The third set of constraints ensures that no possible request is served by an empty node. By the definition, the cost of a sequence of configurations  $\langle B_0, \dots, B_t \rangle$  is  $\sum_{i=1}^t d(B_{i-1}, B_i) + 2 \sum_{i=1}^t d(B_i, r_i)$ . Thus the objective is to minimize



the expression

$$\sum_{\tau} \sum_{u,v} f_{\tau,u,v} d(u,v) + 2 \sum_{\tau} \sum_v \sum_r x_{\tau,v,r} \Pr(z \sim P_{\tau} = r) d(v,r)$$

, where  $\Pr(z \sim P_{\tau} = r)$  denotes the probability that  $r$  is requested at time  $\tau$ .

$$\min. \quad \sum_{\tau} \sum_{u,v} f_{\tau,u,v} d(u,v) + 2 \sum_{\tau} \sum_v \sum_r x_{\tau,v,r} \Pr(z \sim P_{\tau} = r) d(v,r)$$

$$\forall \tau, v \quad b_{\tau+1,v} = b_{\tau,v} + \sum_u f_{\tau,u,v} - \sum_u f_{\tau,v,u}.$$

$$\forall \tau, u, v \quad \sum_v x_{\tau,v,r} \geq 1.$$

$$\forall \tau, v, r \quad x_{\tau,v,r} \leq b_{\tau,v}.$$

$$\forall \tau \quad \sum_v b_{\tau,v} \leq k.$$

$$\forall \tau, v, r \quad x_{\tau,v,r} \in \{0, 1\}.$$

$$\forall \tau, u, v \quad f_{\tau,u,v} \in N.$$

$$\forall \tau, v \quad b_{\tau,v} \in N.$$

Now we consider the following relaxation of the above integer program.

$$\min. \quad \sum_{\tau} \sum_{u,v} f_{\tau,u,v} d(u,v) + 2 \sum_{\tau} \sum_v \sum_r x_{\tau,v,r} \Pr(z \sim P_{\tau} = r) d(v,r)$$

$$\forall \tau, v \quad b_{\tau+1,v} = b_{\tau,v} + \sum_u f_{\tau,u,v} - \sum_u f_{\tau,v,u}.$$

$$\forall \tau, u, v \quad \sum_v x_{\tau,v,r} \geq 1.$$

$$\forall \tau, v, r \quad x_{\tau,v,r} \leq b_{\tau,v}.$$

$$\forall \tau \quad \sum_v b_{\tau,v} \leq k.$$

Next, in Section 8.5 we show how a fractional solution can be rounded to an integral solution.

## 8.5 Reduction from Integral $k$ -server to Fractional $k$ -server

In this section we show how we can obtain an integral algorithm for the stochastic  $k$ -server problem from a fractional algorithm. We first show that every fractional algorithm for the line metric can be modified to an integral algorithm with the same cost. Next, we study the problem on HST metrics; we give a rounding method that produces an integral algorithm from a fractional algorithm while losing a constant factor. Finally, we leverage the previously known embedding techniques to show every metric can be embedded into HST's with a distortion of at most  $O(\log n)$ . This will lead to a rounding method for obtaining an integral algorithm from every fractional algorithm on general metrics while losing a factor of at most  $O(\log n)$ . Combining this with the 3 approximation fractional algorithm that we provide in Section 8.4, we achieve an  $O(\log n)$  approximation algorithm for the stochastic  $k$ -server problem on general graphs.

### 8.5.1 Integrals Are as Strong as Fractionals On the Line

In this section we show every fractional algorithm on the line metric can be derandomized to an integral solution with the same expected cost. The rounding method is as follows: For every fractional configuration  $A$ , we provide an integral configuration  $\mathbb{I}(A)$  such that (i) the distance of two configurations  $A_1$  and  $A_2$  is

equal to the expected distance of two configurations  $\mathbb{I}(A_1)$  and  $\mathbb{I}(A_2)$ . (ii) for every point  $x$  in the metric that  $A$  has a server mass of size at least 1 on  $x$ , there exists a server on point  $x$  in  $\mathbb{I}(A)$ .

Let for every point  $x$  in the metric,  $A(v)$  denote the amount of server mass on node  $v$  of the line. For every fractional configuration  $B$ , we define a mass function  $f_A : (0, k] \rightarrow V$  as follows.  $f_A(x) = v_j$  if and only if  $j$  is the minimum integer such that  $\sum_{i=1}^{j-1} A(i) < x$  and  $\sum_{i=1}^j A(i) \geq x$ . Intuitively, if one gathers the server mass by sweeping the line from left to right,  $f_A(x)$  is the first position on which we have gathered  $x$  amount of server mass. The rounding algorithm is as follows:

- Pick a random real number  $r$  in the interval  $[0, 1)$ .
- $\mathbb{I}(A)$  contains  $k$  servers on positions  $f_A(r), f_A(r+1), f_A(r+2), \dots, f_A(r+k-1)$ .

Note that the rounding method uses the same  $r$  for all of the configurations. More precisely, we draw  $r$  from  $[0, 1)$  at first and use this number to construct the integral configurations from fractional configurations. The following two lemmas show that both of the properties hold for the rounding algorithm we proposed.

**Lemma 8.3.** *Let  $A$  be a fractional configuration and  $x$  be a point such that  $A(x) \geq 1$ . Then  $\mathbb{I}(A)$  has a server on  $x$ .*

**Proof.** Due to the construction of our rounding method, for every two consecutive servers  $a$  and  $b$  in  $\mathbb{I}(A)$ , the total mass of servers after  $a$  and before  $b$  in the fractional solution is less than 1. Therefore,  $\mathbb{I}(A)$  should put a server on point  $x$ , otherwise the total mass of servers in the fractional solution between the first server before  $x$  and the first server after  $x$  would be at least 1. □

The next lemma shows that the rounding preserves the distances between the configurations in expectation.

**Lemma 8.4.** *Let  $A_1$  and  $A_2$  be two fractional configurations and  $|A_1 - A_2|$  be their distance. The following holds for the distances of the configurations*

$$\mathbb{E}|\mathbb{I}(A_1) - \mathbb{I}(A_2)| = |A_1 - A_2|.$$

**Proof.** The key point behind the proof of this lemma is that the distance of two fractional configurations  $A_1$  and  $A_2$  can be formulated as follows

$$|A_1 - A_2| = \int_0^1 |\mathbb{I}_\omega(A_1) - \mathbb{I}_\omega(A_2)| d_\omega$$

where  $\mathbb{I}_\omega(A)$  stands for an integral configurations which places the servers on points  $f_A(\omega), f_A(\omega+1), f_A(\omega+2), \dots, f_A(\omega+k-1)$ . Since at the beginning of the rounding method we draw  $r$  uniformly at random, the expected distance of the two rounded configurations is exactly equal to

$$\int_0^1 |\mathbb{I}_\omega(A_1) - \mathbb{I}_\omega(A_2)| d_\omega$$

which is equal to the distance of  $A_1$  from  $A_2$ . □

**Theorem 8.8.** *For any given fractional online algorithm  $\mathbb{A}$  for the  $k$ -server problem on the line metric, there exists an online integral solution for the same problem with the same expected cost.*

## 8.5.2 Reduction for General Graphs

An HST is a undirected rooted tree in which every leaf represents a point in the metric and the distance of a pair of points in the metric is equal to the distance

of the corresponding leaves in the tree. In an HST, weights of the edges are uniquely determined by the depth of the vertices they connect. More precisely, in a  $\sigma$ -HST the weight of an edges between a vertex  $v$  and its children is equal to  $\sigma^{h-d_v}$  where  $h$  stands for the height of the tree and  $d_v$  denotes the depth of vertex  $v$ .

Since HSTs are very well structured, designing algorithms on HSTs is relatively easier in comparison to a more complex metric. Therefore, a classic method for alleviating the complexity of the problems is to first embed the metrics into HSTs with a low distortion and then solve the problems on these trees.

Perhaps the most important property of the HSTs is the following:

**Observation 8.9.** *For every pair of leaves  $u, v \in T$  of an HST, the distance of  $u$  and  $v$  is uniquely determined by the depth of their deepest common ancestor.*

Note that, the higher the depth of the common ancestor is, the lower the distance of the leaves will be. Therefore, the closest leaves to a leaf  $v$  are the ones that share the most common ancestors with  $v$ . Bansal *et al.* propose a method for rounding every fractional solution of the  $k$ -server problem to an integral solution losing at most a constant factor [BBMN11].

**Theorem 8.10.** [BBMN11] *Let  $T$  be a  $\sigma$ -HST with  $n$  leaves,  $\sigma > 5$ , and let  $A = \langle A_0, A_1, A_2, \dots, A_t \rangle$  be a sequence of fractional configurations. There is an online procedure that maintains a sequence of randomized  $k$ -server configurations  $S = \langle S_0, S_1, S_2, \dots, S_t \rangle$  satisfying the following two properties:*

- *At any time  $i$ , the state  $S_i$  is consistent with the fractional state  $A_i$ .*

- *If the fractional state changes from  $x_{i-1}$  to  $x_i$  at time  $i$ , incurring a movement cost of  $c_i$ , then the state  $S_{i-1}$  can be modified to a state  $S_i$  while incurring a cost of  $O(c_i)$  in expectation.*

Embedding general metrics into trees and in particular HSTs has been the subject of many studies. The seminal work of Fakcharoenphol *et al.* [FRT03] has shown that any metric can be randomly embedded to  $\sigma$ -HSTs with distortion  $O(\frac{\sigma \log n}{\log \sigma})$ .

**Theorem 8.11.** [FRT03] *There exists a probabilistic method to embed an arbitrary metric  $\mathcal{M}$  into  $\sigma$ -HSTs with distortion  $\frac{\sigma \log n}{\log \sigma}$ .*

Therefore, to round a fractional solution on a general metric, we first embed it into  $\sigma$ -HSTs with a distortion of at most  $O(\log n)$  and then round the solution while losing only a constant factor. This will give us an integral algorithm that has an expected cost of at most  $O(\log n)$  times the optimal.

**Theorem 8.12.** *For any given fractional online algorithm  $\mathbb{A}$  for the  $k$ -server problem on an arbitrary metric, there exists an online integral solution for the same problem having a cost of no worse than  $O(\log n)$  times the cost of  $\mathbb{A}$  in expectation.*

## Chapter 9: Survivable Network Design and Prophets

### 9.1 Introduction

In an instance of the *network design* problem, we are given a graph  $G = (V, E)$ , an edge-cost function  $c : E \rightarrow \mathcal{R}^{\geq 0}$ , and a connectivity criteria. The goal is to find a minimum-cost subgraph  $H$  of  $G$  that satisfies the connectivity requirements. An important family of this class is the *survivable network design problem* (SNDP): Given non-negative integers  $r_{uv}$  for each pair  $u, v \in V$ , the solution subgraph  $H$  should contain  $r_{uv}$  edge-disjoint paths for each pair  $u$  and  $v$ . SNDP arises in fault tolerance management and thus is of much interest in design community: the connectivity of nodes  $u$  and  $v$  in  $H$  is resilient to even  $(r_{uv} - 1)$  edge failures. This problem clearly generalizes the *Steiner tree*<sup>1</sup> and *Steiner forest*<sup>2</sup> problems.

For a non-empty cut  $S \subset V$ , let  $\delta(S)$  denote the set of edges with exactly one endpoint in  $S$ . SNDP falls in the general class of network design problems that can be characterized by *proper cut functions*. A function  $f : 2^V \rightarrow \mathcal{Z}^{\geq 0}$  defined over cuts in the graph is *proper*, if it is symmetric ( $f(S) = f(V \setminus S)$  for all  $S \subset V$ ) and

---

<sup>1</sup>In the Steiner tree problem, given a set of terminal nodes  $T \subset V$ , the goal is to find a minimum-cost subgraph connecting all terminals.

<sup>2</sup>In the Steiner forest problem, given a set of pairs of vertices  $s_i, t_i \in V$ , the goal is to find a minimum-cost subgraph in which every pair is connected.

it satisfies maximality ( $f(S \cup T) \leq \max\{f(S), f(T)\}$  for all  $S \cap T = \emptyset$ ). For SNDP, one can choose  $f(S) = \max_{u \in S, v \notin S} r_{uv}$  for every cut  $S$ . Given a proper function  $f$  over cuts in the graph, the goal is to find a minimum-cost subgraph  $H$  such that

$$|E(H) \cap \delta(S)| \geq f(S) \quad \forall \text{ non-empty } S \subset V .$$

Over the past decades, the offline SNDP and proper cut functions have been extensively studied especially as an important testbed for primal-dual and iterative rounding methods (see e.g. [GGP<sup>+</sup>94, GW95, GK11, Jai01, RK93, WGMV95]). In this chapter, we consider SNDP in the *online* setting: we are given a graph  $G = (V, E)$  and an edge-cost function  $c$  in advance. We receive an online sequence of demands in the form of tuples  $(u, v, r_{uv}) \in V \times V \times \mathcal{Z}^{\geq 0}$ . We start with the empty subgraph  $H$ . Upon the arrival of a demand  $(u, v, r_{uv})$ , we need to immediately *augment*  $H$  such that there exist at least  $r_{uv}$  edge-disjoint paths between  $u$  and  $v$  in  $H$ . The goal is to minimize the cost of  $H$ . The *competitive ratio* of an algorithm is defined as the maximum ratio of the cost of its output and that of an optimal offline solution, over all possible input instances.

The online 1-connectivity problems, in which  $r_{uv} \in \{0, 1\}$  for all pairs, have been extensively studied in the last decades. Imase and Waxman [IW91] (SIAM'91) were first to consider the edge-weighted Steiner tree problem. They used a dual-fitting argument to show that the natural greedy algorithm is  $O(\log n)$ -competitive where  $n = |V|$ <sup>3</sup>. Their result is asymptotically tight. Later, Berman and Coul-

---

<sup>3</sup>In fact, the competitive ratio is  $O(\log \min\{n, \mathcal{D}\})$  where  $\mathcal{D}$  is the number of demand requests. However, to simplify the comparison with results for SNDP, in this chapter we ignore the distinction between this factor and  $O(\log n)$ .



ston [BC97] (STOC'97) and Awerbuch, Azar, and Bartal [AAB04] (TCS'04) demonstrated an  $O(\log n)$ -competitive algorithm for the more general Steiner forest problem by designing an elegant online primal-dual technique. The latter also shows that the greedy algorithm achieves the competitive ratio of  $O(\log^2 n)$  for Steiner forest. Indeed, due to the simplicity of greedy approaches, an important open problem is to settle the competitiveness of the greedy algorithm for Steiner forest. In the recent years, several primal-dual techniques are developed for solving node-weighted variants [AAA<sup>+</sup>09, HLP13, NPS11], and prize-collecting variants [QW11, HLP14] of 1-connectivity problems.

Gupta, Krishnaswamy, and Ravi [GKR12] (SIAM'12) were first to consider the online survivable network design problem. They demonstrate an elegant algorithm with competitive ratio of  $\tilde{O}(k \log^3 n)$ , where  $k = \max_{u,v} r_{uv}$ . The crux of their analysis is to use distance-preserving tree-embeddings in an online setting. More precisely, they first pick a random distance-preserving spanning subtree  $T \subseteq G$ . They satisfy a connectivity demand  $r_{uv}$  by iteratively increasing the connectivity of  $u$  and  $v$ . In each iteration, they show that it is sufficient to use cycles that are formed by an edge  $e = (a, b) \notin T$  and the  $\{a, b\}$ -path in  $T$ ; hence, reducing the number of *options* for satisfying a connectivity demand. This would enable them to use a set cover approach to solve the problem in an online manner and achieve the first competitive algorithm for online SNDP.

Single-source SNDP is a variant of SNDP where all demands share a same endpoint. Naor, Panigrahi, and Singh [NPS11] (FOCS'11) partially improve the results of Gupta et al. [GKR12] by demonstrating a *bi-criteria* competitive algo-

rithm for single-source SNDP using structural properties of a single-source optimal solution. A bi-criteria competitive ratio of  $(\alpha, \beta)$  for SNDP implies that the solution produced by the online algorithm achieves a connectivity of  $\lfloor \frac{r_i}{\beta} \rfloor$  for every demand  $\sigma_i$  and is at most a factor of  $\alpha$  more expensive than the optimal offline solution for connectivity of  $r_i$ . The algorithm by Naor et al. achieves the competitive ratio of  $O(\frac{k \log n}{\epsilon}, 2 + \epsilon)$  for any  $\epsilon > 0$ . They also study and give bi-criteria algorithms for the vertex-connectivity problem.

The competitive ratio of algorithms by Gupta et al. and Naor et al. grows linearly in  $k$ . This seems to be inherent to their methods since they may need to solve a set-cover-like problem in each iteration of incrementing the connectivity of a demand; hence, losing a polylogarithmic factor in each iteration. One would need a new approach to break the linear dependency to  $k$ . Indeed, both factors of  $O(\log^3 n)$  and  $O(k)$  are not plausible in practice, and an important open problem in the online community [NPS11, GKR12] is whether the linear dependency to  $k$  can be reduced to a logarithmic dependency. In this chapter we circumvent this problem by two main approaches, first by considering the stochastic setting of the problem, and also by allowing small congestion on the edges.

**Allowing small congestion.** Interestingly, we show that if the online algorithm is allowed to buy three copies of an edge, then a simple greedy approach is  $O(\log^2 n \log k)$ -competitive. More precisely, we demonstrate a deterministic algorithm with a bi-criteria competitive ratio of  $(O(\log^2 n \log_{1+\epsilon} k), 2 + \epsilon)$  for any constant  $\epsilon > 0$ . For the single-rooted variant, the competitive ratio is  $(O(\log n \log_{1+\epsilon} k), 2 + \epsilon)$ . In our analysis of the greedy algorithm, our main technical contributions are two

fold: (i) establishing the connection between online SNDP and the celebrated *Steiner packing* problem; and (ii) demonstrating the optimal packing ratio of  $1/2$  for the relaxed *fractional* variant of Steiner packing problem; which is a decades old open problem for the integral variant.

It is worth mentioning that one can think of our bi-criteria algorithm as a solution to the online SNDP with congestion 2. Relaxing a hard-to-approximate problem by allowing small congestion is quite natural and has been very fruitful over the past decade. Perhaps the most important example of such line of research is the *edge-disjoint path* problem with congestion [RT87, AR01, BS00, KS04, And10, Chu12, KK11, Fra85, CKS09, SCS11], for which the work of Chuzhoy and Li [CL] (FOCS'12) gives a polylogarithmic approximation algorithm with congestion 2.

**Stochastic SNDP.** For many online optimization problems it is natural and fruitful to assume that at each online step the online query is drawn independently from a known probability distribution. We call this model the *i.i.d.* model. This model has been considered for many fundamental problems (see e.g. online stochastic matching [BK10, DJSW11, FMMM09, HMZ11, JL13, MOGS12],  $k$ -server [ADGP<sup>+</sup>10, CL06], set-cover, Steiner network design, facility location [GGLS08], multi-commodity flow [HKLR05], among others). There are two important generalizations of the i.i.d. model. The unknown distribution i.i.d. model, where the queries are again drawn independently from the same probability distribution, but the probability distribution is not known, and also the *prophet setting*. In the prophet setting, instead of only one distribution for every online query, the queries are independently drawn from different distributions, inspired by the classi-

cal *prophet inequality* problem. The prophet setting is also studied for many online problems (see e.g. [AHL12, AHL13, AHL<sup>+</sup>11, FGL15]). In this chapter we consider all three variants of the stochastic SNDP and show that we can achieve significantly more efficient algorithms having a stochastic information about the input. We first provide an oblivious constant competitive algorithm for the i.i.d. SNDP. Using a similar approach to Garg et al. [GGLS08] we provide an  $O(\log n)$ -competitive algorithm for the unknown distribution i.i.d. SNDP. Then we provide a constant competitive algorithm for the prophet SNDP, through a novel technique which leverages the oblivious algorithm provided for the i.i.d. SNDP.

Interestingly we provide a general framework to obtain competitive algorithms for online optimization problems in the prophet setting. Indeed it is an interesting open problem that what is the relation between the i.i.d. setting and the prophet setting? We show, by a simple but tricky technique that we can obtain a competitive online algorithm in the prophet setting if we can design an oblivious competitive algorithm for the problem in the i.i.d. setting. Using this technique we provide asymptotically tight competitive algorithms for fundamental online problems in prophet setting, such as set cover and facility location.

### 9.1.1 Our Results and Techniques

Let  $\sigma_i = (u_i, v_i, r_i)$  denote the  $i$ -th connectivity demand. Consider the following intuitive greedy approach. Upon the arrival of  $\sigma_i$ , we augment the solution subgraph  $H$ , by finding the minimum-cost set of edges whose addition to  $H$  cre-

ates  $r_i$  edge-disjoint paths between  $u_i$  and  $v_i$ . Awerbuch et al. [AAB04] (TCS'04) show that if all the demands require 1-connectivity (i.e.,  $r_i = 1$  for every  $i$ ), this algorithm achieves a competitive ratio of  $O(\log^2 n)$ . This leads to a natural question that whether greedy works for higher connectivity problems as well. However, we show an instance of online SND in Section 9.3, for which the greedy algorithm has a competitive ratio of  $\Omega(n)$ . Indeed, the connectivity demands in the instance are either zero or two, hence greedy is not competitive even for low connectivity demands.

Interestingly, our results in this chapter show that *greedy-like* algorithms do surprisingly well, if we consider the stochastic version of the problem, or if we allow a small congestion on the edges. In the following we present our algorithms techniques in a high-level perspective.

**Allowing small congestion.** We show that a greedy algorithm does surprisingly well, if we relax the connectivity requirements by a constant factor. Let  $\alpha$  denote an arbitrary scale factor. We define an  $\alpha$ -scaled variant of the greedy algorithm in which the goal is to find only  $\lfloor \frac{r_i}{\alpha} \rfloor$  disjoint paths between the endpoints of  $\sigma_i$ . Our main result states that the scaled greedy algorithm is polylogarithmic competitive.

**Theorem 9.1.** *For any constant  $\epsilon > 0$ , the  $(2 + \epsilon)$ -scaled greedy algorithm is  $(O(\log^2 n \log_{1+\epsilon} k), 2 + \epsilon)$ -competitive. For the single-source variant, the competitive ratio is  $(O(\log n \log_{1+\epsilon} k), 2 + \epsilon)$ .*

*Furthermore, for uniform SNDP, 2-scaled greedy is  $(O(\log^2 n), 2)$ -competitive.*

We start by demonstrating a deep connection between the greedy method for SNDP and the *Steiner packing problems*. The Steiner packing problems are motivated by vast applications in VLSI-layout and has been used as an algorithmic toolkit in computer science. In the *Steiner tree packing* problem, we are given a graph  $G = (V, E)$  and a set  $S$  of vertices and the goal is to find the *Steiner decomposition number* (SDN), the maximum number of edge-disjoint subgraphs that each connects the vertices of  $S$ . We note that a minimal connecting subgraph is a *Steiner tree* with respect to  $S$ . In the *Steiner forest packing* problem, we are given a set of demand pairs  $u_i, v_i \in V$  and the goal is to find SDN, the maximum number of edge-disjoint subgraphs that in each, the demand pairs are connected.

For simplicity, let us assume we have an *uniform* instance. Let  $\text{opt}$  denote the optimal SNDP solution, with the Steiner decomposition number  $q$ . In Section 9.3, we show that the  $(\frac{k}{q})$ -scaled greedy algorithm approximates  $\text{opt}$  up to logarithmic factors. Intuitively, every forest in the Steiner forest decomposition, gives us a path to satisfy a demand. Hence, we need to bound the overall cost of satisfying demands in all the  $q$  forests. The crux of our analysis is then to charge the cost of the scaled greedy to that of a parallel set of greedy algorithms that solve 1-connectivity instances on every forest. Finally, to get a polylogarithmic competitive algorithm, we need to find a universal lower bound on the SDN number  $q$  with respect to  $k$ .

It is shown that finding SDN is NP-hard and cannot be computed in polynomial time unless  $P=NP$  [CS06] (Algorithmica'06). Given that there exist  $q$  disjoint Steiner forests connecting a set of demands, it is straight forward to show the graph is  $q$ -connected on the demands. Therefore, a natural upper bound on

SDN is the minimum connectivity of the endpoints of demands. For the case of spanning trees (Steiner tree with  $S = V(G)$ ), it is proven that the above upper bound also provides a good approximation guarantee for the problem. In other words, any  $k$ -connected graph can be decomposed into  $k/2$  edge-disjoint spanning trees [Kri03] (JCT'03). This is also followed by a matching upper bound. The problem is much subtler when  $S$  does not encompass all vertices of the graph. The first lower bound for the Steiner tree packing problem was achieved by Petingi and Rodriguez [PR00] (CON'03) who proved every  $S$ - $k$ -connected<sup>4</sup> has  $\lfloor (2/3)^{|V(G)-|S|} k/2 \rfloor$  disjoint Steiner trees. This was later improved by Kriesell [Kri03] (JCT'03), Jain, Mahdian, and Salavatipour [JMS03] (SODA'03), Lua [Lau04] (FOCS'04), and Devos, McDonald, and Pivotto [DMP13] (Man'13), the most recent of which shows for every  $S$ - $(5k + 4)$ -connected graph, we can find  $k$  edge-disjoint Steiner trees. However, the main conjecture is that, similar to the case of spanning trees, every  $S$ - $k$ -connected graph admits a  $k/2$ -disjoint Steiner tree decomposition [Kri03]. In this chapter, we prove this conjecture for the *fractional* variant of the problem.

For a set of demand pairs  $(u_i, v_i)$ 's, let  $\mathcal{T}$  denote the set of Steiner forests that satisfy all the demands. In the *fractional Steiner forest packing* problem, the output is a fractional assignment  $x$  over  $\mathcal{T}$  such that for every edge  $e$ ,  $\sum_{T \in \mathcal{T}: e \in T} x_T$  is not more than one. The goal is to find a fractional Steiner forest decomposition with maximum  $\|x\|$ . In Section 9.2, we prove the fractional variant of the conjecture of Kriesell. We believe this result can be of independent interest in improving upon other problems that rely on Steiner packing problems. We would like point out that

---

<sup>4</sup>A graph which is  $k$ -connected on a set of vertices  $S$

the fractional Steiner forest decomposition is also presented in [DMP13].

**Theorem 9.2.** *Given a set of demand pairs  $(u_i, v_i)$ , if  $G$  is  $k$ -connected for every demand pair, then the fractional Steiner decomposition number is at least  $k/2$ .*

Indeed, in Section 9.3, we use a dependent rounding method to show that the connection between SDN and the competitiveness of the greedy approach holds even for the stronger fractional variant of SDN. Hence, Theorem 9.2 implies that the 2-scaled greedy algorithm, achieves a polylogarithmic competitive ratio for the uniform SNDP. Finally, in Section 9.4, we prove Theorem 9.9 by showing that the scaled greedy is also competitive for the non-uniform variant, if one is willing to lose an extra  $\log k$  factor in the competitive ratio.

**Stochastic SNDP.** A single-source *uniform* instance of online SNDP is an instance in which for every demand  $\sigma_i$ ,  $u_i = u$ ,  $r_i = k$  for some vertex  $u \in V$  and integer  $k$ . For a non-uniform variant, let  $k = \max_i r_i$ . Let  $D$  be a given probability distribution over  $V$ . In i.i.d. SNDP at each online step  $i$ , a random connectivity demand  $\sigma_i = (u, v_i, k)$  arrives, where  $v_i$  is drawn independently at random from distribution  $D$ . We call the problem unknown distribution SNDP if the probability distribution  $D$  is not given in advance. Another interesting generalization of the i.i.d. model, which we call the prophet SNDP is defined as follows. In prophet SNDP, instead of only a single probability distribution  $D$ , we are given  $T$  probability distributions  $D_1, \dots, D_T$ , such that the  $i$ -th demand is  $\sigma_i = (u, v_i, k)$ , where  $v_i$  is drawn independently at random from distribution  $D_i$ . In all three variants of the stochastic SNDP, the competitive-ratio is defined as the the expected cost of



an algorithm  $\mathcal{A}$  over the expected cost of an optimal offline algorithm while the distribution(s) is chosen by an adversary. More precisely let  $E[A(\omega)]$  and  $E[opt(\omega)]$  denote the expected cost of an algorithm  $\mathcal{A}$  and the expected cost of an optimal offline algorithm for an online scenario  $\omega$ , respectively. Thus the competitive-ratio of algorithm  $\mathcal{A}$  is defined as follows.

$$cr(\mathcal{A}) := \max_D \frac{E_{\omega \sim D}[A(\omega)]}{E_{\omega \sim D}[opt(\omega)]}.$$

We first provide an oblivious algorithm for the i.i.d. SNDP. The algorithm has two steps. First we realize a random scenario from distribution  $D$ . Then we compute a 2-approximate solution for this random offline instance using Jain [Jai01]. We buy all the edges in the offline solution. Now considering the edges already selected, for each vertex  $v$  we compute a minimum-cost  $k$ -flow to the root  $u$ . In the second step, upon arrival of each demand, we buy the computed minimum-cost  $k$ -flow to satisfy the  $k$ -connectivity to the root. While the algorithm is similar to the algorithm in Garg et al. [GGLS08] for the 1-connectivity case, the analysis needs careful consideration of the  $k$ -connected graphs. In fact, despite previous works on the online survivable network design which do not take the structures of  $k$ -connected graphs into account, we obtain a structural result about  $k$ -connected graphs. Then we leverage the structural result to analyze our algorithm, and to prove that our algorithm is 4-competitive. This is quite surprising, since it is dramatically improving the known  $O(k \log^3 n)$ -competitive-ratio.

**Theorem 9.3.** *There exist a 4-competitive algorithm for i.i.d. SNDP.*

Afterwards, we show that a greedy algorithm is  $O(\log n)$ -competitive if the

input is drawn from an even unknown distribution. This is very interesting since, in the adversary setting where there is no stochastic information about the input, we show that a greedy algorithm may be  $\Omega(n)$ -competitive. Note that due to [GGLS08] even the 1-connectivity version of this problem is  $\Omega(\frac{\log n}{\log \log n})$ -hard.

**Theorem 9.4.** *There exist an  $O(\log n)$ -competitive algorithm for unknown distribution SNDP.*

Then we consider the prophet SNDP. In fact we provide a general framework to obtain competitive algorithms for online optimization problems in prophet setting.

**From oblivious i.i.d. to prophet.** We show if there exists a competitive oblivious algorithm for an online problem in i.i.d. setting, we can obtain a competitive algorithm for the same problem in prophet setting.

**Theorem 9.5.** *Given an oblivious  $\alpha$ -competitive online algorithm for problem  $\mathcal{P}$  in the i.i.d. setting, there exists a  $\alpha \frac{2e}{e-1}(1 + o(1))$ -competitive online algorithm for  $\mathcal{P}$  in prophet setting.*

Roughly speaking, we show that we can combine different distributions in the prophet setting to obtain a single distribution. Then using the i.i.d. oblivious algorithm, we may not lose more than a constant factor in the competitive ratio. Thus the following is a direct corollary of this technique.

**Corollary 9.1.** *There exists a  $O(1)$ -competitive algorithm for prophet SNDP.*

Using this framework, we can obtain competitive algorithms for many fundamental and classical problems in prophet setting. For example define  $D_1, \dots, D_T$  be

$T$  probability distributions over the elements of a set cover instance. Now let  $i$ -th demand of a set cover problem be an element randomly and independently drawn from distribution  $D_i$ . We call this problem the prophet set cover problem. Similarly one may define prophet facility location the same as the classical facility location problem, with the difference that the  $i$ -th demand is randomly drawn from a known distribution  $D_i$ . Garg et al. [GGLS08] provide oblivious online algorithms for i.i.d. facility location and i.i.d. vertex cover. Grandoni et al. [GGL<sup>+</sup>08] also provide an oblivious online algorithm for i.i.d. set cover. Thus we can directly obtain the following corollaries.

**Corollary 9.2.** *There exists an  $O(1)$ -competitive algorithm for prophet vertex cover.*

**Corollary 9.3.** *There exists an  $O(1)$ -competitive algorithm for prophet facility location.*

**Corollary 9.4.** *There exists an  $O(\log n)$ -competitive algorithm for prophet set cover.*

### 9.1.2 Further Related Work

Over the past decades, SNDP and proper cut functions have been an important testbed for primal-dual and iterative rounding methods. Goemans and Williamson [GW95] (SIAM'95) were first to consider the case of  $\{0, 1\}$ -proper functions. They used a primal-dual method to obtain a 2-approximation algorithm for the problem; which later on got generalized to the celebrated moat-growing framework for solving connectivity problems. Klein and Ravi [RK93] (IPCO'93)

considered the two-connectivity problem and the case of  $\{0, 2\}$ -proper functions. They gave a primal-dual 3-approximation algorithm for the problem. Williamson, Goemans, Mihail, and Vazirani [WGMV95] (Combinatorica'95) were first to consider general proper functions. They too developed a primal-dual algorithm with approximation ratio  $2k$ , where  $k = \max_S f(S)$ . Subsequently, Goemans, Goldberg, Plotkin, Shmoys, Tardos, and Williamson [GGP+94] (SODA'94) presented a primal-dual  $2H(k)$ -approximation algorithm, where  $H(k)$  is the  $k^{\text{th}}$  harmonic number. Finally, in his seminal work [Jai01] (Combinatorica'01), Jain introduced the *iterative rounding* method by developing a 2-approximation algorithm for network design problems characterized by proper cut functions<sup>5</sup>. We refer the reader to [GK11] for a survey of results for (offline) network design problems.

Prophet inequality has been first studied in 70s by Krengel and Sucheston [KS77, KS78]. Hajiaghayi, Kleinberg and Sandholm [HKS07] study the relation between online auctions and prophet inequality. In particular, they show that algorithms used in derivation of prophet inequality can be reinterpreted as truthful mechanisms for online auctions. In the prophet inequality setting, an onlooker is given an online sequence of independent random variables  $X_1, X_2, \dots, X_n$ , such that  $X_i$  is drawn from known probability distribution  $D_i$ . The onlooker has to choose at most one variable from the succession of the values. The onlooker can choose a value only at the time of arrival. The onlookers goal is to maximize her revenue. The onlooker's revenue is compared with the expected revenue of an optimal

---

<sup>5</sup>Indeed, the results in [GGP+94] and [Jai01] applies to the more general class of weakly or skew supermodular cut functions.

offline algorithm which known all the realized random variables in advance, like a prophet. Many online optimization problem have been studied under a prophet type of stochastic setting (see e.g. [AHL12, AHL13, AHL<sup>+</sup>11, FGL15]), i.e. at each online step the demand, or the input is drawn from a specific known distribution.

## 9.2 Steiner Tree Packing

In this section we study a variant of the Steiner tree packing problem which we call “*the fractional Steiner tree packing problem*” and show the conjecture of Kriesell holds for this variant. We use this tool in Section 9.3 to analyze the behavior of the greedy algorithm in survivable network design. An immediate corollary of this theorem is a simple logarithmic approximation algorithm for the Steiner tree packing problem.

### 9.2.1 Fractional Steiner Tree Packing

One way to formulate the Steiner tree packing problem is via an integer program. Let  $\mathcal{T}_S(G)$  be the set of all Steiner trees of  $G$  on the vertices of  $S$ . By definition, the Steiner tree packing problem is the solution of the following integer program.

$$\begin{aligned}
&\text{maximize:} && \sum_{T \in \mathcal{T}_S(G)} x_T \\
&\text{subject to:} && \sum_{T \ni e} x_T \leq 1 \quad \forall e \in E(G) \\
&&& x_T \in \{0, 1\} \quad \forall T \in \mathcal{T}_S(G)
\end{aligned} \tag{9.1}$$

In Program 9.1, for every Steiner tree  $T \in \mathcal{T}_S(G)$  we have a variable  $x_T$  that is either 0 or 1. The goal is to maximize the number of trees  $T$  with  $x_T = 1$  while every edge appears in no more than one of such trees. One way to relax this problem is by lifting the constraint of  $x_T \in \{0, 1\}$  and instead assume  $0 \leq x_T \leq 1$ . This results in the following linear program:

$$\begin{aligned}
&\text{maximize:} && \sum_{T \in \mathcal{T}_S(G)} x_T && (9.2) \\
&\text{subject to:} && \sum_{T \ni e} x_T \leq 1 \quad \forall e \in E(G) \\
&&& 0 \leq x_T \leq 1 \quad \forall T \in \mathcal{T}_S(G)
\end{aligned}$$

We call the optimal solution of LP 9.2 the fractional Steiner tree packing problem. We show in Section 9.2.2 that for every graph  $G$  that is  $k$ -connected on a set  $S$  of vertices, the answer of the fractional Steiner tree packing problem on set  $S$  is greater than or equal to  $k/2$ .

## 9.2.2 Fractional Steiner Tree Packing of $k$ -connected Graphs

In this section we show the conjecture of Kriesell for Steiner tree packing holds when we relax the problem to the fractional case. More precisely, we show any graph

which is  $k$ -connected on a set  $S$  of vertices, has a fractional Steiner tree packing of at least  $k/2$ . We begin the proof by an auxiliary lemma.

**Lemma 9.1.** *Let  $\mathcal{R}$  be a subspace of  $\mathbb{R}^n$  that contains all points of  $\mathbb{R}^n$  with non-negative coordinates and  $P$  be a convex set of points in  $\mathcal{R}$ . If for every point  $\hat{x} = (x_1, x_2, \dots, x_n) \in \mathcal{R}$  there exists a point  $\hat{p} \in P$  such that*

$$\hat{p} \cdot \hat{x} \leq k \sum_{i=1}^n x_i$$

*then  $P$  contains a point  $\hat{r}$  such that  $\max_{i=1}^n r_i \leq k$ .*

**Proof.** We define  $P'$  as the set of all points in  $\mathbb{R}^n$  whose all indices are greater than or equal to the corresponding indices of a point in  $P$ . In other words

$$P' = \{\hat{p} \in \mathbb{R}^n \mid \exists \hat{q} \in P \text{ such that } p_i \geq q_i \text{ for all } 1 \leq i \leq n\}.$$

We show in the rest that  $(k, k, \dots, k) \in P'$  which immediately implies the lemma.

To this end, suppose for the sake of contradiction that  $(k, k, \dots, k) \notin P'$ . Note that, since  $P$  is a convex set, so is  $P'$ . Therefore, there exists a hyperplane that separates all points of  $P'$  from point  $(k, k, \dots, k)$ . More precisely, there exist coefficients  $h_0, h_1, \dots, h_n$  such that

$$\sum_{i=1}^n h_i p_i > h_0 \tag{9.3}$$

for all points  $\hat{p} \in P'$  and

$$\sum_{i=1}^n h_i k < h_0. \tag{9.4}$$

Due to the construction of  $P'$  we are guaranteed that all coefficients  $h_1, h_2, \dots, h_n$  are non-negative numbers since otherwise for any index  $i$  such that  $h_i < 0$  there

exists a point in  $P'$  whose  $i$ 'th index is large enough to violate Inequality (9.3).

Now let  $\hat{x} = (h_1, h_2, \dots, h_n)$ . By Inequalities (9.3) and (9.4) we have

$$\hat{p} \cdot \hat{x} = \sum_{i=1}^n x_i p_i = \sum_{i=1}^n h_i p_i > h_0 \geq k \sum_{i=1}^n h_i = k \sum_{i=1}^n x_i$$

for every  $\hat{p} \in P'$  which means there is no  $\hat{p} \in P$  such that  $\hat{p} \cdot \hat{x} \leq k \sum_{i=1}^n x_i$ . This contradicts the assumption of the lemma.  $\square$

Now based on Lemma 9.1, we give a lower bound on the fractional Steiner tree packing of any  $S$ - $k$ -connected graph. Let  $G$  be a graph which is  $k$  connected on a set  $S$ , and  $\mathbf{R}$  be the set of all randomized algorithms that randomly select a Steiner tree of  $G$ . In other words, every element of  $\mathbf{R}$  is an algorithm that can be specified with a distribution of probabilities over the Steiner trees of  $G$ . We associate every element of  $\mathbf{R}$  to a point in a  $|E(G)|$  dimensional space in the following way:

$$f(A) = \langle \hat{x}_1, \hat{x}_2, \dots, \hat{x}_{|E(G)|} \rangle$$

where  $A$  is an algorithm and for every edge  $e \in E(G)$ ,  $\hat{x}_e$  is the probability that  $e$  appears in a random tree of algorithm  $A$ . Now, let  $\mathcal{R} = \bigcup_{A \in \mathbf{R}} f(A)$  be the set of all points associated to the elements of  $\mathbf{R}$ . Convexity of  $\mathbf{R}$  is a direct consequence of its definition; For every two algorithms  $A, B \in \mathbf{R}$ , one can construct a randomized procedure  $C$  that selects a random Steiner tree based on each of the procedures with probability  $1/2$  and hence  $f(C) = (f(A) + f(B))/2$ . Thus, for every two points  $\hat{x}, \hat{y} \in \mathcal{R}$ ,  $(\hat{x} + \hat{y})/2$  is also in  $\mathcal{R}$ . Moreover, all indices of the points in  $\mathcal{R}$  are non-negative. Next, we show the following important property of  $\mathcal{R}$ .



**Lemma 9.2.** *For any point  $\hat{y} \in \mathbb{R}^{|E(G)|}$  with non-negative indices, there exists a point  $\hat{x}$  in  $\mathcal{R}$  such that*

$$\hat{x} \cdot \hat{y} \leq \sum 2\hat{y}_i/k.$$

**Proof.** To prove this lemma, we assume every edge  $e$  of  $G$  has a weight equal to  $\hat{y}_e$ . Moreover, let  $W = \sum_{e \in E(G)} w_e$  be the total sum of the weights of the edges. We show the minimum Steiner tree of  $G$  on set  $S$ , has a weight of at most  $2W/k$ . This implies the lemma since the point associated to an algorithm that deterministically selects the minimum Steiner tree of  $G$ , trivially satisfies the condition of the lemma. Therefore, it only suffices to show that the minimum Steiner tree of  $G$  has a weight of at most  $2w/k$ . To this end we write the LP relaxation of the Steiner tree problem as follows:

$$\begin{aligned} \text{minimize:} & \quad x_e w_e \\ \text{subject to:} & \quad \sum_{e \in \sigma} x_e \geq 1 \quad \forall \text{ cut } \sigma \text{ that separates the vertices of } S \\ & \quad 0 \leq x_e \leq 1 \quad \forall e \in E(G) \end{aligned} \quad (9.5)$$

One feasible solution to LP 9.5 can be achieved by setting  $x_e = 1/k$  for all edges of the graph. The reason such a solution is feasible is that every cut that separates two vertices of  $S$  has at least  $k$  edges and therefore the summation of  $x_e$ 's for every separating cut is at least 1. Thus, the optimal solution of LP 9.5 is bounded by  $W/k$ . It has been shown that the integrality gap of the Steiner tree problem is less than 2 [HRW92]. Therefore, the weight of the minimum Steiner tree of graph  $G$  is at most  $2W/k$  which concludes the lemma.

□

According to Lemmas 9.1 and 9.2, and the fact that  $\mathcal{R}$  is convex, there exists a point  $\hat{x} \in \mathcal{R}$  such that  $\hat{x}_e \leq 2/k$  for every edge  $e$  of the graph. Now, we construct a solution to LP 9.2 in the following way: Let  $A^{\hat{x}}$  be a randomized algorithm which is associated to  $\hat{x}$  ( $f(A^{\hat{x}}) = \hat{x}$ ). For every  $T \in \mathcal{T}_S(G)$ , we set  $x_T = k/2 A^{\hat{x}}(T)$ , where  $A^{\hat{x}}(T)$  is the probability that  $A^{\hat{x}}$  selects tree  $T$ . Since the probability that every edge appears in a tree of  $A^{\hat{x}}$  is bounded by  $2/k$ , then all constraints of LP 9.2 are satisfied. Note that the objective function of LP 9.2 for solution  $x$  is equal to  $k/2$  since we multiplied the probabilities by  $k/2$ .

**Theorem 9.6.** *The fractional Steiner tree packing problem for any graph which is  $k$ -connected on a set  $S$  of vertices is at least  $k/2$ .*

### 9.2.3 Steiner Forest Packing

In this section we generalize the Steiner tree packing problem and again, give a lower bound for the fractional variant of this problem. Let  $G$  be a graph and  $S$  be a sequence of vertex pairs  $(u_1, v_1), (u_2, v_2), \dots, (u_k, v_k)$ . We call a subgraph of  $G$  a Steiner forest for  $S$ , if it connects all of the pairs in  $S$  and is minimal (it contains no cycles). Now, the Steiner forest packing problem is defined as follows: Given a graph  $G$  and a sequence of vertex pairs  $S$ , what is the maximum number of edge-disjoint Steiner forests in  $G$  with respect to set  $S$ ? It is trivial to show that the Steiner forest packing problem is a generalization of the Steiner tree packing problem and hence this problem is also NP-hard. Similar to Section 9.2.1, we formulate the Steiner

forest problem as follows:

$$\begin{aligned}
& \text{maximize:} && \sum_{F \in \mathcal{F}_S(G)} x_F \\
& \text{subject to:} && \sum_{F \ni e} x_F \leq 1 && \forall e \in E(G) \\
& && x_F \in \{0, 1\} && \forall F \in \mathcal{F}_S(G)
\end{aligned} \tag{9.6}$$

where  $\mathcal{F}_S(G)$  stands for the set of all Steiner forests of  $G$  with respect to  $S$ . Again, we relax the integer constraints to obtain the following linear program:

$$\begin{aligned}
& \text{maximize:} && \sum_{F \in \mathcal{F}_S(G)} x_F \\
& \text{subject to:} && \sum_{F \ni e} x_F \leq 1 && \forall e \in E(G) \\
& && 0 \leq x_F \leq 1 && \forall F \in \mathcal{F}_S(G)
\end{aligned} \tag{9.7}$$

We call the optimal solution of LP 9.7 “the fractional Steiner forest packing” problem. A similar analysis to what we present in Section 9.2.2 yields to Theorem 9.2.

### 9.3 Uniform SNDP

In this section we consider the uniform-connectivity version of the online survivable Steiner network design problem. The assumption of this version is that all connectivity requirements are equal to a given number. For this problem we first give a very simple algorithm and then analyze it using the tools introduced in the previous section. The next section explains how we generalize our algorithm to make it work for inputs with non-uniform connectivity requirements.

In the online uniform-connectivity survivable Steiner forest problem we are given an offline graph  $G = \langle V(G), E(G) \rangle$ , an integer  $k$ , and an online stream of demands  $S = (s_1, t_1), (s_2, t_2), \dots$ . Every time a demand  $(s_i, t_i)$  arrives we have to add some of the edges of  $G$  to our current solution  $H$  in order to make  $k$  edge-disjoint paths between  $s_i$  and  $t_i$  in  $H$ . The online uniform-connectivity survivable Steiner tree problem is a special case of the forest problem in which the second endpoints of all demands are fixed at some vertex  $root$ . The objective of the problems is to minimize the cost of the selected subgraph  $H$  according to a given cost function.

A simple approach to solve these problems is to choose edges based on the following greedy method: for every demand add a minimum-cost subset of edges that satisfies the  $k$ -connectivity between its endpoints. In this section we show that this algorithm is not competitive to the optimum offline solution. This is shown by Lemma 9.6 in which we give an instance graph and a series of demands for which the greedy algorithm gives a solution of cost  $\Omega(n)$  times the cost of the optimum offline solution.

However, we show a modified version of the greedy algorithm can be a viable approach for these problems if we lose some factor on the connectivity requirement. This can be done by satisfying half of the required connectivity. In particular, for every demand we add a minimum-cost subset of the edges that makes the current solution  $(k/2)$ -connected between the endpoints of that demand. Let us call this algorithm  $GA$ . In this section we show the cost of the edges  $GA$  selects is poly-logarithmically competitive to that optimum offline solution which satisfies  $k$ -connectivity for every demand.

---

**Algorithm 5** 2-scaled Greedy

---

**Input:** A graph  $G$ , an integer  $k$ , and an online stream of demands

$(s_1, t_1), (s_2, t_2), \dots$

**Output:** A set  $H$  of edges such that every given demand  $(s_i, t_i)$  is connected through  $k$  edge-disjoint paths in  $H$ .

**Offline Process:**

1: Initialize  $H = \emptyset$ .

**Online Scheme; assuming a demand  $(s_i, t_i)$  is arrived:**

1:  $P_i =$  A minimum-cost subset of edges, such that  $s_i$  is  $k/2$ -connected to  $t_i$  in  $H \cup P_i$ .

2: Update  $H = H \cup P_i$ .

---

**Theorem 9.7.** *For the online survivable Steiner forest problem, the output of GA satisfies  $(k/2)$ -connectivity for every demand and its cost is  $O(\log^2 n)$ -competitive.*

**Theorem 9.8.** *For the online survivable Steiner tree problem, the output of GA satisfies  $(k/2)$ -connectivity for every demand and its cost is  $O(\log n)$ -competitive.*

As a direct consequence of adding edges according to GA, the  $(k/2)$ -connectivity is guaranteed for every demand. To complete the proof of the theorems, we need to show that the cost of the solution produced by GA is upper bounded by a factor of  $O(\log^2 n)$  for forests, and  $O(\log n)$  for trees.

Let  $c : E(G) \rightarrow \mathcal{R}^{\geq 0}$  be the cost function on the edges. With some abuse of notation, we also use  $c(Y)$  for a subset of edges  $Y \subseteq E(G)$  as the sum of the cost of the edges in  $Y$ . With this notation we can say at every step  $i$  GA chooses a subset

of edges  $P_i$  that satisfies  $(k/2)$ -connectivity and minimizes  $c(P_i)$ .

The overall idea of the proofs is as follows. We take an optimum solution and charge every  $c(P_i)$  to  $c(L_i)$ , where  $L_i$  is a set of edges chosen from the optimum solution. The way we define  $L_i$ 's allows them to have overlapping edges, but we show that their total cost is limited by the desired poly-logarithmic factor of the cost of the optimum solution. More specifically, we charge  $c(L_i)$  to the cost of a fractional routing  $Q_i$  between  $s_i$  and  $t_i$ . Every  $Q_i$  is itself a linear combination of routes on different Steiner forests of the optimum solution. The coefficients of this linear combination are achieved from an Steiner forest packing of the optimum solution. In this fashion, the problem boils down to finding an upper bound for the total cost of routings on each Steiner forest. In the following we formally prove every step in detail.

Let  $OPT$  be an optimum offline solution of the survivable Steiner forest problem on graph  $G$ , a stream of demands  $S$ , and the connectivity requirement  $k$ . Now we define  $L_i$  for every demand  $i$  as a minimum-cost set of edges in  $OPT$  that is  $(k/2)$ -connected between  $s_i$  and  $t_i$  assuming the endpoints of every previous demand are contracted. In particular, we call a set of edges a *pseudo-path* between  $s_i$  and  $t_i$  if there is a path between these vertices using those edges and the edges in  $\{(s_j, t_j) | \forall j < i\}$ . A *pseudo-routing* between  $s_i$  and  $t_i$  is hence a set of pseudo-paths between  $s_i$  and  $t_i$ . With these definitions,  $L_i$  is a minimum-cost pseudo-routing between  $s_i$  and  $t_i$  in  $OPT$  that consists of  $k/2$  pseudo-paths. The following lemma shows the relation between the costs of  $L_i$  and  $P_i$ .

**Lemma 9.3.** *For every demand  $i$ ,  $c(P_i) \leq c(L_i)$ .*

**Proof.** Every time a demand  $i$  arrives, GA finds a set  $P_i$  with the minimum cost and adds it to  $H$  in order to satisfy  $(k/2)$ -connectivity between  $s_i$  and  $t_i$ . Note that the endpoints of every demand  $j < i$  are already connected with  $k/2$  disjoint paths in  $H$ . Besides,  $L_i$  is a pseudo-routing between  $s_i$  and  $t_i$  which is  $(k/2)$ -connected between  $s_i$  and  $t_i$  if we contract the two endpoints of every previous demand. Therefore adding  $L_i$  to  $H$  makes  $H$   $(k/2)$ -connected between  $s_i$  and  $t_i$ . Since GA finds a minimum-cost set of edges that satisfies  $(k/2)$ -connectivity in  $H$ ,  $c(P_i)$  never exceeds  $c(L_i)$ . □

In the remaining we show how to charge the total cost of  $L_i$ 's to  $c(OPT)$ . As a property of an optimum solution,  $OPT$  contains  $k$  edge-disjoint paths between the endpoints of every demand  $(s_i, t_i) \in S$ . Therefore, according to Theorem 9.2 there exists a solution for the fractional Steiner forest packing of  $OPT$  and demand set  $S$  with value at least  $k/2$ . Let  $\mathbf{z}$  be a Steiner forest packing of  $OPT$  with value  $k/2$ . In the following we use  $\mathcal{F}_S(OPT)$  to denote the collection of all Steiner forests of  $OPT$  with respect to demand set  $S$ . The theorem states there exists a vector  $\mathbf{z}$  such that

$$\sum_{F \in \mathcal{F}_S(OPT)} z_F = k/2 \tag{9.8}$$

$$\sum_{F \in \mathcal{F}_S(OPT): e \in F} z_F \leq 1 \quad \forall e \in OPT \ . \tag{9.9}$$

Moreover, the following inequality holds for the summation of the costs of these forests.

**Lemma 9.4.**  $\sum_{F \in \mathcal{F}_S(OPT)} z_F \cdot c(F) \leq c(OPT)$  .

**Proof.** For each forest we replace its cost with the sum of the cost of its edges.

$$\begin{aligned} \sum_{F \in \mathcal{F}_S(OPT)} z_F \cdot c(F) &= \sum_{F \in \mathcal{F}_S(OPT)} z_F \sum_{e \in F} c(e) \\ &= \sum_{e \in OPT} \sum_{F \in \mathcal{F}_S(OPT): e \in F} z_F c(e) \\ &= \sum_{e \in OPT} c(e) \left( \sum_{F \in \mathcal{F}_S(OPT): e \in F} z_F \right) . \end{aligned}$$

Now we use the fact that the load on every edge in the fractional Steiner forest packing is no more than 1.

$$\begin{aligned} \sum_{F \in \mathcal{F}_S(OPT)} z_F \cdot c(F) &\leq \sum_{e \in OPT} c(e) && \text{Inequality (9.9)} \\ &= c(OPT) . \end{aligned}$$

□

Now for every forest  $F \in \mathcal{F}_S(OPT)$  and every demand  $i$  we define  $Q_i(F)$  as a minimum-cost pseudo-path between  $s_i$  to  $t_i$  in  $F$ . This definition allows using an edge  $e \in F$  multiple times in  $Q_i(F)$  of different demands. Note that  $Q_i(F)$  can be considered as a fractional pseudo-routing between  $s_i$  and  $t_i$  with value  $z_F$ . Considering this for all forests in  $\mathcal{F}_S(OPT)$ , we achieve a fractional pseudo-routing between  $s_i$  and  $t_i$  that has a value of  $k/2$ . We use  $Q_i$  to refer to this fractional pseudo-routing and  $c(Q_i) = \sum_{F \in \mathcal{F}_S(OPT)} z_F \cdot c(Q_i(F))$  to refer to its cost.

For every demand  $i$  we have mentioned two different pseudo-routings between  $s_i$  and  $t_i$  in  $OPT$  with value  $k/2$ : an integral pseudo-routing  $L_i$ , and a fractional



pseudo-routing  $Q_i$ . The following lemma shows the relation between the costs of these two.

**Lemma 9.5.** *For every  $L_i$  and  $Q_i$  pseudo-paths defined as above, we have:*

$$c(L_i) \leq c(Q_i)$$

**Proof.** Let  $\mathcal{P}$  be the family of all pseudo-paths that connects  $s_i$  to  $t_i$  in  $OPT$ . Now Consider the following  $LP$ :

$$\begin{aligned} \text{minimize:} \quad & \sum_{p \in \mathcal{P}} x_p c(p) \\ \text{subject to:} \quad & \sum_{p \in \mathcal{P}: e \in p} x_p \leq 1 \quad \forall e \in OPT \\ & \sum_{p \in \mathcal{P}} x_p = k/2 \\ & 0 \leq x_p \leq 1 \quad \forall p \in \mathcal{P} \end{aligned} \tag{9.10}$$

A feasible solution to this LP is in fact a pseudo-routing between  $s_i$  and  $t_i$  in  $OPT$  with value  $k/2$ . Since every  $F \in \mathcal{F}_S(OPT)$  is a subset of  $OPT$ , the set of pseudo-paths between  $s_i$  and  $t_i$  in  $F$  is a subset of  $\mathcal{P}$ . As a result, every  $Q_i(F)$  is also a member of  $\mathcal{P}$  and thus  $Q_i$  corresponds to a feasible fractional solution to LP 9.10 with an objective function equal to  $c(Q_i)$ . Similarly,  $L_i$  is corresponding to a feasible integral solution to this LP. Due to the definition,  $L_i$  is an optimum integral solution of this LP, meaning that  $c(L_i)$  is the minimum among the objective functions of all integral solutions. Note that this LP is essentially a minimum-cost flow which has an integrality gap of 1. Therefore,  $c(L_i)$  equals an optimum solution of the LP, and thus does not exceed  $c(Q_i)$ .  $\square$

Finally for a particular  $F \in \mathcal{F}_S(OPT)$  we show an upper bound for the sum of  $c(Q_i(F))$  over all demands. First let us take a closer look at every  $Q_i(F)$  on a particular  $F$ . Every time a new demand  $(s_i, t_i)$  arrives  $Q_i(F)$  connects its endpoints through a pseudo-path in  $F$ . This can be generalized to an algorithm for the online single-connectivity Steiner forest problem that greedily connects the endpoints of every demand by fully buying a minimum-cost pseudo-path between  $s_i$  and  $t_i$ . This is very similar to the greedy algorithm proposed in [AAB04]. Theorem 2.1 of that paper states that their greedy algorithm is  $O(\log^2 n)$ -competitive. The statement of that theorem is slightly different than Claim 9.1, but the same proof verifies the correctness of the claim.

**Claim 9.1.** *For the online Steiner forest problem, the algorithm that connects every demand with a minimum-cost pseudo-path is  $O(\log^2 n)$ -competitive.*

Now we are ready to wrap up the proof of Theorem 9.7.

**Proof of Theorem 9.7:** Let  $ALG$  denote the output of GA. The cost of  $ALG$  is the sum of the cost of  $P_i$ 's over all demands. Therefore, by applying lemmas 9.3 and 9.5 we have

$$\begin{aligned}
c(ALG) &= \sum_{(s_i, t_i) \in S} c(P_i) \\
&\leq \sum_{(s_i, t_i) \in S} c(L_i) && \text{Lemma 9.3} \\
&\leq \sum_{(s_i, t_i) \in S} c(Q_i) && \text{Lemma 9.5}
\end{aligned}$$

Now we replace  $c(Q_i)$  with the weighted sum of  $c(Q_i(F))$ 's with respect to  $\mathbf{z}$ .

$$\begin{aligned} c(ALG) &\leq \sum_{(s_i, t_i) \in S} \sum_{F \in \mathcal{F}_S(OPT)} z_F \cdot c(Q_i(F)) \\ &= \sum_{F \in \mathcal{F}_S(OPT)} z_F \sum_{(s_i, t_i) \in S} c(Q_i(F)) \end{aligned} \tag{9.11}$$

By applying Claim 9.1 to Inequality (9.11) we achieve an  $O(\log^2 n)$ -competitive ratio for GA.

$$\begin{aligned} c(ALG) &\leq \sum_{F \in \mathcal{F}_S(OPT)} z_F \left( O(\log^2 n) c(F) \right) && \text{Claim 9.1} \\ &\leq O(\log^2 n) \sum_{F \in \mathcal{F}_S(OPT)} z_F \cdot c(F) \\ &\leq O(\log^2 n) c(OPT) . && \text{Lemma 9.4} \end{aligned}$$

□

Finally, for the survivable Steiner tree problem we show that GA is  $O(\log n)$ -competitive. In other words, if one endpoint of every demand is fixed at the root, then the output of GA is at most  $O(\log n)$  times the optimum offline solution. To complete the proof of Theorem 9.8 we use a result from [NPS11]. In that paper the authors prove a competitive ratio of  $O(\log n)$  for the algorithm which satisfies every demand using a minimum-cost pseudo-path. The following claim is a restatement of their result.

**Claim 9.2.** *For the online Steiner tree problem, the algorithm that satisfies each demand with a minimum-cost pseudo-path is  $O(\log n)$ -competitive.*

**Proof of Theorem 9.8:** Note that the tree problem is a special case of the forest problem, hence Inequality (9.11) also holds for it. By applying Claim 9.2 to that

inequality the proof is complete.

$$\begin{aligned}
c(ALG) &\leq \sum_{F \in \mathcal{F}_S(OPT)} z_F \left( O(\log n) c(F) \right) && \text{Claim 9.2} \\
&\leq O(\log n) \sum_{F \in \mathcal{F}_S(OPT)} z_F \cdot c(F) \\
&\leq O(\log n) c(OPT) . && \text{Lemma 9.4}
\end{aligned}$$

□

The following Lemma shows that there exists a graph  $G$  and a sequence of demands  $\sigma$  such that Greedy algorithm performs  $\Omega(n)$  times worse than the optimal solution.

**Lemma 9.6.** *The competitive ratio of the greedy algorithm for survivable Steiner network design is  $\Omega(n)$ , even if every connectivity requirement is exactly 2.*

**Proof.** First we provide an online instance of the survivable network design problem where every connectivity requirement is exactly 2 and show the greedy algorithm performs poorly in comparison with the optimal solution. We construct a graph  $G$  of size  $n$  as follows. For each  $1 \leq i \leq n - 1$ , there exist two undirected edges from node  $i$  to node  $i + 1$  of weights 1 and  $n - i - \epsilon$  for some small  $\epsilon > 0$ . There exist two undirected edges from node  $n$  to node 1 with weights 1 and  $n - \epsilon$ . Thus  $G$  is the union of two cycles of size  $n$ . Figure 9.1 illustrates graph  $G$ . We construct a set of demands  $S$  as follows. For each  $1 \leq i \leq n - 1$ , let  $(i, i + 1)$  be the  $i$ 'th demand in  $S$ .

Now we analyze the output of the greedy algorithm for the input instance. We claim that after satisfying demand  $i$  the greedy algorithm has selected both edges between  $j$  and  $j + 1$  for every  $j \leq i$ . We prove this claim by induction. For the

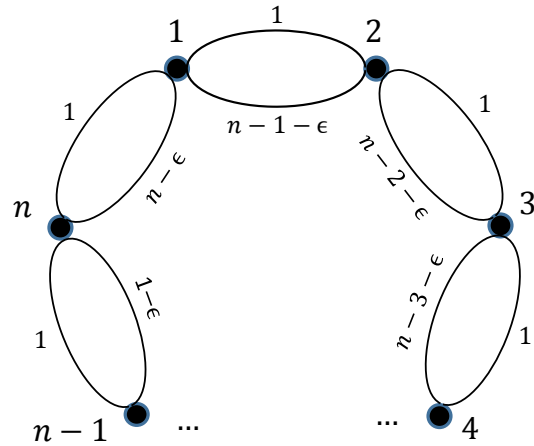


Figure 9.1: An example graph illustrating that the greedy algorithm has  $\Omega(n)$ -competitive ratio.

base case, when the first demand arrives the greedy algorithm chooses both edges between nodes 1 and 2 which costs  $n - \epsilon$ . Now assume the greedy algorithm has selected every edge between  $j$  and  $j + 1$  for every  $j < i$  before the arrival of the  $i$ 'th demand. When the  $i$ 'th demand arrives, the set of edges with minimum cost that provides two edge-disjoint paths from  $i$  to  $i + 1$  is the two edges between  $i$  and  $i + 1$  which costs  $n - i - \epsilon$ . Thus the total cost of the greedy algorithm at the end is  $\frac{n(n-1)}{2} - \epsilon n$ . However, the optimum offline solution chooses the cycle containing all edges of weight 1. Thus the competitive ratio of the greedy algorithm is  $\Omega(n)$ .

□

## 9.4 Non-Uniform SNDP

In Section 9.3 we considered uniform online survivable network design, where all connectivity requirements are the same, or in other words, for every  $\sigma_i = \langle s_i, t_i, r_i \rangle$ ,  $r_i$  equals some fixed integer  $k$ . Theorems 9.8 and 9.7 show a greedy algorithm which satisfies  $k/2$  edge connectivity of the demands, is  $O(\log n)$ -competitive for online survivable Steiner tree, and  $O(\log^2 n)$  competitive for online survivable Steiner forest, respectively. Now we consider the non-uniform case where connectivity requirements are arbitrary numbers between 1 and some value  $k$ . In particular, each demand  $\sigma_i = \langle s_i, t_i, r_i \rangle$  indicates an  $r_i$  edge-connectivity requirement between  $s_i$  and  $t_i$ . Theorem 9.9 shows one can use algorithms provided in Section 9.4 to obtain an online competitive algorithm that satisfies  $\frac{r_i}{2+\epsilon}$  connectivity of the demands.

**Theorem 9.9.** *Given an  $\alpha$ -competitive online algorithm  $\mathcal{A}$  for online survivable Steiner network design with equal connectivity demands, which partially satisfies every demand of  $k$  connectivity with  $\frac{k}{2}$  edge-disjoint paths, there exists an  $O(\frac{\alpha \log k}{\log(1+\epsilon)})$ -competitive algorithm for online survivable Steiner network design that provides a solution that partially satisfies every demand of  $r_i$  connectivity with  $\frac{r_i}{2+\epsilon}$  edge-disjoint paths.*

**Proof.** Let  $l$  be  $\lceil \frac{\log k}{\log(1+\epsilon/2)} \rceil + 1$ . Roughly speaking we define  $l$  subgraphs of  $G$ ,  $H_1, \dots, H_l$ , such that  $H_j$  is supposed to maintain a  $\lceil (1 + \epsilon/2)^{j-1} \rceil$ -connected graph on the subset of demand pairs with connectivity requirement  $(1 + \epsilon/2)^{j-1} \leq r_i < (1 + \epsilon/2)^j$ .

We use  $l$  parallel and independent greedy algorithms  $\mathcal{A}_1, \dots, \mathcal{A}_l$ , such that  $H_j$  is the solution of  $\mathcal{A}_j$  at every time. Let  $\sigma_i = \langle s_i, t_i, r_i \rangle$  be the  $i$ -th demand. Assume  $(1 + \epsilon/2)^{j-1} \leq r_i < (1 + \epsilon/2)^j$ , then we use  $\mathcal{A}_j$  for demand  $i$ . At arrival of  $\sigma_i$ , we define another request  $\sigma'_i$  as follows. Set  $\sigma'_i = \langle s_i, t_i, r'_i \rangle$ , where  $r'_i = \lceil (1 + \epsilon/2)^{j-1} \rceil$ . Now we use  $\mathcal{A}_j$  to satisfy demand  $\sigma'_i$ . Let the solution of the algorithm  $S$  be the union of the selected edges in  $H_1, \dots, H_l$ .

First we prove  $S$  partially satisfies every demand of  $r_i$  connectivity with  $\frac{r_i}{2+\epsilon}$  edge-disjoint paths. Since  $\mathcal{A}_j$  provides  $k/2$  edge-disjoint paths for a connectivity demand of  $k$ , if  $\sigma'_i$  is assigned to  $\mathcal{A}_j$ ,  $\mathcal{A}_j$  provides  $\frac{\lceil (1+\epsilon/2)^{j-1} \rceil}{2}$  edge-disjoint paths between  $s_i$  and  $t_i$ . Moreover, since we assign  $\sigma'_i$  to  $\mathcal{A}_j$  only if  $(1 + \epsilon/2)^{j-1} \leq r_i < (1 + \epsilon/2)^j$ ,  $\frac{r_i}{\lceil (1+\epsilon/2)^{j-1} \rceil} \leq (1 + \epsilon/2)$ . Thus  $\frac{\lceil (1+\epsilon/2)^{j-1} \rceil}{2} \geq \frac{r_i}{2+\epsilon}$ , hence there are at least  $\frac{r_i}{2+\epsilon}$  edge-disjoint paths between  $s_i$  and  $t_i$  in  $H_j$ . Since  $H_j \subseteq S$ , there are at least  $\frac{r_i}{2+\epsilon}$  edge-disjoint paths between  $s_i$  and  $t_i$  in  $S$ .

Let  $\text{opt}$  denote the cost of an optimal offline solution which maintains edge-connectivity of  $r_i$  for each demand  $i$ . Let  $|(H)$  denote the total cost of edges in graph  $H$ , where  $H$  is a subset of  $G$ . We prove  $|(S)$  is no more than  $O(\frac{\alpha \log k}{\log(1+\epsilon/2)}) \cdot \text{opt}$ . Let  $\text{opt}_j$  denote the cost of an optimal solution for maintaining edge-connectivity of  $r_i$  for each demand  $i$  such that  $(1 + \epsilon/2)^{j-1} \leq r_i < (1 + \epsilon/2)^j$ . Since the set of such demands is a subset of all demands  $\text{opt}_j \leq \text{opt}$ .  $\mathcal{A}_j$  is  $\alpha$ -competitive to  $\text{opt}_j$ , thus

$$|(H_j) \leq \alpha \cdot \text{opt}_j \leq \alpha \cdot \text{opt} . \quad (9.12)$$

Since  $S = \bigcup_{j=1}^l H_j$ ,  $|(S) = \sum_{j=1}^l |(H_j)$ . Thus by Equation (9.12),

$$|(S) \leq \sum_{j=1}^l \alpha \cdot \text{opt}_j \leq \sum_{j=1}^l \alpha \cdot \text{opt} .$$

Since  $l \leq O\left(\frac{\log k}{\log(1+\epsilon/2)}\right)$ ,

$$|S| \leq O\left(\frac{\alpha \log k}{\log(1+\epsilon)}\right) \cdot \text{opt} .$$

□

Using Theorems 9.8 and 9.7 for online survivable Steiner tree and forest in Section 9.3 and Theorem 9.9, we can immediately imply the following corollaries.

**Corollary 9.5.** *There exists an algorithm for the Survivable Steiner tree network design problem that: (i) provides a solution that partially satisfies every demand of  $r_i$  connectivity with  $\frac{r_i}{2+\epsilon}$  edge-disjoint paths, and (ii) is  $O\left(\frac{\log n \log k}{\log(1+\epsilon)}\right)$ -competitive to an optimal solution that maintains  $r_i$ -connectivity for every demand.*

**Corollary 9.6.** *There exists an algorithm for the Survivable Steiner forest network design problem that: (i) provides a solution that partially satisfies every demand of  $r_i$  connectivity with  $\frac{r_i}{2+\epsilon}$  edge-disjoint paths, and (ii) is  $O\left(\frac{\log^2 n \log k}{\log(1+\epsilon)}\right)$ -competitive to an optimal solution that maintains  $r_i$ -connectivity for every demand.*

## 9.5 From Oblivious I.I.D. to Prophet and Applications to Online Problems

In this section we show how one can use an oblivious competitive algorithm for an online optimization problem in the i.i.d. setting to obtain a competitive algorithm for that problem in the prophet setting. We first define and formulate a general set of online problems. Note that for simplicity, we only consider cost minimization problems, but one can similarly obtain the same statements for welfare



maximization problems as well.

Let  $\mathcal{P}$  be an online problem. Let  $Q$  be a set of queries or demands. Let  $\mathcal{E}$  be a set of elements, such that each response of an algorithm to a demand  $\sigma \in Q$  is a subset of  $\mathcal{E}$ . At each online step  $i$ , an online algorithm is given a demand  $\sigma_i \in Q$ . Then the algorithm needs to provide a response  $R_i \subseteq \mathcal{E}$  that satisfies the given demand. Finally let  $C : 2^{\mathcal{E}} \rightarrow \mathbb{R}$  denote a monotone and sub-additive cost function that maps each subset of  $\mathcal{E}$  to a real number denoting the cost. The overall cost of an online algorithm that responds  $R_1, \dots, R_T$  to demands  $\sigma_1, \dots, \sigma_T$  is computed as  $C(\bigcup_{i=1}^T R_i)$ . To clarify the notations, consider the online SNDP. Given a graph  $G = (V, E)$ ,  $Q$  is the set of triples  $(u, v, r)$  such that  $u, v \in V$  and  $r$  is an integer such that there exist at least  $r$  edge-disjoint paths from  $u$  to  $v$  in  $G$ . Let  $\mathcal{E}$  be the set of edges  $E$ . Now  $R_i \subseteq E$  is a feasible response to a given demand  $\sigma_i = (u_i, v_i, r_i)$ , if adding  $R_i$  to the existing graph guarantees  $r_i$  edge-disjoint paths from  $u_i$  to  $v_i$ . The overall solution of an algorithm up to time  $T$  is  $R = \bigcup_{i=1}^T R_i$ . The cost function is defined as the total cost of edges in the solution, i.e.  $C(R) := \sum_{e \in R} C(e)$ , where  $C(e)$  denotes the cost of a single edge  $e$ . Similarly one can formulate other fundamental online optimization problems such as online set-cover, online facility location, etc. in this way.

For an online problem in the i.i.d. setting we are given a probability distribution  $D$  over  $Q$ . At each time  $i$ , a random demand  $\sigma_i$  is drawn randomly and independently from distribution  $D$ . Let  $\mathcal{P}_{iid}^D$  denote problem  $\mathcal{P}$  in the i.i.d. setting given probability distribution  $D$ . For an online problem in the prophet setting we are given  $T$  probability distributions over  $Q$ ,  $D = \langle D_1, \dots, D_T \rangle$ . At each time  $i$ ,

a random demand  $\sigma_i$  is drawn randomly and independently from distribution  $D_i$ . Let  $\mathcal{P}_{pht}^D$  denote problem  $\mathcal{P}$  in the prophet setting given the sequence of probability distributions  $D$ . Now given  $\mathcal{P}_{pht}^D$ , we define a corresponding i.i.d. instance of the problem as follows. Define  $D^*$  to be the average of all distributions  $D_1, \dots, D_T$ , i.e.  $D^* = \sum_{i=1}^T \frac{D_i}{n}$ . Let  $\mathcal{A}$  be an oblivious  $\alpha$ -competitive algorithm for  $\mathcal{P}_{iid}^{D^*}$ . In the following we show that  $\mathcal{A}$  is  $\frac{2e}{e-1}(1 + o(1))\alpha$ -competitive for  $\mathcal{P}_{iid}^{D^*}$ .

First we need to define another problem  $W_{pht}^D$ , which is the same as  $\mathcal{P}_{pht}^D$ , but in the beginning, with probability  $e^{-\frac{T(1-\frac{1}{e})}{8}}$  we do not provide any demand at all, and with the remaining probability we remove  $\frac{1}{2}(1 - \frac{1}{e})$  fraction of the  $T$  distributions uniformly at random, i.e. we do not provide any demands at those times.

Consider a subset of demands  $\sigma = \sigma_1, \dots, \sigma_{\frac{1}{2}(1-\frac{1}{e})}$  in  $W_{pht}^D$ . We prove that the probability that a super set of  $\sigma$  is an online scenario for  $\mathcal{P}_{iid}^{D^*}$  is no less than the probability that  $\sigma$  is an online scenario for  $W_{pht}^D$ , or roughly speaking if one ignores the order of the online demands,  $W_{pht}^D$  is an easier problem than  $\mathcal{P}_{iid}^{D^*}$ .

**Lemma 9.7.** *For every subset of demands  $\sigma = \sigma_1, \dots, \sigma_{\frac{1}{2}(1-\frac{1}{e})}$ , the probability the online scenario for  $\mathcal{P}_{iid}^{D^*}$  is a super set of  $\sigma$  is no less than the probability that the online scenario for  $W_{pht}^D$  is  $\sigma$ .*

**Proof.** Without loss of generality we can assume the demands in  $D_1, \dots, D_T$  are different. In other words each demand either arrive at a specific time, or never arrives. We can easily add dummy demands, if a demand can possibly arrive at two or more different time steps. For drawing a random online scenario in  $\mathcal{P}_{iid}^{D^*}$ , we define an equivalent random process as follows. At each time, first we draw a distribution

from all  $T$  distributions uniformly at random, and then we draw a random demand from that distribution. Now for every subset  $S$  of distributions of size  $\frac{1}{2}(1 - \frac{1}{e})T$  we show that the probability that  $S$  is drawn in  $W_{pht}^D$  is less than or equal to the probability that a super set of  $S$  is drawn in  $\mathcal{P}_{iid}^{D*}$ . Note that this shows that for every subset of demands  $\sigma = \sigma_1, \dots, \sigma_{\frac{1}{2}(1-\frac{1}{e})}$ , the probability that a super set of  $\sigma$  is an online scenario for  $W_{pht}^D$  is less than or equal to the probability that  $\sigma$  is an online scenario for  $\mathcal{P}_{iid}^{D*}$ . Since if the set of distributions is fixed, we can use the same random coin for drawing random demands from the same distributions in  $W_{pht}^D$  and  $\mathcal{P}_{iid}^{D*}$ .

Using a Chernoff bound we show that with probability  $e^{-\frac{T(1-\frac{1}{e})}{8}}$  there are at least  $\frac{1}{2}(1 - \frac{1}{e})T$  distinct distributions drawn by  $\mathcal{P}_{iid}^{D*}$ . Let  $Y$  be the number of distinct distributions that are drawn. Let  $X_i$  be 1 if distribution  $i$  is drawn and 0 if not. We have

$$E[Y] = \sum_{i=1}^T E[X_i] = \sum_{i=1}^T P[X_i = 1] = T - \frac{(T-1)^T}{T^{T-1}} \geq T(1 - \frac{1}{e}). \quad (9.13)$$

Since  $X_i$ 's are negatively correlated we can use a Chernoff bound to bound the probability that we have less than  $\frac{1}{2}T(1 - \frac{1}{e})$  distinct distributions in  $\mathcal{P}_{iid}^{D*}$ .

$$Pr[Y \leq \frac{1}{2}T(1 - \frac{1}{e})] \leq e^{-\frac{T(1-\frac{1}{e})}{8}}. \quad (9.14)$$

Since there is a symmetry across distributions in the random process, the probability for every subset of distributions of size  $k$  to be drawn is the same. Thus for every subset of distributions  $S$  of size  $\frac{1}{2}(1 - \frac{1}{e})T$  the probability that  $S$  is drawn in  $W_{pht}^D$  is less than or equal to the probability that a super set of  $S$  is drawn in  $\mathcal{P}_{iid}^{D*}$ .  $\square$

Let  $C_W^D(\mathcal{A})$  and  $C_{iid}^{D^*}(\mathcal{A})$  denote the expected cost of algorithm  $\mathcal{A}$  for online random scenarios drawn by  $W_{pht}^D$  and  $\mathcal{P}_{iid}^{D^*}$ , respectively. By Lemma 9.7, since  $\mathcal{A}$  is oblivious and indifferent to the order of the input,

$$C_W^D(\mathcal{A}) \leq C_{iid}^{D^*}(\mathcal{A}). \quad (9.15)$$

Recall that in  $W_{pht}^D$  with probability  $e^{-\frac{T(1-\frac{1}{e})}{s}}$  we do not provide any demand at all, and with probability  $1 - e^{-\frac{T(1-\frac{1}{e})}{s}}$  we remove  $\frac{1}{2}(1 - \frac{1}{e})$  fraction of the  $T$  distributions uniformly at random. Moreover  $\mathcal{A}$  is oblivious and the cost function is monotone and subadditive. Thus

$$\frac{1 - e^{-\frac{T(1-\frac{1}{e})}{s}}}{\frac{1}{2}(1 - \frac{1}{e})} C_{pht}^D(\mathcal{A}) \leq C_W^D(\mathcal{A}), \quad (9.16)$$

where  $C_{pht}^D(\mathcal{A})$  is the expected cost of algorithm  $\mathcal{A}$  for an online random scenario drawn by  $\mathcal{P}_{pht}^D$ .

Now we are ready to prove that one can use  $\mathcal{A}$  to obtain a competitive algorithm for  $\mathcal{P}_{pht}^D$ . Let  $opt_{pht}^D$  and  $opt_{iid}^{D^*}$  denote the expected cost of an optimal offline solution for a random online scenario drawn in  $\mathcal{P}_{pht}^D$  and  $\mathcal{P}_{iid}^{D^*}$ , respectively.

$$\begin{aligned} \frac{\frac{1 - e^{-\frac{T(1-\frac{1}{e})}{s}}}{\frac{1}{2}(1 - \frac{1}{e})} C_{pht}^D(\mathcal{A})}{opt_{pht}^D} &\leq \frac{\frac{1 - e^{-\frac{T(1-\frac{1}{e})}{s}}}{\frac{1}{2}(1 - \frac{1}{e})} C_{pht}^D(\mathcal{A})}{opt_{iid}^{D^*}} && \text{Since } opt_{iid}^{D^*} \leq opt_{pht}^D \\ &\leq \frac{C_W^D(\mathcal{A})}{opt_{iid}^{D^*}} && \text{By Inequality (9.16)} \\ &\leq \frac{C_{iid}^{D^*}(\mathcal{A})}{opt_{iid}^{D^*}} && \text{By Inequality (9.15)} \\ &\leq \alpha. && \text{Since } \mathcal{A} \text{ is } \alpha\text{-competitive} \end{aligned}$$

Thus  $\frac{C_{pht}^D(\mathcal{A})}{opt_{pht}^D} \leq \frac{2e}{e-1}(1 + o(1))\alpha$ .

**Theorem 9.10.** *Given an oblivious  $\alpha$ -competitive online algorithm for an online problem in the i.i.d. setting, there exists a  $\frac{2e}{e-1}(1 + o(1))\alpha$ -competitive online algorithm for the problem in the prophet setting, where the competitive ratio approaches  $\frac{2e}{e-1}\alpha$  exponentially fast as the number of online steps  $T$  grows.*

Interestingly, we can use Theorem 9.10 to obtain online competitive algorithms for other fundamental problems in the prophet setting. Using the oblivious i.i.d. algorithms in [GGLS08] for i.i.d. vertex cover and i.i.d. facility location, we may obtain  $O(1)$ -competitive online algorithms for prophet vertex cover and prophet facility location. Moreover using the oblivious i.i.d. algorithm for i.i.d. set cover in [GGL+08], we can obtain  $O(\log n)$ -competitive algorithm for the prophet set cover problem.

## 9.6 Stochastic Survivable Network Design

In this section we study the stochastic variant of survivable network design. Recall that in this model, the input consists of both offline and online data. The offline input is given in advance to the algorithm and specifies a graph  $G = \langle V(G), E(G) \rangle$ , a source node  $s$ , an integer  $k$  denoting the connectivity requirement, a distribution  $D$  of probabilities over the vertices of the graph, and an integer  $l$  denoting the number of demands. Next, an online stream of demands  $t_1, t_2, \dots, t_l$  arrive one by one, upon every arrival of which we are required to update our solution to make sure the newly arrived vertex is  $k$ -connected to the source node  $s$ . No prior information about the demands is given in advance, however, we're guaranteed that

the demands are randomly and independently drawn from the given distribution  $D$ .

In this section we show that a slight variation of the greedy algorithm performs almost optimally in this setting. This improves upon the result of Garg *et al.* [GGLS08] which is a constant bound for the case where the connectivity is equal to 1. This is surprising since the proven bounds of the online algorithms for survivable network design are much worse than that of single the connectivity [GKR12]. From a high-level perspective our method is similar to the greedy algorithm in Garg *et al.* [GGLS08], however, such a generalization requires a deep and innovative study of the  $k$ -connected graphs. In Section 9.6.2 we present a structural lemma which basically simplifies the analysis of our greedy algorithm.

**Theorem 9.11.** *There exists an oblivious 4-approximation algorithm for the stochastic survivable network design problem.*

Moreover, we generalize the algorithm to the setting in which the demands are drawn from independently from an unknown distribution  $D$ .

**Theorem 9.12.** *There exists an oblivious  $O(\log n)$ -approximation algorithm for the stochastic survivable network design problem with an unknown distribution.*

Note that achieving a constant factor approximation algorithm is not possible for an unknown distribution due to the work of Garg *et al.* [GGLS08]. In particular, they show there is an  $\Omega(\frac{\log n}{\log \log n})$  lower bound for Steiner tree, therefore our greedy algorithm is almost tight.

The rest of this section is summarized in the following. In Section 9.6.1 we describe our algorithm and show a sufficient condition for obtaining the constant

approximation factor. We also prove the approximation factor of the algorithm for unknown distribution. Finally, in Section 9.6.2 we provide a study of  $k$ -connected graphs and prove the structural lemma.

### 9.6.1 Algorithm

In this section we explain our algorithm and outline the analysis. The algorithm is as follows. Before any demand arrives, we simulate a stream  $t_1^*, t_2^*, \dots, t_l^*$  of demands by drawing  $l$  random vertices from probability distribution  $D$ . Next, we find a 2-approximation Steiner network of the graph that  $k$ -connects all the simulated demands to the source node via the algorithm of Jain [Jai01]. Let  $H$  denote this network. Based on this solution, for every node  $v$  of the graph, we find a minimum cost  $k$ -flow from  $v$  to  $s$  that uses the edges of  $H$  for free and call that the *partial solution* of  $v$ . Now we're ready to run the algorithm on the actual demands. We start with an empty graph for our initial solution. Every time a demand  $t_i$  arrives, we update our solution by adding all of the edges of the partial solution of  $t_i$  to our current solution. Notice that our solution is *oblivious* in the sense that the  $k$ -flow of each demand is regardless of the queries that have arrived prior to that demand.

In the rest we show that the approximation factor of our algorithm is bounded by 4. Let for every list  $L$  of vertices,  $\text{sol}(L)$  be the set of all subsets of  $E(G)$  that  $k$ -connect all vertices of  $L$  to  $s$ . Moreover, let for a subset of edges  $Q$ ,  $||Q$  denote the total cost of the edges in  $Q$ . We define a pseudo-cost function for a list of vertices

$L = \langle v_1, v_2, \dots, v_{|L|} \rangle$  and another vertex  $u$  as follows:

$$\beta(L, u) = \max_{F_1 \in \text{sol}(L)} \min_{F_2 \in \text{sol}(L \cup \{u\})} (|F_2 \setminus F_1|)$$

In words,  $\beta(L, u)$  is the smallest cost that we need to pay in order to  $k$ -connect  $u$  to the source in **any** solution that already  $k$ -connects all vertices of  $L$  to the source. Monotonicity of  $\beta$  follows from its definition; the more vertices we add to  $L$ , the less costly it will be to satisfy another node in any solution that satisfies  $L$ . In other words, by adding more vertices to  $L$ , the max in the formulation of  $\beta$  will be more constrained.

The main observation that enables us to prove a constant approximation factor for our algorithm is a bound on the pseudocost of a pair  $(L, u)$ . More precisely, in Section 9.6.2 we prove the following theorem.

**Theorem 9.13** [to be proven in Section 9.6.2]. *For any set  $S$  of vertices we have*

$$\sum_{v \in S} \beta(S \setminus \{v\}, v) \leq 2c(\text{opt}_S)$$

where  $\text{opt}_S$  is a minimum cost Steiner network that  $k$ -connects all vertices of  $S$  to the source node  $s$ .

Roughly speaking, the idea is to consider a minimum weight subgraph that  $k$ -connects all vertices of  $S$  to the source node. Next, we find a  $k$ -flow for each node in  $S$  that  $k$ -connects this node to other vertices of the set. We do this in a way that  $k$ -flows use only the edges of the Steiner network, and that every edge appears in at most two  $k$ -flows. This in turn implies that summation of the  $\beta$  functions for all nodes is bounded by two times  $c(\text{opt}_S)$ . This is discussed in details in Section 9.6.2.



An immediate corollary of Theorem 9.13 is that if we randomly draw  $l$  demands  $d_1, d_2, \dots, d_l$  from  $D$ ,  $\beta(\{d_1, d_2, \dots, d_{l-1}\}, d_l)$  is no more than  $2/l$  times the minimum cost of  $k$ -connecting all vertices of  $d_1, d_2, \dots, d_l$  to the source.

Now, recall that before the stream of demands arrives, our algorithm randomly draws  $l$  demands  $t_1^*, t_2^*, \dots, t_l^*$  and finds a 2-approximation solution for these demands. We refer to this subgraph as  $H$ . Since we use a 2-approximation algorithm and all demands are drawn from  $D$ , the expected cost of  $H$  is bounded by  $2c(\text{opt})$ . Moreover, for every actual demand  $t_i$ , the total cost of the edges in the partial solution of  $t_i$  that are not in  $H$  is bounded by  $\beta(\{t_1^*, t_2^*, \dots, t_l^*\}, t_i)$  which is by monotonicity bounded by  $\beta(\{t_1^*, t_2^*, \dots, t_{i-1}^*, t_{i+1}^*, \dots, t_l^*\}, t_i)$ . Notice that all  $t_i^*$ 's and  $t_i$ 's are independently drawn from  $D$  and thus the expected cost of such edges in the partial solution of a demand  $t_i$  is no more than  $2c(\text{opt})/l$  in expectation. Therefore, if we add the cost of all such edges for all of the demands to the cost of  $H$ , it yields an upperbound on the total cost of our algorithm as follows

$$c(T) + l \frac{2c(\text{opt})}{l} \leq 4c(\text{opt}) .$$

This proves a 4-approximation bound on the cost of the greedy algorithm.

Now suppose the distribution of demands is unknown. In this case, the greedy algorithm is simply  $k$ -connecting new demand  $t_i$  to the source by buying a minimum cost set of edges. Let  $T_i$  denote the set of these edges. We show that as we serve more demands the expected cost of  $T_i$  decreases for larger  $i$ . For every demand  $i$  let  $L_i = \{t_1, \dots, t_i\}$  and  $p(i) = 2^{\lceil \log i \rceil}$ . We have

$$c(T_i) \leq \beta(L_{i-1}, t_i) \leq \beta(L_{p(i)-1}, t_i) .$$

Using Theorem 9.13 and the fact that all items are drawn from independently from the same distribution, in expectation we have

$$\beta(L_{p^{(i)}-1}, t_i) \leq 2c(\text{opt}_{L_{p^{(i)}-1}})/p_i .$$

Now we use  $c(\text{opt})$  as an upper bound for  $c(\text{opt}_{L_{p^{(i)}-1}})$ . Therefore,

$$\sum_{i=1}^l c(T_i) \leq \sum_{i=1}^l \frac{2c(\text{opt})}{2^{\lfloor \log i \rfloor}} \leq 2 \log(l) c(\text{opt}) .$$

The number of demands is at most  $n$ , and thus the greedy algorithm is  $O(\log n)$ -approximation for unknown distribution.

## 9.6.2 Structural Lemma for $k$ -connected Graphs

In this section we provide a study of  $k$ -connected Steiner graphs. Roughly speaking, we state a lemma that shows the  $k$ -connectivity property suffices for the existence of concurrent  $k$ -flows for all the vertices such that the congestions on the edges are bounded by a factor of 2. We formally define  $k$ -flows in the remainder. As a result of this lemma, we can prove the following theorem, which has been used in Section 9.6.1.

**Theorem 9.13.** *For any set  $S$  of vertices we have*

$$\sum_{v \in S} \beta(S \setminus \{v\}, v) \leq 2c(\text{opt}_S)$$

where  $\text{opt}_S$  is a minimum cost Steiner network that  $k$ -connects all vertices of  $S$  to the source  $s$ .

The theorem states that the overall cost of  $k$ -connecting every vertex  $v \in S$  to the optimum solution that  $k$ -connects  $S \setminus \{v\}$  is no more two times the cost of

the optimum solution that  $k$ -connects all vertices in  $S$ . We prove this theorem via a structural lemma on unweighted  $k$ -connected Steiner graphs. To this end let us first define  $k$ -flows.

**Definition 9.1.** *Consider an  $S$ - $k$ -connected graph  $G$  which is undirected and unweighted. A  $k$ -flow for a vertex  $v \in S$  in  $G$  is the union of  $k$  edge-disjoint directed paths that  $k$ -connects  $v$  to  $S \setminus \{v\}$ .*

Let  $\text{opt}_S$  be a minimum cost solution that  $k$ -connects every vertex to the source node  $s$ . Since every vertex is  $k$ -connected to  $s$  it follows that every other pairs of vertices are  $k$ -connected too. Therefore  $\text{opt}_S$  is an  $S$ - $k$ -connected graph. Now for every  $v \in S$  let  $F(v)$  be a  $k$ -flow in  $\text{opt}_S$  from  $v$  to  $S \setminus \{v\}$ . We note that  $\beta(S \setminus \{v\}, v) \leq c(F(v))$  because one can  $k$ -connect  $v$  to  $S \setminus \{v\}$  through  $F(v)$ . As a result we have

$$\sum_{v \in S} \beta(S \setminus \{v\}, v) \leq \sum_{v \in S} c(F(v)) .$$

The following structural lemma states that one can find such  $k$ -flows in an  $S$ - $k$ -connected graph for every vertex in  $S$  in a way that every edge appears in at most two  $k$ -flows. Therefore, for such set of  $k$ -flows in  $\text{opt}_S$  we have

$$\sum_{v \in S} c(F(v)) \leq 2c(\text{opt}_S)$$

which completes the proof of Theorem 9.13.

**Lemma 9.8.** *In every  $S$ - $k$ -connected graph  $G$  there exists a set of  $k$ -flows  $\{F(v) | v \in S\}$  such that every edge is used at most once in each direction.*

**Proof.** Without loss of generality we can assume that the graph is minimal, i.e. removing any edge decreases the connectivity of  $S$  to  $k - 1$ . This minimality assumption implies that every edge participates in a separating cut of  $S$  with size  $k$ . As a result, every minimum size separating cut of  $S$  has exactly  $k$  edges. We prove the lemma by induction on the number of vertices in  $G$ . In particular, we find a minimum size separating cut on  $S$  and considering the following two cases:

- *Basis:* In every min-cut  $C = [A, B]$  either  $A$  or  $B$  has one vertex. Recall that every edge  $e$  belongs to a min-cut of  $S$ . For such cut, we now that one of the sides has size one, therefore at least one endpoints of  $e$  belongs to  $S$ .

Now we explain how to find the  $k$ -flows for every  $v \in S$ . Take a neighbor  $u$  of  $v$ . If  $u \in S$  then we draw a direct flow from  $v$  to  $u$ . If  $u \notin S$ , then every neighbor of  $u$  belongs to  $S$ , because every edge  $(u, w)$  has to have at least one endpoint in  $S$ . Without loss of generality assume that the neighbors of  $u$  are ordered such that each of them has a unique *next*. Let  $w$  be the next vertex after  $v$ . We draw a flow from  $v$  to  $u$  and from  $u$  to  $w$ . Since the degree of  $v$  is at least  $k$  the total number of outflows from  $v$  to  $S \setminus \{v\}$  is at least  $k$ . Moreover, in this manner every edge has at most one flow in each direction.

- *Inductive step:* There exists a min-cut  $C = [A, B]$  of  $S$  such that both  $A$  and  $B$  have more than one vertex. In this case we proceed with the following two actions. We once contract all the vertices in  $B$  into a vertex  $v_B$ . Let  $S_A = \{S \cap A\} \cup \{v_B\}$  be the set of Steiner vertices in the new graph. As a consequence, this contracted graph is  $S_A$ - $k$ -connected and its size is smaller

than  $G$ . Therefore, due to the induction we can find  $k$ -flows for every  $v \in S_A$  to  $S_A \setminus \{v\}$  such that every edge in  $A$  has a flow of at most one in each direction. Similarly, we can find flows in  $B$  by contracting  $A$  and then using the induction.

In this way, for every  $v \in S$  we get  $k$  flows that leave  $v$ , but may not end up to a vertex in  $S \setminus \{v\}$ . This is because there are some flows that go to  $v_B$  or  $v_A$  in the contracted graphs, and become incomplete after mapping them to the original graph. However, we show that the flows of the other side can be used in order to continued the incomplete flows to reach  $S$ .

Consider the graph achieved from contracting  $B$ . Let  $in_A(v_B)$  be the set of flows from  $S_A \setminus \{v_B\}$  to  $v_B$  and  $out_A(v_B)$  be the set of flows from  $v_B$  to  $S_A \setminus \{v_B\}$ . Note that  $out_a(v_B)$  is of size  $k$  and  $in_A(v_B)$  is of size at most  $k$ , because there are  $k$  edges in the cut. Likewise, we we define  $in_B(v_A)$  and  $out_B(v_A)$  for the graph achieved from contracting  $A$

Now consider a flow of  $in_A(v_B)$ . This flow ends up to an edge  $(x, y)$  of the cut, where  $x \in A$  and  $y \in B$ . Since  $out_B(v_A)$  is of size  $k$ , for every such  $y$  there exists a flow from  $y$  to  $S_B$ . Therefore, we can continue that flow in  $A$  such that it reaches a vertex of  $S$ . By doing this for all such flows in  $in_A(v_B)$  and  $in_B(v_A)$  every  $v \in S$  has a  $k$ -flow such that every edge of  $G$  is used at most once in each direction.

□

## Appendices

### A 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 generates a much larger weight than the optimum.

Consider a graph  $G$  as shown in Figure 2, 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.

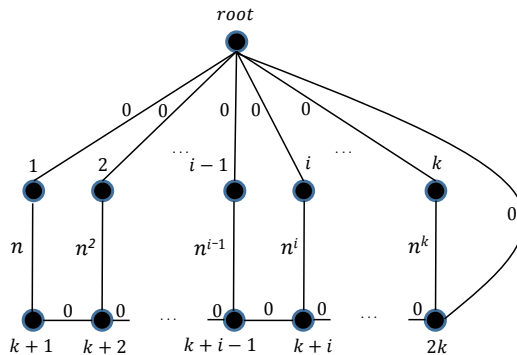


Figure 2: The graph  $G$  consists of  $2k + 1$  nodes and  $3k$  edges.

**Proof of Theorem 4.3:** The adversary consecutively presents terminals starting

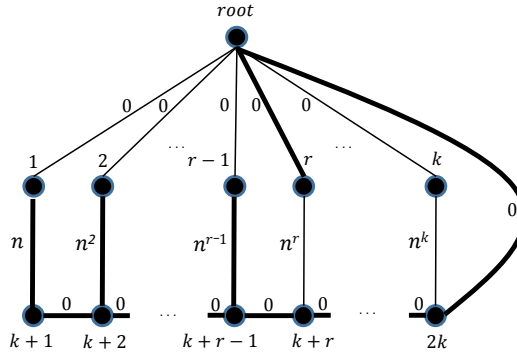


Figure 3: The highlighted subtree represents an optimum solution  $OPT_3$ .

from node 1. At each step  $i$  the algorithm  $A$  adds some edges to its current solution such that the  $i^{th}$  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

$T$ :

$$\begin{aligned} T &= \{\{i, k+i\} | 1 \leq i \leq r-1\} \\ &\cup \{\{i, i+1\} | k \leq i < 2k\} \\ &\cup \{\{2k, root\}, \{r, root\}\}. \end{aligned}$$

As we can observe in Figure 3,  $T$  meets the degree bounds and connects the first  $r$  terminals to the root, so forms a valid solution. We have

$$w(OPT) \leq w(T) = \sum_{i=1}^{r-1} n^i \in O(n^{r-1}) .$$

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

$$\mathbb{E}[w(X_r)] = \sum_{i=1}^r q_{ir} \cdot n^i \geq q_{rr} \cdot n^r \geq \frac{n^r}{2} .$$

Hence,  $\mathbb{E}[w(X)] \geq \Omega(n) \cdot w(OPT)$ .

2. The process continues until step  $k$ , i.e., for every  $i \leq k$ ,  $q_{ii} \leq \frac{1}{2}$ .

$$\begin{aligned} \mathbb{E}[\deg(root)] &\geq \sum_{i=1}^k p_{ik} \geq \sum_{i=1}^k p_{ii} \geq \sum_{i=1}^k (1 - q_{ii}) \\ &\geq \frac{k}{2} = \frac{n-1}{4} . \end{aligned}$$

Hence,  $\mathbb{E}[\deg(root)] \geq \Omega(n) \cdot b$ .

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.  $\square$



## B 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.

**Proof 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  $\{v_2, v_3, \dots, v_n\} \setminus C$ . We do this until  $\{v_2, v_3, \dots, v_n\} \setminus C = \emptyset$ , 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  $\mathcal{A}$  add all edges in  $G$  to  $H$ . Hence  $\deg(v_1) = n - 1$ .

Now for 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  $g_k \subset g_{k-1} \subset \dots \subset g_1$ . There is a single node that exists in all group demands. The optimal offline algorithm 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  $n - 1$ .  $\square$

## C Omitted Proofs

**Proof of Lemma 6.2:** For every  $x \in \mathcal{R}^{\geq 0}$  we have:

$$\begin{aligned}
 Pr[\max X_i \geq x] &= 1 - F(x)^n \\
 &= 1 - (1 - G(x))^n \\
 &= 1 - \left( \frac{1}{1 - G(x)} \right)^{-n} \\
 &= 1 - \left( 1 + \frac{G(x)}{1 - G(x)} \right)^{-n} \\
 &\leq 1 - \exp\left( \frac{-nG(x)}{1 - G(x)} \right) .
 \end{aligned} \tag{17}$$

We complete the proof by proving for every  $\epsilon > 0$  that there exists an  $n_\epsilon$  such that for every  $n \geq n_\epsilon$  and  $0 \leq z \leq 1$ , the ratio between  $A(n, z) = 1 - \exp(-nz/(1 - z))$  and  $B(n, z) = 1 - \exp(-nz)$  is no more than  $1/(1 - \epsilon)$ .

For every  $n$  and  $z$  there are two cases:

- If  $\ln(n)/n \leq z \leq 1$  then we have:

$$\frac{A(n, z)}{B(n, z)} \leq \frac{1}{B(n, z)} \leq \frac{1}{1 - \exp(-\ln(n))} = \frac{1}{1 - 1/n} . \tag{18}$$

- If  $0 \leq z \leq \ln(n)/n$ , we use partial derivatives of the functions to find an upper bound of their ratio. In the following the derivative of a function is

with respect to variable  $z$ .

$$\begin{aligned}
\frac{A(n, z)}{B(n, z)} &= \frac{\int_0^z A'(n, w)dw}{\int_0^z B'(n, w)dw} \\
&= \frac{\int_0^z B'(n, w) \frac{A'(n, w)}{B'(n, w)} dw}{\int_0^z B'(n, w)dw} \\
&\leq \frac{\int_0^z B'(n, w)dw}{\int_0^z B'(n, w)dw} \cdot \max_{0 \leq w \leq z} \left\{ \frac{A'(n, w)}{B'(n, w)} \right\} \\
&= \max_{0 \leq w \leq z} \left\{ \frac{A'(n, w)}{B'(n, w)} \right\} \\
&= \max_{0 \leq w \leq z} \left\{ \frac{n \exp(-nz/(1-z))/(1-z)^2}{n \exp(-nz)} \right\} \\
&= \max_{0 \leq w \leq z} \left\{ \frac{\exp(-nz^2/(1-z))}{(1-z)^2} \right\} \\
&\leq \frac{1}{(1-z)^2} \\
&\leq \frac{1}{1 - 2 \ln(n)/n + \ln^2(n)/n^2} . \tag{19}
\end{aligned}$$

Note that the denominators of both (18) and (19) become greater than  $1 - \epsilon$  as  $n$  becomes greater than some  $n_\epsilon$ , thus the proof of the lemma follows.  $\square$

**Proof of Lemma 6.3:** Recall that  $s(x) = j(x)/n$  is a number from 0 to 1. We prove the correctness of the lemma by analyzing it for two different ranges of  $s(x)$ . For simplicity we may refer to  $s(x)$  as  $s$  in different parts of the proof. Suppose  $s_0 = \min(0.5, \alpha)\epsilon$ . For  $0 \leq s \leq s_0$  we have:

$$\begin{aligned}
1 - h(s) + nG(x) \int_s^1 h(r)dr &\geq nG(x) \int_s^1 h(r)dr \\
&\geq nG(x) \int_{s_0}^1 h(r)dr \\
&= nG(x) \left( \int_0^1 h(r)dr - \int_0^{s_0} h(r)dr \right) \\
&\geq nG(x) \left( \int_0^1 h(r)dr - s_0 \right) . \tag{20}
\end{aligned}$$

From the second property of  $\alpha$ -strong functions we have  $\int_0^1 h(r)dr \geq \alpha$ . Also, for every  $z \in \mathcal{R}^{\geq 0}$  it holds that  $z \geq 1 - \exp(-z)$ . By using these two inequalities in Inequality (20) the lemma is proved for this case:

$$\begin{aligned}
1 - h(s) + nG(x) \int_s^1 h(r)dr &\geq nG(x) \left( \int_0^1 h(r)dr - s_0 \right) \\
&\geq (1 - \exp(-nG(x))) (\alpha - s_0) \\
&\geq (1 - \exp(-nG(x))) \alpha \left( 1 - \frac{s_0}{\alpha} \right) \\
&\geq (1 - \exp(-nG(x))) \alpha (1 - \epsilon) \quad s_0 \leq \alpha \epsilon
\end{aligned}$$

Now what remains is the case that  $s_0 < s \leq 1$ . Again, for this case we want the following inequality to hold:

$$\frac{1 - h(s) + nG(x) \int_s^1 h(r)dr}{1 - \exp(-nG(x))} \geq (1 - \epsilon) \alpha . \quad (21)$$

Recall that for every  $x \geq \theta_n$ ,  $s(x) = j(x)/n$  where  $j(x)$  is the greatest index for which  $\theta_j > x \geq \theta_{j+1}$ . Since  $G$  is a strictly decreasing function, we have  $G(\theta_j) < G(x) \leq G(\theta_{j+1})$ . Recall that for every  $1 \leq i \leq n$  we have  $q_i = q_{i-1}(1 - G(\theta_i))$ , or equivalently  $G(\theta_i) = 1 - q_i/q_{i-1}$ . Therefore we can bound  $G(x)$  as follows:

$$1 - q_j/q_{j-1} < G(x) \leq 1 - q_{j+1}/q_j . \quad (22)$$

Now, finding a lower bound for  $1 - q_j/q_{j-1}$  and an upper bound for  $1 - q_{j+1}/q_j$  in Inequality (22) gives us a lower bound and an upper bound for  $G(x)$ . For the lower bound we have

$$G(\theta_j) = 1 - q_j/q_{j-1} = \frac{q_{j-1} - q_j}{q_{j-1}} = \frac{-(q_j - q_{j-1})}{q_{j-1}} = \frac{-(h(s) - h(s - 1/n))}{h(s - 1/n)} .$$

By multiplying this fraction by  $n/n$  we get:

$$G(\theta_j) = \frac{-\left(\frac{h(s)-h(s-1/n)}{1/n}\right)}{n h(s-1/n)} \geq \frac{-h'(s-1/n)}{n h(s-1/n)}, \quad (23)$$

where the last inequality in (23) comes from the concavity of  $h$ . From the second property of threshold functions there exists some  $\delta_0 \leq s_0$  such that for every  $n \geq 1/\delta_0$ ,  $s_0 + 1/n \leq s \leq 1$  the following inequality holds:

$$\frac{-h'(s-1/n)}{h(s-1/n)} \geq (1-s_0) \frac{-h'(s)}{h(s)}. \quad (24)$$

By using Inequality (24) in Inequality (23), and by using that inequality in Inequality (22), we get:

$$G(x) > (1-s_0) \frac{-h'(s)}{n h(s)}. \quad (25)$$

Similarly one can show the following upper bound on  $G(x)$ :

$$\begin{aligned} G(x) &\leq G(\theta_{j+1}) \\ &= 1 - \frac{q_{j+1}}{q_j} \\ &= \frac{-(h(s+1/n) - h(s))}{h(s)} \\ &= \frac{-\frac{h(s+1/n)-h(s)}{1/n}}{n h(s)} && \text{multiplying by } \frac{n}{n} \\ &\leq \frac{-\frac{h(s+1/n)-h(s)}{1/n}}{n h(s+1/n)} && \text{since } h \text{ is decreasing} \\ &\leq \frac{-h'(s+1/n)}{n h(s+1/n)} && \text{concavity of } h \quad (26) \\ &\leq \frac{1}{1-s_0} \cdot \frac{-h'(s)}{n h(s)}. && \text{second property of threshold functions} \quad (27) \end{aligned}$$

Using these bounds and the following auxiliary lemma we prove the correctness of Inequality (21).

**Lemma .9.** For every  $z < 0$  and  $t \geq 1$  we have:  $(1 - \exp(zt))/(1 - \exp(z)) \leq t$ .

**Proof.** Let  $A(z) = 1 - \exp(z)$  and  $B(z) = 1 - \exp(zt)$ . In the following the derivatives are with respect to  $z$ . For the ratio of these functions we have:

$$\begin{aligned} \frac{A(z)}{B(z)} &= \frac{\int_0^z A'(w)dw}{\int_0^z B'(w)dw} \\ &= \frac{\int_0^z B'(w) \frac{A'(w)}{B'(w)} dw}{\int_0^z B'(w)dw} \\ &\leq \max_{z \leq w \leq 0} \left\{ \frac{A'(w)}{B'(w)} \right\} \frac{\int_0^z B'(w)dw}{\int_0^z B'(w)dw} \\ &\leq \max_{z \leq w \leq 0} \left\{ \frac{-t \exp(tw)}{-\exp(w)} \right\} \\ &= \max_{z \leq w \leq 0} \left\{ t \exp(w(t-1)) \right\} . \end{aligned}$$

Since  $w \leq 0$  and  $t \geq 1$  the  $\exp(w(t-1)) \leq 1$ , and the proof follows.  $\square$  By applying the bounds of Inequalities (25) and (27) to the left hand side of Inequality (21) we have:

$$\begin{aligned} \frac{1 - h(s) + nG(x) \int_s^1 h(r)dr}{1 - \exp(-nG(x))} &\geq \frac{1 - h(s) - (1 - s_0)h'(s)/h(s) \int_s^1 h(r)dr}{1 - \exp(-nG(x))} && \text{Inequality (25)} \\ &\geq \frac{(1 - s_0)(1 - h(s) - h'(s)/h(s) \int_s^1 h(r)dr)}{1 - \exp(-nG(x))} && 1 - h(s) \geq 0 \\ &\geq \frac{(1 - s_0)(1 - h(s) - h'(s)/h(s) \int_s^1 h(r)dr)}{1 - \exp(\frac{h'(s)}{h(s)(1-s_0)})} && \text{Inequality (27)} . \end{aligned}$$

By applying Lemma .9 to the denominator we have:

$$\frac{1 - h(s) + nG(x) \int_s^1 h(r)dr}{1 - \exp(-nG(x))} \geq (1 - s_0)^2 \cdot \frac{1 - h(s) - h'(s)/h(s) \int_s^1 h(r)dr}{1 - \exp(h'(s)/h(s))} .$$

From the third property of  $\alpha$ -strong functions, the fraction at the right hand side of the above inequality is at least  $\alpha$ . Moreover, since  $s_0 \leq 0.5\epsilon$  it holds that  $(1 - s_0)^2 \geq (1 - \epsilon)$ , and thus Inequality (21) holds and the proof of the lemma is complete.  $\square$

**Proof of Lemma 6.4:** Let us first take a look at the first three derivatives of  $A(w)$  which are all continuous and bounded in range  $[0, \tan(a)]$ :

$$A'(w) = 2\alpha a(1 - \exp(-aw))(1 - \alpha + \alpha aw + \alpha \exp(-aw)) - 2w,$$

$$A''(w) = 2\alpha a^2 \exp(-aw)(1 - 3\alpha + \alpha aw + 2\alpha \exp(-aw)) + 2(\alpha^2 a^2 - 1),$$

$$A'''(w) = -2\alpha a^3 \exp(-aw)(1 - 4\alpha + \alpha aw + 4\alpha \exp(-aw)) .$$

In this part of the proof we frequently use one of the implications of intermediate value theorem: if  $f(x)$  and  $f'(x)$  are two continuous and bounded functions, then there exists a root of  $f'(x)$  between every two roots of  $f(x)$ . This also implies that the number of the roots of  $f(x)$  is at most one plus the number of the roots of  $f'(x)$ .

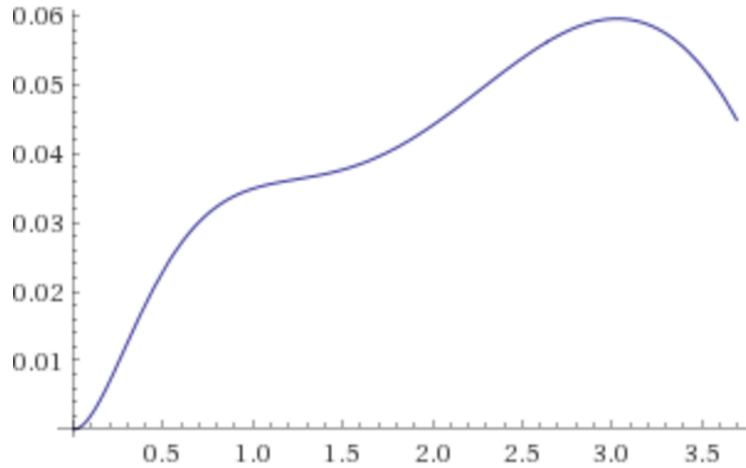


Figure 4: The plot shows function  $A(w)$  for values of  $w$  from 0 to  $\tan(a) \approx 3.7$ .

We claim that  $A'''(w)$  has at most two roots. The reason for this is because  $-2\alpha a^3 \exp(-aw)$  is always non-zero, and  $1 - 4\alpha + \alpha aw + 4\alpha \exp(-aw)$  has at most two roots, because its derivative,  $\alpha a(1 - 4 \exp(-aw))$  has exactly one root, which is

$\ln(4)/a$ .

The fact that  $A'''(w)$  has at most two roots implies that  $A''(w)$  has at most three roots, which are  $w_1 \approx 0.28157$ ,  $w_2 \approx 1.24251$ , and  $w_3 \approx 2.27082$ . We note that  $A'(w)$  is positive at all these points. Therefore  $A'(w)$  has at most two roots, because otherwise there would be a point in which  $A'(w) \leq 0$  and  $A''(w) = 0$  which is impossible.

Note that  $A'(0) = 0$ , therefore  $A'(w)$  has at most one root in  $\mathcal{R}^+$ . Now we note that  $A(0^+) > 0$  because  $A'(0) = 0$  and  $A''(0) = 2(\alpha a^2(1 - \alpha) + \alpha^2 a^2 - 1) > 0$ . Also  $A(\tan(a)) > 0$ . Now if  $A(w) < 0$  for some  $0 < w < \tan(a)$ , then  $A(w)$  would have at least two roots in range  $(0, \tan(a))$  which results in  $A'(w)$  having two roots in  $\mathcal{R}^+$ . Since this is not true, we have  $A(w) \geq 0$  for every  $0 \leq w \leq \tan(a)$ .  $\square$

**Proof of Lemma 6.5:** Let  $X_1, \dots, X_n$  be random variables representing the items, and let  $Y_1, \dots, Y_m$  be iid random variables with distribution function  $F(x) = \prod_{i=1}^k F_i(x)$ . For the expectation of the maximum of these variables we have:

$$\begin{aligned}
\mathbb{E}[\max_{i=1}^k Y_i] &= \int_0^\infty Pr[\max_{i=1}^k Y_i \geq x] dx \\
&= \int_0^\infty \left(1 - \prod_{i=1}^m F(x)\right) dx \\
&= \int_0^\infty \left(1 - \prod_{i=1}^m \prod_{j=1}^k F_j(x)\right) dx \\
&= \int_0^\infty \left(1 - \prod_{i=1}^n F_i(x)\right) dx \\
&= \int_0^\infty Pr[\max_{i=1}^n X_i \geq x] dx \\
&= \mathbb{E}[\max_{i=1}^n X_i] .
\end{aligned} \tag{28}$$



This shows that the optimum solution is the same for both sets of items. Let  $\tau_Y$  and  $\tau_X$  be random variables that denotes the index of the picked items in  $Y_1, \dots, Y_m$  and  $X_1, \dots, X_n$  respectively. Theorem 6.5 states that there exists some  $s$  such for every  $m \geq s$ , we have  $E[Y_{\tau_Y}] \geq (1 - \epsilon/2)\alpha E[\max_{i=1}^m Y_i]$ . In the following we show that there exist some  $m_2$  such that for every  $m \geq m_2$ ,  $E[X_{\tau_X}] \geq (1 - \epsilon/2)E[Y_{\tau_Y}]$ . This proves the lemma for every  $m \geq m_\epsilon = \max\{s, m_2\}$ , i.e.

$$\mathbb{E}[X_{\tau_X}] \geq (1 - \epsilon/2)\mathbb{E}[Y_{\tau_Y}] \geq (1 - \epsilon/2)^2 \alpha \mathbb{E}[\max_{i=1}^m Y_i] \geq (1 - \epsilon)\alpha \mathbb{E}[\max_{i=1}^m Y_i] = (1 - \epsilon)\alpha \mathbb{E}[\max_{i=1}^n X_i] .$$

Since for every non-negative random variable  $Z$ ,  $\mathbb{E}[Z] = \int_0^\infty Pr[Z \geq z]dz$ , therefore, in order to show  $\mathbb{E}[X_{\tau_X}] \geq (1 - \epsilon/2)\mathbb{E}[Y_{\tau_Y}]$  we show  $Pr[X_{\tau_X} \geq (1 - \epsilon/2)Pr[Y_{\tau_Y}]]$  for every  $x \geq 0$ .

In the following, we use  $G_i(x)$  to denote  $1 - F_i(x)$ . For every non-negative  $x$  we have:

$$\begin{aligned} Pr[X_{\tau_X} \geq x] &= \sum_{i=1}^n Pr[X_{\tau_X} \geq x | \tau_X = i] Pr[\tau_X = i] \\ &= \sum_{i=1}^n \prod_{j=1}^{i-1} F_j(\theta_{\lceil j/k \rceil}) G_i(\max\{x, \theta_{\lceil i/k \rceil}\}) . \end{aligned}$$

By rewriting the above sum with respect to the  $m$  partitions we have:

$$\begin{aligned} Pr[X_{\tau_X} \geq x] &= \sum_{i=0}^{m-1} \sum_{j=1}^k \prod_{l=1}^{ik+j-1} F_l(\theta_{\lceil l/k \rceil}) G_{ik+j}(\max\{x, \theta_{i+1}\}) \\ &= \sum_{i=0}^{m-1} \sum_{j=1}^k \left( \prod_{l=1}^{ik} F_l(\theta_{\lceil l/k \rceil}) \prod_{p=1}^{j-1} F_{ik+p}(\theta_{i+1}) G_{ik+j}(\max\{x, \theta_{i+1}\}) \right) \\ &= \sum_{i=0}^{m-1} \left( \prod_{l=1}^{ik} F_l(\theta_{\lceil l/k \rceil}) \sum_{j=1}^k \prod_{p=1}^{j-1} F_{ik+p}(\theta_{i+1}) G_{ik+j}(\max\{x, \theta_{i+1}\}) \right) . \quad (29) \end{aligned}$$

Note that  $X_1, \dots, X_n$  are  $m$ -partitioned, hence for every partition  $0 \leq i < m$  and  $x \geq 0$  we have  $\prod_{l=ik+1}^{ik+k} F_l(x) = F(x)$ . Therefore  $\prod_{l=1}^{ik} F_l(\theta_{\lceil l/k \rceil}) = \prod_{l=1}^i F(\theta_l)$ .

By this replacement, Inequality (29) can be written as follows:

$$Pr[X_{\tau_X} \geq x] = \sum_{i=0}^{m-1} \left( \prod_{l=1}^i F(\theta_l) \sum_{j=1}^k \prod_{p=1}^{j-1} F_{ik+p}(\theta_{i+1}) G_{ik+j}(\max\{x, \theta_{i+1}\}) \right). \quad (30)$$

Moreover, for every  $0 \leq i < m$  and  $1 \leq j \leq k$  we have:

$$\begin{aligned} \prod_{p=1}^{j-1} F_{ik+p}(\theta_{i+1}) &\geq \prod_{p=1}^k F_{ik+p}(\theta_{i+1}) && \text{every } F_t(x) \text{ is at most } 1 \\ &= F(\theta_{i+1}) \\ &= 1 - G(\theta_{i+1}) \\ &\geq 1 - \frac{a \tan(a)}{m} && \text{inequality 26 for } h(s) = \cos(as) \\ &\geq 1 - \epsilon/2 && \text{for every } m \geq m_2 = \frac{2a \tan(a)}{\epsilon}. \end{aligned} \quad (31)$$

Inequality (31) shows that for a large enough  $m$ , the left hand side of the inequality becomes close enough to 1. By using this inequality in Inequality (30) we have:

$$\begin{aligned} Pr[X_{\tau_X} \geq x] &\geq \sum_{i=0}^{m-1} \left( \prod_{l=1}^i F(\theta_l) \sum_{j=1}^k (1 - \epsilon/2) G_{ik+j}(\max\{x, \theta_{i+1}\}) \right) \\ &= (1 - \epsilon/2) \sum_{i=0}^{m-1} \left( \prod_{l=1}^i F(\theta_l) \sum_{j=1}^k G_{ik+j}(\max\{x, \theta_{i+1}\}) \right). \end{aligned} \quad (32)$$

Let  $r = \max\{x, \theta_{j+1}\}$ . By multiplying every term in Inequality (32) by

$\prod_{p=1}^{j-1} F_{ik+p}(r)$ , which is less than or equal to 1, we have:

$$\begin{aligned}
Pr[X_{\tau_X} \geq x] &\geq (1 - \epsilon/2) \sum_{i=0}^{m-1} \left( \prod_{l=1}^i F(\theta_l) \sum_{j=1}^k G_{ik+j}(r) \right) \\
&\geq (1 - \epsilon/2) \sum_{i=0}^{m-1} \left( \prod_{l=1}^i F(\theta_l) \sum_{j=1}^k \prod_{p=1}^{j-1} F_{ik+p}(r) G_{ik+j}(r) \right) \\
&= (1 - \epsilon/2) \sum_{i=0}^{m-1} \left( \prod_{l=1}^i F(\theta_l) \sum_{j=1}^k \prod_{p=1}^{j-1} F_{ik+p}(r) (1 - F_{ik+j}(r)) \right) \\
&= (1 - \epsilon/2) \sum_{i=0}^{m-1} \left( \prod_{l=1}^i F(\theta_l) \sum_{j=1}^k \left( \prod_{p=1}^{j-1} F_{ik+p}(r) - \prod_{p=1}^j F_{ik+p}(r) \right) \right) \\
&= (1 - \epsilon/2) \sum_{i=0}^{m-1} \left( \prod_{l=1}^i F(\theta_l) \left( 1 - \prod_{p=1}^k F_{ik+p}(r) \right) \right) \quad \text{telescoping series} \\
&= (1 - \epsilon/2) \sum_{i=0}^{m-1} \left( \prod_{l=1}^i F(\theta_l) \left( 1 - F(r) \right) \right) \\
&= (1 - \epsilon/2) \sum_{i=0}^{m-1} \prod_{l=1}^i F(\theta_l) G(r) . \tag{33}
\end{aligned}$$

Note that  $Pr[Y_{\tau_Y} \geq x] = \sum_{i=0}^{m-1} \prod_{l=1}^i F(\theta_l) G(\max\{x, \theta_{i+1}\})$ . Using this in Inequality (33) results that  $Pr[X_{\tau_X} \geq x] \geq (1 - \epsilon/2) Pr[Y_{\tau_Y} \geq x]$ , hence the proof is complete.  $\square$

**Proof of Lemma 6.6:** Let  $\rho$  and  $\rho'$  be random variables that denote the index of the maximum with the smallest index amongst  $X_1, \dots, X_n$  and  $X_{S_1}, \dots, X_{S_r}$ , respectively. Then,

$$\begin{aligned}
\mathbb{E}[\max_{i \in S} X_i] &= \sum_{i \in S} \mathbb{E}[X_i | i = \rho'] Pr[i = \rho'] \\
&\geq \sum_{i \in S} \mathbb{E}[X_i | i = \rho] Pr[i = \rho] \\
&\geq \frac{p}{k} \sum_{i=1}^n \mathbb{E}[X_i | i = \rho] Pr[i = \rho] \\
&= \frac{p}{k} \mathbb{E}[\max_{1 \leq i \leq n} X_i] .
\end{aligned}$$

Therefore, the proof is complete. □

## D Missing Calculations in Example 7.1

It is a key observation that the probability that the item is sold before  $t$  can only go down by conditioning on  $T_i = t$ . The reason is that this effectively removes buyer  $i$  from the process. Furthermore, if buyer  $i$  arrives at time  $t$  then the event that the item is sold before time  $t$  cannot depend on  $v_i$ . This gives us

$$\begin{aligned} \mathbb{E}[u_i \mid T_i = t] &= \mathbb{E}[\mathbf{1}_{\text{item not sold before } t} \cdot (v_i - \alpha(t) \cdot b)^+ \mid T_i = t] \\ &= \Pr[\text{item not sold before } t \mid T_i = t] \cdot \mathbb{E}[(v_i - \alpha(t) \cdot b)^+] \\ &\geq \Pr[\text{item not sold before } t] \cdot \mathbb{E}[(v_i - \alpha(t) \cdot b)^+]. \end{aligned}$$

## E Extension of FTA's to Bipartite Matchings

Here we study the set of algorithms that use  $m$  fixed thresholds  $\tau_1, \dots, \tau_m$  for the items and have a recommendation strategy which at the arrival of every buyer offers her an item at its fixed price. In particular, when buyer  $i$  arrives the algorithm recommends an unsold item  $k$  to her and she accepts to buy it if  $v_{i,k} \geq \tau_k$ , i.e. her valuation for the item is greater than or equal to its price.

**Lemma .10.** *For every instance of matching prophet secretary there exists a sequence of fixed thresholds  $\tau_1, \dots, \tau_k$  and a randomized algorithm which is  $(1 - 1/e)$ -approximation in expectation.*

**Proof.** Our general approach is to extend the methods we have for single item FTA's to an algorithm for matchings. This generalization is similar to a reduction

from matchings to single items. However, there are some details that we have to consider. We first show a stronger claim than the statement of Theorem 7.4 which holds for a specific inputs class of prophet secretary. Then we propose a randomized algorithm which exploits the single item algorithm for that class in order to find such fixed thresholds that lead to a  $(1 - 1/e)$ -approximation algorithm for matchings.

We note that the analysis of Theorem 7.4 indicates we can find a single threshold such that every item will be seen with probability at least  $1 - 1/e$ . More precisely, the analysis shows  $\sum_{j=1}^n \frac{q(j)}{n} \geq 1 - 1/e$ , and consequently Inequality (7.13) shows the algorithm gets the same approximation factor from the utility of every buyer, i.e.  $\mathbb{E}[\text{Utility}] \geq (1 - 1/e) \sum_{i=1}^n \mathbb{E}[v_i \cdot \mathbf{1}_{v_i \geq \tau}]$ . Now, if an input instance guarantees  $\sum_{i=1}^n \text{Pr}[v_i > 0] \leq 1$  then we will have

$$\mathbb{E}[\text{Revenue}] = \left(1 - \frac{1}{e}\right) \tau \geq \left(1 - \frac{1}{e}\right) \tau \sum_{i=1}^n \text{Pr}[v_i > 0] \geq \left(1 - \frac{1}{e}\right) \sum_{i=1}^n \mathbb{E}[v_i \cdot \mathbf{1}_{v_i < \tau}] .$$

This results in

$$\begin{aligned} \mathbb{E}[\text{Alg}] &= \mathbb{E}[\text{Revenue}] + \mathbb{E}[\text{Utility}] \\ &\geq \left(1 - \frac{1}{e}\right) \sum_{i=1}^n (\mathbb{E}[v_i \cdot \mathbf{1}_{v_i < \tau}] + \mathbb{E}[v_i \cdot \mathbf{1}_{v_i \geq \tau}]) \\ &= \left(1 - \frac{1}{e}\right) \sum_{i=1}^n \mathbb{E}[v_i] . \end{aligned}$$

The following claim formally states the above result.

**Claim .3.** *If the input of prophet secretary guarantees  $\sum_{i=1}^n \text{Pr}[v_i > 0] \leq 1$ , then there exists an FTA such that*

$$\mathbb{E}[\text{Alg}] \geq \left(1 - \frac{1}{e}\right) \sum_{i=1}^n \mathbb{E}[v_i] .$$

Now we demonstrate how the matching problem reduces to the instances describe above. Let us assume we already know thresholds  $\tau_1, \dots, \tau_m$  for the  $m$  items. Upon the arrival of buyer  $i$  and realizing  $v_i$  we use the following algorithm to recommend an item  $k$  to buyer  $i$ . We first calculate probabilities  $p_1, \dots, p_m$  where  $p_k(v_i) := Pr_{\mathbf{v}_{-i}}[(i, k) \in \mathcal{M}(\mathbf{v}_i \cup \mathbf{v}_{-i})]$  and  $\mathcal{M}(B)$  is the maximum matching<sup>6</sup> of a bipartite graph  $B$ . These are in fact the probabilities of each of those edges belonging to the maximum matching. Then, by drawing a random number  $r \in [0, 1]$  we select a candidate item  $k$  if  $\sum_{l=1}^{k-1} p_l < r \leq \sum_{l=1}^k p_l$ . In this way, we dependently select a candidate such that every  $k$  becomes selected with probability  $p_k$ . Note that the algorithm might sometimes select none of the items, in which cases there will be no candidate. Finally we recommend item  $k$  to buyer  $i$  if the item is still unsold, and she buys it if  $v_{i,k} \geq \tau_k$ .

The above method for candidate selection has a close relationship with the optimum solution. To put it into perspective, let us define a new distribution  $\hat{D}_i : \mathbb{R}^m \rightarrow [0, 1]$  for every buyer  $i$ . This distribution is supposed to show the valuations of  $i$  on the items when they are selected as candidates. In other words, for every vector  $x = \langle x_1, \dots, x_m \rangle$  in which at most one of  $x_k$ 's is non-zero we have  $Pr_{\hat{v}_i \sim \hat{D}_i}[\hat{v}_i = x] := \mathbb{E}_{v_i \sim D_i}[\mathbf{1}_{v_{i,k}=x_k} \cdot \mathbf{1}_{k \text{ is a candidate}}]$ . Equivalently,  $\hat{D}_i$  can be interpreted as the distribution of the value of the edge incident to  $i$  in the maximum matching. This is true because we select a candidate with the probability that it belongs to the maximum matching.

---

<sup>6</sup>WLOG we can assume it is unique for every graph.

Therefore:

$$\mathbb{E}[OPT] = \mathbb{E}_{\mathbf{v} \sim D}[M(\mathbf{v})] = \sum_{i=1}^n \mathbb{E}_{\hat{\mathbf{v}}_i \sim \hat{D}_i} \left[ \sum_{k=1}^m \hat{v}_{i,k} \right]. \quad (34)$$

Now we reduce the problem to the single item case. By looking at a scenario of the problem from the viewpoint of item  $k$  we notice that the whole scenario and the algorithm run equivalent to the single item case. This item observes the buyers in a random order such that the valuation of buyer  $i$  comes from  $\hat{D}_i$ . These scenarios occur in parallel for all the items, because no two items are offered to buyers at the same time. In addition, every item  $k$  is offered to a buyer with an overall probability of

$$\sum_{i=1}^n Pr_{\hat{\mathbf{v}}_i \sim \hat{D}_i} [\hat{v}_{i,k} > 0] = \sum_{i=1}^n \mathbb{E}_{\mathbf{v}_i, \mathbf{v}_{-i}} [\mathbf{1}_{v_{i,k} > 0} \cdot \mathbf{1}_{(i,k) \in M(\mathbf{v}_i \cup \mathbf{v}_{-i})}] = Pr_{\mathbf{v} \sim D} [k \text{ is matched}] \leq 1.$$

Now we can use the result of Claim .3. The right hand side of Equality (34) can be written as  $\sum_{k=1}^m \sum_{i=1}^n \mathbb{E}_{\hat{\mathbf{v}}_i \sim \hat{D}_i} [\hat{v}_{i,k}]$ . The claim states that there exists a threshold  $\tau_k$  such that the FTA achieves at least  $(1 - 1/e) \sum_{i=1}^n \mathbb{E}_{\hat{\mathbf{v}}_i \sim \hat{D}_i} [\hat{\mathbf{v}}_{i,k}]$  for every item  $k$ . Therefore our algorithm is  $(1 - 1/e)$ -approximation for the matching of all items.

□

## F Correlated Setting

In this section, we study the  $k$ -server problem when the probability distributions are not independent. Recall that in the independent setting the sequence of requests is referred to by  $\rho = \langle r_1, \dots, r_t \rangle$ . In the correlated model we assume all different possibilities for  $\rho$  have been given in the form of a set  $\mathcal{R} = \{\rho_1, \dots, \rho_m\}$  of

$m$  sequences  $\rho_i = \langle r_{i,1}, \dots, r_{i,t} \rangle$ . Moreover, we assume the probability of each scenario  $\rho_i$  is denoted by  $p_i$  and given in advance. Given the list of different scenarios and probabilities, the goal is to design an online algorithm to serve each request  $r_{i,j}$  prior to arrival of the next request such that the overall movement of the servers is minimized.

We model this problem by an integer program. We first write an integer program and show that every solution of this program is uniquely mapped to a deterministic online algorithm for the problem. Moreover, every online algorithm can be mapped to a feasible solution of the program. More precisely, each solution of the program is equivalent to an online algorithm of the problem. Furthermore, we show how to derive an online algorithm from the solution of the integer program. These two imply that the optimal deterministic online algorithm can be obtained from the optimal solution of the program.

## G Program

To better convey the idea behind the integer program, we first introduce the tree  $T$  which is a trie containing all sequences  $\rho_1$  to  $\rho_m$ . Let us use  $w(v)$  to denote the path from the root to a node  $v$ . With these notations, a node  $v \in T$  represents a request which may occur conditioning all requests in  $w(v)$  occur beforehand. Besides, every leaf of  $T$  uniquely represents one of the  $\rho_i$ 's. Let us use  $l(v)$  to denote the set of those indices  $i$  for which  $\rho_i$  is a leaf of the subtree of  $v$ . At each step  $t$ , only those  $\rho_i$ 's can be a final option for  $R$  that  $\langle \rho_{i,1}, \dots, \rho_{i,t} \rangle = \langle r_1, \dots, r_t \rangle$ . Hence, a new



request  $r_t$  can be informative since we know that none of the  $\rho_i$ 's in  $l(r_{t-1}) \setminus l(r_t au)$  will occur anymore. For a node  $v$  we define  $Pr(v)$  as the probability of all requests in  $w(v)$  happening i.e.  $Pr(v) = \sum_{i \in l(v)} Pr(R = \rho_i)$ .

We extend the tree  $T$  by adding  $k - 1$  additional nodes. As shown in Figure 1, these nodes form a path leading to the root of  $T$ . These nodes plus the root represent the initial configuration of the  $k$  servers. Let us call these nodes the initial set  $I$ . Now we can show the movement of the servers in our metric space by means of  $k$  tokens in  $T$ . To do so, we begin with putting one token on each of the  $k$  nodes of  $I$ . Each token corresponds to one of the servers. After a server moves to serve a request  $r_t$ , we move its corresponding token to a node of  $T$  which represents the request  $r_t$ . Note that at this step, there is no discrimination between any of the sequences in  $l(r_t)$  in terms of occurrence. This causes a deterministic online algorithm  $A$  to serve the first  $|w(r_t)|$  requests of  $R$  in the same way if  $R$  is going to be one of  $\rho_i$ 's ( $i \in l(r_t)$ ). A result of this uniquely serving is that we can use some downward links on  $T$  in order to show how each request  $v$  gets served. In the next paragraphs we explain about these links and how we construct the integer program.

Let us use  $x_{u,v}$  to denote a link from a node  $u \in T$  to its descendant  $v$ .  $x_{u,v}$  is one if and only if  $A$  uses the same server to serve  $u$  and then  $v$  without using that server to serve any other request between  $u$  and  $v$ . This consecutive serving may occur with probability  $Pr(v) = Pr(u)Pr(v|u)$ . In this case, the algorithm moves a server from  $u$  to  $v$  and pays  $|u - v|$  as the distance cost between the two points of the metric space corresponding to  $u$  and  $v$ .

There are two conditions for these links that we must care about. First, since

each request  $v$  should be served with a server, at least one of the  $x_{u,v}$ 's should be one for all  $u$  in  $w(v)$ . Without loss of generality, we assume this is exactly one of them, i.e. there is no need to serve a request with more than one server. Second, after serving a request  $u$ , a server can go for serving at most one other request. That is, for each  $i \in l(u)$ , there should be at most one  $v \in \rho_i$  such that  $x_{u,v} = 1$ . This condition guarantees that in serving the sequence of requests  $R$ , a server which serves  $r_{t_1} \in R$  has always at most one other request  $r_{t_2} \in R$  as the next serving request.

The following integer program maintains both conditions for  $x_{u,v}$ 's and has the expected overall movement of all servers as the objective function:

$$\begin{aligned}
& \min. && \sum_{u,v \in T; u \in w(v)} Pr(v)|u - v|x_{u,v} \\
& \forall v \in T \setminus I && \sum_{u \in w(v)} x_{u,v} = 1. \\
& \forall u \in T, i \in l(u) && \sum_{v \in \rho_i} x_{u,v} \leq 1. \\
& \forall u, v \in T, u \in w(v) && x_{u,v} \in \{0, 1\}
\end{aligned}$$

Next, we can relax the constraints of the program to make it linear. Therefore, instead of assigning either  $\{0\}$  or  $\{1\}$ , to each  $x_{u,v}$  we let it be a real number between 0 and 1. Thus, the integer program turns to the following linear program with the

same objective function but more relaxed constraints.

$$\begin{aligned}
\min. \quad & \sum_{u,v \in T; u \in w(v)} Pr(v) |u - v| x_{u,v} \\
\forall v \in T \setminus I \quad & \sum_{u \in w(v)} x_{u,v} = 1. \\
\forall u \in T, i \in l(u) \quad & \sum_{v \in \rho_i} x_{u,v} \leq 1. \\
\forall u, v \in T, u \in w(v) \quad & x_{u,v} \leq 1 \\
\forall u, v \in T, u \in w(v) \quad & x_{u,v} \geq 0
\end{aligned}$$

Note that every feasible solution of the linear program is corresponding to a fractional solution of the problem. Since the optimal solution of the linear program can be found in polynomial time, using the rounding methods presented in Section 8.5 we obtain an optimal online algorithm for the line metric and a  $O(\log n)$  approximation algorithm for general metrics as stated in Theorem 8.6.

## H Experimental Results

The goal of this section is to make an evaluation of our method for the line on a real world data set. The line can be an appropriate model for a plenty of applications. For example, it could be sending road maintenance trucks to different points of a road or sending emergency vehicles to accident scenes along a highway. For this experiment, we take the case of car accidents.

**Data sets.** We use Road Safety Data<sup>7</sup> to find the distribution of the accidents

---

<sup>7</sup><https://data.gov.uk/dataset/road-accidents-safety-data/>

along the A1<sup>8</sup> road in Great Britain. In 2015, over 1600 accidents occurred on this highway, with an average of 140 accidents per month. We assume a point every 10 miles along the highway. That is 40 points in total. Then we build the distributions with respect to how the accidents are spread over the days of month. In this way, we achieve 30 distributions for 40 points along the line.

**Algorithms.** We compare the performance of our method to that of the optimum algorithm. To find the optimum solution we use backtracking. The running time of the algorithm is exponential to  $k$ . However, we use techniques such as *branch and bound* and *exponential dynamic programming* to get a fast implementation.

**Results.** We run different experiments with  $k$  from 2 to 11 on the line and distributions explained above. In previous sections we showed an upper bound of 3 for the approximation factor of our algorithm. Interestingly, in these experiments we can observe a better performance as shown by Figure 5. We compare the running time of the algorithms in Table 1. Note that the size of our LP our method solves does not vary by  $k$ . This is in fact the reason behind why its running time remains almost the same. In contrast, the running time of the optimum algorithm grows exponentially.

---

<sup>8</sup>[https://en.wikipedia.org/wiki/A1\\_road\\_\(Great\\_Britain\)](https://en.wikipedia.org/wiki/A1_road_(Great_Britain))

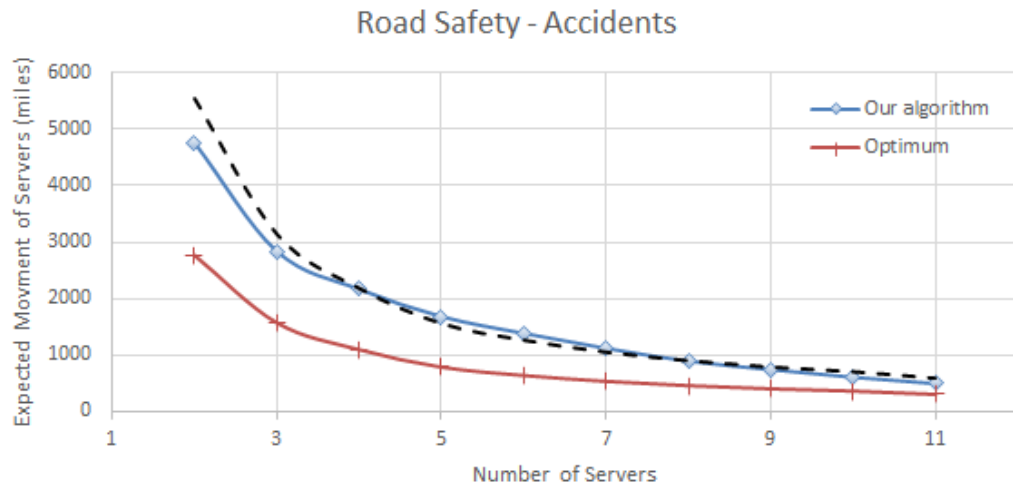


Figure 5: Performance of our algorithm compared to the optimum. The dashed curve indicates two times the optimum.

Number of Servers	2	3	4	5	6	7	8	9	10	11
Algorithm	6.5	7.6	6.7	7.1	7.5	8.3	8.5	8.4	9.3	8.2
Optimum	0.2	0.8	3.1	8.4	29.4	57.9	126.3	406.7	1477.1	6173.6

Table 1: The time performance of our algorithm vs. the optimum algorithm.

## Bibliography

- [AAA<sup>+</sup>09] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM Journal on Computing*, 39(2):361–370, 2009.
- [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.
- [AAB04] Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized steiner problem. *Theoretical Computer Science*, 324(2):313–324, 2004.
- [AAF<sup>+</sup>97] James Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM (JACM)*, 44(3):486–504, 1997.
- [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.
- [ACN00] Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234(1):203–218, 2000.
- [ADGP<sup>+</sup>10] Aris Anagnostopoulos, Clément Dombry, Nadine Guillotin-Plantard, Ioannis Kontoyiannis, and Eli Upfal. Stochastic analysis of the k-server problem on the circle. *DMTCS Proceedings*, (01):21–34, 2010.
- [AEE<sup>+</sup>17] Melika Abolhassani, Soheil Ehsani, Hossein Esfandiari, Mohammad-Taghi HajiAghayi, Robert Kleinberg, and Brendan Lucier. Beating  $1-1/e$  for ordered prophets. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 61–71. ACM, 2017.
- [AFG02] Susanne Albers, Lene M Favrhøldt, and Oliver Giel. On paging with locality of reference. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 258–267. ACM, 2002.
- [AHL<sup>+</sup>11] Saeed Alaei, Mohammad T Hajiaghayi, Vahid Liaghat, Dan Pei, and Barna Saha. Adcell: Ad allocation in cellular networks. In *Algorithms-ESA 2011*, pages 311–322. 2011.

- [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*, pages 18–35, 2012.
- [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. Springer, 2013.
- [AKR91] Ajit Agrawal, Philip Nathan Klein, and R Ravi. How tough is the minimum-degree steiner tree?: A new approximate min-max equality. *Technical Report CS-91-49, Brown University*, 1991.
- [Ala11] S. Alaei. Bayesian combinatorial auctions: Expanding single buyer mechanisms to many buyers. In *FOCS*. 2011.
- [AM06] Lawrence M Ausubel and Paul Milgrom. The lovely but lonely vickrey auction. *Combinatorial auctions*, 17:22–26, 2006.
- [And10] Matthew Andrews. Approximation algorithms for the edge-disjoint paths problem via raecke decompositions. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 277–286. IEEE, 2010.
- [AR01] Yossi Azar and Oded Regev. Strongly polynomial algorithms for the unsplittable flow problem. In *Integer Programming and Combinatorial Optimization*, pages 15–29. Springer, 2001.
- [BBK99] Avrim Blum, Carl Burch, and Adam Kalai. Finely-competitive paging. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, 1999.
- [BBMN11] Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. 2011.
- [BBN12] Nikhil Bansal, Niv Buchbinder, and Joseph Seffi Naor. A primal-dual randomized algorithm for weighted paging. *Journal of the ACM (JACM)*, 59(4):19, 2012.
- [BBS06] Douglas Bauer, Hajo Broersma, and Edward Schmeichel. Toughness in graphs—a survey. *Graphs and Combinatorics*, 22(1):1–35, 2006.
- [BC97] Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 344–353. ACM, 1997.

- [BCL00] Yair Bartal, Marek Chrobak, and Lawrence L Larmore. A randomized algorithm for two servers on the line. *Information and Computation*, 158(1):53–69, 2000.
- [Bec04] Luca Becchetti. Modeling locality: A probabilistic analysis of lru and fwf. In *Algorithms–ESA 2004*, pages 98–109. Springer, 2004.
- [BEY05] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.
- [BHZ13] MohammadHossein Bateni, Mohammad Taghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. *ACM Transactions on Algorithms*, 9(4):32, 2013.
- [BIK07] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443. Society for Industrial and Applied Mathematics, 2007.
- [BIRS95] Allan Borodin, Sandy Irani, Prabhakar Raghavan, and Baruch Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50(2):244–258, 1995.
- [BJN07] Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, pages 253–264. Springer, 2007.
- [BK04] Yair Bartal and Elias Koutsoupias. On the competitive ratio of the work function algorithm for the k-server problem. *Theoretical computer science*, 324(2):337–345, 2004.
- [BK10] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *European Symposium on Algorithms*, pages 170–181. Springer, 2010.
- [BKN09] Nikhil Bansal, Rohit Khandekar, and Viswanath Nagarajan. Additive guarantees for degree-bounded directed network design. *SIAM J. Comput.*, 39(4):1413–1431, 2009.
- [BN09] 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.
- [BS00] Alok Baveja and Aravind Srinivasan. Approximation algorithms for disjoint paths and related routing and packing problems. *Mathematics of Operations Research*, 25(2):255–280, 2000.



- [BV95] Fred Bauer and Anujan Varma. Degree-constrained multicasting in point-to-point networks. In *INFOCOM'95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People. Proceedings. IEEE*, pages 369–376. IEEE, 1995.
- [CEKP15] Deeparnab Chakrabarty, Alina Ene, Ravishankar Krishnaswamy, and Debmalya Panigrahi. Online buy-at-bulk network design. In *FOCS*, 2015.
- [CFH<sup>+</sup>17] José Correa, Patricio Foncea, Ruben Hoeksma, Tim Oosterwijk, and Tjark Vredeveld. Posted price mechanisms for a random stream of customers. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 169–186. ACM, 2017.
- [CG82] Paolo M Camerini and Giulia Galbiati. The bounded path tree problem. *SIAM Journal on Algebraic Discrete Methods*, 3(4):474–484, 1982.
- [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.
- [CHK07] Shuchi Chawla, Jason D. Hartline, and Robert D. Kleinberg. Algorithmic pricing via virtual valuations. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, pages 243–251, 2007.
- [CHMS10a] Shuchi Chawla, Jason Hartline, David Malec, and Balasubramanian Sivan. Multi-parameter mechanism design and sequential posted pricing. 2010.
- [CHMS10b] Shuchi Chawla, Jason D. Hartline, David L. Malec, and Balasubramanian Sivan. Multi-parameter mechanism design and sequential posted pricing. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 311–320, 2010.
- [Chu12] Julia Chuzhoy. Routing in undirected graphs with constant congestion. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 855–874. ACM, 2012.
- [Chv73] Vašek Chvátal. Tough graphs and hamiltonian circuits. *Discrete Mathematics*, 5(3):215–228, 1973.
- [CKPV91] Marek Chrobak, H Karloff, T Payne, and S Vishwnathan. New results on server problems. *SIAM Journal on Discrete Mathematics*, 4(2):172–181, 1991.

- [CKS09] Chandra Chekuri, Sanjeev Khanna, and F Bruce Shepherd. Edge-disjoint paths in planar graphs with constant congestion. *SIAM Journal on Computing*, 39(1):281–301, 2009.
- [CL] Julia Chuzhoy and Shi Li. A polylogarithmic approximation algorithm for edge-disjoint paths with congestion 2. In *FOCS 2012*.
- [CL91] Marek Chrobak and Lawrence L Larmore. An optimal on-line algorithm for k servers on trees. *SIAM Journal on Computing*, 20(1):144–148, 1991.
- [CL06] Randy Cogill and Sanjay Lall. On decentralized policies for the stochastic k-server problem. *arXiv preprint math/0605188*, 2006.
- [CRRT06] Kamalika Chaudhuri, Satish Rao, Samantha Riesenfeld, and Kunal Talwar. A push-relabel algorithm for approximating degree bounded msts. In *Proceedings of the 33rd international conference on Automata, Languages and Programming-Volume Part I*, pages 191–201. Springer-Verlag, 2006.
- [CRRT09] Kamalika Chaudhuri, Satish Rao, Samantha Riesenfeld, and Kunal Talwar. What would edmonds do? augmenting paths and witnesses for degree-bounded msts. *Algorithmica*, 55(1):157–189, 2009.
- [CS06] Joseph Cheriyan and Mohammad R Salavatipour. Hardness and approximation results for packing steiner trees. *Algorithmica*, 45(1):21–43, 2006.
- [DEH<sup>+</sup>15] Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Saeed Seddighin. Online survivable network design and prophets. 2015.
- [DEH<sup>+</sup>17a] Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, Vahid Liaghat, Harald Racke, and Saeed Seddighin. Online weighted degree-bounded steiner networks via novel online mixed packing/covering. *arXiv preprint arXiv:1704.05811*, 2017.
- [DEH<sup>+</sup>17b] Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Saeed Seddighin. Stochastic k-Server: How Should Uber Work? In *ICALP 2017*, 2017.
- [DEHL16] Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online degree-bounded steiner network design. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 164–175. Society for Industrial and Applied Mathematics, 2016.
- [Den83] Peter J Denning. The working set model for program behavior. *Communications of the ACM*, 26(1):43–48, 1983.

- [DFKL16] Paul Dütting, Michal Feldman, Thomas Kesselheim, and Brendan Lucier. Posted prices, smoothness, and combinatorial prophet inequalities. *arXiv preprint arXiv:1612.03161*, 2016.
- [DH09] Nikhil Devanur and Thomas Hayes. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *EC*, pages 71–78, 2009.
- [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 *Proceedings of the 12th ACM conference on Electronic commerce*, pages 29–38. ACM, 2011.
- [DK15] Paul Dütting and Robert Kleinberg. Polymatroid prophet inequalities. In *Algorithms-ESA 2015*, pages 437–449. Springer, 2015.
- [DMP13] Matt DeVos, Jessica McDonald, and Irene Pivotto. Packing steiner trees. *arXiv preprint arXiv:1307.7621*, 2013.
- [DNS10] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. *Mathematics of Operations Research*, 35(1):1–13, 2010.
- [Dyn63] Eugene B Dynkin. The optimum choice of the instant for stopping a markov process. In *Soviet Math. Dokl*, volume 4, 1963.
- [EHKS17] Soheil Ehsani, MohammadTaghi Hajiaghayi, Thomas Kesselheim, and Sahil Singla. Prophet secretary for combinatorial auctions and matroids. *arXiv preprint arXiv:1710.11213*, 2017.
- [EHLM15] Hossein Esfandiari, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Morteza Monemizadeh. Prophet secretary. In *Algorithms-ESA 2015*, pages 496–508. Springer, 2015.
- [EKM15] Hossein Esfandiari, Nitish Korula, and Vahab Mirrokni. Online allocation with traffic spikes: Mixing adversarial and stochastic models. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 169–186. ACM, 2015.
- [EV14] Alina Ene and Ali Vakilian. Improved approximation algorithms for degree-bounded network design problems with node connectivity requirements. *STOC*, 2014.
- [Fei06] Uriel Feige. On maximizing welfare when utility functions are subadditive. In *STOC*, pages 41–50, 2006.

- [FGL15] Michal Feldman, Nick Gravin, and Brendan Lucier. Combinatorial auctions via posted prices. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 123–135. SIAM, 2015.
- [FKL<sup>+</sup>91] Amos Fiat, Richard M Karp, Michael Luby, Lyle A McGeoch, Daniel D Sleator, and Neal E Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.
- [FM97] Amos Fiat and Manor Mendel. Truly online paging with locality of reference. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 326–335. IEEE, 1997.
- [FM03] Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM Journal on Computing*, 32(6):1403–1422, 2003.
- [FM<sup>+</sup>09] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S Muthukrishnan. Online stochastic matching: Beating  $1-1/e$ . In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 117–126. IEEE, 2009.
- [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.
- [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.
- [FR12] Takuro Fukunaga and R Ravi. Iterative rounding approximation algorithms for degree-bounded node-connectivity network design. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 263–272. IEEE, 2012.
- [Fra85] András Frank. Edge-disjoint paths in planar graphs. *Journal of Combinatorial Theory, Series B*, 39(2):164–178, 1985.
- [FRT03] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455. ACM, 2003.
- [FSZ15] Moran Feldman, Ola Svensson, and Rico Zenklusen. A simple  $o(\log(\text{rank}))$ -competitive algorithm for the matroid secretary problem. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1189–1201. SIAM, 2015.

- [FV06] Uriel Feige and Jan Vondrak. Approximation algorithms for allocation problems: Improving the factor of  $1-1/e$ . In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 667–676. IEEE, 2006.
- [FZ15] Moran Feldman and Rico Zenklusen. The submodular secretary problem goes linear. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 486–505. IEEE, 2015.
- [GGL<sup>+</sup>08] Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. Set covering with our eyes closed. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 347–356. IEEE, 2008.
- [GGLS08] Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *SODA*, pages 942–951. Society for Industrial and Applied Mathematics, 2008.
- [GGP<sup>+</sup>94] Michel X Goemans, Andrew V Goldberg, Serge A Plotkin, David B Shmoys, Eva Tardos, and David P Williamson. Improved approximation algorithms for network design problems. In *SODA*, volume 94, pages 223–232, 1994.
- [GHK<sup>+</sup>14] Oliver Göbel, Martin Hoefer, Thomas Kesselheim, Thomas Schleiden, and Berthold Vöcking. Online independent set beyond the worst-case: Secretaries, prophets, and periods. In *International Colloquium on Automata, Languages, and Programming*, pages 508–519. Springer, 2014.
- [GK11] Anupam Gupta and Jochen Könemann. Approximation algorithms for network design: A survey. *Surveys in Operations Research and Management Science*, 16(1):3–20, 2011.
- [GKR12] Anupam Gupta, Ravishankar Krishnaswamy, and R Ravi. Online and stochastic survivable network design. *SIAM Journal on Computing*, 41(6):1649–1672, 2012.
- [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*, pages 982–991, 2008.
- [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.

- [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.
- [GRST10] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: offline and secretary algorithms. available at <http://www.cs.cmu.edu/alroth/submodularesecretaries.html>, 2010.
- [GS17] Guru Prashanth Guruganesh and Sahil Singla. Online matroid intersection: Beating half for random arrival. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 241–253. Springer, 2017.
- [GW95] Michel X Goemans and David P Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.
- [HGM03] Yongqiang Huang and Hector Garcia-Molina. Publish/subscribe tree construction in wireless ad-hoc networks. In *Mobile Data Management*, pages 122–140, 2003.
- [HK82] Theodore P Hill and Robert P Kertz. Comparisons of stop rule and supremum expectations of iid random variables. *The Annals of Probability*, pages 336–345, 1982.
- [HKLR05] MohammadTaghi Hajiaghayi, Jeong Han Kim, Tom Leighton, and Harald Räcke. Oblivious routing in directed graphs with random demands. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 193–201. ACM, 2005.
- [HKP04] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C. Parkes. Adaptive limited-supply online auctions. In *EC*, pages 71–80, 2004.
- [HKS07] Mohammad T. Hajiaghayi, Robert Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *AAAI*, pages 58–65, 2007.
- [HLP13] Mohammad Taghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Online node-weighted steiner forest and extensions via disk paintings. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 558–567, 2013.
- [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.

- [HMZ11] Bernhard Haeupler, Vahab S Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In *International Workshop on Internet and Network Economics*, pages 170–181. Springer, 2011.
- [HRW92] Frank K Hwang, Dana S Richards, and Pawel Winter. *The Steiner tree problem*, volume 53. Elsevier, 1992.
- [IKP96] Sandy Irani, Anna R Karlin, and Steven Phillips. Strongly competitive algorithms for paging with locality of reference. *SIAM Journal on Computing*, 25(3):477–497, 1996.
- [IW91] Makoto Imase and Bernard M Waxman. Dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991.
- [Jai01] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.
- [JL13] Patrick Jaillet and Xin Lu. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 39(3):624–646, 2013.
- [JMS03] Kamal Jain, Mohammad Mahdian, and Mohammad R Salavatipour. Packing steiner trees. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 266–274. Society for Industrial and Applied Mathematics, 2003.
- [Ken85] DP Kennedy. Optimal stopping of independent random variables and maximizing prophets. *The Annals of Probability*, 13(2):566–571, 1985.
- [Ken87] DP Kennedy. Prophet-type inequalities for multi-choice optimal stopping. *Stochastic Processes and their Applications*, 24(1):77–88, 1987.
- [Ker86] Robert P Kertz. Comparison of optimal value and constrained maxima expectations for independent random variables. *Advances in applied probability*, pages 311–340, 1986.
- [KK11] Ken-ichi Kawarabayashi and Yusuke Kobayashi. Breaking  $o(n^{1/2})$ -approximation algorithms for the edge-disjoint paths problem with congestion two. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 81–88. ACM, 2011.
- [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.
- [KKRR04] Philip N Klein, Radha Krishnan, Balaji Raghavachari, and R Ravi. Approximation algorithms for finding low-degree subgraphs. *Networks*, 44(3):203–215, 2004.

- [Kle05] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA*, pages 630–631, 2005.
- [KMT11] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 587–596, 2011.
- [KMZ15] Nitish Korula, Vahab Mirrokni, and Morteza Zadimoghaddam. Online submodular welfare maximization: Greedy beats  $1/2$  in random order. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 889–898. ACM, 2015.
- [KP95] Elias Koutsoupias and Christos H Papadimitriou. On the  $k$ -server conjecture. *Journal of the ACM (JACM)*, 42(5):971–983, 1995.
- [KP09] Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *International Colloquium on Automata, Languages and Programming*, pages 508–520. Springer, 2009.
- [KPR00] Anna R Karlin, Steven J Phillips, and Prabhakar Raghavan. Markov paging. *SIAM Journal on Computing*, 30(3):906–922, 2000.
- [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.
- [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.
- [Kri03] Matthias Kriesell. Edge-disjoint trees containing some given vertices in a graph. *Journal of Combinatorial Theory, Series B*, 88(1):53–65, 2003.
- [KRR94] Howard Karloff, Yuval Rabani, and Yiftach Ravid. Lower bounds for randomized  $k$ -server and motion-planning algorithms. *SIAM Journal on Computing*, 23(2):293–312, 1994.
- [KRTV13] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In *European Symposium on Algorithms*, pages 589–600. Springer, 2013.
- [KS77] Ulrich Krengel and Louis Sucheston. Semiamarts and finite values. *Bull. Am. Math. Soc*, 1977.



- [KS78] U. Krengel and L Sucheston. On semiamarts, amarts, and processes with finite value. In *In Kuelbs, J., ed., Probability on Banach Spaces*. 1978.
- [KS04] Stavros G Kolliopoulos and Clifford Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99(1):63–87, 2004.
- [KVV90] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.
- [KW12] Robert Kleinberg and Seth Matthew Weinberg. Matroid prophet inequalities. In *STOC*. ACM, 2012.
- [KWW16] Robert Kleinberg, Bo Waggoner, and E. Glen Weyl. Descending price optimally coordinates search. In *Proceedings of the 17th ACM Conference on Economics and Computation*, pages 23–24, 2016.
- [Lac14] Oded Lachish.  $O(\log \log \text{rank})$  competitive ratio for the matroid secretary problem. In *Proceedings of the Fifty-Fifth Annual IEEE Symposium on Foundations of Computer Science*, pages 326–335, 2014.
- [Lau04] Lap Chi Lau. An approximate max-steiner-tree-packing min-steiner-cut theorem. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 61–70. IEEE, 2004.
- [LB10] Brendan Lucier and Allan Borodin. Price of anarchy for greedy auctions. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 537–553. Society for Industrial and Applied Mathematics, 2010.
- [LNSS09] Lap Chi Lau, Joseph Naor, Mohammad R Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. *SIAM Journal on Computing*, 39(3):1062–1087, 2009.
- [LRS11] Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.
- [LS13] Lap Chi Lau and Mohit Singh. Additive approximation for bounded degree survivable network design. *SIAM Journal on Computing*, 42(6):2217–2242, 2013.
- [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.

- [Meh12] Aranyak Mehta. Online matching and ad allocation. *Theoretical Computer Science*, 8(4):265–368, 2012.
- [Mey01] Adam Meyerson. Online facility location. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 426–431. IEEE, 2001.
- [MGZ12] 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. SIAM, 2012.
- [MMS90] Mark S Manasse, Lyle A McGeoch, and Daniel D Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.
- [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.
- [MRS<sup>+</sup>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.
- [MS91] Lyle A McGeoch and Daniel D Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(1-6):816–825, 1991.
- [MSVV07] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22, 2007.
- [MY11] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 597–606, 2011.
- [NPS11] Joseph Naor, Debmalya Panigrahi, and Mohit Singh. Online node-weighted steiner tree and related problems. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 210–219, 2011.
- [Nut12] Zeev Nutov. Degree-constrained node-connectivity. In *LATIN 2012: Theoretical Informatics*, pages 582–593. 2012.
- [OP05] Carlos AS Oliveira and Panos M Pardalos. A survey of combinatorial optimization problems in multicast routing. *Computers & Operations Research*, 32(8):1953–1981, 2005.

- [PR00] Louis Petingi and J Rodriguez. Bounds on the maximum number of edge-disjoint steiner trees of a graph. *Congressus Numerantium*, pages 43–52, 2000.
- [PS97] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the chernoff–hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.
- [PS06] Konstantinos Panagiotou and Alexander Souza. On adequate performance measures for paging. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 487–496. ACM, 2006.
- [PY82] Christos H Papadimitriou and Mihalis Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM*, 29(2):285–309, 1982.
- [QW11] Jiawei Qian and David P Williamson. An  $o(\log n)$ -competitive algorithm for online constrained forest problems. In *Automata, Languages and Programming*, pages 37–48. 2011.
- [Rag96] Balaji Raghavachari. Algorithms for finding low degree structures. In *Approximation algorithms for NP-hard problems*, pages 266–295. PWS Publishing Co., 1996.
- [RK93] R Ravi and Philip Klein. When cycles collapse: A general approximation technique for constrained two-connectivity problems. In *3rd Conference on Integer Programming and Combinatorial Optimization*, pages 39–56, 1993.
- [RMR<sup>+</sup>01] R Ravi, Madhav V Marathe, SS Ravi, Daniel J Rosenkrantz, and Harry B Hunt III. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31(1):58–78, 2001.
- [Rot07] Michael H Rothkopf. Thirteen reasons why the vickrey-clarke-groves process is not practical. *Operations Research*, 55(2):191–197, 2007.
- [RS06] R Ravi and Mohit Singh. Delegate and conquer: An lp-based approximation algorithm for minimum degree msts. In *Automata, Languages and Programming*, pages 169–180. Springer, 2006.
- [RS17] Aviad Rubinfeld and Sahil Singla. Combinatorial prophet inequalities. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1671–1687. SIAM, 2017.
- [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.

- [RT87] Prabhakar Raghavan and Clark D Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.
- [Rub16] Aviad Rubinfeld. Beyond matroids: Secretary problem and prophet inequality with general constraints. *arXiv preprint arXiv:1604.00357*, 2016.
- [SCS11] L c S guin-Charbonneau and F Bruce Shepherd. Maximum edge-disjoint paths in planar graphs with congestion 2. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 200–209. IEEE, 2011.
- [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.
- [ST85] Daniel D Sleator and Robert E Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.
- [T r05] Duru T rkoglu. *The k-Server Problem and Fractional Analysis*. PhD thesis, Masters Thesis, The University of Chicago, 2005. <http://people.cs.uchicago.edu/~duru/papers/masters.pdf>, 2005.
- [Vo 92] Stefan Vo . Problems with generalized steiner problems. *Algorithmica*, 7(1):333–335, 1992.
- [Wei79] Martin L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–654, 1979.
- [WGMV95] David P Williamson, Michel X Goemans, Milena Mihail, and Vijay V Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. *Combinatorica*, 15(3):435–454, 1995.
- [Win89] Sein Win. On a connection between the existence of  $k$ -trees and the toughness of a graph. *Graphs and Combinatorics*, 5(1):201–205, 1989.
- [Yan11] Qiqi Yan. Mechanism design via correlation gap. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 710–719, 2011.
- [You98] Neal E Young. Bounding the diffuse adversary. In *SODA*, volume 98, pages 420–425, 1998.