ABSTRACT

Title of dissertation: ASSIGNMENT PROBLEMS
AND THEIR APPLICATIONS IN ECONOMICS

Melika Abolhassani, Doctor of Philosophy, 2016

Dissertation directed by: Professor Mohammad Taghi Hajiaghayi
Department of Computer Science

Four assignment problems are introduced in this thesis, and they are approached based on the context they are presented in. The underlying graphs of the assignment problems in this thesis are in most cases bipartite graphs with two sets of vertices corresponding to the agents and the resources. An edge might show the interest of an agent in a resource or willingness of a manufacturer to produce the corresponding product of a market, to name a few examples.

The first problem studied in this thesis is a two-stage stochastic matching problem in both online and offline versions. In this work, which is presented in Chapter 2 of this thesis, a coordinator tries to benefit by having access to the statistics of the future price discounts which can be completely unpredictable for individual customers. In our model, individual risk-averse customers want to book hotel rooms for their future vacation; however, they are unwilling to leave booking to the last minute which might result in huge savings for them since they have to take the risk of all the hotel rooms being sold out. Instead of taking this risk, individual customers make contracts with a coordinator who can spread the risk over many

such cases and also has more information on the probability distribution of the future prices. In the first stage, the coordinator agrees to serve some buyers, and then in the second stage, once the final prices have been revealed, he books rooms for them just as he promised. An agreement between the coordinator and each buyer consists of a set of acceptable hotels for the customer and a single price. Two models for this problem are investigated. In the first model, the details of the agreements are proposed by the buyer, and we propose a bicriteria-style approximation algorithm that gives a constant-factor approximation to the objective function by allowing a bounded fraction of our hotel bookings to overlap. In the second model, the details of the agreements are proposed by the coordinator, and we show the prices yielding the optimal profit up to a small additive loss can be found by a polynomial time algorithm.

In the third chapter of this thesis, two versions of the online matching problem are analyzed with a similar technique. Online matching problems have been studied by many researchers recently due to their direct application in online advertisement systems such as Google Adwords. In the online bipartite matching problem, the vertices of one side are known in advance; however, the vertices of the other side arrive one by one, and reveal their adjacent vertices on the offline side only upon arrival. Each vertex can only be matched to an unmatched vertex once it arrives and we cannot match or rematch the online vertex in the future. In the *online matching problem with free disposal*, we have the option to rematch an already matched offline vertex only if we eliminate its previous online match from the graph. The goal is to maximize the expected size of the matching. We propose a randomized algorithm that achieves a ratio greater than 0.5 if the online nodes have bounded degree. The other problem studied in the third chapter is the *edge-weighted oblivious matching* in which the weights of all the edges in the underlying graph are known but existence of each edge is only revealed upon probing that edge. The weighted version of the problem has applications in pay-per-click online advertisements, in which the revenue for a click on a particular ad is known, but it is unknown whether the user will actually click on that ad. Using a similar technique, we develop an algorithm with approximation factor greater than 0.5 for this problem too.

In Chapter 4, a generalized version of the Cournot Competition (a foundational model in economics) is studied. In the traditional version, firms are competing in a single market with one heterogeneous good, and their strategy is the quantity of good they produce. The price of the good is an inverse function of the total quantity produced in the market, and the cost of production for each firm in each market increases with the quantity it produces. We study Cournot Competition on a bipartite network of firms and markets. The edges in this network demonstrate access of a firm to a market. The price of the good in each market is again an inverse function of the quantity of the good produced by the firms, and the cost of production for each firm is a function of its production in different markets. Our goal is to give polynomial time algorithms to find the quantity produced by firms in each market at the equilibrium for generalized cost and price functions.

The final chapter of this thesis is on analyzing a problem faced by online marketplaces such as Amazon and ebay which deal with huge datasets registering transaction of merchandises between many buyers and sellers. As the size of datasets grow, it is important that the algorithms become more selective in the amount of data they store. Our goal is to develop pricing algorithms for social welfare (or revenue) maximization that are appropriate for use with the massive datasets in these networks. We specially focus on the streaming setting, the common model for big data analysis. Furthermore, we include hardness results (lower bounds) on the minimum amount of memory needed to calculate the exact prices and also present algorithms which are more space efficient than the given lower bounds but *approximate* the optimum prices for the goods besides the revenue or the social welfare of the mechanism.

ASSIGNMENT PROBLEMS AND THEIR APPLICATIONS IN ECONOMICS

by

Melika Abolhassani

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
2016

Acknowledgments

I owe my gratitude to my advisor, Prof. Hajiaghayi, who provided me with the opportunity of working with him, and who has been very supportive since day one and throughout all the years of my PhD program. It has been an honor to work with him during these years, and I am very thankful for his continuous help, motivation and support.

I would like to thank my mentor Dr. Kamal Jain who has been a great source of motivation and inspiration for me during the short period of time I had the pleasure of working with him.

I am very grateful for the opportunity of working with amazing researchers such as Prof. Aravind Srinivasan, Prof. Robert Kleinberg, Prof. Brendan Lucier, Mohammad Hossein Bateni, and my dear friend Hamid Mahini.

Special thanks to my defense committee, Prof. Lawrence Ausubel, Prof. William Gasarch, Prof. Raghu Raghavan and Prof. Ashok Agrawala for their very insightful comments.

During my PhD program, I have had the chance to both work and become friends with an amazing and talented research group whom I have learnt from everyday. I would like to thank my friends Vahid Liaghat, Hossein Esfandiari, David Malec, Soheil Ehsani, Anshul Sawant, Sina Dehghani, Saeed Seddighin, Hadi Yami, Reza Khani. I would also like to thank Hubert Chan, Fei Chen, and X.Wu whom I have had the pleasure to work with but not the chance to meet them in person.

I am very blessed to have found an unbelievably kind and caring group of friends during this time who have helped and supported me through this journey. I would like to express my gratitude to my dearest friends Kiana Roshan Zamir, Ali Shafahi, Ladan Rabiee Kenari, Pouya Samangouei, Soudeh Montazeri, Niloufar Shadab and Faeze Dorri.

Words cannot describe how grateful I am to my beloved parents and brothers for everything they have done for me. Without a doubt, I would not be here without such an amazing family.

Table of Contents

List of Figures vi

1 Introduction 1
- 1.1 Overview . . . 1
- 1.2 Selling Tomorrow's Bargains Today . . . 4
- 1.3 Edge-Weighted Oblivious and Online Matching . . . 5
- 1.4 Network Cournot Competition . . . 8
- 1.5 Market Pricing for Data Streams . . . 9

2 Selling Tomorrow's Bargains Today 12
- 2.1 Introduction . . . 12
  - 2.1.1 Buyer-selected Packs . . . 14
  - 2.1.2 Coordinator-selected Packs . . . 16
  - 2.1.3 Examples . . . 16
  - 2.1.4 Our results . . . 19
  - 2.1.5 Related work . . . 21
- 2.2 Profit Maximization in Buyer-selected Packs Model . . . 23
- 2.3 Regret Minimization in Buyer-selected Packs Model . . . 28
  - 2.3.1 Approximate-optimality via sampling . . . 28
  - 2.3.2 Approximating Regret Function After Sampling . . . 30
- 2.4 Coordinator-selected Packs Model . . . 39
- 2.5 Conclusion . . . 53

3 Edge-Weighted Oblivious and Online Matching 55
- 3.1 Introduction . . . 55
  - 3.1.1 Our Results . . . 57
  - 3.1.2 Related Work . . . 62
- 3.2 Preliminaries: Matching Coverage as an Analysis Technique . . . 66
- 3.3 Edge-Weighted Oblivious Matching Problem with a Bounded Number of Distinct Weights . . . 69
  - 3.3.1 Partitioning $\binom{V}{2}$ into Batches . . . 70
  - 3.3.2 Edge-Weighted OM: Running Unweighted $\mathcal{A}^{un}$ on Each Batch 71

3.4 Edge-weighted Online Bipartite Matching with Free Disposal and Bounded Online Degree 83
3.5 The Hardness of Edge-Weighted Online Bipartite Matching without Free Disposal 96
3.6 Conclusion 98

4 Network Cournot Competition 99
4.1 Introduction 99
4.1.1 Example 101
4.1.2 Related Work 104
4.1.3 Results and techniques 106
4.1.3.1 Linear Inverse Demand Functions 108
4.1.3.2 The general case 110
4.1.3.3 Cournot oligopoly 111
4.2 Notations 113
4.3 Cournot competition and potential games 114
4.4 Finding equilibrium for cournot game with general cost and inverse demand functions 120
4.4.1 Marginal profit function 122
4.4.2 Non-linear complementarity problem 123
4.4.3 Designing a polynomial-time algorithm 125
4.4.4 Price Functions for Monotone Marginal Revenue Function . . 133
4.5 Algorithm for Cournot Oligopoly 136
4.5.1 Polynomial time algorithm 137
4.5.2 Proof of correctness 140
4.6 Conclusion 146

5 Market Pricing for Data Streams 147
5.1 Introduction 147
5.1.1 Our Results 149
5.1.2 Related Work 153
5.2 Pricing problem: Maximizing Social Welfare 155
5.3 Pricing problem: Maximizing Revenue 159
5.4 Improving Space Efficiency While Approximating Social Welfare . . . 162
5.5 Improving Space Efficiency While Approximating Optimum Revenue 169
5.6 A Hardness Proof for Social Welfare Maximization Problem 174
5.7 A Hardness Proof for Revenue Maximization Problem 178
5.8 Conclusion 181

Bibliography 182

List of Figures

2.1.1 Buyer-selected packs example graph 17
2.3.2 The buyer-selected packs integer program 31
2.3.3 Illustration of the instance of the problem with high integrality gap. 33

3.4.1 The graph corresponding to Example 3. 95

4.1.1 The figure showing scenarios in Example 4.1.1 102

5.6.1 An example for Theorem 5.11 176

Chapter 1: Introduction

1.1 Overview

The Assignment Problem is a foundational problem in Theoretical Computer Science, Optimization and Operations Research. In its most simple form, there is a set of people on one side and a set of tasks on the other side. Each person can potentially do a subset of these tasks. Each person can only be assigned one task and each task can be assigned to only one person. The goal is to get as many tasks done as possible. The underlying graph of this problem is a bipartite graph and the assignment which matches maximum number of tasks to people is called a maximum bipartite matching. A more general form of the maximum bipartite matching problem is the weighted matching problem in which edges between people and tasks are weighted and we are interested in finding the matching with the maximum weight. The matching problem is a special case of linear programming and the min cost flow problem.

Recent technologies such as online marketing, network allocations and social networks have raised up many new assignment problems with more complicated structures. Many different versions of the bipartite matching problem have been considered by researchers to solve fundamental problems in Theoretical Computer Science, Economics and Business. Online matching introduced by Karp, Vazirani and Vazirani in 1990, has gained a lot of attention due to its direct relation with internet ad allocations and search-based advertising. In the online matching problem, a set of the vertices L of the left side of a bipartite graph $G(L, R, E)$ is known to us. However, the vertices on the right side, denoted by R, arrive one vertex at a time and the neighbors of the incoming vertices are revealed upon their arrival. Each arriving vertex can be matched at the time of arrival to an unmatched neighbor in the left side, and if it is not matched immediately, it cannot be matched in the future. The goal is to maximize the number of the matched vertices. The online matching problem can be generalized to the weighted online matching problem in the same way as the matching problem is generalized to the weighted matching problem. In search-based advertising, the advertisers are known in advance, and upon each word's search, they reveal how much they are willing to bid if their ad is shown to the user who searched the word. The advertisers can be viewed as the set of offline vertices, the searched keywords can be considered as the set of online vertices, and the advertisers bids for keywords are the weights on the edges. The most well-known algorithm for the online matching problem is called Ranking which is also introduced by Karp, Vazirani and Vazirani. In this algorithm, the offline vertices are permuted according to a random permutation and the online incoming vertices are matched to the first available neighbor in the order of this permutation. Online matching problem has many different aspects and variations itself. One example is when multiple ad slots can be assigned to one advertiser but the advertisers' budgets are limited. The advertisers might also have contracts for the maximum number of search keywords that gets matched to them. The problem has also been considered widely in the online mechanism design. Another variant of the online bipartite matching is the online stochastic matching problem which was introduced in a paper published in FOCS 2012 by Mehta and Panigrahi. In this specific version, each edge has a probability of becoming a successful match. That is if an online vertex is matched to an offline vertex by an algorithm, the matching is successful with a certain probability shown on the edge. They consider the problem when the probabilities are equal and derive different algorithms and competitive ratios depending on whether this probability is vanishingly small or not. The algorithms and their analysis also depend strongly on the assumptions about the available information on the requests arriving online. The four models studied so far about these assumptions are called *Adversarial*, *Random Order Arrival*, *Unknown Distribution* and *Known Distribution*.

The assignment problems have also been studied by many researchers in the streaming setting which is the common model for working with large datasets. Over the past decade, massive datasets from huge and growing graphs such as the social networks, web pages and their links and the citations of academic work, have required a change in the nature of some algorithms. Algorithms should be more time and specially space efficient as the size of the datasets increase. In the streaming setting, a stream of data (for example set of edges of the bipartite graph) arrives sequentially and must be analyzed by an algorithm with limited memory. These streams can only be read once (or a limited number of times), and so the algorithms must be selective in the data they choose to store. An application of these algorithmic techniques is in the market design problems. For instance, one might imagine that there is a set of items (e.g., goods for sale) and a set of potential buyers (e.g., individual consumers) to which they should be matched.

In each of the following four chapters, an assignment problem is introduced along with its application in economics, and our algorithms for achieving specific goals such as revenue maximization, or to calculate properties of the network such as its equilibria are presented along with the proof of their correctness and efficiency. Brief explanations of these problems are given below.

1.2 Selling Tomorrow's Bargains Today

In Chapter 2, we consider a two stage stochastic matching problem in both online and offline cases. In this work, we study how a reseller of hotel rooms can profit from having access to forecasts of the future price discounts. The price to book a hotel can change dramatically as the date of the booking approaches. An individual looking for a hotel room who is aware of this fact might choose to gamble, delaying booking until the last minute in the hope of getting heavy discounts. While this gamble can yield large savings, it also carries large risk. Consider an online hotel booking business which offers customers a compromise by assuming the risk of last minute bookings but sharing the savings via discounted rates. Here we explore how this company could best profit from forecasts of such future discounts.

In this two-stage optimization problem, the coordinator (the online hotel booking business), first agrees to serve some buyers, and then later books rooms for them once the final prices have been revealed. Agreements with buyers consist of a set of acceptable hotels and a price. We investigate two models, one where the details of the agreements are proposed by the coordinator, and one where the details of the agreements are proposed by the buyer. In the former, we show that even when buyers arrive online, we can find prices yielding the optimal profit up to a small additive loss. In the latter case, we propose a bicriteria-style approximation algorithm that gives a constant-factor approximation to the minimal regret by allowing a fraction of our hotel bookings to overlap. Importantly, however, we show that our algorithm provides a strong, uniform bound on the amount of the reservation overlap per hotel rooms. The paper of this work, "Selling Tomorrow's Bargains Today", has appeared on *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015).*

1.3 Edge-Weighted Oblivious and Online Matching

In Chapter 3, we consider the weighted versions of the *oblivious matching problem* along with the *online bipartite matching with free disposal.* In the oblivious matching problem, an adversary decides on a simple undirected graph $G = (V, E)$ with non-negative weights on the edges. The adversary then reveals the nodes V and the weights of all pairs to the (randomized) algorithm. However, the edges E are kept secret. The output of the algorithm is a permutation of all $\binom{V}{2}$ pairs of nodes. At the end of the algorithm, the pairs of nodes are probed one by one in the order specified by the output to form a matching greedily. Upon probing pair (u, v) if both nodes are currently unmatched and the edge (u, v) is in E, the two nodes will be matched to each other; otherwise, we simply continue to the next pair. The goal is to maximize the performance ratio of the (expected) sum of the weights of the edges in the matching produced by the algorithm to that of a maximum weight matching in G.

We prove the first non-trivial performance ratios strictly above 0.5 for the weighted versions of the oblivious matching problem. Even for the unweighted version, achieving a ratio above 0.5 is quite challenging. Since Aronson, Dyer, Frieze, and Suen first proved a non-trivial ratio above 0.5 in the mid-1990s, during the next twenty years several attempts have been made to improve this ratio, until Chan, Chen, Wu and Zhao successfully achieved a significant ratio of 0.523 very recently (SODA 2014). This work is the first in the literature that considers the node-weighted and the edge-weighted versions of the problem in arbitrary graphs (as opposed to the bipartite graphs). For arbitrary node weights, we prove that a weighted version of the *Ranking* algorithm has ratio strictly above 0.5. This property allows us to form LP constraints for both the node-weighted and the unweighted oblivious matching problems. Consequently, we prove that the ratio for the node-weighted case is at least 0.501512, and improve the ratio for the unweighted case to 0.526823 (from the previous best 0.523166 in SODA 2014). For a bounded number of distinct edge weights, we show that a ratio strictly above 0.5 can be achieved by partitioning edges carefully according to the weights, and running the (unweighted) Ranking algorithm on each part. Our analysis is based on a new primal-dual framework called *the matching coverage*, in which the dual feasibility is bypassed, and instead, only dual constraints corresponding to the edges in an optimal matching are satisfied.

The *online bipartite matching problem with free disposal* is the same as the weighted online matching problem except that a matched offline vertex can be rematched in case we decide to discard the previously matched edge to it. More formally, in this problem an adversary fixes an edge-weighted bipartite graph $G(U \cup V, E)$ between a set U of online nodes and a set V of offline nodes, and determines the arrival order of the online nodes. Upon arrival of an online node u, all the weights of the edges between u and the offline nodes in V are revealed to the algorithm. However, the algorithm matches u to one of the offline nodes $v \in V$, even if this node is already matched to a previous online node. That is the algorithm is allowed to dispose of the previously matched edge to v, and match u to v instead. The goal is to maximize the performance ratio, which is the (expected) sum of the weights of the edges in the final matching to that of a maximum weight matching in hindsight. It has been proven that a greedy algorithm can achieve ratio 0.5, however, no approximation algorithm has been found to achieve a better ratio. We propose a randomized algorithm that achieves a ratio strictly greater than 0.5 for the case in which each online node has bounded degree using the matching coverage framework. Using the same framework, we also design and analyze an algorithm for the edge-weighted online bipartite matching problem with free disposal. We prove that for the case of bounded online degrees, the ratio is strictly above 0.5. Our paper "Beating Ratio 0.5 for Weighted Oblivious Matching Problems" has been accepted for publication in *European Symposium on Algorithms (ESA 2016).*

1.4 Network Cournot Competition

In Chapter 4, we consider a general version of the *Cournot Competition.* Cournot competition, introduced in 1838 by Antoine Augustin Cournot, is a fundamental economic model that represents firms competing in a single market of a homogeneous good. Each firm tries to maximize its utility (naturally a function of the production cost as well as the market price of the product) by deciding on the amount of production. This problem has been studied comprehensively in Economics and Game Theory; however, in today's dynamic and diverse economy, many firms often compete in more than one market simultaneously, i.e., each market might be shared among a subset of these firms. In this situation, a bipartite graph models the access restriction where the firms are on one side, the markets are on the other side, and the edges demonstrate whether a firm has access to a market or not. We call this game *Network Cournot Competition* (NCC). Computation of the equilibrium, taking into account a network of markets and firms and the different forms of cost and price functions, makes challenging and interesting new problems. In this work, we propose algorithms for finding pure Nash equilibria of the NCC games in different situations. We design a potential function for NCC, when the price functions for the markets are linear functions of the production in that market. This result lets us leverage the optimization techniques for a single function rather than multiple utility functions of many firms. However, for nonlinear price functions, this approach is not feasible—there is indeed no single potential function that captures the utilities of all firms for the case of nonlinear price functions. We model the problem as a *nonlinear complementarity problem* in this case, and design a polynomial-time algorithm that finds an equilibrium of the game for strongly convex cost functions and strongly monotone revenue functions. We also explore the class of the price functions that ensures strong monotonicity of the revenue function, and show a large number of price functions belong to this class. Moreover, we discuss the uniqueness of the equilibria in both of these cases which means our algorithms find the unique equilibria of the games. When the cost of the production in one market is independent from the cost of production in other markets for all firms, the problem can be separated into several independent classical *Cournot Oligopoly* problems in which the firms compete over a single market. We give the first combinatorial algorithm for this widely studied problem. Interestingly, our algorithm is much simpler and faster than the previous optimization-based approaches. The paper of this work, "Network Cournot Competition", has appeared on the *10th Conference on Web and Internet Economics (WINE 2014).*

1.5 Market Pricing for Data Streams

Finally, in the last chapter, we again deal with an assignment problem in a network of sellers and buyers, however, the problem is examined in *the streaming setting.*

The motivation for using streaming setting in this chapter is that internet-enabled marketplaces such as Amazon deal with huge datasets registering transaction of merchandises between lots of buyers and sellers. We specially focus on the streaming setting, the common model for big data analysis. As explained before, in the streaming setting, a stream of data arrives sequentially and must be analyzed by an algorithm with *limited* memory. The number of passes on the data stream is also limited (once or a limited number of times), and hence streaming algorithms must be frugal in the amount and nature of data that they choose to store. Here, the development of the pricing algorithms that are appropriate for use with massive datasets is studied.

An assignment of the prices to the items and the items to the buyers in our setting is called *envy-free*, if no buyer prefers his outcome to the outcome of the other buyers. An envy-free mechanism would specially be important here since it guarantees stability and fairness.

We consider both social-welfare maximization and revenue maximization versions of the envy-free pricing problem. We present an envy-free mechanism for social welfare maximization problem in the streaming setting using $O(k^2 l)$ space, where k is the number of different goods and l is the number of available items of each good. We also provide an α-approximation mechanism for revenue maximization in this setting given an α-approximation mechanism for the corresponding offline problem. Moreover, we provide mechanisms to approximate the optimum social welfare (or revenue) within $1 - \epsilon$ factor, in space independent of l which would be favorable in case l is large compared to k. Finally, we present hardness results showing approximation of *optimal prices* that maximize social welfare (or revenue) in the streaming setting needs $\Omega(l)$ space. The resulting paper, "Market Pricing for Data Streams", has been accepted for publication in *Conference on Artificial Intelligence (AAAI 20017)*.

Chapter 2: Selling Tomorrow's Bargains Today

2.1 Introduction

Google makes 96% of its revenue by offering free advertising services to its customers. Google AdWords, an online auction-based advertising system, lets the advertisers bid on keywords for showing their ads in Google's search results. The *cost per click* (CPC) amount that an advertiser pays for users clicks on its ads heavily depends on the online demand and other competitors' bids and is not known to the advertisers at the bidding time.

Many risk-averse advertisers do not like to take this risk and prefer to have a predetermined contract which guarantees a fixed price for a fixed number of clicks.

Customers of many other businsessses face a similar situation due to the uncertainly of future demand, future costs, and competitors' behavior. As another example, consider a family that decides, on Monday, that they would like to go on a vacation the following weekend. Perhaps they do some research, and find a convenient location that seems both pleasant and affordable. The only thing left to do is to actually reserve their accommodations. However, this involves an interesting dilemma: should they book a room now, or wait until late in the week? Booking now assures them a place to stay that is affordable. On the other hand, many hotels offer last-minute deals, which could save the potential vacationers some money if they decide to wait. Unfortunately, the latter option carries not only the chance for large savings, but the risk that prices go up, perhaps even to the point where the vacation becomes impossible.

In this work, we study how a company might profit by offering customers a compromise between these options. While dealing with online prices typically carries too much risk and requires too much effort to appeal to individual customers, a coordinator has the advantage of spreading the risk across many contracts. By expending the effort to collect pricing data and forming estimates of future prices, a company could reasonably hope to monetize this advantage by offering customers a reliable contract with an affordable price, while executing the contract when prices are as favorable as possible. In fact, while not every contract may be profitable, good price estimates should provide a profit in aggregate.

In fact, this opportunity arises more generally – the key relevant aspects of our examples are uncertain future prices. Thus, one could hope to exploit this sort of future arbitrage when selling airline tickets, rental cars, event tickets, temporary labor contracts, or any product or service that typically faces price fluctuation. Our goal in this work is to answer this question: Given estimates of future prices, what is the best way for an enterprising coordinator to offer contracts to the buyers?

Two-stage optimization. We have a coordinator who can provide options from a set H, and who will have a chance to offer these options to a set of potential buyers B. This process, however, takes places in stages: in the first stage, the coordinator negotiates agreements; in the second stage, the prices will be realized, and the coordinator must serve options in the realized *scenario* to fulfill all of the previously made agreements. Each agreement with a buyer $b \in B$ specifies a *pack* $P \subseteq H$ of options that are acceptable to the buyer, and a *value* v_b the buyer must pay. The coordinator may satisfy the agreement by getting any option in the pack to the buyer, and it does not matter which one. The two-stage nature of our problem arises because the coordinator must make binding decisions about what agreements to make before prices are revealed.

2.1.1 Buyer-selected Packs

First stage: agreements. The first stage of our optimization problem models the formation of agreements. We mainly study a buyer-selected packs model where the pack P and value v_b are whatever pack and price the buyer included in their offer. Note that agreements are only formed when an offer is made and the coordinator accepts; therefore, we refer to the set S of buyers the coordinator formed agreements with as the *served set*.

Second stage: execution. In the second stage, the coordinator must match each buyer $b \in S$ to a option in their associated pack. At this point, the prices are revealed, and the coordinator's problem becomes one of maximum-weight matching. We call the collection of revealed prices a *scenario*, and denote it by I; we denote the full set of possible scenarios are $\mathcal{I}$. We denote the price of option h in scenario I by c_h^I. The I seen in the second stage is drawn according to a probability distribution, and the coordinator has the ability to sample from this distribution.

Objectives. The coordinator's objective is to maximize *profit*. We denote the profit from a served set S as

$$P(S) = \sum_{b \in S} v_b + [\sum_{h \notin \mathcal{M}^I(S)} c_h^I],$$

where $\mathcal{M}^I(S)$ is the cheapest set of options that buyers in S can be matched to in scenario I, and the expectation is over which I occurs. The first term is the profit that is extracted from agreements in S, e.g., set of contracts in the Google advertising example. The second term is the profit that is made by selling the remaining options in the future, e.g., selling through online Google AdWords system.

In some applications such as the hotel booking system the value c_h^I can be interpreted as the cost of providing option h in scenario I. In these situations, we study the regret minimization problem rather than the profit maximization one. Therefore, we also consider a modified objective we call *regret*, which has the form

$$R(S) = \sum_{b \in B \setminus S} v_b + [\sum_{h \in \mathcal{M}^I(S)} c_h^I].$$

Note that $R(S)$ is an affine transformation of $P(S)$. Intuitively, the regret objective tries to capture the idea of lost revenue, where we can lose revenue either by choosing not to serve a buyer, or by having to spend it to pay for an option.

2.1.2 Coordinator-selected Packs

We also investigate the setting where the coordinator decides about the offers. In particular, the coordinator partitions the set of available options into packs such that all the options inside a pack have similar properties, e.g., all keywords related to the word "insurance" in the Google advertising example.The packs are formed entirely based on external criteria. The customers then arrive one-by-one and choose a pack they like, and the coordinator offers each of them a non-negotiable price which they can only accept or reject.

2.1.3 Examples

Example 1 ***Buyer-selected packs****: In this example, there are two buyers* b_1 *and* b_2*, three hotels* h_1*,* h_2*, and* h_3*, and three possible weekend scenarios. Each weekend scenario can be represented by a vector of* 3 *elements indicating the weekend costs of the three hotels. Assume the weekend scenarios are* $I_1 = \{125, 250, 25\}$*,* $I_2 = \{200, 25, 225\}$*, and* $I_3 = \{75, 150, 100\}$*, and they happen with probabilities* 0.4*,* 0.3*, and* 0.3*, respectively. The first buyer is willing to pay a price equal to* 125 *dollars for being served, while the second buyer will pay* 100 *dollars. Figure 2.1.1 illustrates this example.*

In this case, if we only serve b_1*, the best matches to this buyer in weekend scenarios* I_1, I_2*, and* I_3 *cost* 125*,* 25*, and* 75 *dollars, respectively. Therefore, our expected revenue from choosing only* b_1 *to serve would be* $125-(0.4(125)+0.3(25)+0.3(75)) =$

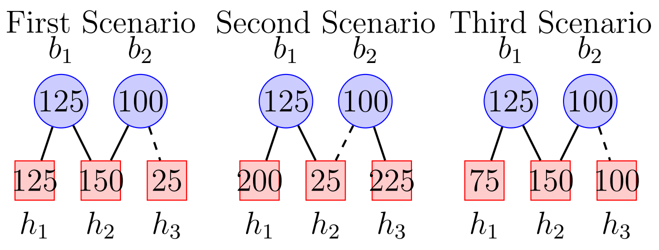

Figure 2.1.1: Buyer-selected packs example graph

45 dollars. On the other hand, if we only choose b_2, our expected revenue would be $100 - (0.4(25) + 0.3(25) + 0.3(100)) = 52.5$ dollars. Finally, if we choose both buyers to serve, we may no longer be able to serve each buyer with their cheapest feasible hotel. In the first scenario, the best hotels to book for b_1 and b_2 would be h_1 and h_3, respectively, for a total cost of 150 dollars. Similarly, it would cost us 225 and 175 dollars to serve both buyers in the second and third weekend scenarios, respectively. Therefore, the expected total profit from serving both customers would be $125+100-(0.4\times 150+0.3\times 225+0.3\times 175) = 45$ dollars. Thus, our best option is to only serve b_2 for a total profit of 52.5 dollars, even though her offered price is less than the price b_1 is willing to pay us.

Explanation of Figure 2.1.1: Each graph corresponds to one scenario. The upper vertices show the buyers and the price they are willing to pay. The lower vertices show the hotels and their cost at the weekend in each scenario. The edges indicate the buyers' interest in hotels. In this example, the best Monday decision is to choose b_2, for which our best weekend matches are shown with dashes.

Example 2 ***Coordinator-selected packs***: *Suppose there is only one hotel with three rooms available. A key feature of the seller-selected packs model is that the* *coordinator only needs to know the expected marginal cost of serving each agent to make decisions; therefore, we can assume that the expected cost to rent a single room is* 50 *dollars, and the expected additional cost to rent second and third rooms, respectively, is* 100 *dollars and* 200 *dollars.*

All the customers in this example have the same budget probability distribution which is 50 *dollars with probability* 0.4*,* 100 *dollars also with probability* 0.4*, or* 150 *dollars with probability* 0.2*. Recall that the coordinator only knows the distribution, and does not know the exact budget of the buyer when choosing a price to offer her.*

If the coordinator expects to face exactly 3 *buyers, what is the best way to set prices for the hotel rooms? We can immediately observe that there is no benefit in picking a price strictly between two possible amounts in customers' budget probability distribution, or in picking a price that is not strictly greater than the (expected) marginal cost to rent another room. Thus, the coordinator should charge either* 100 *or* 150 *dollars for the first hotel room rented,* 150 *dollars for the second hotel room rented, and never sell three hotel rooms. The only decision left for the coordinator is how to price the initial room. Case analysis indicates that the best strategy is to start at the higher price of* 150*, and if this is rejected by the first buyer, immediately switch to the lower price of* 100 *for future buyers.*

To see what actually happens in such an instance, consider the case where the first customer's budget is 50 *dollars, and the second and third customers' budgets are* 150 *dollars. The coordinator offers a price of* 150 *to the first buyer which will be rejected. Next, the coordinator offers a price of* 100 *to the second buyer which will* *be accepted, and finally, he offers a price of* 150 *dollars to the last buyer which will also be accepted. Note the price is increased again for the third customer, since the coordinator's marginal cost increased when the second customer accepted the offer. Since the expected costs for the coordinator to rent the first and the second hotel room are* 50 *and* 100 *dollars respectively, a profit of* $250 - 150 = 100$ *is made from these particular customers.*

2.1.4 Our results

We mainly study the buyer-selected packs model where our problem becomes fundamentally complicated. Since buyers may specify any pack of hotels they like, the marginal value of serving a particular customer becomes hard to quantify – they may conflict with other buyers in complex and arbitrary ways. Nevertheless, we prove the profit function is submodular, and thus a polynomial time algorithm approximates the optimum value within a factor of 0.42 [1–3].

Theorem (Section 2.2, Theorem 2.1) The profit function is submodular in the buyer-selected packs model.

We also study the regret objective function that has been used in the associated literature [4–7]. The regret objective has the form of "missed" value; note that exactly minimizing regret is equivalent to exactly maximizing profit. Unfortunately, the regret object functions is supermodular and does not have the nice structural property of the profit object function. We begin by showing that we can use sampling to construct an integer program that, with high probability, provides a $(1 + O(\epsilon))$-approximation to the regret objective. Unfortunately, this does not directly lead to an approximation, as we show that the integrality gap of the corresponding linear program is quite high.

Theorem (Section 2.3.1, Theorem 2.3) The integrality gap of the buyer-selected packs IP is at least $\Omega(\log N)$.

The high integrality gap of aforementioned linear program leads us to consider bicriteria-style approximations; our main result is the following, which provides an approximation to the regret objective by relaxing the matching constraints between buyers and options.

Theorem (Section 2.3.1, Theorem 2.4) Any fractional solution to the buyer-selected packs integer program can be rounded to an integral solution while increasing the regret objective value by at most a factor of $1/f$, while ensuring that no option is matched to more than 2 buyers and at most a $\min\{\frac{f}{1-2f}, \frac{1}{2}\}$ fraction of buyers cannot be uniquely matched to an option, for any $0 < f < \frac{1}{2}$.

Last but not least, we study the coordinator-selected hotel packs model and present an approximation to the optimal *profit*, the value the coordinator receives in the first stage plus the value that she achieves in the second stage. The nice structure on packs ensures the marginal value of serving an additional customer is well-defined; this, in turn, makes it reasonable to analyze the trade-offs involved in deciding whether to serve a buyer even in online settings with very general value models for options. Our main theorem for this model is the following:

Theorem (Section 2.4, Theorem 2.7) Let ϵ and δ be arbitrary small positive numbers, N_P be the number of packs, and $N^{\max}$ be the maximum number of buyers that can ever arrive. Assume every pack contains a constant number of options. If every option price lies in the range $[\ell, h]$, then using $4N_P(N^{\max})^3h/\ell\epsilon^2\delta$ samples of scenarios we can generate a set of offers which produce profit that approximates the optimal profit to an additive factor of ϵh with probability $1 - \delta$.

In the above, the requirement that the number of options in a pack be constant arises from allowing general cost functions; we could remove this requirement by restricting value functions. The key technical challenge in this section arises from the two-stage nature of our problem: given exact values for expected marginal values, we can use a dynamic program to compute the profit maximizing prices to offer. However, we only have sample access to the distributions, and so must rely on estimates of the expected marginal values. In essence, our main theorem above says that we can do so in our dynamic program with only a small additive penalty.

2.1.5 Related work

Our problem falls into the framework of two-stage stochastic optimization. This framework formalizes hedging against uncertainty into two stages: in the first, decisions have low cost but the exact input is uncertain; in the seceond, the input is known but decisions have high cost. Many problems have been cast in this framework, e.g., set cover, minimum spanning tree, Steiner tree, maximum weighted matching, facility location, and knapsack [8–11]. Prior work has considered linear programming approaches in this framework [12,13], for example the Sample Average Approximation (SAA) method to reduce the size of a linear program [14,15]. Ensuring the reduced linear program is representative of the original problem is generally hard and requires problem-specific techniques for most combinatorial optimization settings, however, and so no unified framework has been developed so far.

The buyer-selected packs version of our problem is most closely related to bipartite matching problems in this literature. Katriel et al. [16] consider such a problem where the goal is to buy an edge set containing a maximum matching, and balance fixed first-stage edge costs against the potential risks and rewards of random second-stage edge costs. They propose a polynomial-time deterministic algorithm which approximates the expected cost of minimum weight maximum matching within a factor of $O(n^2)$, where n is the size of the input graph. They also design a polynomial-time bicriteria randomized algorithm which returns, with probability $1-e^{-n}$, a matching of size at most $(1-\beta)n$ which approximates the optimum cost within a factor of $1/\beta$. In our setting, however, we *must* book a room for every buyer chosen in the first stage, and this bicriteria algorithm gives no guarantee on the set of chosen but unmatched buyers – they might even all have demanded the exact same option. We seek an algorithm assigning a limited number of customers to each option, even in the worst case, an objective that requires significant new insight compared to the setting of [16]. We design an algorithm which assigns at most two customers to each option. Kong and Schaefer [17] give results for the maximum-weighted matching problem, but this objective fails to capture either of our problems.

The seller-selected packs model is closely related to pricing problems in the algorithmic game theory literature. How to optimally form packs of items (of which, the buyer will receive only one) has been studied both in models where buyers have distributional knowledge of which item they will receive from a pack [18, 19], and where buyers have no such knowledge [20]. The focus in these works is different from our own, as we assume packs capture external characteristics and only prices are controlled by the coordinator. The problem of choosing optimal prices to offer a sequence of buyers has also been studied extensively (see, e.g., [21,22] and references therein), but the uncertainty of supply costs in our setting presents novel challenges. A related question is how to set prices when the *number* of items, rather than their cost, is unknown [23]; the challenges in the two settings require different techniques, however. Our problem also bears similarity to secretary problems, where non-linear objectives [24, 25] and even value-minus-cost objectives [26] have been considered. The differences between revenue and value achieved by a pricing, however, makes these results hard to apply in our setting.

2.2 Profit Maximization in Buyer-selected Packs Model

In this subsection, we consider the profit function from *Buyer-selected Packs* model. Our first step is to consider the second-stage of the coordinator's optimization problem more closely. Note that we let customers form any pack of options they like.

Since packs can now intersect in arbitrary ways, the problem of choosing how to assign buyers to options once prices are revealed becomes more complicated. We shall show, however, the coordinator's objective function has good structural properties. In particular, we show that the profit objective function is submodular. In order to show the submodularity of the profit function, we first prove the expected cost for satisfying a set of buyers $S \subseteq B$ in the second stage is supermodular in S, where the cost of satisfying a set of buyers $S \subseteq B$ is defines as follows:

$$\mathrm{C}(S) = [\textstyle\sum_{h \in \mathcal{M}^I(S)} c_h^I],$$

where $\mathcal{M}^I(S)$ is the minimum matching that covers buyers in S in scenario I, and the expectation is over which I occurs. We then leverage the supermodularity of the cost function and prove the profit function is submodular.

We now show that the expected cost for reserving a set of buyers $S \subseteq B$ in the second stage is supermodular in S. We begin by showing that for any fixed future scenario $I \in \mathcal{I}$, the cost $\mathrm{C}_I(S) = \sum_{h \in \mathcal{M}^I(S)} c_h^I$ of reserving options for a set $S \subseteq B$ is supermodular in S. Since our expected cost overall is just a weighted sum of the costs in each possible scenario, it immediately follows that the expected cost of serving a set of buys is supermodular as well. Thus, for the rest of this section, our discussion and arguments fall within the context of a single fixed future scenario $I \in \mathcal{I}$, and so omit it from our notation. Before we begin our proof, however, we first define some notation that will prove useful. First, given a set of buyers S, let $\mathcal{M}(S)$ denote the minimum-cost matching of buyers S to options. Note that even after fixing a scenario, this may not be well weekend, since multiple matchings may give the same cost; careful tie-breaking is critical to our proofs, and so we defer further discussion of this matter until later. Lastly, we use $\mathrm{C}(S) = \sum_{h \in \mathcal{M}(S)} c_h$ to denote the minimum cost to serve a set of buyers S in our fixed scenario.

We now proceed to show that the function $\mathrm{C}(S)$ is supermodular in S, that is

$$\mathrm{C}(T \cup \{b\}) - \mathrm{C}(T) \geq \mathrm{C}(S \cup \{b\}) - \mathrm{C}(S)$$

for any $S \subseteq T \subseteq B$ and $b \in B \setminus T$. We start by finding a clean characterization of how adding buyers to our served set changes the optimal matching to options.

Lemma 2.1 *For any $S \subseteq T \subseteq B$ and any choice of $\mathcal{M}(S)$, there exists a choice of $\mathcal{M}(T)$ such that $\mathcal{M}(S) \triangle \mathcal{M}(T)$ consists of $k = T \setminus S$ disjoint paths of odd length. Furthermore, each of these paths has one endpoint in $T \setminus S$.*

Proof: Choose $\mathcal{M}(T)$ to be the minimum-cost matching covering T such that $\mathcal{M}(S) \triangle \mathcal{M}(T)$ is minimized. First, note that in $\mathcal{M}(S) \triangle \mathcal{M}(T)$, every element of $T \setminus S$ has degree exactly one; every element of S has degree either zero or two; every other element of B has degree zero; and every element of H has degree zero, one, or two. As such, we can immediately see that $\mathcal{M}(S) \triangle \mathcal{M}(T)$ can be decomposed into a disjoint union of paths and cycles, and the latter must all be of even length since our underlying graph is bipartite. We shortly show that if an even length path or cycle exists, we can use it to modify $\mathcal{M}(T)$ and get a minimum-cost matching that covers T but has strictly smaller symmetric difference with $\mathcal{M}(S)$. The claim immediately follows, since this means $\mathcal{M}(S) \triangle \mathcal{M}(T)$ is a disjoint union of paths of odd length, and as we already observed the set of vertices in B with degree one is precisely $T \setminus S$.

Let $\mathcal{C}$ be any cycle of even length in $\mathcal{M}(S) \triangle \mathcal{M}(T)$. Consider what it represents in the context of our original problem. It means that both of our matchings assigned the customers incident to $\mathcal{C}$ to the options incident to $\mathcal{C}$, just in a different order. Thus, $\mathcal{M}(T) \triangle \mathcal{C}$ would still be a minimum-cost matching, but have strictly smaller symmetric difference with $\mathcal{M}(S)$. Similarly, let $\mathcal{P}$ be an even length path in $\mathcal{M}(S) \triangle \mathcal{M}(T)$. Note that the endpoints of the path must lie in H – otherwise, the set of buyers served by $\mathcal{M}(S)$ and $\mathcal{M}(T)$ would be incomparable, rather than the former being a subset of the latter. Thus, we can see that in the context of our problem, the path $\mathcal{P}$ represents that the two matchings used served the incident buyers using slightly different sets of options. If an option has degree one in $\mathcal{M}(S) \triangle \mathcal{M}(T)$, however, we may conclude that it is used in precisely one of the matchings. Thus, it follows that both $\mathcal{M}(S) \triangle \mathcal{P}$ and $\mathcal{M}(T) \triangle \mathcal{P}$ are valid matchings covering S and T, respectively. Since both $\mathcal{M}(S)$ and $\mathcal{M}(T)$ are minimum-cost matchings, however, we may conclude either of these assignments of the buyers incident to $\mathcal{P}$ to options have the same cost. As such, $\mathcal{M}(T) \triangle \mathcal{P}$ is a minimum-cost matching that has strictly smaller symmetric difference with $\mathcal{M}(S)$, contradicting our choice of $\mathcal{M}(T)$. Thus, we may conclude that no paths of cycle of even length exist in $\mathcal{M}(S) \triangle M(T)$.

□

We may use the above lemma to show that the cost function is, in fact, supermodular.

Lemma 2.2 *For any $S \subseteq T \subseteq B$, and any $b \in B \setminus T$, we have that* $\mathrm{C}(T \cup \{b\}) - \mathrm{C}(T) \geq \mathrm{C}(S \cup \{b\}) - \mathrm{C}(S)$.

Proof: Consider applying Lemma 2.1 to the sets $T \cup \{b\}$ and S, and some minimum-cost matching $\mathcal{M}(S)$. Let $\mathcal{P}_b$ be the resulting path with endpoint b, and let $\mathcal{P}_{T\setminus S}$ be the union of the paths with endpoints in $T \setminus S$. Observe that each of these paths is an alternating path with respect to $\mathcal{M}(S)$, and that since they are disjoint they can be applied one-by-one to $\mathcal{M}(S)$ in any order to produce a sequence of matchings. Since every path has odd length, we can see that it will increase the size of the matching by one and the cost of the matching by precisely the cost of the option that is one of the path's endpoints. But then, $\mathcal{M}(S) \triangle \mathcal{P}_b$ is a matching covering $S \cup \{b\}$, and so has cost at least $\mathrm{C}(S \cup \{b\})$. Similarly, $\mathcal{M}(S) \triangle \mathcal{P}_{T\setminus S}$ is a matching covering T, and so has cost at least $\mathrm{C}(T)$. But then we can see that

$$\mathrm{C}(T \cup \{b\}) - \mathrm{C}(S) \geq (\mathrm{C}(S \cup \{b\}) - \mathrm{C}(S)) + (\mathrm{C}(T) - \mathrm{C}(S)).$$

Rearranging terms gives precisely the desired inequality. □

Theorem 2.1 *The profit function is submodular in the buyer-selected packs model.*

Proof: We first write the profit objective function as follows:

$$P(S) = \sum_{b \in S} v_b + [\sum_{h \in H} c_h^I] - \mathrm{C}(S).$$

Knowing facts that $\mathrm{C}(S)$ is supermodular (based on Lemma 2.2), $[\sum_{h\in H} c_h^I]$ is a constant independent of S, and $\sum_{b\in S} v_b$ is just an additive function we can conclude the profit function is submodular. □

2.3 Regret Minimization in Buyer-selected Packs Model

2.3.1 Approximate-optimality via sampling

Charikar et al. consider general 2-stage stochastic models. In these models, one must make a decision in the first stage which leads to a known cost in the first stage and an unknown cost in the second stage. For our problem, this first stage decision is choosing which customers to serve. In terms of the regret objective, our first stage cost is the values of customers we do not choose to serve, and our second stage cost is buying options for the customers we choose to serve. To that end, we use

$$g(S) = \sum_{b \notin S} v_b \qquad \text{and} \qquad w(S, I) = \sum_{h \in \mathcal{M}^I(S)} c_h^I$$

to denote the first and second stage costs, respectively, of choosing to serve a set of customers $S \subseteq B$ when second stage scenario $I \in \widehat{\mathcal{I}}$ happens. Recall that $\mathcal{M}_I(S)$ is the minimum-cost matching between customers in S and the options in second stage scenario I. Thus the regret objective for a future scenario I is $R_I(S) = g(S) + w(S, I)$. The goal is to find a first stage decision $S \subseteq B$ so that $E_I\left(R_I(S)\right) = g(S) + E_I\left(w(S, I)\right) = R(S)$ is minimized. space $\mathcal{I}$ might be very large it is hard to solve the problem of minimizing the function R over the full space $\mathcal{I}$. Instead, we define another function $\hat{R}$ from R as follows. Given N independent samples of scenarios $I_1, I_2, ..., I_N$ from the space $\mathcal{I}$, we estimate the function R by $\hat{R}(S) = g(S) + \frac{1}{N}\sum_{1 \leq i \leq N} w(S, I_i)$.

In order to apply Charikar et al.'s theorem, we need to prove some properties on the first and second stage costs. These properties are as follows.

1. Both first and second stage costs must always be nonnegative for all first stage decisions and all future scenarios.

2. There must exist a first stage decision for which the first stage cost is zero and the second stage cost is more than that of any other first stage decision for any future scenarios. That is there must exist a first stage decision $S_0 \subseteq B$ for which $g(S_0) = 0$ and $w(S, I) \leq w(S_0, I)$ for all $S \subseteq B$ and all $I \in \widehat{\mathcal{I}}$. We call this S_0 the *null* decision.

3. There must be a bounded inflation factor. That is if S_0 is the null decision from the previous property, then $w(S_0, I) - w(S, I) \leq \eta g(S)$ should hold for all $S \subseteq B$ and a fixed finite real number η. This means the penalty that we have to pay in the second stage because of making a null first stage decision compared to any other first stage decision is no more than a constant factor of the cost of the other first decision.

In our problem, $g(S) = \sum_{b \notin S} v_b$. Since v_b is nonnegative for all customers $b \in B$, $g(S)$ should also be nonnegative for all S. Moreover, $w(S, I)$ is equal to the cost of the matching $\mathcal{M}_I(S)$; since the weights in this matching represent nonnegative option costs, this must be nonnegative as well. Thus, the first property holds. For the second property, we claim B gives the desired null decision for the first stage. Now, $g(B) = \sum_{b \notin B} v_b = 0$, and for a fixed future scenario I, the optimization problem on the future would be matching all the customers, which must be more costly than matching any other subset of customers. The third property holds for our problem with $\eta = \frac{Max^H}{Min^B}$, where Max^H is the maximum possible option price and Min^B is the minimum value of customers. We can see this because

$$\begin{aligned}
\sum_{h \in \mathcal{M}_I(B)} c_h^I - \sum_{h \in \mathcal{M}_I(S)} c_h^I &\leq \sum_{b \notin S} Max^H \\
&\leq \eta \sum_{b \notin S} Min^B \\
&\leq \eta \sum_{b \notin S} v_b = \eta g(S).
\end{aligned}$$

Thus, we may apply the following theorem, which is a restatement of a theorem from [15], specialized to our setting.

Theorem 2.2 *Let $\hat{R}$ be an estimate of R using $\Theta(\eta^2 \frac{1}{\epsilon^4} \log(|\mathcal{I}|) \log(\frac{1}{\delta}))$ samples of scenarios. Let $\hat{S}$ be the set of buyers which minimizes $\hat{R}$. Then $R(\hat{S})$ is a $(1+O(\epsilon))$-approximate of the optimum value of R with probability at least $1-2\delta$. That is with probability at least $1-2\delta$, the inequality $R(\hat{S}) \leq (1+O(\epsilon))R(S)$ holds for all $S \subseteq B$.*

2.3.2 Approximating Regret Function After Sampling

Unfortunately, the regret objective given by $\hat{R}$ remains hard to approximate as well. While we can phrase our problem as an integer program, we can show that the integrality gap of this program is quite large. This motivates us to try relaxing some of the constraints in our problem, and find a bicriteria-style approximation.

We investigate how the coordinator can minimize the regret objective $R(S)$, in the buyer-selected packs model. We can find an approximate solution $\hat{R}$ to function

$$\textbf{minimize:}\quad \left(\sum_{b\in B}(1-Y_b)v_b + \frac{1}{N}\sum_{1\le k\le N}\sum_{(b,h)\in E} x_{hbk}c_{hk}\right)$$

$$\begin{array}{lll}
\textbf{subject to:} & \sum_{h\in H} x_{hbk} \ge Y_b & \forall b\in B, \forall 1\le k\le N \\
 & \sum_{b\in B} x_{hbk} \le 1 & \forall h\in H, \forall 1\le k\le N \\
 & \mathrm{Y}_b, x_{hbk} \in \{0,1\} & \forall h\in H, \forall b\in B, \forall 1\le k\le N
\end{array}$$

Figure 2.3.2: The buyer-selected packs integer program

R using polynomially-many samples of scenarios and solving the problem simultaneously for these samples based on the previous section. Therefore, our goal is to minimize function $\hat{R}$. The integer program for the problem of minimizing function $\hat{R}$ can be written as in Figure 2.3.2. We call it the buyer-selected packs IP.

In the buyer-selected packs IP: (i) N is the number of samples; (ii) c_{hk} is the known price of hotel h in the k^{th} sample; (iii) variable Y_b is 1 if and only if $b \in S$, and $Y_b = 0$ otherwise; and (iv) variable x_{hbk} is 1 if and only if hotel h is assigned to buyer b in the k^{th}, sample and is 0 otherwise. Constraint (2.3.2) requires that if $Y_b = 1$, then at least one hotel should be assigned to this buyer in every sample $1 \le k \le N$. We call this family of constraints the *capacity* constraints. Constraint (2.3.2) for each hotel $h \in H$ and each scenario k requires that hotel h in sample k can be assigned to no more than one buyer; we call this family of constraints the *assignment* constraints. The objective function of this IP is exactly equal to $\hat{R}(S)$. Let $\sum_{b\in B}(1-Y_b)v_b$ be the *lost* term, and $\sum_k \sum_{(b,h)\in E} x_{hbk}c_{hk}$ be the *cost* term of the objective function. In the following, we relax the last two constraints of this IP to their linear counterparts $x_{hbk} \in [0,1]$ and $Y_b \in [0,1]$ to obtain an linear program. Edge between buyer b and hotel h in sample k is a *fractional* edge in a LP solution $\langle \mathbf{x}, \mathbf{Y} \rangle$ if and only if $0 < x_{hbk} < 1$.

The first question that comes to mind when trying to find a minimizer of function $\hat{R}$ is whether we can use a solution to the LP to find an exact or an approximate solution to the IP. However, we show that the integrality gap between the IP and LP solutions can be quite high by the following theorem:

Theorem 2.3 *The integrality gap of the buyer-selected packs IP is at least* $\Omega(\log N)$*.*

Proof: Assume there are $2n$ buyers and $(2n-2)\binom{2n}{n}$ hotels. Suppose there are only two possible prices for each hotel at the weekend: the *low* price for all hotels is 0 and the *high* price for all hotels is $(2n+1)\binom{2n}{n}$. For each buyer b, $v_b = 1$. Partition the hotels into $\binom{2n}{n}$ groups of size $2n-2$ each. Consider all the subsets of buyers of size n. There are $\binom{2n}{n}$ such subsets. Let γ denote any one to one mapping from these subsets to hotel groups. For each subset S and the group of hotels $\gamma(S)$ mapped to it add edges from each $b \in S$ to all the first $n-1$ hotels in $\gamma(S)$ and add edges from each $b \notin S$ to all the last $n-1$ hotels in $\gamma(S)$. Figure 2.3.3 illustrates the edges between the buyers and the group of hotels $\gamma(S)$ for an arbitrary subset of buyers S of size n. Consider $\binom{2n}{n}$ samples and in sample i suppose all the hotels in the i-th group have the low price (which is 0) and all other hotels have the high

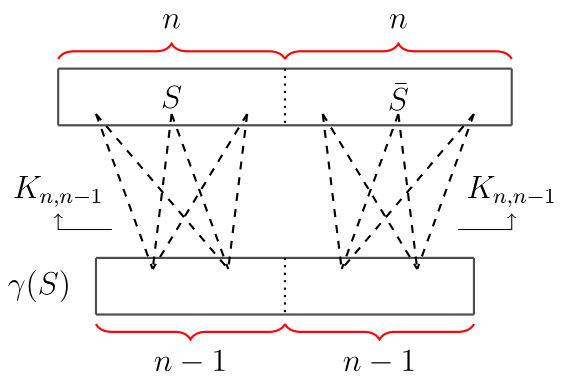

Figure 2.3.3: Illustration of the instance of the problem with high integrality gap.

price.

1

Since the high price for hotels is very large, we can never assign a person to a hotel that has a high price. This is true because a trivial solution to the IP is when all the variables are equal to 0 in which case the objective value would be equal to $2n$. However, if we assign a person to a hotel with high price for even one sample, the objective would become at least $2n + 1$, which is worse than the trivial solution. Therefore, in any optimal solution to the IP we can assume x_{hbk} is 1 only if hotel h in sample k has price 0. Thus, the second term in the objective function is always

[1]Figure 2.3.3 explanation: In this graph the upper vertices show the buyers with a subset of size n labeled as S and the rest of buyers labeled as $\bar{S}$. The lower vertices show the group of hotels $\gamma(S)$. Note that in exactly one scenario all the hotels in $\gamma(S)$ have a zero cost and they have very high costs in all other scenarios. Therefore, we can only match the buyers to hotels in $\gamma(S)$ in one scenario, and no matching can use this set of hotels in any other scenario. The best LP solution requires us to assign an amount $\frac{1}{n}$ to each edge in this figure in the aforementioned scenario, and to assign a value equal to zero to all these edges in all other scenarios.

equal to 0, and the objective function for this example simplifies to $\sum_{b \in B} (1 - Y_b)$, the number of buyers that we fail to serve.

Note that the objective of the IP in this example is to maximize the number of buyers served. Recall, however, our earlier observation that an optimal integral solution can never use a hotel with high cost. By construction, this means that an optimal IP solution can only serve at most $n-1$ buyers. Note this property is much less restrictive for LP solutions. Simply serving every buyer at a rate of $(1 - 1/n)$ will satisfy this requirement while serving a (fractional) total of $2n - 2$ customers. We now proceed to formalize the above discussion and demonstrate that it yields the desired gap.

We present a feasible LP solution for which the objective value is equal to 2. Then we show no IP solution can achieve an objective value of less than $n + 1$ and we conclude the integrality gap between the LP and IP solutions can be very large.

Here we present a feasible LP solution.

$$Y_b = \frac{n-1}{n} \qquad \forall b \in B \tag{2.3.1}$$

$$x_{hbk} = \begin{cases} \frac{1}{n} & \text{if } c_{hk} = 0 \\ 0 & \text{otherwise} \end{cases} \qquad \forall h \in H, \forall b \in B, \forall 1 \leq k \leq N$$

Consider the first set of constraints in the LP. For each buyer $b \in B$ and each sample, b has edges to $n-1$ hotels with low prices. Therefore, for $n-1$ hotels $x_{hbk} = \frac{1}{n}$. This means $\sum_{h \in H} x_{hbk}$ in each sample for each buyer b is $\frac{n-1}{n}$, which is equal to Y_b and the constraint holds. Now consider the second set of constraints. For each hotel

$h \in H$ and each sample, at most n buyers have an edge to this hotel. The maximum possible value of the variables x is $\frac{1}{n}$. Therefore, $\sum_{b \in B} x_{hbk}$ cannot exceed 1. Thus the set of variables form a feasible LP solution. The objective value with this set of variables would be equal to $\sum_{b \in B} (1 - \frac{n-1}{n}) = 2$.

On the other hand, assume an IP solution can achieve an objective value less than $n+1$. Since the objective function is equal to $\sum_{b \in B} (1 - Y_b)$, this means for at least n buyers $Y_b = 1$. Consider S to be the set of these n buyers. Consider the sample which has the group of hotels $\gamma(S)$ as its low price hotels. In this sample, these n buyers have edges to only $n-1$ hotels with low prices. It means at least one of them is matched to a hotel with the high price. Therefore, Y_b can be equal to 1 for at most $n-1$ of them contradicting the fact that $Y_b = 1$ for all the buyers $b \in S$. Therefore, no IP solution can achieve an objective value less than $n+1$.

Therefore, while the optimum LP solution is less than or equal to 2, the optimum IP solution cannot be less than $n+1$. Since, the number of vertices is $2n+(2n-2)\binom{2n}{n} = O(4^n)$, we conclude the integrality between our LP and IP is logarithmic. $\square$

The result of Theorem 2.3 leads us to consider relaxations of our problem. In particular, we consider relaxing the constraint that requires matching at most one customer to each hotel. We will allow ourselves to match **up to two** buyers to a hotel, but try to minimize the fraction of buyers who are not matched uniquely. We say a buyer is **multi-covered** in a scenario if she is matched to the same hotel as a previous buyer in that scenario: if we match 2 buyers to a hotel then one of them is multi-covered. We formally define the bicriteria-style approximation below.

Definition 2.1 *An (α, β)-approximate solution to the Buyer-selected packs IP is a solution which has an objective value at most α times the objective value of the optimal solution to this IP while the number of buyer vertices that it multi-covers in all graphs overall is no more than β times the number of buyer vertices that it covers in all graphs overall.*

Theorem 2.4 *For any given f such that $0 < f < 1/2$, we can find in deterministic polynomial time, an $(1/f, \min\{\frac{f}{1-2f}, \frac{1}{2}\})$-approximate solution to the Buyer-selected packs IP in which in every scenario, any hotel is matched to at most two buyers.*

Proof:

The four-step algorithm supporting Theorem 2.4 is parametrized by $0 < f < 1/2$ and is described next. The primary work done is for Step 4, as seen below.

Step 1: Solving the LP. Solve the LP relaxation; let $x^{(1)}$ and $y^{(1)}$ denote the vectors x and Y of the LP, that occur as the optimal solution-vectors.

Step 2: Filtering. Update $y^{(1)}$ to $y^{(2)}$ as follows: for all b such that $y_b^{(1)} \leq 1 - f$, set $y_b^{(2)} := 0$, with $y_b^{(2)} = y_b^{(1)}$ for all other b. Let $x^{(2)} := x^{(1)}$.

Step 3: Scaling up. Update $y^{(2)}$ to $y^{(3)}$ as follows: for all b such that $y_b^{(2)} > 0$, set $y_b^{(3)} := 1$ (we have $y_b^{(3)} = 0$ for all other b). Next update $x^{(2)}$ to $x^{(3)}$ in two sub-steps as follows:

- for all b such that $y_b^{(3)} = 1$, and for all (h, k), set $x_{hbk}^{(3)} := x_{hbk}^{(2)}/y_b^{(1)}$, so that the constraints (2.3.2) are satisfied; for all other (h, b, k), initialize $x_{hbk}^{(3)} := x_{hbk}^{(2)}$;

- arbitrarily decrease the $x^{(3)}_{hbk}$ values (subject to non-negativity) such that equality now holds in the constraints (2.3.2).

Step 4: Derandomized Dependent Rounding. Separately for each scenario k, we apply a certain derandomized version of the bipartite dependent-rounding procedure of [27] to the vector $x^{(3)}$ (restricted to the index k): the details are as follows. Let $\ell_k(h) = \sum_b x^{(3)}_{hbk}$ denote the fractional load on hotel h. This procedure rounds x_{hbk} for each (h, b) – recall that we are considering any fixed k now – to some $X_{hbk} \in \{0, 1\}$, such that the following properties hold, among others:

(P1) For all (h, b), $E[X_{hbk}] = x^{(3)}_{hbk}$;

(P2) For all b such that $y^{(3)}_b = 1$, $\sum_h X_{hbk} = 1$ with probability one, and

(P3) For all h, $\sum_b X_{hbk} \in \{\lfloor \ell_k(h) \rfloor, \lceil \ell_k(h) \rceil\}$ with probability one.

We will run a derandomized version of this procedure as follows. For $i = 1, 2, 3$, let L_i and C_i denote the "lost" and "cost" values of the objective function for scenario k, at the end of step i above. That is, for $i = 1, 2, 3$, at the **end** of Step i above, let

$$L_i = \sum_{b \in B} (1 - y^{(i)}_b) v_b \ \text{ and } \ C_i = \sum_{(b,h)} x^{(i)}_{hbk} c_{hk}.$$

Let $t = \sum_b y^{(3)} b$ be the final number of buyers chosen, and define $H_k = \{h : \ell_k(h) > 1\}$; let $s = |H_k|$. Consider the potential function

$$\Phi = \frac{f}{1-f} \cdot \frac{\sum_{(b,h)} X_{hbk} c_{hk}}{C_3} + \frac{1-2f}{1-f} \cdot \frac{\sum_{h \in H_k} [(\sum_b X_{hbk}) - 1]}{\min\{tf, sf/(1-f)\}}.$$

At every step of the dependent-rounding procedure of [27] – which randomizes among two choices and continually updates the vector X which initially starts at $x^{(3)}$ –

deterministically make the choice that never increases Φ. As pointed out in [27], this is indeed possible ((P1) and the linearity of expectation, along with the nature of the choices made in [27], justify this).

Analysis of the algorithm. Let us start with L_i. It is easy to see that $L_2 \leq L_1/f$, and that L_i does not decrease any further. Thus, the "lost" value gets blown up by a factor of at most $1/f$, as compared to the initial LP value.

Note next that $x^{(3)}_{hbk} \leq x^{(1)}_{hbk}/(1-f)$. Combined with (2.3.2), this shows that $\ell_k(h) \leq 1/(1-f) \leq 2$ for all (h, k). Thus, property (P3) assures us that the final load $\sum_b X_{hbk}$ on hotel h in scenario k will be at most two.

To analyze the cost and overbooking, we first claim that for all $h \in H_k$,

$$\sum_h (\ell_k(h) - 1) \leq \min\{tf, sf/(1-f)\}. \tag{2.3.2}$$

To see this, start by recalling that $\ell_k(h) \leq 1/(1-f)$ and note that: (i) the LHS of (2.3.2) is

$$\sum_h \ell_k(h) \cdot (1 - 1/\ell_k(h)) \leq \sum_b \ell_k(h) \cdot (1 - (1-f)) \leq tf,$$

and (ii) since $\ell_k(h) \leq 1/(1-f)$, the LHS of (2.3.2) is at most $s \cdot (1/(1-f) - 1) = sf/(1-f)$. Thus we have (2.3.2).

Therefore we see that Φ is initially at most $\frac{f}{1-f} \cdot 1 + \frac{1-2f}{1-f} \cdot = 1$, and thus never exceeds 1. Thus, the final cost value is at most $\sum_k C_3 \cdot (1-f)/f$. However, since $x^{(3)}_{hbk} \leq x^{(1)}_{hbk}/(1-f)$, this implies that the final total cost is at most the LP's cost times $((1-f)/f) \cdot 1/(1-f)$; thus, just like the "lost" function, the "cost" function again gets blown up by a factor of at most f.

Finally for the multi-covering. It is easy to see that the fraction of people multi-covered at the end is at most $U = (1/t) \cdot \sum_{h \in H_k} [(\sum_b X_{hbk}) - 1]$. Since $\Phi \leq 1$ at the end, this implies that

$$U \leq \frac{1-f}{1-2f} \cdot (1/t) \cdot \min\{tf, sf/(1-f)\}. \tag{2.3.3}$$

However, property (P3) shows an additional upper-bound on U:

$$U \leq \frac{\min\{t, 2s\} - s}{t} \tag{2.3.4}$$

A case analysis of the minimum of these two upper-bounds (e.g., based on whether s/t is at least or at most $1/2$), we get the bound

$$U \leq \min\{\frac{f}{1-2f}, \frac{1}{2}\}$$

as desired. □

2.4 Coordinator-selected Packs Model

In this section, we consider the setting where hotel packs are selected and suggested by the coordinator. In this setting, the coordinator will face a stream of buyers, each interested in a single (predetermined) pack of hotels. Upon arrival of each buyer, the coordinator decides what price should be offered to this buyer, and the buyer would reject or accept the offer immediately based on his or her budget. The coordinator wants to offer prices to buyers in a way that maximizes his own revenue and he must take into account both profit from the current customer and the expected profit obtained in the future. Note that in this setting the buyers arrive one by one and the coordinator only has statistics (rather than the exact values) of how many buyers will arrive in the future, what hotel packs they are interested in, and the costs and deals of hotel reservations. We assume that the coordinator forms packs of hotels based on fixed external qualities, such as review scores and location; i.e, the choice of hotel packs is predetermined, and is not part of the coordinator's optimization problem.

Before giving a general solution, we consider the problem with just one single hotel. In this setting, we interact with customers in an online fashion. Our cost to reserve rooms at this hotel is uncertain and can be quite general. For example, a hotel might normally charge guests \$150 per room, change its rate to \$100 per room if at least 10 rooms are reserved together(e.g by a coordinator), and in some rare cases offer a flat rate of \$125 per room. Thus, a coordinator must handle not only uncertainty over the prices, but over whether enough customers will appear to make bulk pricing feasible. This online problem is closely related to prophet inequality problems (see, e.g., [22] and references therein), but with hard supply constraints replaced by costs. The uncertainties present in our cost functions give a unique variant of the problem: for example, achieving approximately optimal profit may require us to incur negative profit initially in settings where we expect to see bulk price discounts.

In order to handle types of hotel costs in this setting, we need to consider the marginal cost of reserving a room. Let $\text{Total}_P(r)$ be the minimum cost of reserving r rooms from pack P, and let $\text{Cost}_P(r) = \text{Total}_P(r) - \text{Total}_P(r-1)$ be the the marginal cost of the r^{th} room. Both $\text{Total}_P(r)$ and $\text{Cost}_P(r)$ are random variables. When we consider multiple scenarios at once, we denote these quantities in scenario I as $\text{Total}_P^I(r)$ and $\text{Cost}_P^I(r)$, respectively. Finally, let $\text{Price}_P^{\min}$ and $\text{Price}_P^{\max}$ be the minimum and maximum possible prices, respectively, of reserving a room in pack P, in any scenario I and at any level of demand. Let $\lambda = \left(\text{Price}_P^{\max}/\text{Price}_P^{\min}\right)$ be the ratio of these prices.

In this setting, we expect to see a stream of n (upper bounded by constant $N^{\max}$) buyers, each interested in precisely one of the packs. Customers are characterized by the pack they are interested in and their willingness to accept an offered price; both are random variables that are independently and identically distributed across customers. The probability that a buyer is interested in pack P is denoted by q_P. The budget of the buyer has tail distribution function g_P; i.e, the buyer accepts an offered price of s with probability $g_P(s)$.

Each time a customer appears, the coordinator must choose a price to offer to him, balancing profit from the current sale against the effect on future profit. Future profits from a pack P of hotels depend both on how many customers we have seen and how many of them have bought rooms. If we have seen i customers and sold r rooms from pack P, we use $V_P(r, i)$ to denote the optimal (expected) profit we can achieve from future customers interested in P. Thus, a coordinator who wants to offer an optimal price to the $(i+1)^{\text{th}}$ customer must both consider the cost $[\text{Cost}_P(r)]$ of providing them a room and compare the values of $V_P(i+1, r)$ and $V_P(i+1, r+1)$. Let $\text{Offer}_P(r, i)$ denote the optimal price for the coordinator to offer in this situation.

Algorithm 1 OnlineDP

```
0: for  $r \leftarrow$ number of rooms in pack $P, \dots, 0$ do
0:     $V_P(r, N^{\max}) \leftarrow 0$
0:     for $i \leftarrow N^{\max} - 1, \dots, 0$ do
0:         $\mathrm{Rev}_P(r,i) \leftarrow \max_s\{\, g_P(s)(s - [\mathrm{Cost}_P(r+1)] + V_P(r+1, i+1)) + (1 - g_P(s))V_P(r, i+1)\}$
0:         $\mathrm{Offer}_P(r,i) \leftarrow \mathrm{argmax}_s\{\, (g_P(s)(s - [\mathrm{Cost}_P(r+1)] + V_P(r+1, i+1)) + (1 - g_P(s))V_P(r, i+1))\}$
0:         $V_P(r,i) \leftarrow \Pr(n > i | n \geq i)((1 - q_P)V_P(r, i+1) + q_P\mathrm{Rev}_P(r,i))$
0:     end for
0: end for=0
```

Algorithm 1 in is based around the optimization process referred to above, and as the following Theorem states, computes the optimal price for a coordinator to offer in all situations that can arise.

Theorem 2.5 *Given the value of* $[\mathrm{Cost}_P(r)]$ *for all packs* P *and numbers* r*, the OnlineDP Algorithm computes* $V_P(r,i)$ *and* $\mathrm{Offer}_P(r,i)$ *for all* r*,* i*, and* P*.*

Proof: Assume the $(i+1)^{\text{th}}$ customer has just arrived and is interested in pack P, and r of the previous i customers purchased rooms from pack P. We want to understand how to compute the optimal price to offer the $(i+1)^{\text{th}}$ customer. Say we offer them a price of s. If they accept, they pay us s and our expected cost to serve them is $[\mathrm{Cost}_P(r+1)]$; if they reject, we receive no payment and incur no cost. In the former case, our expected profit from future rounds is $V_P(i+1, r+1)$, while in the latter it is $V_P(i+1, r)$. Recall that the probability a customer interested in pack P will accept an offer of s is $g_P(s)$. Thus, we can see that if there is an $(i+1)^{\text{th}}$ buyer and they are interested in pack P, the best profit we can hope to receive in this and future rounds is precisely given by $\text{Rev}_P(r, i)$ in Figure 1, and is achieved by offering price $\text{Offer}_P(r, i)$.

Once we know how to compute optimal prices, however, we can compute the optimal profit we can achieve from pack P in this and future rounds. We have only three cases to consider: if i was the last customer, we receive no further profit from this (or indeed any) pack; if customer $(i+1)$ is interested in a pack other than P, we receive no profit in this round, but $V_P(r, i+1)$ from future rounds; and if customer $(i+1)$ is interested in pack P, we receive profit $\text{Rev}_P(r, i)$ as we just discussed. Note that the first happens with probability $\Pr[n = i | n \geq i]$; the second with probability $(1 - q_P) \Pr[n > i | n \geq i]$; and the third with probability $q_P \Pr[n > i | n \geq i]$, we can see that $V_P(r, i)$ is precisely as defined in our dynamic algorithm. □

While the previous theorem requires that we know $[\text{Cost}_P(r)]$ exactly, we can only estimate this quantity in our setting. The next two theorems show, however, that we can compute good estimates of $[\text{Cost}_P(r)]$ with few samples, and that using sufficiently good samples in Algorithm 1 provides revenue that is within a small additive loss of optimal.

Before proceeding to show that we can estimate the values of $[\text{Cost}_P(r)]$, we first state two concentration bounds that we need in our proof.

Lemma 2.3 *(Bhatia-Davis Inequality [28]) Suppose a random variable X is bounded* *between m and M, i.e. $m \leq X \leq M$ always. Then if μ is the expected value of X and σ^2 is the variance of X, we have that $\sigma^2 \leq (M-\mu)(\mu-m)$.*

Lemma 2.4 *(Chebyshev's Inequality) Let X be a random variable with expected value μ and non-zero variance σ^2. Then for any real number $k > 0$, we have that* $\Pr\left[|X-\mu| \geq k\sigma\right] \leq \frac{1}{k^2}$.

First, we use the Bhatia-Davis Inequality to bound the variance of $\text{Cost}_P(r)$; then we can use Chebyshev's Inequality to show that for any set $\widehat{\mathcal{I}}$ of $|\widehat{\mathcal{I}}| = \lambda/\delta\epsilon^2$ random and independent samples of future scenarios from $\mathcal{I}$, $\frac{1}{|\widehat{\mathcal{I}}|}\sum_{I\in\widehat{\mathcal{I}}} \text{Cost}_P^I(r)$ provides a good estimate of $[\text{Cost}_P(r)]$ with high probability. Note, however, that this is of little use if we cannot compute $\text{Cost}_P^I(r)$ efficiently for fixed I and r. The following lemma shows that we can do so whenever the pack P contains only a constant number of hotels.

Lemma 2.5 *Let P be a pack with a constant number of hotels. We can compute $\text{Cost}_P^I(r)$ in polynomial time for any fixed I.*

Proof: First, note that in any valid reservation of r rooms from a pack of hotels, at most r rooms can be reserved in any single hotel. Hence, the number of ways of reserving r rooms from a pack is no more than the number of ways of reserving between 0 and r rooms in each hotel in the pack. But there are precisely $(r+1)^{C_P}$ ways to do the latter, where C_P is the number of hotels in pack P. Also, once we fix the rooms being reserved, it is trivial to compute the associated total cost. Thus, by iterating over all possible reservations, we can computer $\mathrm{Total}^I_P(r)$, and hence $\mathrm{Cost}^I_P(r)$, in polynomial time for any fixed I. $\square$

Theorem 2.6 *Let ϵ and δ be arbitrary small positive numbers and let P be a pack with a constant number of hotels. Using $\lambda/\delta\epsilon^2$ samples of scenarios, with probability $1-\delta$ we can compute $[\mathrm{Cost}_P(r)]$ with relative error of ϵ.*

Proof: Let $\widehat{\mathcal{I}}$ be a set of $|\widehat{\mathcal{I}}| = \lambda/\delta\epsilon^2$ random and independent samples of scenarios. Since the number of hotels in pack P is constant, Lemma 2.5 tells us we can compute $\mathrm{Cost}^I_P(r)$ in polynomial time for any fixed scenario I. In the rest of the proof we show that $\widehat{Cost}_p(r) = \frac{1}{|\widehat{\mathcal{I}}|}\sum_{I\in\widehat{\mathcal{I}}}\mathrm{Cost}^I_P(r)$ provides a good estimate of $[\mathrm{Cost}_P(r)]$ with high probability. Our proof is based around the following two bounds:

$$Var(\mathrm{Total}_P(r)) \leq \mathrm{Price}^{\max}_P[\mathrm{Cost}_P(r)]; \text{ and} \tag{2.4.5}$$

$$\mathrm{Price}^{\min}_P \leq \mathrm{Cost}_P(r) \leq \mathrm{Price}^{\max}_P\,. \tag{2.4.6}$$

First, we show how our main claim follows from (2.4.5) and (2.4.6), and then we prove these two inequalities hold. Our goal is to show that

$$\Pr[|\,\widehat{\mathrm{Cost}}_P(r) - [\mathrm{Cost}_P(r)]| \geq \epsilon[\mathrm{Cost}_P(r)]] \leq \delta.$$

Since $\widehat{Cost}_p(r)$is the average of $|\widehat{\mathcal{I}}|$ independent samples of $\mathrm{Cost}^I_P(r)$, we know that $[\mathrm{Cost}_P(r)] = E[\widehat{\mathrm{Cost}}_P(r)]$ and $Var(\widehat{\mathrm{Cost}}_P(r)) = Var(\mathrm{Cost}_P(r))/|\widehat{\mathcal{I}}|$. So if we denote

$\sigma^2 = Var(\text{Cost}_P(r))$ and $\widehat{\sigma}^2 = Var(\widehat{\text{Cost}}_P(r))$, we can see that

$$\begin{aligned}
[\text{Cost}_P(r)] &\geq \sqrt{[\text{Cost}_P(r)]\text{Price}_P^{\min}} \\
&= \frac{\sqrt{\text{Price}_P^{\max}[\text{Cost}_P(r)]}}{\sqrt{\frac{\text{Price}_P^{\max}}{\text{Price}_P^{\min}}}} \\
&\geq \frac{\sigma}{\sqrt{\lambda}} = \widehat{\sigma}\sqrt{\frac{|\widehat{\mathcal{I}}|}{\lambda}},
\end{aligned}$$

where the first inequality follows from (2.4.6) and the second follows from (2.4.6). Thus we can upper bound the probability that our estimate $\widehat{\text{Cost}}_P(r)$ has relative error exceeding ϵ as

$$\Pr[|\widehat{\text{Cost}}_P(r) - [\text{Cost}_P(r)]| \geq \epsilon[\text{Cost}_P(r)]] \leq$$
$$\Pr\left[|\widehat{\text{Cost}}_P(r) - [\text{Cost}_P(r)]| \geq \epsilon\widehat{\sigma}\sqrt{\frac{|\widehat{\mathcal{I}}|}{\lambda}}\right] \leq \frac{\lambda}{\epsilon^2|\widehat{\mathcal{I}}|},$$

where the second inequality follows by an application of Chebyshev's Inequality. But recall that we chose $|\widehat{\mathcal{I}}| = \lambda/\delta\epsilon^2$, and hence $\lambda/\epsilon^2|\widehat{\mathcal{I}}|$ is precisely δ as desired.

Now that we have shown that the theorem holds given inequalities (2.4.5) and (2.4.6), we now prove that these inequalities do in fact hold.

Recall that $\text{Total}_P(r)$ is the least possible cost to reserve r hotel rooms. Since one way of reserving n rooms is to take the optimal set of $r-1$ rooms and add an arbitrary room to it, we can see that

$$\text{Total}_P(r) \leq \text{Total}_P(r-1) + \text{Price}_P^{\max}.$$

Similarly, since we can form a set of $r-1$ rooms by taking the optimal set of r rooms and just removing one, we get that

$$\text{Total}_P(r-1) \leq \text{Total}_P(r) - \text{Price}_P^{\min}.$$

Recalling that $\text{Cost}_P(r) = \text{Total}_P(r) - \text{Total}_P(r-1)$, we see that by rearranging terms in the above inequalities we may conclude that

$$\text{Price}_P^{\min} \le \text{Cost}_P(r) \le \text{Price}_P^{\max},$$

which is precisely inequality (2.4.6). We can derive inequality (2.4.5) as

$$\begin{aligned} Var(\text{Total}_P(r)) &\le (\text{Price}_P^{\max} - [\text{Cost}_P(r)]) \\ &\quad ([\text{Cost}_P(r)] - \text{Price}_P^{\min}) \\ &\le \text{Price}_P^{\max}[\text{Cost}_P(r)], \end{aligned}$$

where the first inequality follows by the Bhatia-Davis Inequality and the second by observing that $\text{Cost}_P(r)$ and $\text{Price}_P^{\min}$ are always nonnegative. □

Theorem 2.7 *Let ϵ and δ be arbitrary small positive numbers and N_P be the number of hotel packs. Assume every pack contains a constant number of hotels. Using $4N_P(N^{\max})^3\lambda/\epsilon^2\delta$ samples of scenarios we can generate a set of offers which produce revenue that approximates the optimal revenue to an additive factor of $\epsilon\,\text{Price}_P^{\max}$ with probability $1-\delta$.*

Proof: We apply Theorem 2.6 using $4N_P(N^{\max})^3\lambda/\epsilon^2\delta$ samples of scenarios. With probability $1 - \frac{\delta}{N^{\max}N_P}$ we can compute $[\text{Cost}_P(r)]$ with relative error of $\frac{\epsilon}{2N^{\max}}$. For all $r \le N^{\max}$ and all packs P we will compute $[\text{Cost}_P(r)]$ using the same set of samples. Let $\widehat{E}_P(r)$ be the resulting estimate of $[\text{Cost}_P(r)]$. For all $r \le N^{\max}$ and all packs P, we have $\Pr[|\widehat{E}_P(r) - [\text{Cost}_P(r)]| \ge \frac{\epsilon[\text{Cost}_P(r)]}{2N^{\max}}] \le \frac{\delta}{N^{\max}N_P}$, so by union bound we have

$$\Pr[\exists_{P,r\leq N^{\max}}\text{s.t.}|\widehat{E}_P(r)-[\text{Cost}_P(r)]|\geq\frac{\epsilon[\text{Cost}_P(r)]}{2N^{\max}}]\leq\delta$$

Consider the result of using the approximations $\widehat{E}_P(r)$ in OnlineDP instead of the true values $[\text{Cost}_P(r)]$. Let $\widehat{V}_P(r,i)$ and $\widetilde{\text{Offer}}_P(r,i)$ be the computed future profits and optimal offers, respectively.

Since we did not use the exact values of $\widehat{E}_P(r)$ in OnlineDP, Theorem 2.5 gives us no guarantee on the profit from offering prices $\widetilde{\text{Offer}}_P(r,i)$. As we shall now show, however, the resulting profit is approximately optimal. Let $\widetilde{V}_P(r,i)$ be the expected value we achieve by offering price $\widetilde{\text{Offer}}_P(r,i)$. Later we show that when all $\widehat{E}_P(r)$ have relative error of $\frac{\epsilon}{2N^{\max}}$, we have $\widetilde{V}_P(0,0)\geq\widehat{V}_P(0,0)-q_P\frac{\epsilon\text{Price}_P^{\max}}{2}$ and $\widehat{V}_P(0,0)\geq V_P(0,0)-q_P\frac{\epsilon\text{Price}_P^{\max}}{2}$. Combining this gives $\widetilde{V}_P(0,0)\geq V_P(0,0)-q_P\epsilon\text{Price}_P^{\max}$, and so we have

$$\begin{aligned}\sum_{\forall_P}\widetilde{V}_P(0,0)&\geq\sum_{\forall_P}(V_P(0,0)-q_P\epsilon\text{Price}_P^{\max})\\&=\sum_{\forall_P}V_P(0,0)-\sum_{\forall_P}q_P\epsilon\text{Price}_P^{\max}\\&=\sum_{\forall_P}V_P(0,0)-\epsilon\text{Price}^{\max}.\end{aligned}$$

The above inequality says that by offering prices $\widetilde{\text{Offer}}_P(r,i)$, we achieve expected profit of at least that of the best online algorithm minus $\epsilon\text{Price}^{\max}$. It is sufficient to show when all $\widehat{E}_P(r)$ have relative error of $\frac{\epsilon}{2N^{\max}}$, we have

$$\widehat{V}_P(0,0)-q_P\frac{\epsilon\text{Price}_P^{\max}}{2}\leq\widetilde{V}_P(0,0)\tag{2.4.7}$$

and

$$V_P(0,0) - q_P \frac{\epsilon \text{Price}_P^{\max}}{2} \leq \widehat{V}_P(0,0). \tag{2.4.8}$$

Recall from Inequality 2.4.6 that $\text{Cost}_P(r) \leq \text{Price}_P^{\max}$. Thus, we have that

$$|\widehat{E}_P(r) - [\text{Cost}_P(r)]| \leq \frac{\epsilon}{2N^{\max}}[\text{Cost}_P(r)] \leq \frac{\epsilon\, \text{Price}_P^{\max}}{2N^{\max}},$$

meaning that each $\widehat{E}_P(r)$ has additive error of at most $\frac{\epsilon\, \text{Price}_P^{\max}}{2N^{\max}}$. In the two following lemmas we prove Inequalities 2.4.7.

Lemma 2.6 *When each $\widehat{E}_P(r)$ has additive error of $\frac{\epsilon\, \text{Price}_P^{\max}}{2N^{\max}}$ we have $V_P(r,i) - q_P \frac{(N^{\max}-i)\epsilon\, \text{Price}_P^{\max}}{2N^{\max}} \leq \widehat{V}_P(r,i)$.*

Proof: We prove this lemma by induction on $N^{\max} - i$. The base case occurs when $i = N^{\max}$, where both $V_P(r, N^{\max})$ and $\widehat{V}_P(r, N^{\max})$ are zero. For readability let, $\epsilon' = q_P \frac{\epsilon\, \text{Price}_P^{\max}}{2N^{\max}}$, $j = (N^{\max} - i - 1)$, and let $P_{i+1|i}$ be equal to $\Pr[N > i | N \geq i]$. To prove the induction step, we begin with

$$\begin{aligned}\widehat{V}_P(r,i) = P_{i+1|i}((1-q_P)\widehat{V}_P(r,i+1) \\ +q_P \max_s(g_P(s)(s - \widehat{E}_P(r+1) + \widehat{V}_P(r+1,i+1)) \\ +(1-g_P(s))\widehat{V}_P(r,i+1))).\end{aligned}$$

Now, the induction hypothesis gives us that for any r, $V_P(r,i+1) - j\epsilon' \leq \widehat{V}_P(r,i+1)$. Substituting this into the above equality and using the fact that $\widehat{E}_P(r+1) \geq E_P(r+$

$1) - \frac{\epsilon'}{q_P}$ we have

$$\begin{aligned}\widehat{V}_P(r,i) \geq & P_{i+1|i}((1-q_P)(V_P(r,i+1) - j\epsilon') \\ & + q_P \max_s(g_P(s)(s - E_P(r+1) - \frac{\epsilon'}{q_P} + V_P(r+1,i+1) - j\epsilon') \\ & \quad + (1-g_P(s))(V_P(r,i+1) - j\epsilon'))).\end{aligned}$$

By separating out the terms which contains ϵ' and using the fact that for any positive function f and h, $\max_s(f(s) - h(s)) \geq \max_s(f(s)) - \max_s(h(s))$ we have

$$\begin{aligned}\widehat{V}_P(r,i) \geq & P_{i+1|i}((1-q_P)V_P(r,i+1) \\ & \quad + q_P \max_s(g_P(s)(s - E_P(r+1) + V_P(r+1,i+1)) \\ & \quad\quad + (1-g_P(s))V_P(r,i+1))) \\ & - P_{i+1|i}((1-q_P)(j\epsilon') \\ & \quad + q_P \max_s(g_P(s)(\frac{\epsilon'}{q_P} + j\epsilon') + (1-g_P(s))j\epsilon')).\end{aligned}$$

Note the first part is precisely $V_P(r,i)$ and the absolute value of the second part is clearly less than $(j+1)\epsilon'$. Hence we have

$$\begin{aligned}\widehat{V}_P(r,i) & \geq V_P(r,i) - (j+1)\epsilon' \\ & = V_P(r,i) - q_P \frac{(N^{\max} - i)\epsilon \, \mathrm{Price}_P^{\max}}{2N^{\max}},\end{aligned}$$

as claimed. □

Lemma 2.7 *When each* $\widehat{E}_P(r)$ *has additive error of at most* $\frac{\epsilon \, \mathrm{Price}_P^{\max}}{2N^{\max}}$*, we have* $\widehat{V}_P(r,i) - q_P \frac{(N^{\max}-i)\epsilon \, \mathrm{Price}_P^{\max}}{2N^{\max}} \leq \widetilde{V}_P(r,i)$.

Proof: We prove this lemma by induction on $N^{\max} - i$. The base case occurs when $i = N^{\max}$, where both $\widehat{V}_P(r, N^{\max})$ and $\widetilde{V}_P(r, N^{\max})$ are zero. For more readability let $\epsilon' = q_P \frac{\epsilon \, \text{Price}_P^{\max}}{2N^{\max}}$, $j = (N^{\max} - i - 1)$, and let $P_{i+1|i}$ be equal to $\Pr[N > i | N \geq i]$. Let s realize the maximum

$$\max_s (g_P(s)(s - \widehat{E}_P(r+1) + \widehat{V}_P(r+1, i+1)) \\ +(1 - g_P(s))\widehat{V}_P(r, i+1)).$$

For the induction step we begin with

$$\widetilde{V}_P(r,i) = P_{i+1|i}((1 - q_P)\widetilde{V}_P(r, i+1) \\ +q_P(g_P(s)(s - E_P(r+1) + \widetilde{V}_P(r+1, i+1)) \\ + (1 - g_P(s))\widetilde{V}_P(r, i+1))).$$

By the induction hypothesis, we have that for any r, $\widehat{V}_P(r, i+1) - j\epsilon' \leq \widetilde{V}_P(r, i+1)$. Substituting this into the above equality yields

$$\widetilde{V}_P(r,i) \geq P_{i+1|i}((1 - q_P)(\widehat{V}_P(r, i+1) - j\epsilon') \\ +q_P(g_P(s)(s - \widehat{E}_P(r+1) - \frac{\epsilon'}{q_P} + \widehat{V}_P(r+1, i+1) - j\epsilon') \\ +(1 - g_P(s))(\widehat{V}_P(r, i+1) - j\epsilon'))).$$

Collecting the terms involving ϵ' gives us

$$\begin{aligned}\widetilde{V}_P(r,i) \geq P_{i+1|i}((1-q_P)\widehat{V}_P(r,i+1) \\ +q_P(g_P(s)(s-\widehat{E}_P(r+1)+\widehat{V}_P(r+1,i+1)) \\ +(1-g_P(s))\widehat{V}_P(r,i+1))) \\ -P_{i+1|i}((1-q_P)(j\epsilon') \\ +q_P(g_P(s)(\epsilon'/q_P+j\epsilon')+(1-g_P(s))j\epsilon')).\end{aligned}$$

Now, by the definition of s we may conclude that the first part above is precisely $\widehat{V}_P(r,i)$. Since the absolute value of the second part is clearly less than $(j+1)\epsilon'$, we conclude that

$$\begin{aligned}\widetilde{V}_P(r,i) &\geq \widehat{V}_P(r,i)-(j+1)\epsilon' \\ &= \widehat{V}_P(r,i)-q_P\frac{(N^{\max}-i)\epsilon\,\mathrm{Price}_P^{\max}}{2N^{\max}},\end{aligned}$$

exactly as desired. □ □

2.5 Conclusion

We studied the problem faced by a reseller of options which are prone to price fluctuations to risk-averse customers. The reseller wants to make a profit by having access to information such as probability distribution of future option prices and spreading the risk over his many customers. Two variants of this problem were investigated in this chapter. In the first variant, buyers could form their own option packs and inform us about their budget. The arbitrary overlap between customers chosen packs in this case gives the problem a matching aspect, and it aligns most closely with the two stage stochastic matching literature. In this case, we gave an algorithm which helps the coordinator choose the most profitable set of customers. This profit is determined by taking expectation over all possible future price scenarios. For this model, we defined a profit function and showed it is submodular, and thus, it can be approximated using the approximation algorithms for submodular functions. We also defined a regret function and showed we can minimize it within an approximation factor if we solve the problem for a sample of future scenarios. Minimizing the regret objective function would was then modeled by an Integer Program which we showed has a big integrality gap and therefore, cannot be solved or approximated efficiently. We then defined a bicriteria style approximation algorithm and provided an algorithm which approximates the minimum regret value possible by at most a factor of $\frac{1}{f}$, while ensuring that no option is matched to more than 2 buyers and at most a $min\{\frac{f}{1-2f}, 0.5\}$ of buyers cannot be uniquely matched to an option, for any $0 < f < 0.5$. The future work in this setting is to improve these approximation factors and to provide hardness results to show the approximation factors cannot be improved any further and are tight.

In the second variant, packs of options are chosen by the coordinator and are always disjoint, and so we may solve the problem independently for each pack. We consider this variant of our problem in an online setting (the customers arrive one by one) , where it resembles online selection problems such as secretary problems and the prophet inequalities but presents additional challenges. In this setting, we assume the coordinator only knows the distribution of customers' budget and should provide the customer with a take it or leave it price immediately. The customer then might reject or accept the offer based on her actual budget. This version of our problem, though easier, provided a more realistic model for many online services, and we provided an online algorithm which decides price offers to the customers such that the maximum profit is approximated within a factor close to one with high probability.

Chapter 3: Edge-Weighted Oblivious and Online Matching

3.1 Introduction

The primal-dual LP framework has been vastly used by many researchers to show approximation factors of algorithms designed for many different versions of matching problem including *online bipartite matching* and *online budgeted allocation problem* [29, 30]. In this framework, the algorithm builds both a primal and a dual solution during each run. To analyze the approximation ratio, the value of the primal solution returned by the algorithm is compared with that of the dual solution. When the primal LP is a maximization problem, any feasible dual value provides an upper bound on the optimal primal value and can guarantee some approximation ratio. Hence, it is crucial in this framework to establish the feasibility of the dual solution returned by the algorithm or even its feasibility when scaled by a constant factor. In most matching problems, dual feasibility requires sum of the dual values of every edge's end-points to be large enough.

We observe that this strict requirement of dual feasibility is an artifact of the approximation analysis, and instead explore a new analysis method in which dual feasibility can be bypassed. We call this new framework *Matching Coverage* and show how our algorithms when analyzed by this framework improve the previous approximation factors for two different variations of edge-weighted maximum matching problem. We use a *vector* to mean an assignment of non-negative values to each node. A vector $\vec{\alpha}$ is a *matching coverage* for a matching M if for each edge in M, the sum of the values of its incident nodes in $\vec{\alpha}$ is at least the edge weight; in other words, if the vector satisfies only the dual constraints corresponding to the edges in matching M. In our new analysis, alongside the execution of the algorithm, we assume that a matching coverage vector is constructed with the knowledge of the optimal solution. This is a major departure from the conventional primal-dual framework in which a feasible dual solution (satisfying every single constraint) is returned by the algorithm. We also compare the primal solution returned by the algorithm with this matching coverage vector. Since every dual constraint corresponding to every edge in the primal solution (a matching) returned by the algorithm is satisfied, the sum of the vector's values on each matching's edge is greater than the weight of that edge and thus, the total sum of the values in the vector gives an upper bound on the optimal weight of a primal solution.

Since we no longer need to return a feasible dual solution, we gain more flexibility in the design and the analysis of our algorithms. In particular, we apply this matching coverage framework to two well-known problems: (1) edge-weighted oblivious matching on general graphs [31, 32], (2) edge-weighted online bipartite matching with free disposal [33].

3.1.1 Our Results

We propose a new matching coverage framework for analyzing algorithms on the following graph matching problems.

(1) Edge-Weighted Oblivious Matching on General Graphs. An undirected graph $G(V, E)$ is fixed by the adversary in advance; moreover, each pair $e \in \binom{V}{2}$ of nodes has some weight w_e. The node set V and the weights of all edges are revealed to the algorithm as its input, whereas the edge set E is unknown initially. The (possibly randomized) algorithm returns an ordering on all $\binom{V}{2}$ pairs for probing. The edges are probed one by one according to this order. Upon probing the edge $e = \{u, v\}$, if both nodes are currently unmatched and an edge is revealed between them once probed, the end-points of the edge, u and v are matched to each other; otherwise, we skip to the next pair, until all pairs in the list are probed. The goal is to maximize the *competitive ratio*, which is the ratio between the expected sum of weights of edges in the matching produced by the algorithm and the maximum weight of an optimal matching in the graph. The *greedy* algorithm lists the edges in non-increasing weight order, and can be shown to achieve a ratio of $\frac{1}{2}$.

The unweighted version of the problem has applications in the Kidney Exchange Problem [34], in which donor-recipient pairs are probed and greedily matched when two pairs are compatible. The weighted version of the problem has applications in pay-per-click online advertisements, in which the revenue for a click on a particular ad showing on a particular page is known, but it is unknown whether the user will actually click on that ad.

Adaptive Algorithms. An algorithm could in general be *adaptive*, i.e., after seeing the result of a probe, the algorithm can change the order of the pairs in the remaining list. However, we only consider non-adaptive algorithms in this work.

Number of Distinct Weights. The unweighted oblivious matching problem is a special case when all pairs have the same weight. In this case, the work of Aronson et al. [31] and Chan et al. [32] states that there exists $\xi_1 > 0$ such that there is an algorithm with a competitive ratio of $\frac{1}{2} + \xi_1$. Observe that in [31], it was shown that $\xi_1 \geq \frac{1}{400000}$, and since then there have been some attempts [35, 36] (which we discuss below) to improve the ratio. However, almost twenty years have passed before the improvement $\xi_1 \geq 0.023$ was proved recently [32] by considering the Rankingalgorithm, which is easy to describe. A permutation on V is picked uniformly at random, and this induces a lexicographical order on the node pairs that is used for probing. No results on the edge-weighted version of the problem are known previously even for two distinct weights. We extend the result to the case when the number of distinct weights is bounded.

Theorem 3.1 (Edge-Weighted OM with Bounded Number of Distinct Weights) *Suppose there is an algorithm on unweighted OM with competitive ratio $\frac{1}{2}+\xi_1$. Then, for each positive integer $k > 1$, there exists $\xi_k = \Omega(\xi_1)^{O(k^2)}$ such that the following holds. There exists an algorithm on the edge-weighted oblivious matching problem such that on instances with k distinct edge weights, the competitive ratio is $\frac{1}{2} + \xi_k$.*

(2) Edge-weighted Online Bipartite Matching (OBM) with Free Disposal. Suppose V is the set of *offline* nodes, each of which has capacity 1, i.e., it can be matched to at most one online node. The adversary fixes a bipartite graph between a set U of *online* nodes and V, and the weights of edges between U and V. The adversary determines the order of arrival for the online nodes. When an online node u arrives, all the weights w_{uv}'s of edges between u and the offline nodes v in V are revealed to the (possibly randomized) algorithm. The algorithm matches u to one of the offline nodes v. Even if an offline node v is already matched to a previous online node u', the algorithm is allowed to dispose of the edge $\{u', v\}$ and include the edge $\{u, v\}$ in the matching, provided that $w_{uv} > w_{u'v}$, where the quantity $w_{uv} - w_{u'v}$ is known as the *benefit* of edge $\{u, v\}$. The goal is to maximize the competitive ratio, which is the expected sum of weights of edges in the final matching to that of a maximum weight matching in hindsight. We show (in Section 3.5) that without the *free disposal* assumption, no randomized algorithm can achieve any non-trivial constant guarantee on the ratio. Hence, when we refer to the edge-weighted version of OBM, unless otherwise stated, we implicitly assume that we have free disposal. The greedy algorithm matches an online node u to an offline node v with the largest benefit, and can be shown to achieve a ratio of $\frac{1}{2}$.

Degree of Online Nodes. Despite previous research on OBM, it is unknown whether there is an algorithm with ratio strictly larger than $\frac{1}{2}$ on the edge-weighted version with free disposal, for the hard instances where each offline node has capacity 1. We design an algorithm which beats the ratio $\frac{1}{2}$ when each online node has bounded degree, i.e., each online node is incident to a bounded number of edges with positive weights. Observe that we do not place any degree constraints on the offline nodes.

Theorem 3.2 (Edge-weighted OBM with Bounded Online Degree) *There exists an algorithm for edge-weighted online bipartite matching with free disposal such that on instances in which every online node has degree at most Δ, the competitive ratio is $\frac{1}{2} + \Omega(\frac{1}{\Delta^2})$.*

Our Techniques. Our matching algorithms for both problems are analyzed using the matching coverage framework. In the analysis, we assume the knowledge of an optimal matching, and construct a (random) vector alongside the execution of the (randomized) algorithm to ensure that the vector is always a matching coverage of the optimal matching. The technical part is to show that the (expected) sum of assigned values by the vector over all nodes is small compared to the (expected) weight of the matching produced by the algorithm. We next outline the techniques specific to each problem.

Edge-weighted OM. The idea is to group pairs of similar weights in batches, and run each batch using an algorithm on unweighted OM with competitive ratio $\frac{1}{2} + \xi_1$, for some $\xi_1 > 0$. For instance, it is proved in [32] that Rankinghas $\xi_1 \geq 0.023$.

Consider the simple case when there are only two batches contained in the corresponding intervals $I_2 = [a_2, b_2]$ and $I_1 = [a_1, b_1]$, where $b_2 \leq a_1$. If $\frac{a_1}{b_1}$ is close to 1, then running Rankingon edges with weights in I_1 should produce a matching M_1 with ratio (with respect to the optimal matching using edges with weights in I_1)

close to $\frac{1}{2} + \xi_1$. However, the matching M_1 might eliminate some edges from the optimal matching with weights in I_2 or smaller. When $\frac{b_2}{b_1}$ is small, we can argue that this damage is small. Using the framework of matching coverage, we can express the competitive ratio in terms of the quantities $\mu_1 = \frac{a_1}{b_1}$ and $\mu_2 = \frac{b_2}{b_1}$. If the number of distinct weights are bounded, we show that it is possible to group the edges such that there is some constant gap between the μ_1 and μ_2, which is required to obtain a competitive ratio strictly larger than $\frac{1}{2}$.

Edge-weighted OBM. The idea is to modify the greedy algorithm such that, when an online node u arrives, the incident edges are grouped into two sets whose benefits are in intervals $I_2 = [a_2, b_2]$ and $I_1 = [a_1, b_1]$, where $b_2 \leq a_1$. Depending on the degree d_u of u, the intervals I_1 and I_2 can be chosen such that $\frac{a_1}{b_1}$ is at least some threshold, and at the same time $\frac{b_2}{a_1}$ is smaller than another threshold, where both thresholds depend on d_u. The edges with benefits in I_1 are known as *active* edges. The algorithm picks an active edge uniformly at random. Using the framework of matching coverage, we can express the competitive ratio in terms of the maximum online degree.

As discussed below, there are lots of related work on similar problems, but not much progress has been made so far for the weighted versions of the two particular problems that we consider. We believe that our result is a sensible progress towards these very difficult problems by considering bounds on the number of distinct weights or the degree (which are both reasonable assumptions in practice). Our main technical contribution is the introduction of the novel matching coverage technique, which is important in its own right and we believe will have potential applications for other problems as well.

3.1.2 Related Work

Online Bipartite Matching. The bipartite matching problem is a classical problem in computer science. Karp, Vazirani, and Vazirani introduced the (unweighted) online bipartite matching problem in their seminal work [37] (STOC 1990). They used a complicated analysis to show that the Rankingalgorithm (for OBM which also involves a random permutation) has optimal competitive ratio $1-\frac{1}{e}$. Subsequent works by Goel and Mehta [38] (SODA 2008), and Birnbaum and Mathieu [39] (SIGACT News 2008) simplified the proof. Aggarwal, Goel, Karande, and Mehta [40] (SODA 2011) generalized the Rankingalgorithm when each offline node has a weight, and showed that its competitive ratio is $1 - \frac{1}{e}$ for the corresponding node-weighted version of the problem. Generalizing online biparite matching, Mehta, Saberi, Vazirani, and Vazirani [41] (FOCS 2005) introduced the online budgeted allocation problem to model sponsored search auctions, and proposed a $1 - \frac{1}{e}$-competitive algorithm when the bid-to-budget ratio tends to zero.

Primal-Dual Framework. Using the primal-dual approach, Buchbinder, Jain, and Naor [29] (ESA 2007) designed a deterministic online algorithm for the aforementioned allocation problem with the same competitive ratio, where a feasible dual solution is also returned by the algorithm.

Devanur, Jain, and Kleinberg [30] (SODA 2013) considered a randomized primal-dual approach for matching problems. In particular, they re-analyzed the Rankingalgorithm [40] for the online bipartite matching problem with weighted offline nodes. The algorithm is randomized and can be augmented to return a (random) dual solution, whose expectation is feasible. Moreover, it is interesting that this randomized primal-dual framework can be adapted to analyze the deterministic algorithm [29] for the online budgeted allocation problem.

Jain et al. [42] (JACM 2003) analyzed greedy facility location algorithms using *dual-fitting* with factor-revealing LP. Although the dual solution involved is not feasible initially, it can be *fitted* by scaling with some appropriate factor to achieve feasibility. In contrast, our matching coverage approach ignores some dual constraints and does not attempt to achieve any kind of dual feasibility.

Oblivious Matching. In the mid-1990s, Dyer and Frieze first considered whether a randomized greedy approach can be used to solve the maximum matching problem, although they did not use the same name for the problem. They showed that returning a permutation of unordered pairs uniformly at random cannot achieve a constant ratio strictly larger than $\frac{1}{2}$ [43] (Random Struct. Algorithms 1991). Aronson et al. [31] (Random Struct. Algorithms 1995) achieved the first non-trivial ratio by showing that the modified randomized greedy (MRG) algorithm has performance ratio at least $\frac{1}{2} + \xi$, where $\xi = \frac{1}{400{,}000}$. Even though the ratio is only slightly above 0.5, the proof used very sophisticated combinatorial arguments.

While the problem on general graphs seems difficult, there are some results on the special case of bipartite graphs. The Rankingalgorithm by Karp et al. [37] for online bipartite matching can be readily adapted for oblivious matching problem on bipartite graphs to give the same ratio.

Since running Rankingfor OM on bipartite graphs is equivalent to running Rankingfor OBM with random arrival order, the result by Karande, Mehta, and Tripathi [44] (STOC 2011) implies that Rankingachieves a ratio of 0.653 for OM on bipartite graphs, and is later improved to 0.696 by Mahdian and Yan [45] (STOC 2011) using the technique of strongly factor-revealing LP.

Since the first result by Aronson, Dyer, Frieze and Suen [31], no attempts at improving the $\frac{1}{2} + \xi$ ratio for general graphs have been made until two papers were published in FOCS 2012. Goel and Tripathi [36] claimed a ratio of 0.56, but later announced the withdrawal of the paper on arXiv [46] due to a bug in their proof. Poloczek and Szegedy [35] claimed a ratio of 0.5039, but, according to personal communication with the authors, they are currently bridging some gaps in their proof at the time of writing.

Chan, Chen, Wu and Zhao [32] (SODA 2014) analyzed the Rankingalgorithm on general graphs. They also employed an LP framework to give a bound on the performance ratio. To analyze the limiting behavior of the LP as the number of nodes grows, they developed new primal-dual and complementary slackness characterizations for continuous LP and obtained the currently best theoretical ratio of 0.523.

Edge-weighted Online Bipartite Matching. Feldman, Korula, Mirrokni, Muthukrishnan, and Pál [33] (WINE 2009) proposed the free disposal feature for edge-weighted online bipartite matching. They considered the setting in which each offline node v

has capacity $n(v)$, and an online algorithm benefits from the $n(v)$ highest-weighted edges matched to v. They proposed an $1 - \frac{1}{e_k}$-competitive online algorithm, where $e_k = (1 + \frac{1}{k})^k$, and k is a lower bound on capacities. Thus, the proposed algorithm has competitive ratio $\frac{1}{2}$ for the classic weighted version, when all capacities are 1.

Other Stochastic Settings. There are several attempts to study the online matching problem in a stochastic setting. In the *known distribution model*, the online algorithm knows the online vertices are drawn i.i.d. from a known distribution. Feldman, Mehta, Mirrokni, and Muthukrishnan [47] (FOCS 200) first studied the online unweighted matching problem in the known distribution model, and proposed a 0.67-competitive online algorithm which beats $1 - \frac{1}{e}$. Bahmani and Kapralov [48] (ESA 2010) and Manshadi, Oveis-Gharan, and Saberi [49] (SODA 2011) improved competitive ratio to 0.699 and 0.702 respectively. Haeupler, Mirrokni, and Zadimoghaddam [50] (WINE 2011) considered the weighted version in the same input model and proposed an online algorithm with competitive ratio 0.667. They computed multiple offline solutions and used them as a guideline for the online solution. The authors applied the same technique for designing a 0.7036-competitive online algorithm for the unweighted bipartite graph. A stricter input model is the *unknown distribution model*. In this model, online vertices are drawn i.i.d. from an unknown distribution, i.e., the algorithm has no information about the distribution. Karande, Mehta, and Tripathi [44] (STOC 2011) analyzed the Ranking algorithm and showed it has competitive ratio 0.653 for the unweighted version in the unknown distribution model.

3.2 Preliminaries: Matching Coverage as an Analysis Technique

Given an undirected graph $G = (V, E)$ with non-negative edge weights, we recall the standard maximum weight matching LP relaxation, together with its dual (also known as vertex cover), as follows.

$$\begin{aligned} \max \quad & w(x) := \sum_{\{u,v\}\in E} w_{uv} x_{uv} && (3.2.1)\\ \text{s.t} \quad & \sum_{u:\{u,v\}\in E} x_{uv} \le 1 && \forall v \in V\\ & x_{uv} \ge 0 && \forall \{u,v\} \in E \end{aligned}$$

$$\begin{aligned} \min \quad & C(\alpha) := \sum_{u\in V} \alpha_u && (3.2.2)\\ \text{s.t} \quad & \alpha_v + \alpha_u \ge w_{uv} && \forall \{u,v\} \in E\\ & \alpha_v \ge 0 && \forall v \in V \end{aligned}$$

If x is an integral primal feasible solution, then x corresponds to some matching M, and we write $w(M) := w(x)$. When G is a bipartite graph between U and V, we use α_u for the variables corresponding to the nodes in U and β_v for those corresponding to the nodes in V.

Definition 3.1 *Let M be a matching in graph G. A vector $\alpha \in^V$ is a* matching coverage *for matching M if α is non-negative, and the dual constraints of LP 3.2.2 corresponding to the edges of M are satisfied. In other words, for each $\{u, v\} \in M$, $\alpha_u + \alpha_v \ge w_{uv}$.*

The following lemma is a trivial consequence of the fact that any two distinct edges in a matching do not share any node.

Lemma 3.1 *If a vector α is a matching coverage for a matching M, then $C(\alpha) \ge w(M)$.*

General Framework of Matching Coverage. We describe this general framework for analyzing a matching algorithm. A typical primal-dual algorithm implicitly constructs a feasible primal-dual pair of solutions, and the approximation analysis compares the pair of solutions. In our new analysis framework, the algorithm does not actually return any dual solution (not even an infeasible one). In the analysis, we imagine that as an algorithm ALG is executed, a vector α is constructed alongside with the knowledge of an optimal matching M^*. The idea is that the values in α are increased just enough to make sure that α is a matching coverage for M^*.

Why does this help the analysis? Since the vector α is a matching coverage for M^*, by Lemma 3.1, we have $w(M^*) \leq C(\alpha)$. As α does not have to be feasible for all edge constraints, it is possible that the resulting value $C(\alpha)$ could be smaller than that of a feasible dual. Therefore, we can hope to get a smaller value of B when we compare $C(\alpha) \leq B \cdot w(M_{\text{ALG}})$ with the weight of the matching M_{ALG} returned by ALG, thereby getting a larger competitive ratio $w(M_{\text{ALG}}) \geq \frac{1}{B} \cdot C(\alpha) \geq \frac{1}{B} \cdot w(M^*)$.

It is known that the minimum solution to the dual LP can be as large as 1.5 times the maximum solution to the matching problem. For example, any maximum matching of a complete graph with three vertices has just one edge. Although, the minimum vertex cover of this graph has the cost 1.5. This ratio compare to the available approximation ratios of matching problems is pretty high. The question raised here is that, do we really need to satisfy all of the constraints in the dual LP to be an upper bound for the matching problem?

Consider, in the dual LP, for each edge (u, v) we have a constraint states $\alpha_v + \alpha_u \geq$

w_{uv}. But, in the matching problem lots of edges do not play any role. In fact, if we just write the dual LP for a subgraph of the actual graph which has the same maximum matching, it is still an upper bound for the maximum matching problem. As a minimal subgraph we can write the dual LP for a graph just consist of the edges in a fixed maximum matching of the actual graph. However, this LP can be a conceptual LP that just helps us to analyze our algorithm better. We call a solution to this LP, a matching coverage.

Definition 3.2 *Let M be a matching in graph G. Vector $C(M) = \overrightarrow{\alpha}$ is a* matching coverage *of matching M if the dual constraints of LP 3.2.2 corresponding to the edges of M are satisfied by this vector. Note there is no need for other constraints to be satisfied. Let the* weight *of $C(M)$ be the total sum of all values in vector $\overrightarrow{\alpha}$, i.e. $w(C(M)) = \sum_{u \in U} \alpha_u$.*

The following lemma shows that size of any matching can be bounded by a matching coverage of the that matching.

Lemma 3.2 *The weight of any matching M is at most that of any matching coverage of M.*

Proof: Let $C(M) = \overrightarrow{\alpha}$ be a matching coverage of M. For each edge $e = (u, v)$ in M we have $w_e \leq \alpha_u + \alpha_v$ due to the corresponding dual constraint. By summing it up over all edges in M we have

$$\sum_{e \in M} w_e \leq \sum_{(u,v) \in M} \alpha_u + \alpha_v.$$

The left-hand side is the weight of matching M. Since edges in M do not share an endpoint, the right-hand side is the sum of α_u over a subset of vertices. Hence, the right-hand side is at most the weight of $C(M)$. $\square$

3.3 Edge-Weighted Oblivious Matching Problem with a Bounded Number of Distinct Weights

We consider the edge-weighted oblivious matching problem where the number of distinct weights is k. We give an algorithm whose competitive ratio is $\frac{1}{2} + \xi_k$, where ξ_k only depends on k. As a subroutine, we use an algorithm $\mathcal{A}^{un}$for the unweighted version of the problem with competitive ratio $\frac{1}{2} + \xi_1$, where $\xi_1 > 0$. For instance, it was proved in [32] that the Rankingalgorithm achieves $\xi_1 \geq 0.023$.

Running $\mathcal{A}^{un}$on a Subset $H \subseteq \binom{V}{2}$. This means that $\mathcal{A}^{un}$is first run to produce a random order L of node pairs. Only pairs in H are kept in L, while pairs not in H are removed. Then, the list L is used for probing as before.

High Level Idea. We partition the pairs in $\binom{V}{2}$ into batches $\{H_i\}_{i \geq 1}$, where the weights of pairs in each batch are similar. Then, starting from the batch with largest weights, we run $\mathcal{A}^{un}$on each batch H_i to produce a list L_i, and return the concatenated list used for probing. An alternative view is that $\mathcal{A}^{un}$is used to probe pairs in a batch with higher weights before those in one with lower weights.

3.3.1 Partitioning $\binom{V}{2}$ into Batches

Given a set W of k distinct weights, the algorithm first returns disjoint intervals $\{I_i\}_{i\geq 1}$ whose union contains W. The intervals will be used to partition $\binom{V}{2}$ into batches such that H_i is the batch containing pairs with weights in interval I_i. We use the convention that an interval with a smaller index i contains larger values, i.e., I_1 is the interval containing the largest weights. The following lemma describes the properties of the intervals picked by the algorithm. Recall that $\mathcal{A}^{un}$has competitive ratio $\frac{1}{2}+\xi_1$ on unweighted OM. Given two real numbers $a \leq b$, we denote $\mathsf{dist}(a, b) := 1 - \frac{a}{b}$.

Lemma 3.3 (Partitioning Weights into Batches) *Given a set W of k distinct weights, there exists an integer $r = O(k^2)$ and $\epsilon = \frac{\xi_1}{2}$ such that the algorithm can return disjoint intervals $\{I_i := [a_i, b_i]\}_{i\geq 1}$, whose union contains W, and for each $i \geq 1$, $\mathsf{dist}(a_i, b_i) \leq \epsilon^r$ and $\mathsf{dist}(b_{i+1}, b_i) \geq \epsilon^{r-1}$.*

Proof: The algorithm first finds the smallest integer $r \geq 1$ such that there are no distinct $x < y$ in W satisfying $\epsilon^r < \mathsf{dist}(x, y) \leq \epsilon^{r-1}$. Observe that there are only $\binom{k}{2}$ such distinct pairs (x, y), and hence by the Pigeon Hole Principle, the algorithm can start testing from $r = 1$ and eventually will find an $r \leq \binom{k}{2} + 1$ with this property.

The algorithm next partition weights from W into batches using the following simple rule: two weights $x < y$ in W are in the same batch *iff* $\mathsf{dist}(x, y) \leq \epsilon^r$. To show that this is well-defined, it suffices to prove the following transitivity property.

Claim. Suppose $x < y < z$ are three distinct weights in W. Then, $\mathsf{dist}(x, y) \leq \epsilon^r$ and $\mathsf{dist}(y, z) \leq \epsilon^r$ implies that $\mathsf{dist}(x, z) \leq \epsilon^r$.

Proof of Claim. Observe that $\mathsf{dist}(x, y) \leq \epsilon^r$ and $\mathsf{dist}(y, z) \leq \epsilon^r$ implies that both $\frac{x}{y}$ and $\frac{y}{z}$ are at least $1 - \epsilon^r$. Hence, it follows that $\frac{x}{z} \geq (1 - \epsilon^r)^2 > 1 - \epsilon^{r-1}$, because $\epsilon = \frac{\xi_1}{2} \leq \frac{1}{2}$. Therefore, $\mathsf{dist}(x, z) < \epsilon^{r-1}$, and by the choice of r, $\mathsf{dist}(x, z)$ cannot be in the interval $(\epsilon^r, \epsilon^{r-1}]$. So, we conclude that $\mathsf{dist}(x, z) < \epsilon^r$. ■

Therefore, each batch of weights defines an interval I_i, where a_i is the minimum weight in the batch and b_i is the maximum one. From the definition of a batch, it follows that $\mathsf{dist}(a_i, b_i) \leq \epsilon^r$.

To show that weights from different batches are far apart, for each $i \geq 1$, since $b_{i+1} < b_i$ are in different batches, it must be the case that $\mathsf{dist}(b_{i+1}, b_i) > \epsilon^r$. By the choice of r, this implies that $\mathsf{dist}(b_{i+1}, b_i) \geq \epsilon^{r-1}$. □

3.3.2 Edge-Weighted OM: Running Unweighted $\mathcal{A}^{un}$ on Each Batch

Given an instance of edge-weighted oblivious matching problem, the algorithm receives a node set V, and the weight w_e for each pair $e \in \binom{V}{2}$. Here is the outline of the algorithm to return a probing list.

Suppose W is the set of weights of pairs. Using the procedure in Lemma 3.3, disjoint intervals are constructed to partition W, which induces a partition of $\binom{V}{2}$ into batches $\{H_i\}_{i\geq 1}^k$, where H_1 is the batch containing pairs with largest weights.

Starting from $i = 1$ to k, the unweighted $\mathcal{A}^{un}$is run (using independent randomness)

Algorithm 2 Algorithm for Edge-Weighted OM

Input: Set of vertices V with w_e for any $e \in \binom{V}{2}$

Output: The probing list L.

```
0: $W \leftarrow \{w_e : e \in \binom{V}{2}\}$
0: $\{I_i := [a_i, b_i]\}_{i=1}^{K} \leftarrow$ Disjoint intervals as given in Lemma 3.3 to partition $W$,
   where $I_1$ is the interval with the largest weights.
0: for $i$ from 1 to $K$ do
0:     $H_i \leftarrow$ Pairs in $\binom{V}{2}$ with weights in $I_i$
0:     $L_i \leftarrow$ List produced by running unweighted $\mathcal{A}^{un}$ on $H_i$ using independent
       randomness
0: end for
0: return concatenated list $L := L_1 \oplus L_2 \oplus \cdots \oplus L_K$ =0
```

on H_i to produce list L_i.

The lists are concatenated to produce a list $L := L_1 \oplus L_2 \oplus \cdots \oplus L_k$ that is used for probing.

Analysis: Assume we are given an optimal matching OPT, we construct a matching coverage $\alpha \in^V$ for OPTduring an execution of the algorithm. For a matching M, we use $|M|$ to denote its cardinality and $w(M)$ to denote the sum of weights of its edges. We say an edge e in $OPT$is *destroyed* by a matching M if edge e is not in M but at least one end-point of e is matched in M. Moreover, two edges *intersect* if they share at least one end-point.

For each $i \geq 1$, we define the following edge sets.

ALG_i is the set of the edges the algorithm includes in the matching when list L_i is probed. Observe that ALG_i is maximal in the sense that every edge in L_i intersects with an edge in ALG_i.

OPT_i is the set of edges in OPT that intersect with edges in ALG_i, but do not intersect with edges in ALG_j, for all $j < i$. Intuitively, those are the edges in OPT that are either selected or first destroyed by ALG_i. Observe that each edge in OPT_i has weight at most b_i.

$OPT_i^H := OPT_i \cap H_i$, each of which has weight in $[a_i, b_i]$; we shall make sure that exactly one end-point of such an edge will receive value b_i in α.

The matching resulting from the probing list L returned by the algorithm is $\mathrm{ALG} := \cup_i \mathrm{ALG}_i$. Since ALG is a maximal matching in G, it follows that every edge in OPT

appears in exactly one OPT_i.

Remark. Observe that conditioning on $\{\mathrm{ALG}_j\}_{j<i}$, OPT_i^H are exactly those edges in $OPT \cap H_i$ that have not been destroyed by $\{\mathrm{ALG}_j\}_{j<i}$. The reason is that each of these edges will either be selected or destroyed by ALG_i. Hence, OPT_i^H is (conditionally) independent of the randomness used in running $\mathcal{A}^{un}$on H_i to produce ALG_i.

Constructing a Matching Coverage. In addition to the optimal matching OPT, the proposed vector α also depends on the randomness of the algorithm. For each $i \geq 1$, we apply the following analysis. Suppose V_i is the set of nodes matched in ALG_i. We define α_v for each $v \in V_i$ such that the resulting α is sufficient to be a matching coverage for OPT_i. We will compare $C(\alpha_{V_i}) := \sum_{v \in V_i} \alpha_v$ with $w(\mathrm{ALG}_i)$ to analyze the competitive ratio, where α_{V_i} is the vector α restricted to coordinates corresponding to V_i.

Recall that every edge in H_i has weight $[a_i, b_i]$, and every edge in OPT_i has weight at most b_i or at most b_{i+1} if it is not in H_i. We first define a subset $\widehat{V_i} \subseteq V_i$ as follows. Since ALG_i is maximal with respect to edges in H_i, each edge in OPT_i^H has at least one end-point in V_i; we arbitrarily pick one such an end-point and include it in $\widehat{V_i}$. Note that $|\widehat{V_i}| = |OPT_i^H|$. For each $v \in V_i$, we set α_v to be b_i if $v \in \widehat{V_i}$, and b_{i+1} otherwise.

Checking Matching Coverage Requirement. For each edge $e \in OPT_i$, if it is in H_i, then one of its end-points will receive value b_i in α, and hence the dual constraint corresponding to e is satisfied. Otherwise, $e \in OPT_i \setminus H_i$, and by definition, e intersects with an edge in ALG_i and has weight at most b_{i+1}; hence, at least one of its end-points will receive value b_{i+1} in α, and the corresponding dual constraint is also satisfied. Combining the analysis over all $i \geq 1$, it follows that the resulting vector α is a matching coverage for OPT.

Lemma 3.4 (Local Competitive Ratio) *Suppose the weights of H_i are in $[a_i, b_i]$, where $\eta :=$ **dist**(a_i, b_i); moreover, let $\lambda :=$ **dist**(b_{i+1}, b_i). Then, $[w(ALG_i)] \geq (1-\eta) \cdot (\frac{1}{2} + \frac{\xi_1 \lambda}{1+2\xi_1}) \cdot [C(\alpha_{V_i})]$.*

Proof: From the construction, $C(\alpha_{V_i}) = (2|ALG_i| - |OPT_i^H|) \cdot b_{i+1} + |OPT_i^H| \cdot b_i$.

Suppose we condition on $\{\mathrm{ALG}_j\}_{j<i}$. Then, OPT_i^H is determined. Observe that since $\mathcal{A}^{un}$has competitive ratio $\frac{1}{2} + \xi_1$, and when $\mathcal{A}^{un}$is run on H_i, all edges in OPT_i^H are still available. Hence, we have $[|ALG_i|] \geq (\frac{1}{2} + \xi_1) \cdot [OPT_i^H]$, where the latter expectation is taken over the randomness in $\{\mathrm{ALG}_j\}_{j<i}$.

Since $\mathsf{dist}(b_{i+1}, b_i) = \lambda$, it follows that $b_{i+1} = (1-\lambda) b_i$. Moreover, every edge in ALG_i has weight at least a_i, and $\mathsf{dist}(a_i, b_i) = \eta$. Hence, we have

$$\begin{aligned}
[C(\alpha_{V_i})] &= 2(1-\lambda) b_i \cdot [|ALG_i|] + \lambda b_i \cdot [|OPT_i^H|] \leq (2(1-\lambda) + \frac{\lambda}{\frac{1}{2}+\xi_1}) \cdot b_i \cdot [|\mathrm{ALG}_i|] \\
&= \left(\frac{1}{2} + \frac{\xi_1 \lambda}{1 + 2\xi_1(1-\lambda)}\right)^{-1} \cdot b_i \cdot [|\mathrm{ALG}_i|] \leq (\frac{1}{2} + \frac{\xi_1 \lambda}{1+2\xi_1})^{-1} \cdot b_i \cdot [|\mathrm{ALG}_i|] \\
&\leq (\frac{1}{2} + \frac{\xi_1 \lambda}{1+2\xi_1})^{-1} \cdot (1-\eta)^{-1} \cdot [w(\mathrm{ALG}_i)],
\end{aligned}$$

as required. □

Finally, we are ready to prove the competitive ratio of the algorithm.

Proof of Theorem 3.1: From Lemma 3.3, it follows that the parameters in Lemma 3.4 satsify $\lambda \geq \epsilon^{r-1}$ and $\eta \leq \epsilon^r$, where $r = O(k^2)$. Observing that $\epsilon = \frac{\xi_1}{2} \leq \frac{1}{4}$, it follows that the local competitive ratio is

$$\begin{aligned} \frac{[w(\text{ALG}_i)]}{[C(\alpha_{V_i})]} &\geq (1-\eta)\cdot(\frac{1}{2}+\frac{\xi_1\lambda}{1+2\xi_1}) \geq (1-\epsilon^r)(\frac{1}{2}+\frac{\xi_1\epsilon^{r-1}}{1+2\xi_1}) = (1-\epsilon^r)(\frac{1}{2}+\frac{2\epsilon^r}{1+2\xi_1}) \\ &= \frac{1}{2}+\frac{\epsilon^r}{2+4\xi_1}\cdot(3-4(\epsilon+\epsilon^r)) \geq \frac{1}{2}+\frac{1}{2+4\xi_1}(\frac{\xi_1}{2})^r, \end{aligned}$$

where the last inequality follows because $r \geq 1$ and $\epsilon \leq \frac{1}{4}$.

Hence, we have $[\text{ALG}] = \sum_i[\text{ALG}_i] \geq (\frac{1}{2}+\frac{1}{2+4\xi_1}(\frac{\xi_1}{2})^r)\sum_i[C(\alpha_{V_i})]$. Finally, observe that $\sum_i[C(\alpha_{V_i})] = E[C(\alpha)] \geq w(OPT)$, as α is a matching coverage for OPT. Therefore, we conclude that the competitive ratio for the whole algorithm is $\frac{1}{2}+\xi_k$, where $\xi_k = \frac{1}{2+4\xi_1}(\frac{\xi_1}{2})^r = \Omega(\xi_1)^{O(k^2)}$, as required. ■

Let $W = \{w_1, w_2, ..w_m\}$ be the set of weights appears in the the graph G where $w_1 > w_2 > \cdots > w_m$. We define the distance between two weight w_i and w_j for $w_i \leq w_j$ as $dis(w_i, w_j) = 1 - \frac{w_i}{w_j}$.For any matching M, we show the number of edges of matching M by $|M|$. Weighted size of a matching M, is $\sum_{e\in M} w(e)$ where $w(e)$ is the weight of edge e. We show the weighted size of the matching M by $w(M)$.

The following is our suggested algorithm for weighted oblivious matching problem. This algorithm uses an algorithm UWAlg for un-weighted oblivious matching problem.

Algorithm 1

Find the smallest integer $x \geq 0$ with the following property.

There is no two weights w_i and w_j such that $\epsilon^x < dis(w_i, w_j) \leq \epsilon^{x-1}$.

replace weight w of each edge with the highest weight in set W which is at most $\frac{w}{1-\epsilon^x}$.

Run algorithm 2 on the new graph.

Algorithm 2

Find a permutation π based on the algorithm UWAlg. First output the edges of weight w_1 according to π and then edges of weight w_2 and so on.

Lemma 3.5 *Assume $\epsilon \leq 0.5$ and let x be the variable selected by Algorithm 1. If $dis(w_i, w_j) \leq \epsilon^x$ then Algorithm 1 replaces w_i and w_j with the same weight.*

Proof: Assume Algorithm 1 replace w_i with w_k and w_j with w_l. Suppose that $w_k \neq w_l$. Without loss of generality, we can assume that $w_k < w_l$. We show that w_l is at most $\frac{w_i}{1-\epsilon^x}$. Assumption $w_k < w_l$ and fact $w_l \leq \frac{w_i}{1-\epsilon^x}$ together contradict the choice of w_k.

Since Algorithm 1 replace w_j with w_l, we have $w_l \leq \frac{w_j}{1-\epsilon^x}$. Also, we have $dis(w_i, w_j) \leq \epsilon^x$ which means $1 - \frac{w_i}{w_j} \leq \epsilon^x$. By putting these two together we have

$$\frac{w_i}{w_l} = \frac{w_i}{w_j}\frac{w_j}{w_l} \geq (1-\epsilon^x)(1-\epsilon^x) = 1 - 2\epsilon^x + (\epsilon^x)^2 \geq 1 - 2\epsilon^x.$$

Using the above inequality and assumption $\epsilon \leq 0.5$, we have

$$dis(w_i, w_l) = 1 - \frac{w_i}{w_l} \leq 1 - (1 - 2\epsilon^x) = 2\epsilon^x \leq \epsilon^{x-1}.$$

However, we know that the distance of non of weight pairs falls in the range $(\epsilon^x, \epsilon^{x-1}]$. Hence, we have $1 - \frac{w_i}{w_l} = dis(w_i, w_l) \leq \epsilon^x$. This is equivalent to $w_l \leq \frac{w_i}{1-\epsilon^x}$ which together with $w_k \leq w_l$ contradict the choice of w_k. $\square$

One immediate conclusion of lemma 3.5 is that Algorithm 1 divides the weights to some bunches. The distance between weights inside a bunch is at most ϵ^x and the distance of any two weights from two different bunches is at least ϵ^{x-1}. Algorithm 1 increase the weight w of each edge to the maximum weight in its bunch. We call this weight representer of weight w.

Lemma 3.6 *Let G be the input graph and let G_r be the graph that Algorithm 1 produce by replacing the weights with their representers. If Algorithm 2 finds an f-approximation solution to the oblivious weighted matching problem for graph G_r, it is a $(1-\epsilon^x)f$-approximation solution for the oblivious weighted matching problem for G.*

Proof: Since we just increase the weight of edges, the size of the maximum matching does not decreased. Hence, in order to bound the approximation ratio, we can compare the size of the resulted matching in graph G with the maximum matching in graph G_r. Let w_e be the weight of edge e in graph G and let w_e^r be the weight of edge e in graph G_r. We know that for any edge e, $dist(w_e, w_e^r) \leq \epsilon^x$ which means $1 - \frac{w_e}{w_e^r} \leq \epsilon^x$. Thus, we have

$$w_e = w_e^r \frac{w_e}{w_e^r} \geq w_e^r(1-\epsilon^x). \tag{3.3.3}$$

Let M_r be an arbitrary matching of graph G_r and let M be the matching consist of edges of M_r in graph G. By summing up the inequality 3.3.3 for all edges in M we have

$$w(M) = \sum_{e\in M} w_e \geq \sum_{e\in M} w_e^r(1-\epsilon^x) = w(M_r)(1-\epsilon^x) \tag{3.3.4}$$

Which means the size of any matching in graph G is at least $1-\epsilon^x$ times the size of the same matching in graph G_r. Let $Alg^2(G_r)$ denote the random weighted matching resulted by applying Algorithm 2 on graph G_r and let $Alg(G)$ be the random weighted matching resulted by applying Algorithm 1 on graph G. By Inequality 3.3.4, we know that $w(Alg(G)) \geq w(Alg^2(G_r))(1-\epsilon^x)$. Thus, we have

$$\begin{aligned}\frac{E[w(Alg(G))]}{Opt(G)} &\geq \frac{E[w(Alg(G))]}{Opt(G_r)} \\ &\geq \frac{E[w(Alg^2(G_r))(1-\epsilon^x)]}{Opt(G_r)} \\ &= (1-\epsilon^x)f\end{aligned}$$

which means $Alg(G)$ is a $(1-\epsilon^x)f$-approximation solution for the oblivious weighted matching problem for G. This completes the proof of the lemma. □

Lemma 3.7 *The value of x in Algorithm 1 is at most $\binom{m}{2}+1$ where m is the number of deferent weights.*

Proof: There is at most $\binom{m}{2}$ different distance of weight pairs and each falls in one range of the form $[\epsilon^y, \epsilon^{y-1})$. Thus, there exist is a $z \leq \binom{m}{2}+1$ such that $[\epsilon^z, \epsilon^{z-1})$ does not contain any distance $dis(w_i, w_j)$. This directly implies that $x \leq \binom{m}{2}+1$. □

Let Opt be a fixed maximum weighted matching of input graph G, and Alg be the random matching resulted by algorithm 2 where the randomness comes from the random decision in algorithm UWAlg. Let Alg_i be the set of edges in Alg with weight w_i. We define Opt_i^{Alg} as the set of edges of Opt with following three properties.

- has weight at most w_i.
- Share an end point with an edge of Alg_i.
- Do not share an end point with an edge of Alg_j where $j < i$.

We define $SubOpt_i^{Alg}$ as the subset of edges in Opt_i^{Alg} which has the weight w_i.

Lemma 3.8 *Each edge of Opt is exactly in one of the sets* Opt_i^{Alg}.

Proof: Consider an edge e of an arbitrary weight w_i in Opt. Algorithm 2 outputs all the edges of weight more than w_i before e which means for all $j < i$, all the edges in Alg_j are matched. Also, Algorithm 2 may outputs some of the edges of weight w_i. This means some of the edges in Alg_i are matched; we call this subset of the edges $PAlg_i$. When Algorithm 2 outputs e one of the following three cases happen.

- Non of the endpoints of e is covered. In this case, e is in Opt_i^{Alg}.
- One of the endpoints of e is in $\bigcup_{\forall j<i} Alg_j$. In this case, for a number $j < i$, e is in Opt_j^{Alg}
- Non of the endpoints of e is in $\bigcup_{\forall j<i} Alg_j$ but at least one of the endpoints of e is in $PAlg_i$. Again, in this case, e is in Opt_i^{Alg}

There for each edge in Opt is exactly in one of the sets Opt_i^{Alg} as desired. □

Theorem 3.3 *Given a* $\frac{1}{2}+\xi$*-approximation Algorithm UWAlg for un-weighted oblivious matching. Algorithm 2 is a* $\frac{1}{2} + \frac{\xi\lambda}{1+2\xi}$*-approximation algorithm for weighted* *oblivious matching where λ is the minimum distance between each two weights in the input graph.*

Proof: In order to prove this theorem for each i we present a matching coverage $C(Opt_i^{Alg})$ with weight $\frac{1+2\xi-2\xi\lambda}{0.5+\xi}w(Alg_i)$. Recall from lemma **??**, each edge of Opt is exactly in one of the sets Opt_i^{Alg}. Also, by definition each edge of Alg is exactly in one Alg_i. Hence, $\bigcup_{\forall i} C(Opt_i^{Alg}) = C(Opt)$ is a matching coverage of Opt with the weight $\sum_{\forall i} \frac{1+2\xi-2\xi\lambda}{0.5+\xi}w(Alg_i) = \frac{1+2\xi-2\xi\lambda}{0.5+\xi}w(Alg)$. Recall lemma 3.2 says $w(C(Opt)) \geq w(Opt)$. By putting these two fact together one can see

$$\begin{aligned}\frac{w(Alg)}{w(Opt)} \geq \frac{w(Alg)}{C(Opt)} &= \frac{w(Alg)}{\frac{1+2\xi-2\xi\lambda}{0.5+\xi}w(Alg)} \\ &= \frac{0.5+\xi}{1+2\xi-2\xi\lambda} = \frac{1}{2} + \frac{\xi\lambda}{1+2\xi-2\xi\lambda} \\ &\geq \frac{1}{2} + \frac{\xi\lambda}{1+2\xi}\end{aligned}$$

That means Algorithm 2 is a $\frac{1}{2} + \frac{\xi\lambda}{1+2\xi}$-approximation algorithm. It remains to present a matching coverage $C(Opt_i^{Alg})$ with weight $\frac{1+2\xi-2\xi\lambda}{0.5+\xi}w(Alg_i)$.

For each edge $(u,v) \in SubOpt_i^{Alg}$, we know (u,v) share an end point with one edge in Alg_i. Assume v is the intersection of (u,v) and an edge in Alg_i (If both are so, select one of them arbitrary). We set $\alpha_v = w_i$. For all the other vertices in Alg_i we set $\alpha_u = w_i(1-\lambda)$. This clearly covers all the edges in $SubOpt_i^{Alg}$ and since any other edge in Opt_i^{Alg} has weight at most $w_{i+1} \leq w_i(1-\lambda)$, it covers all the edges in Opt_i^{Alg}. For $|SubOpt_i^{Alg}|$ number of vertices, we set $\alpha = w_i$ and for rest of them we set $\alpha = w_i(1-\lambda)$. Hence, we have

$$\begin{aligned} w(C(Opt_i^{Alg})) &= |SubOpt_i^{Alg}|w_i + (2|Alg_i| - |SubOpt_i^{Alg}|)w_i(1-\lambda) \\ &= 2|Alg_i|w_i - \lambda(2|Alg_i| - |SubOpt_i^{Alg}|)w_i. \end{aligned} \tag{3.3.5}$$

Consider the Algorithm 2 runs UWAlg on a subgraph that contains the matching $SubOpt_i^{Alg}$. Hence, we have $|Alg_i| \geq (\frac{1}{2}+\xi)|SubOpt_i^{Alg}|$. Putting this together with equality 3.3.5 gives us

$$\begin{aligned} w(C(Opt_i^{Alg})) &= 2|Alg_i|w_i - \lambda(2|Alg_i| - |SubOpt_i^{Alg}|)w_i \\ &\leq 2|Alg_i|w_i - \lambda(2|Alg_i| - \frac{|Alg_i|}{0.5+\xi})w_i \\ &= \frac{|Alg_i| + 2\xi|Alg_i| - 2\xi\lambda|Alg_i|}{0.5+\xi}w_i \\ &= \frac{1+2\xi-2\xi\lambda}{0.5+\xi}w(Alg_i). \end{aligned}$$

This means the weight of cover $C(Opt_i^{Alg})$ is $\frac{1+2\xi-2\xi\lambda}{0.5+\xi}w(Alg_i)$ which complete the proof of the theorem. □

Theorem 3.4 *Given a $\frac{1}{2}+\xi$-approximation Algorithm UWAlg for un-weighted oblivious matching. Algorithm 1 is a $\frac{1}{2}+\frac{1}{2+4\xi}(\frac{\xi}{2})^{\binom{m}{2}+1}$-approximation algorithm for weighted oblivious matching where m is the number of different weights in the input graph.*

Proof: In Algorithm 1 we set $\epsilon = \frac{\xi}{2}$. Using this ϵ, Observation 3.3.2 says that the minimum distance between each two weights in the graph reported to Algorithm 2 is at least $(\frac{\xi}{2})^{x-1}$. By Theorem 3.3 we know that Algorithm 2 finds a $\frac{1}{2}+\frac{\xi(\frac{\xi}{2})^{x-1}}{1+2\xi}$-approximation solution of this graph. At the end, applying Lemma 3.6 says that

Algorithm 1 is a $(1-(\frac{\xi}{2})^{x-1})(\frac{1}{2}+\frac{\xi(\frac{\xi}{2})^{x-1}}{1+2\xi})$-approximation algorithm for weighted oblivious matching. We have

$$\begin{aligned}
(1-(\frac{\xi}{2})^{x})(\frac{1}{2}+\frac{\xi(\frac{\xi}{2})^{x-1}}{1+2\xi}) &= \frac{1}{2}+\frac{\xi-(\frac{\xi}{4}+\frac{\xi^2}{2}+\frac{\xi^{x+1}}{2^x})}{1+2\xi}(\frac{\xi}{2})^{x-1} \qquad \xi \le \frac{1}{2} \text{ and } x \ge 1 \\
&\ge \frac{1}{2}+\frac{\xi}{4+8\xi}(\frac{\xi}{2})^{x-1} \\
&= \frac{1}{2}+\frac{\xi}{4+8\xi}(\frac{\xi}{2})^{x-1} \\
&= \frac{1}{2}+\frac{1}{2+4\xi}(\frac{\xi}{2})^{x}.
\end{aligned}$$

This means Algorithm 1 is a $\frac{1}{2}+\frac{1}{2+4\xi}(\frac{\xi}{2})^{x}$-approximation algorithm for weighted oblivious matching. From Lemma 3.7 we know that $x \le \binom{m}{2}+1$. Therefore, Algorithm 1 is a $\frac{1}{2}+\frac{1}{2+4\xi}(\frac{\xi}{2})^{\binom{m}{2}+1}$-approximation algorithm for the weighted oblivious matching problem. This completes the proof of the theorem. □

3.4 Edge-weighted Online Bipartite Matching with Free Disposal and Bounded Online Degree

As a warmup, we restate a well-known greedy algorithm for the edge-weighted OBM, where V is the set of offline nodes, and the online nodes U arrive in an arbitrary order. Upon arrival of an online node u, consider all its neighbors among offline nodes. The greedy algorithm picks an offline node which maximizes the *benefit* of assigning the new online node, where *benefit* is the difference between the corresponding edge weight and the maximum edge weight previously assigned to this offline node.

The following is a well-known proposition, and its proof can be given by augmenting the algorithm to construct a feasible dual solution (α, β) along the way, where α gives the dual values for the online nodes in U, and β gives the dual values for the offline nodes in V. We will give a brief review and explain how the matching coverage approach can suggest a better algorithm.

Proposition 3.1 *Greedy algorithm has a competitive ratio of* $\frac{1}{2}$.

Algorithm 3 Greedy(Augmented)

```
0: for each offline node $v \in V$ do
0:     $\beta_v \leftarrow 0$
0: end for
0: for all arriving online node $u \in U$ do
0:     Assign $u$ to an offline node $v$ for which $w_{uv} - \beta_v$ is maximized.
0:     $\alpha_u \leftarrow w_{uv} - \beta_v$; $\beta_v \leftarrow \beta_v + (w_{uv} - \beta_v)$
0: end for=0
```

Brief Analysis of Greedy. Observe that at every round when an online node u arrives, the increase in the dual solution is $2(w_{uv} - \beta_v)$, which is twice the increase in the weight of the primal solution. Hence, if we can prove that the resulting dual is feasible, then we can conclude readily that the competitive ratio is $\frac{1}{2}$.

Observe that the β values do not decrease during the execution of the algorithm. Hence, it suffices to show that at the end of every round, the dual (α, β) values satisfy all edge constraints that appear in that round. Consider an edge $(u, v) \in E$. If this edge is picked by the algorithm, then $\beta_v = w_{uv}$, and so the dual is feasible for this edge. Otherwise, it must be the case that β_v is unchanged in this round, and based on the choice of Greedy $\alpha_u \geq w_{uv} - \beta_v$, and therefore, $\alpha_u + \beta_v \geq w_{uv}$.

How can the matching coverage approach bring improvement? In the Greedy algorithm, α_u does not play any essential role, but its value needs to be chosen for the sake of analysis in a way that the dual (α, β) is feasible for every edge incident to u. In the framework of matching coverage, we assume the knowledge of an optimal matching OPT, and we construct a (not necessarily feasible) dual solution (α, β) that is feasible only for the edges in OPT. Moreover, if an edge in OPT is picked by the algorithm, then we do not have to increase α_u at all. Since in this framework we could potentially obtain smaller α values, it is possible that we could improve the competitive ratio.

New Randomized Algorithm. We propose a new algorithm based on Greedy algorithm which takes advantage of randomization. The algorithm works as follows. Let d_u denote the degree of online node u. For each arriving online node u, assume i is a chosen natural number which is less than or equal to d_u. The algorithm considers i neighbors of the online vertex with maximum benefits, and assigns the online node to one of these i offline nodes uniformly at random. Our choice of i depends on the values of benefits as well as the degree of the online node.

Before analyzing the competitive ratio of RandGreedy, we present the intuition behind this algorithm. In each step, Greedy algorithm assigns the arriving online node to an offline node with maximum benefit. However, Greedy is a deterministic algorithm, and it is known that no deterministic algorithm can achieve a competitive

Algorithm 4 RandGreedy

```
0: for each offline node $v \in V$ do
0:     $\beta_v \leftarrow 0$
0: end for
0: for all arriving online node $u \in U$ do
0:     Sort neighbors according to the benefit $b_{uv} = w_{uv} - \beta_v$ in non-increasing order.
0:     Let $v_1$,$v_2$,...,$v_{d_u}$ be the offline nodes in the sorted order.
0:     $S(u) \leftarrow \{i | 1 \leq i \leq d_u - 1$ and $b_{uv_i} \times (1 - \frac{1}{4{d_u}^2}) > b_{uv_{i+1}}\}$
0:     if $S(u)$ is not empty then $i \leftarrow$ min element of $S(u)$
0:     else $i \leftarrow d_u$
0:     end if
0:     Let $\mathsf{Act}(u)$ be the set $\{(u, v_1), \ldots, (u, v_i)\}$ of edges.
0:     $(u, v') \leftarrow$ an edge from $\mathsf{Act}(u)$ chosen uniformly at random.
0:     Assign $u$ to $v'$; $\beta_{v'} \leftarrow w_{uv'}$.
0: end for=0
```

ratio better than $\frac{1}{2}$ in this problem[1]. In order to take advantage of randomization, we consider more than one offline node as potential vertices that the arriving online node in each step can be assigned to. However, we only consider those offline nodes for which the benefit is relatively high.

Matching Coverage. In the description of $\mathsf{RandGreedy}$, the values in β are already decided. Moreover, the sum $\sum_{v \in V} \beta_v$ is exactly the weight of the matching returned by the algorithm. We shall later construct α in the analysis to give a matching coverage for an optimal matching, and show that the expectation $[\sum_{u \in U} \alpha_u]$ is not too large.

Let the set of *active edges* of the arriving online node u be the set of edges which might be assigned to u by $\mathsf{RandGreedy}$. We denote the set of active edges of online vertex u by $\mathsf{Act}(u)$. An edge of u which is not in $\mathsf{Act}(u)$ is called a *non-active edge*. The algorithm chooses an edge from set $\mathsf{Act}(u)$ uniformly at random and assigns u to the vertex on the other end of this edge.

Lemma 3.9 *For any e_1 and e_2 in $\mathsf{Act}(u)$, we have $b_{e_1} \leq (1+\frac{1}{d_u})b_{e_2}$; in other words, their benefits are within a multiplicative factor of $1+\frac{1}{d_u}$ from each other.*

Proof: For convenience, we define $d = d_u$. Let $m = |\mathsf{Act}(u)|$ be the size of $\mathsf{Act}(u)$, and let $\overrightarrow{r} = (r_1, r_2, \ldots, r_m)$ be the sorted benefits of edges in $\mathsf{Act}(u)$ in non-increasing order, i.e., $r_i = b_{uv_i}$. From definition for each $1 \leq i < m$, we have

[1]Note that edge-weighted OBM is a generalization of OBM, and there is no deterministic algorithm for OBM with competitive ratio above $\frac{1}{2}$.

$r_i(1-\frac{1}{4d^2}) \leq r_{i+1}$. Hence, we have $r_1(1-\frac{1}{4d^2})^{m-1} \leq r_m$. Since m is at most d, we replace $m-1$ with d and conclude $r_1 \leq \left(1-\frac{1}{4d^2}\right)^{-d} \cdot r_m \leq (1+\frac{1}{d}) \cdot r_m$, where the last inequality follows because for $d \geq 1$, one can verify that $(1-\frac{1}{4d^2})^{-d} \leq (1+\frac{1}{2d^2})^d \leq e^{\frac{1}{2d}} \leq (1-\frac{1}{2d})^{-1} \leq 1+\frac{1}{d}$. We give the detailed calculation in Lemma 3.10.

This means the maximum benefit among the edges in $\mathsf{Act}(u)$ is at most $1+\frac{1}{d}$ times the minimum benefit among the edges in $\mathsf{Act}(u)$. Thus, we immediately conclude this holds for any two edges in $\mathsf{Act}(u)$. $\square$

The following lemma is used in the calculation in the proof of Lemma 3.9.

Lemma 3.10 *For $d \geq 1$, $(1-\frac{1}{4d^2})^{-d} \leq 1+\frac{1}{d}$.*

Proof: We first show

$$\frac{1}{1-\frac{1}{4d^2}} \leq 1+\frac{1}{2d^2}. \tag{3.4.6}$$

First, we note that

$$\frac{1}{1-\frac{1}{4d^2}} = \frac{1}{1-\frac{1}{4d^2}} \times \frac{1+\frac{1}{2d^2}}{1+\frac{1}{2d^2}} = \frac{1+\frac{1}{2d^2}}{1+\frac{1}{4d^2}-\frac{1}{8d^4}}. \tag{3.4.7}$$

Since d is at least 1, we have $\frac{1}{4d^2} \geq \frac{1}{8d^4}$. Thus, we have $1+\frac{1}{4d^2}-\frac{1}{8d^4} \geq 1$. By putting this and Equality 3.4.7 together, we obtain Inequality 3.4.6.

Next, we show

$$\left(1+\frac{1}{2d^2}\right)^d \leq (1+\frac{1}{d}). \tag{3.4.8}$$

To prove it, first we have

$$\begin{aligned}\left(1+\frac{1}{2d^2}\right)^d &\le e^{\frac{1}{2d}} = \frac{1}{e^{-\frac{1}{2d}}} \\ &\le \frac{1}{1-\frac{1}{2d}} && \text{(since } e^{-x} \ge 1-x > 0\text{, for all } x<1\text{)} \\ &= \frac{1+\frac{1}{d}}{1+\frac{1}{2d}-\frac{1}{2d^2}}. && \text{(multiplying both the numerator and the denominator by } 1+\frac{1}{d}\text{)}\end{aligned} \tag{3.4.9}$$

Since d is at least 1, we have $\frac{1}{2d} \ge \frac{1}{2d^2}$. Thus, we have $1+\frac{1}{2d}-\frac{1}{2d^2} \ge 1$. By putting this and Inequality 3.4.9 together, we obtain Inequality 3.4.8. By putting Inequalities 3.4.6 and 3.4.8 together, we obtain the required inequality. □

Construction of Matching Coverage. Suppose ALG is the matching returned by RandGreedy, and we assume that the knowledge of an optimal weight matching *OPT* is given. Without loss of generality, we can assume that every online node u is matched in both ALG and *OPT*, because for every online node u, we can add a dummy offline node v_u, which connects to u with an edge of weight zero. Observe that the values in β are already constructed in the execution of RandGreedy.

Lemma 3.11 (Construction of Matching Coverage for *OPT*) *Given an optimal matching OPT, we can set values in α such that (α, β) is a matching coverage for OPT. Moreover, $[C(\alpha,\beta)] \le (2-\frac{1}{4\Delta^2})[w(ALG)]$, where $C(\alpha,\beta) := \sum_{u\in U}\alpha_u + \sum_{v\in V}\beta_v$, and Δ is the maximum online degree.*

Proof: We describe how values in α are assigned. For an online vertex u, let $OPT(u)$ be the edge in *OPT* incident on u and ALG(u) be the corresponding edge in M. Observe that the β values do not decrease during the execution of the algorithm. Hence, in order to show that (α, β) is a matching coverage for OPT, it suffices to check, at the end of each round in which online node u arrives, the vector (α, β) is feasible for the edge $OPT(u)$.

We shall also prove that $[\alpha_u] \leq (1 - \frac{1}{4\Delta^2})[b_{\mathrm{ALG}(u)}]$. We next condition on the randomness $\mathcal{F}$ used by the algorithm before online node u arrives. (Specifically, we prove that $E[\alpha_u|\mathcal{F}] \leq (1 - \frac{1}{4\Delta^2})b_{\mathrm{ALG}(u)}$, and hence we can take expectation again over randomness $\mathcal{F}$.)

Case 1: $\mathsf{Act}(\mathbf{u})$ does not contain $\boldsymbol{OPT}(\mathbf{u})$. In this case we set $\alpha_u := b_{OPT(u)}$. Hence, if $OPT(u) = (u, v)$, we have $\alpha_u + \beta_v = (w_{uv} - \beta_v) + \beta_v = w_{uv}$.

Recall that the benefit of a non-active edge of an online vertex u is at most $1 - \frac{1}{4{d_u}^2}$ times the minimum benefit of any active edge of u. Since in this case $OPT(u)$ is not an active edge and also $\mathrm{ALG}(u)$ is an active edge, we have $\alpha_u = b_{OPT(u)} \leq (1 - \frac{1}{4{d_u}^2})b_{\mathrm{ALG}(u)} \leq (1 - \frac{1}{4\Delta^2})b_{\mathrm{ALG}(u)}$, which also holds in expectation.

Case 2: $\mathsf{Act}(\mathbf{u})$ contains $\boldsymbol{OPT}(\mathbf{u})$. In this case if $OPT(u) = \mathrm{ALG}(u)$, we set $\alpha_u := 0$; when an offline node v is picked, we have $\beta_v = w_{uv}$, which clearly makes the vector feasible for the edge $OPT(u) = (u, v)$. Otherwise, $OPT(u) \neq \mathrm{ALG}(u)$, and we set $\alpha_u := b_{OPT(u)}$; similar to case 1, the resulting vector is feasible for $OPT(u)$.

Recall that $\mathsf{RandGreedy}$ selects an active edge of u uniformly at random. Also, $|\mathsf{Act}(u)|$ is at most d_u. Hence, it selects $OPT(u)$ with probability at least $\frac{1}{d_u}$. It means that with probability at least $\frac{1}{d_u}$, we set $\alpha_u := 0$, and with probability at most $1 - \frac{1}{d_u}$ we set $\alpha_u := b_{OPT(u)}$. Therefore, $[\alpha_u|\mathcal{F}] \leq (1 - \frac{1}{d_u})b_{OPT(u)}$.

Since both $OPT(u)$ and $\mathrm{ALG}(u)$ are active neighbors of u, we can apply Lemma 3.9 and conclude that $b_{OPT(u)} \leq (1 + \frac{1}{d_u})b_{\mathrm{ALG}(u)}$.

Combining the last two inequalities together, we conclude that $[\alpha_u|\mathcal{F}] \leq (1-\frac{1}{d_u})(1+\frac{1}{d_u})b_{\mathrm{ALG}(u)} = (1 - \frac{1}{{d_u}^2})b_{\mathrm{ALG}(u)} \leq (1 - \frac{1}{\Delta^2})b_{\mathrm{ALG}(u)} \leq (1 - \frac{1}{4\Delta^2})b_{\mathrm{ALG}(u)}$.

To summarize, we have shown that (α, β) is always a matching coverage for OPT.

Moreover, $[\sum_{u\in U} \alpha_u + \sum_{v\in V} \beta_v] \leq (1 - \frac{1}{4\Delta^2})E[\sum_{u\in U} b_{\mathrm{ALG}(u)}] + [\sum_{v\in V} \beta_v]$.

Next, observe that in each round when online u is matched, we see that among all offline nodes, only the value of the offline node in $\mathrm{ALG}(u)$ is increased by $b_{\mathrm{ALG}(u)}$. Hence, the total sum $\sum_{v\in V} \beta_v$ increases by exactly $b_{\mathrm{ALG}(u)}$, which means $\sum_{u\in U} b_{\mathrm{ALG}(u)} = \sum_{v\in V} \beta_v$.

Finally, observe that whenever an offline node v is assigned to an online node u, we set β_v to w_{uv}. Hence, it is immediate that $\sum_{v\in V} \beta_v = w(\mathrm{ALG})$.

Therefore, we conclude that $[\sum_{u\in U} \alpha_u+\sum_{v\in V} \beta_v] \leq (2-\frac{1}{4\Delta^2})[w(\mathrm{ALG})]$, as required.

□

We are now ready to complete the proof for the competitive ratio of RandGreedy.

Proof of Theorem 3.2: From Lemma 3.11, we have $w(OPT) \leq [C(\alpha,\beta)] \leq (2-\frac{1}{4\Delta^2})[w(\mathrm{ALG})]$, which means the competitive ratio is $\frac{[w(\mathrm{ALG})]}{w(OPT)} \geq \frac{1}{2-\frac{1}{4\Delta^2}} = \frac{1}{2}\cdot(1-\frac{1}{8\Delta^2})^{-1} \geq \frac{1}{2}\cdot(1+\frac{1}{8\Delta^2}) = \frac{1}{2} + \frac{1}{16\Delta^2}$, as required. ■

Theorem 3.5 *RandGreedy is $\frac{1}{2}+\frac{1}{16\Delta^2}$-competitive, where Δ is an upper bound on degree of online vertices.*

Proof: Let M be the output matching of RandGreedy on graph G and let OPT be a maximum weighted matching of graph G. We show there exist a matching coverage $C(OPT) = (\overrightarrow{\alpha}, \overrightarrow{\beta})$ for each run of RandGreedy such that, $E[w(C(OPT))] \leq (2 - \frac{1}{4\Delta^2})E[w(M)]$. By applying Lemma 3.2, we conclude $E[w(M)] \geq \frac{1}{2-\frac{1}{4\Delta^2}} w(OPT)$. Thus, we have

$$\begin{aligned}
\frac{E[w(M)]}{w(OPT)} &\geq \frac{1}{2-\frac{1}{4\Delta^2}} \\
&= \frac{1}{2-\frac{1}{4\Delta^2}} \times \frac{1+\frac{1}{8\Delta^2}}{1+\frac{1}{8\Delta^2}} \\
&= \frac{1+\frac{1}{8\Delta^2}}{2-\frac{1}{32\Delta^4}} \\
&\geq \frac{1+\frac{1}{8\Delta^2}}{2} = \frac{1}{2} + \frac{1}{16\Delta^2}
\end{aligned}$$

That means RandGreedy has a competitive ratio $1 + \frac{1}{16\Delta^2}$.

In order to define $C(OPT) = (\overrightarrow{\alpha}, \overrightarrow{\beta})$, initially for all u and v we set α_u and β_v to 0. Upon arrival of online vertex u, RandGreedy matches it to offline vertex v with edge $e = (u, v)$; We increase β_v by b_e for offline vertex v where b_e is the benefit of edge e. Recall that c_v is the weight of the last edge matched to v and we have $b_e = w_e - c_v$. Thus, $\beta_v = c_v$, i.e., β_v is always equal to the weight of the last edge matches to v. Later, we describe how to change α_u for online vertices such that we increase α_u by at most $(1 - \frac{1}{4\Delta^2})b_e$ in expectation.

At the beginning, for each online vertex, we add a *dummy* offline vertex and connect it to the online vertex with an edge of weight 0. Since the weight of new edges are

0, selecting them have no effect on the size of the final matching. On the other hand, each online vertex has an edge with non-negative weight to a distinct offline dummy vertex. Thus, without loss of generality, we can assume each online vertex has a match in both M and OPT. For an online vertex u, let $OPT(u)$ be the edge of OPT adjacent to u and $\mathrm{ALG}(u)$ be the edge that RandGreedy matches to u.

Case 1: $\mathsf{Act}(\mathbf{u})$ does not contain $\boldsymbol{OPT}(\mathbf{u})$. In this case we increase α_u by $b_{OPT(u)}$.

Recall that the benefit of a non-active edge of an online vertex u is at most $1 - \frac{1}{4{d_u}^2}$ times the minimum benefit of any active edge of u. Since in this case $OPT(u)$ is not an active edge and also $\mathrm{ALG}(u)$ is an active edge, we have

$$b_{OPT(u)} \leq (1 - \frac{1}{4{d_u}^2}) b_{\mathrm{ALG}(u)}.$$

This means in this case we increase α_u by at most $(1 - \frac{1}{4{d_u}^2}) b_{\mathrm{ALG}(u)} \leq (1 - \frac{1}{4\Delta^2}) b_{\mathrm{ALG}(u)}$ as desired. Recall that β_v is equal to c_v and for each edge $e = (u, v)$ we have $b_e = w_e - c_v$. Let v be the offline vertex that OPT assigns to u. We have

$$w_{OPT(u)} = b_{OPT(u)} + c_v = \alpha_u + \beta_v.$$

That means the dual constraint for edge $OPT(u)$ is feasible.

Case 2: $\mathsf{Act}(\mathbf{u})$ contains $\boldsymbol{OPT}(\mathbf{u})$. In this case if $OPT(u) = \mathrm{ALG}(u)$ we do not increase α_u otherwise we increase α_u by $b_{OPT(u)}$.

Recall that RandGreedy selects an active edge of u uniformly at random. Also, $|\mathsf{Act}(u)|$ is at most d_u. Hence, it selects $OPT(u)$ with probability at least $\frac{1}{d_u}$. It means with probability at least $\frac{1}{d_u}$ we do not increase α_u and with probability at most $1 - \frac{1}{d_u}$ we increase α_u by $b_{OPT(u)}$. Thus, in expectation we increase α_u by at most

$$(1 - \frac{1}{d_u})b_{OPT(u)}. \tag{3.4.10}$$

Since both $OPT(u)$ and $\text{ALG}(u)$ are active neighbors of u we can apply Lemma 3.9 and conclude

$$b_{OPT(u)} \leq (1 + \frac{1}{d_u})b_{\text{ALG}(u)}.$$

Putting the above inequality and Formula 3.4.10 together we conclude we increase α_u by at most $(1 - \frac{1}{d_u})(1 + \frac{1}{d_u})b_{\text{ALG}(u)} = (1 - \frac{1}{{d_u}^2})b_{\text{ALG}(u)} \leq (1 - \frac{1}{\Delta^2})b_{\text{ALG}(u)}$ in expectation. Let v be the offline vertex that OPT assigns to u. Recall that β_v is always equal to the largest edge matched to v. Hence, when $OPT(u) = \text{ALG}(u)$, we have $\beta_v \geq w_{OPT(v)}$ which means the dual constraint for edge $OPT(u)$ is feasible since $\alpha_u \geq 0$ and thus $\alpha_u + \beta_v \geq w_{OPT(v)}$. On the other hand when $OPT(u) \neq \text{ALG}(u)$, we have $\alpha_u = b_{OPT(u)}$. Thus, we have

$$w_{OPT(u)} = b_{OPT(u)} + c_v = \alpha_u + \beta_v.$$

This means the dual constraint for edge $OPT(u)$ is feasible and thus $C(OPT) = (\overrightarrow{\alpha}, \overrightarrow{\beta})$ is a coverage. This finishes the proof. □

]

We finish this section with an example to describe how a matching coverage is constructed in the course of the algorithm.

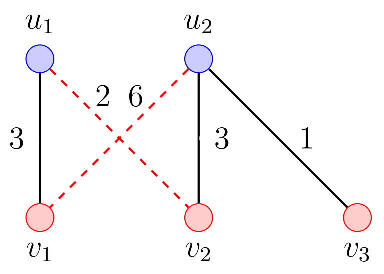

Figure 3.4.1: The graph corresponding to Example 3.

Example 3 *Assume $U = \{u_1, u_2\}$ and $V = \{v_1, v_2, v_3\}$. Online vertex u_1 has two edges with weights 3 and 2 to v_1 and v_2 respectively. Online vertex u_2 has three edges with weights 6, 3, and 1 to v_1, v_2, and v_3 respectively. Note that $d_{u_1} = 2$ and $d_{u_2} = 3$. The optimum matching is $OPT = \{(u_1, v_2), (u_2, v_1)\}$ with $w(OPT) = 8$. This instance has been shown in Figure 3.4.1.* [2]

When u_1 arrives online it has two options. However, we know $3 \times (1 - \frac{1}{16}) > 2$ which means $b_{u_1 v_1}(1 - \frac{1}{4d_{u_1}^2}) > b_{u_1 v_2}$ and $\mathsf{Act}(u_1) = \{(u_1, v_1)\}$. Therefore, RandGreedy matches u_1 to v_1 for sure. As the $OPT(u_1) = (u_1, v_2)$ is not in active set $\mathsf{Act}(u_1)$ we increase α_{u_1} by $b_{OPT(u_1)} = 2$. We also set $\beta_{v_1} = 3$

In the next step u_2 arrives and it has three edges. We have $b_{u_2 v_1} = 6 - 3 = 3$, $b_{u_2 v_2} = 3$, and $b_{u_2 v_3} = 1$ which means $\mathsf{Act}(u_2) = \{(u_2, v_1), (u_2, v_2)\}$. Hence, RandGreedy selects one of edges in $\mathsf{Act}(u_2)$ uniformly at random and constructs a matching coverage based on each selection. If the algorithm selects $(u_2, v_1) \in OPT(u_2)$, it only increases β_{v_1} to 6 and builds matching coverage $C(OPT) = (\overrightarrow{\alpha}, \overrightarrow{\beta}) = ((2, 0), (6, 0, 0))$ in this case. If the algorithm selects $(u_2, v_2) \notin OPT(u_2)$, it in-

[2]Explanation of Figure 3.4.1: Dashed lines represent the optimum matching. RandGreedy constructs two matching coverages $((2, 0), (6, 0, 0))$ and $((2, 3), (3, 3, 0))$ in the course of algorithm.

creases β_{v_2} *by* 3 *and* α_{u_2} *by* $b_{OPT(u_2)} = b_{u_2v_1} = 6-3 = 3$. *Matching coverage* $C(OPT) = (\overrightarrow{\alpha}, \overrightarrow{\beta}) = ((2,3),(3,3,0))$ *will be constructed in this case. Note that*

$$\begin{aligned} w(OPT) &= 8 \\ E[w(C(OPT))] &= \frac{1}{2}\times 8 + \frac{1}{2}\times 11 = 9.5 \\ E[w(M)] &= \frac{1}{2}\times 6 + \frac{1}{2}\times 6 = 6 \end{aligned}$$

3.5 The Hardness of Edge-Weighted Online Bipartite Matching without Free Disposal

In this section, we show without the free disposal assumption, there is no randomized online algorithm with bounded competitive ratio for edge-weighted online bipartite matching. Assume there is an online algorithm $\mathcal{A}$ with competitive ratio $\frac{1}{k}$ for the problem. Consider offline vertex v and a sequence of online vertices $u_1, u_2, \cdots, u_N$. We will define N later. Online vertex u_i has an edge to v with weight $w_{u_iv} = k^{i-1}$. Assume online vertices arrive in this order: $u_1, u_2, \cdots, u_N$. Consider online vertex u_i and assume offline vertex v is unmatched by Algorithm $\mathcal{A}$ upon arrival of u_i. Let p_i be the probability that $\mathcal{A}$ match u_i to v in this situation.

Let graph G_i be the subgraph of G which consists of offline vertex v and online vertices $u_1, \cdots, u_i$. The expected value of the output of online algorithm $\mathcal{A}$ should be at least $\frac{1}{k}$ the maximum weighted matching of G_i for all $1 \le i \le N$. Note that the maximum weighted matching for graph G_i is the edge (u_i, v) which has weight k^{i-1}. Therefore, the expected value of the output of the online algorithm should be at least k^{i-2} on graph G_i. Note that the expected weight of the output matching of $\mathcal{A}$ is $T_i = \sum_{j=1}^{i} q_{j-1}p_jk^{j-1}$, where $q_j = \prod_{r=1}^{j}(1-p_r)$ is the probability that vertex v is unmatched upon arrival of u_{j+1}. Online algorithm $\mathcal{A}$ is $\frac{1}{k}$-competitive which means $T_i \geq k^{i-2}$. Now we prove p_i should be at least $\frac{1}{k}$. Note that $T_i = \sum_{j=1}^{i} q_{j-1}p_jk^{j-1}$ and k^j is an increasing function in j. Therefore, we have

$$\begin{aligned} T_i = \sum_{j=1}^{i} q_{j-1}p_jk^{j-1} &= \sum_{j=1}^{i-1} q_{j-1}p_jk^{j-1} + q_{i-1}p_ik^{i-1} \\ &\leq (1-q_{i-1})k^{i-2} + q_{i-1}p_ik^{i-1}. \end{aligned}$$

where $\sum_{j=1}^{i-1} q_{j-1}p_j = 1 - q_{i-1}$ is the probability that v is matched upon arrival of u_i. On the other hand, we have $T_i \geq k^{i-2}$. Hence, we conclude

$$(1-q_{i-1})k^{i-2} + q_{i-1}p_ik^{i-1} \geq k^{i-2} \Rightarrow p_i \geq \frac{1}{k}. \tag{3.5.11}$$

Let N be a big number such that $(1-\frac{1}{k})^N < \frac{1}{2k}$, and W be the sum of weights of all edges in graph G_N. Construct graph G' by adding another online node u' with $w_{u'v} = 2kW$ to graph G_N. The optimum offline solution is $2kW$ for graph G'. Thus, the expected value of the output of online algorithm $\mathcal{A}$ should be at least $2W$ on graph G'. Assume u' be the last online vertex which arrives online. The probability that v remains unmatched upon arrival of u' is $q_N = \prod_{r=1}^{N} 1 - p_r$. As $p_r \geq \frac{1}{k}$ for all $1 \leq r \leq N$ and $(1-\frac{1}{k})^N < \frac{1}{2k}$, we conclude $q_N < \frac{1}{2k}$. It means the expected weight of the output of $\mathcal{A}$ is at most $(1-q_N)W + q_N 2kW < W + W = \frac{2kW}{k}$. Therefore, online algorithm $\mathcal{A}$ is not $\frac{1}{k}$-competitive.

3.6 Conclusion

We introduced a new framework called the *matching coverage* which is a more powerful tool than the primal dual framework for analyzing some approximation algorithms. We utilize this framework to show our algorithms for two variations of matching problem can achieve a competitive ratio strictly greater than 0.5 for the first time. There are many areas where we can continue this work in the future.

Can we achieve a better approximation ratio for any of these two problems? How can we use the matching coverage framework for other online problems (not necessarily matching problems)? and Can we analyze some previous algorithms using this framework to show that they actually have a higher competitive ratio than we thought?

Chapter 4: Network Cournot Competition

4.1 Introduction

In this paper we study selling a utility with a distribution network, e.g., natural gas, water and electricity, in several markets when the clearing price of each market is determined by its supply and demand. The distribution network fragments the market into different regional markets with their own prices. Therefore, the relations between suppliers and submarkets form a complex network [51–57]. For example, a market with access to only one supplier suffers a monopolistic price, while a market having access to multiple suppliers enjoys a lower price as a result of the price competition.

Antoine Augustin Cournot introduced the first model for studying the duopoly competition in 1838. He proposed a model where two individuals own different springs of water, and sell it independently. Each individual decides on the amount of water to supply, and then the aggregate water supply determines the market price through an inverse demand function. Cournot characterizes the unique equilibrium outcome of the market when both suppliers have the same marginal costs of production, and the inverse demand function is linear. He argued that in the unique equilibrium outcome, the market price is above the marginal cost.

Joseph Bertrand Bertrand1883 criticized the Cournot model, where the strategy of each player is the quantity to supply, and in turn suggested to consider prices, rather than quantities, as strategies. In the Bertrand model each firm chooses a price for a homogeneous good, and the firm announcing the lowest price gets all the market share. Since the firm with the lowest price receives all the demand, each firm has incentive to price below the current market price unless the market price matches its cost. Therefore, in an equilibrium outcome of the Bertrand model, assuming all marginal costs are the same and there are at least two competitors in the market, the market price will be equal to the marginal cost.

The Cournot and Bertrand models are two basic tools for investigating the competitive market price, and have attracted much interest for modeling real markets; see, e.g., [53–56]. While these are two extreme models for analyzing the price competition, it is hard to say which one is essentially better than the other. In particular, the predictive power of each strongly depends on the nature of the market, and varies from application to application. For example, the Bertrand model explains the situation where firms literally set prices, e.g., the cellphone market, the laptop market, and the TV market. On the other hand, Cournot's approach would be suitable for modeling markets like those of crude oil, natural gas, and electricity, where firms decide about quantities rather than prices.

There are several attempts to find equilibrium outcomes of the Cournot or Bertrand competitions in the oligopolistic setting, where a small number of firms compete in only one market; see, e.g., [58–63]. Nevertheless, it is not entirely clear what equilibrium outcomes of these games are when firms compete over more than one market. In this paper, we investigate the problem of finding equilibrium outcomes of the Cournot competition in a network setting where there are several markets for a homogeneous good and each market is accessible to a subset of firms.

4.1.1 Example

We start with the following warm-up example[1]. This is a basic example for the Cournot competition in the network setting. It consists of three scenarios. We assume firm $i \in \{A, B\}$ produces quantity q_{ij} of the good in market $j \in \{1, 2\}$. Let $\mathbf{q}$ be the vector of all quantities. The detailed computations are in the Appendix.

Scenario 1: Consider the Cournot competition in an oligopolistic setting with two firms and one market (see Figure 5.6.1). Let $p(\mathbf{q}) = 1 - q_{A1} - q_{B1}$ be the market price (the inverse demand function), and $c_i(\mathbf{q}) = \frac{1}{2}q_{i1}^2$ be the cost of production for firm $i \in \{A, B\}$. The profit of a firm is what it gets by selling all the quantities of good it produces in all markets minus its cost of production. Therefore, the profit of firm i denoted by $\pi_i(\mathbf{q})$ is $q_{i1}(1 - q_{A1} - q_{B1}) - \frac{1}{2}q_{i1}^2$. In a Nash equilibrium of the game, each firm maximizes its profit assuming its opponent does not change its strategy. Hence, the unique Nash equilibrium of the game can be found by solving the set of

[1]Explanation of Figure 5.6.1 : This figure represents the three scenarios of our example. Vector $\mathbf{q} = (\frac{1}{4}, \frac{1}{4})$ represents the unique equilibrium in the first scenario. Vector $\mathbf{q} = (\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8})$ is the unique equilibrium of the second scenario. Finally, Vector $\mathbf{q} = (0.18, 0.1, 0.16)$ is the unique equilibrium in the third scenario.

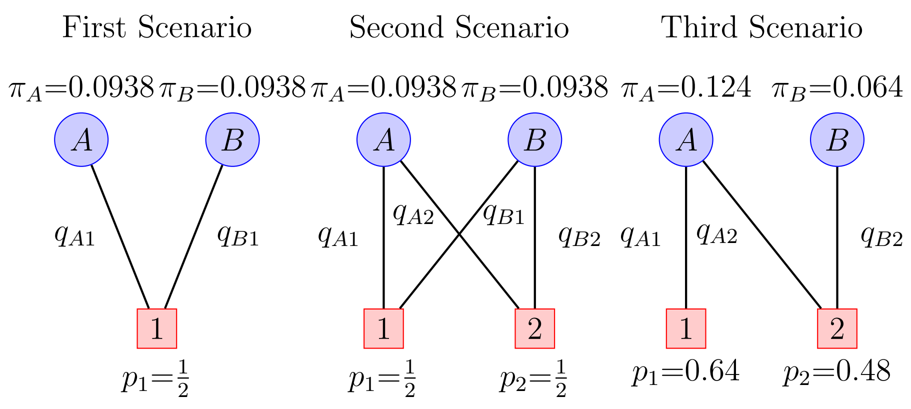

Figure 4.1.1: The figure showing scenarios in Example 4.1.1

equations $\frac{\partial \pi_A}{\partial q_{A1}} = \frac{\partial \pi_B}{\partial q_{B1}} = 0$. So $q_{A1} = q_{B1} = \frac{1}{4}$ is the unique Nash equilibrium where $p(\mathbf{q}) = \frac{1}{2}$, and $\pi_A(\mathbf{q}) = \pi_B(\mathbf{q}) = 0.9375$.

Scenario 2: We construct the second scenario by splitting the market in the previous scenario into two identical markets such that both firms have access to both markets (see Figure 5.6.1). Since the demand is divided between two identical markets, the price for market j would be $p_j(\mathbf{q}) = 1 - 2q_{Aj} - 2q_{Bj}$, i.e., the clearance price of each market is the same as the clearance price of the market in Scenario 1, when the supply is half of the supply of the market in Scenario 1. In this scenario, the profit of firm $i \in \{A, B\}$ is $\pi_i(\mathbf{q}) = \sum_j q_{ij}(1 - 2q_{Aj} - 2q_{Bj}) - \frac{1}{2}(q_{i1} + q_{i2})^2$. Any Nash equilibrium of this game satisfies the set of equations $\frac{\partial \pi_A}{\partial q_{A1}} = \frac{\partial \pi_A}{\partial q_{A2}} = \frac{\partial \pi_B}{\partial q_{B1}} = \frac{\partial \pi_B}{\partial q_{B2}} = 0$. By finding the unique solution to this set of equations, one can verify that $\mathbf{q} = (\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8})$ is the unique equilibrium of the game where $p_1(\mathbf{q}) = p_2(\mathbf{q}) = \frac{1}{2}$, and $\pi_A(\mathbf{q}) = \pi_B(\mathbf{q}) = 0.09375$. Since we artificially split the market into two identical markets, this equilibrium is, not surprisingly, the same as the equilibrium in the previous scenario.

Scenario 3: Consider the previous scenario, and suppose firm 2 has no access to the first market (see Figure 5.6.1). Let the demand functions and the cost functions be the same as the previous scenario. The profits of firms 1 and 2 can be written as follows:

$$\begin{aligned}\pi_A(\mathbf{q}) &= q_{A1}(1-2q_{A1}) + q_{A2}(1-2q_{A2}-2q_{B2}) - \frac{1}{2}(q_{A1}+q_{A2})^2,\\ \pi_B(\mathbf{q}) &= q_{B2}(1-2q_{A2}-2q_{B2}) - \frac{1}{2}q_{B2}^2.\end{aligned}$$

The unique equilibrium outcome of the game is found by solving the set of equations $\frac{\partial \pi_A}{\partial q_{A1}} = \frac{\partial \pi_A}{\partial q_{A2}} = \frac{\partial \pi_B}{\partial q_{B2}} = 0$. One can verify that vector $\mathbf{q} = (q_{A1}, q_{A2}, q_{B2}) = (0.18, 0.1, 0.16)$ is the unique equilibrium outcome of the game where $p_1(\mathbf{q}) = 0.64$, $p_2(\mathbf{q}) = 0.48$, $\pi_A(\mathbf{q}) = 0.124$, and $\pi_B(\mathbf{q}) = 0.064$. The following are a few observations worth mentioning.

- Firm A has more power in this scenario due to having a captive market[2].
- The equilibrium price of market 1 is higher than the equilibrium price in the previous scenarios.
- The position of firm B affects its profit. Since it has no access to market 1, it is not as powerful as firm A.
- The equilibrium price of market 2 is smaller than the equilibrium price in the previous scenarios.

[2]A captive market is one in which consumers have limited options and the seller has a monopoly power.

4.1.2 Related Work

There are several papers that investigate the Cournot competition in an oligopolistic setting (see, e.g., [58, 59, 61–63]). In spite of these works, little is known about the Cournot competition in a network. [64] studies the Cournot competition in a network setting, and considers a network of firms and markets where each firm chooses a quantity to supply in each accessible market. He studies the competition when the inverse demand functions are linear and the cost functions are quadratic (functions of the total production). In this study, we consider the same model when the cost functions and the demand functions may have quite general forms. We show the game with linear inverse demand functions is a potential game and therefore has a unique equilibrium outcome. Furthermore, we present two polynomial-time algorithms for finding an equilibrium outcome for a wide range of cost functions and demand functions. While we investigate the Cournot competition in networks, there is a recent paper which considers the Bertrand competition in network setting [65], albeit in a much more restricted case of only two firms competing in each market.

The final price of each market in the Cournot competition is a market clearing price; i.e, the final price is set such that the market becomes clear. Finding a market clearance equilibrium is a well-established problem, and there are several papers which propose polynomial-time algorithms for computing equilibriums of markets in which the price of each good is defined as the price in which the market clears. Examples of such markets include Arrow-Debreu market and its special case Fisher market (see related work on these markets [66–72]). clear4 design an approximation scheme which computes the market clearing prices for the Arrow-Debreu market, and clear6 improve the running time of the algorithm. The first polynomial-time algorithm for finding an Arrow-Debreu market equilibrium is proposed by clear2 for a special case with linear utilities. The Fisher market, a special case of the Arrow-Debreu market, attracted a lot of attention as well. clear7 present the first polynomial-time algorithm by transferring the problem to a concave cost maximization problem. clear3 design the first combinatorial algorithm which runs in polynomial time and finds the market clearance equilibrium when the utility functions are linear. This result is later improved by clear5.

For the sake of completeness, we refer to recent works in the computer science literature [73, 74], which investigate the Cournot competition in an oligopolistic setting. wine10 study a coalition formation game in a Cournot oligopoly. In this setting, firms form coalitions, and the utility of each coalition, which is equally divided between its members, is determined by the equilibrium of a Cournot competition between coalitions. They prove the price of anarchy, which is the ratio between the social welfare of the worse stable partition and the social optimum, is $\Theta(n^{2/5})$ where n is the number of firms. [74] consider a Cournot competition where agents may decide to be non-myopic. In particular, they define two principal strategies to maximize revenue and profit (revenue minus cost) respectively. Note that in the classic Cournot competition all agents want to maximize their profit. However, in their study each agent first chooses its principal strategy and then acts accordingly.

The authors prove this game has a pure Nash equilibrium and the best response dynamics will converge to an equilibrium. They also show the equilibrium price in this game is lower than the equilibrium price in the standard Cournot competition.

4.1.3 Results and techniques

We consider the problem of Cournot competition on a network of markets and firms for different classes of cost and inverse demand functions. Adding these two dimensions to the classical Cournot competition which only involves a single market and basic cost and inverse demand functions yields an engaging but complicated problem which needs advanced techniques for analyzing. For simplicity of notation we model the competition by a bipartite graph rather than a hypergraph: vertices on one side denote the firms, and vertices on the other side denote the markets. An edge between a firm and a market demonstrates willingness of the firm to compete in that specific market. The complexity of finding the equilibrium, in addition to the number of markets and firms, depends on the classes that inverse demand and production cost functions belong to.

We summarize our results in the following table.

Cost functions	**Inverse demand functions**	**Running time**	**Technique**
Convex	Linear	$O(E^3)$	Convex optimization, formulation as an ordinal potential game
Convex	Strongly monotone marginal revenue function[3]	poly(E)	Reduction to a non-linear complementarity problem
Convex, separable	Concave	$O(n \log^2 Q_{\max})$	Supermodular optimization, nested binary search

In the above table, E denotes the number of edges of the bipartite graph, n denotes the number of firms, and $Q_{\max}$ denotes the maximum possible total quantity in the oligopoly network at any equilibrium. In our results we assume the inverse demand functions are nonincreasing functions of total production in the market. This is the basic assumption in the classical Cournot Competition model: As the price in the

[3]Marginal revenue function is the vector function which maps production quantities for an edge to marginal revenue along that edge.

market increases, it is reasonable to believe that the buyers drop out of the market and demand for the product decreases. The classical Cournot Competition model as well as many previous works on Cournot Competition model assumes linearity of the inverse demand function [64, 73]. In fact there is little work on generalizing the inverse demand function in this model. The second and third row of the above table shows we have developed efficient algorithms for more general inverse demand functions satisfying concavity rather than linearity. This can be accounted as a big achievement. The assumption of monotonicity of the inverse demand function is a standard assumption in Economics [75–77]. We assume cost functions to be convex which is the case in many works related to both Cournot Competition and Bertrand Network [78, 79]. In a previous work [64], the author considered Cournot Competition on a network of firms and markets; however, assumed that inverse demand functions are linear and all the cost functions are quadratic function of the total production by the firm in all markets which is quite restrictive. Most of the results in other related works in Cournot Competition and Bertrand Network require linearity of the cost functions [65, 73]. A brief summary of our results presented in three sections is given below.

4.1.3.1 Linear Inverse Demand Functions

In case inverse demand functions are linear and production costs are convex, we present a fast and efficient algorithm to obtain the equilibrium. This approach works by showing that Network Cournot Competition belongs to a class of games called *ordinal potential games*. In such games, the collective strategy of the independent players is to maximize a single potential function. The potential function is carefully designed in such a way that changes made by one player reflects in the same way in the potential function as in their own utility function. We design a potential function for the game, which depends on the network structure, and show how it captures this property. Moreover, in the case where the cost functions are convex, we prove concavity of this designed potential function (Theorem 4.6) concluding convex optimization methods can be employed to find the optimum and hence, the equilibrium of the original Cournot competition. We also discuss uniqueness of equilibria in case the cost functions are strictly concave. Our result in this section is specifically interesting since we find the unique equilibrium of the game. We prove the following theorems in Section 4.3.

Theorem 4.1 *The Network Cournot Competition with linear inverse demand functions forms an ordinal potential game.*

Theorem 4.2 *Our designed potential function for the Network Cournot Competition with linear inverse demand functions is concave provided that the cost functions are convex. Furthermore, the potential function is strictly concave if the cost functions are strictly convex, and hence the equilibria for the game is unique. In addition, a polynomial-time algorithm finds the optimum of the potential function which describes the market clearance prices.*

4.1.3.2 The general case

Since the above approach does not work for nonlinear inverse demand functions, we design another interesting but more involved algorithm to capture more general forms of inverse demand functions. We show that an equilibrium of the game can be computed in polynomial time if the production cost functions are convex and the revenue function is monotone. Moreover, we show under strict monotonicity of the revenue function, the solution is unqiue, and therefore our results in this section is structural; i.e. we find the one and only equilibria. For convergence guarantee we also need Lipschitz condition on derivatives of inverse demand and cost functions. We start the section by modeling our problem as a complementarity problem. Then we prove how holding the aforementioned conditions for cost and revenue functions yields satisfying *Scaled Lipschitz Condition* (SLC) and semidefiniteness for matrices of derivatives of the profit function. SLC is a standard condition widely used in convergence analysis for scalar and vector optimization [80]. Finally , we present our algorithm, and show how meeting these new conditions by inverse demand and cost functions helps us to guarantee polynomial running time of our algorithm. We also give examples of classes of inverse demand functions satisfying the above conditions. These include many families of inverse demand functions including quadratic functions, cubic functions and entropy functions. The following theorem is the main result of Section 4.4 which summarizes the performance of our algorithm.

Theorem 4.3 *A solution to the Network Cournot Competition can be found in* *polynomial number of iterations under the following conditions:*

1. *The cost functions are (strongly) convex.*
2. *The marginal revenue function is (strongly[4]) monotone.*
3. *Ther first derivative of cost functions and inverse demand functions and the second derivative of inverse demand functions are Lipschitz continuous.*

Furthermore, the solution is unique assuming only the first condition. Therefore, our algorithm finds the unique equilibria of NCC.

4.1.3.3 Cournot oligopoly

Another reasonable model for considering cost functions of the firms is the case where the cost of production in a market depends only on the quantity produced by the firm in that specific market (and not on quantities produced by this firm in other markets). In other words, the firms have completely independent sections for producing different goods in various markets, and there is no correlation between cost of production in separate markets. Interestingly, in this case the competitions are separable; i.e. equilibrium for Network Cournot Competition can be found by finding the quantities at equilibrium for each market individually. This motivates us for considering Cournot game where the firms compete over a single market. We

[4]For at least one of the first two conditions, strong version of condition should be satisfied, i.e., either cost functions should be strongly convex or the marginal revenue function should be strongly monotone.

present a new algorithm for computing equilibrium quantities produced by firms in a Cournot oligopoly, i.e., when the firms compete over a single market. Cournot Oligopoly is a well-known model in Economics, and computation of its Cournot Equilibrium has been subject to a lot of attention. It has been considered in many works including [81–85] to name a few. The earlier attempts for calculating equilibrium for a general class of inverse demand and cost functions are mainly based on solving a Linear Complementarity Problem or a Variational Inequality. These settings can be then turned into convex optimization problems of size $O(n)$ where n is the number of firms. This means the runtime of the earlier works cannot be better than $O(n^3)$ which is the best performance for convex optimization [86]. We give a novel combinatorial algorithm for this important problem when the quantities produced are integral. We limit our search to integral quantities for two reasons. First, in real-world all commodities and products are traded in integral units. Second, this algorithm can easily be adapted to compute approximate Cournot-Nash equilibrium for the continuous case and since the quantities at equilibrium may not be rational numbers, this is the best we can hope for. Our algorithm runs in time $O(n \log^2(Q_{\max}))$ where $Q_{\max}$ is an upper bound on total quantity produced at equilibrium. Our approach relies on the fact that profit functions are supermodular when the inverse demand function is nonincreasing and the cost functions are convex. We leverage the supermodularity of inverse demand functions and Topkis' Monotonicity Theorem [87] to design a nested binary search algorithm. The following is the main result of Section 4.5.

Theorem 4.4 *A polynomial-time algorithm successfully computes the quantities produced by each firm at an equilibrium of the Cournot oligopoly if the inverse demand function is non-increasing, and the cost functions are convex. In addition, the algorithm runs in $O(n \log^2(Q_{\max}))$ where $Q_{\max}$ is the maximum possible total quantity in the oligopoly network at any equilibrium.*

4.2 Notations

Suppose we have a set of n firms denoted by $\mathcal{F}$ and a set of m markets denoted by $\mathcal{M}$. A single good is produced in each of these markets. Each firm might or might not be able to supply a particular market. A bipartite graph is used to demonstrate these relations. In this graph, the markets are denoted by the numbers $1, 2, \ldots, m$ on one side, and the firms are denoted by the numbers $1, 2, \ldots, n$ on the other side. For simplicity, throughout the paper we use the notation $i \in \mathcal{M}$ meaning the market i, and $j \in \mathcal{F}$ meaning firm j. For firm $j \in \mathcal{F}$ and market $i \in \mathcal{M}$ there exists an edge between the corresponding vertices in the bipartite graph if and only if firm j is able to produce the good in market i. This edge will be denoted (i, j). The set of edges of the graph is denoted by $\mathcal{E}$, and the number of edges in the graph is shown by E. For each market $i \in \mathcal{M}$, the set of vertices $N_{\mathcal{M}}(i)$ is the set of firms that this market is connected to in the graph. Similarly, $N_{\mathcal{F}}(j)$ denotes the set of neighbors of firms j among markets. The edges in $\mathcal{E}$ are sorted and numbered $1, \ldots, E$, first based on the number of their corresponding market and then based on the number of their corresponding firm. More formally, edge $(i, j) \in \mathcal{E}$ is ranked above edge

$(l,k) \in \mathcal{E}$ if $i < l$ or $i = l$ and $j < k$. The quantity of the good that firm j produces in market i is denoted by q_{ij}. The vector $\mathbf{q}$ is an $E \times 1$ vector that contains all the quantities produced over the edges of the graph in the same order that the edges are numbered.

The demand for good i, denoted D_i, is the sum of the total quantity of this good produced by all firms, i.e., $D_i = \sum_{j \in N_{\mathcal{M}}(i)} q_{ij}$. The price of good i, denoted by the function $P_i(D_i)$, is only a decreasing function of total demand for this good and not the individual quantities produced by each firm in this market. For a firm j, the vector $\vec{s_j}$ denotes the strategy of firm j, which is the vector of all quantites produced by this firm in the markets $N_{\mathcal{F}}(j)$. Firm $j \in \mathcal{F}$ has a cost function related to its strategy denoted by $c_j(\vec{s_j})$. The profit that firm j makes is equal to the total money that it obtains by selling its production minus its cost of production. More formally, the profit of firm j, denoted by π_j, is

$$\pi_j = \sum_{i \in N_{\mathcal{F}}(j)} P_i(D_i) q_{ij} - c_j(\vec{s_j}). \tag{4.2.1}$$

4.3 Cournot competition and potential games

In this section, we design an efficient algorithm for the case where the price functions are linear. More specifically, we design an innovative *potential function* that captures the changes of all the utility functions simultaneously, and therefore, show how finding the quantities at the equilibrium would be equivalent to finding the set of quantities that maximizes this function. We use the notion of *potential games* as introduced in monderer. In that paper, the authors introduce *ordinal potential games* as the set of games for which there exists a *potential function* P^* such that the pure strategy equilibrium set of the game coincides with the pure strategy equilibrium set of a game where every party's utility function is P^*.

In this section, we design a function for the Network Cournot Competition and show how this function is a potenial function for the problem if the price functions are linear. Interestingly, this holds for any cost function meaning Network Cournot Competition with arbitrary cost functions is an ordinal potential game as long as the price functions are linear. Furthermore, we show when the cost functions are convex, our designed potential function is concave, and hence any convex optimization method can find the equilibrium of such a Network Cournot Competition. In case cost functions are strictly convex, the potential function is strictly concave. We restate a well known theorem in this section to conclude that the convex optimization in this case has a unique solution, and therefore the equilibria that we find in this case is the one and only equilibria of the game.

Definition 4.1 *A game is said to be an* ordinal potential game *if the incentive of all players to change their strategy can be expressed using a single global function called the potential function. More formally, a game with n players and utility function u_i for player $i \in \{1, \dots, n\}$ is called ordinal potential with potential function P^* if for all the strategy profiles $q \in R^n$ and every strategy x_i of player i the following holds:*

$$u_i(x_i, q_{-i}) - u_i(q_i, q_{-i}) > 0 \text{ \emph{iff} } P^*(x_i, q_{-i}) - P^*(q_i, q_{-i}) > 0.$$

An equivalent definition of an ordinal potential game is a game for which a potential funciton P^* exists such that the following holds for all strategy profiles $q \in R^n$ and for each player i.

$$\frac{\partial u_i}{\partial q_i} = \frac{\partial P^*}{\partial q_i}.$$

In other words, for each strategy profile q, any change in the strategy of player i has the same impact on its utility function as on the game's potential function.

The pure strategy equilibrium set of any ordinal potential game coincides with the pure strategy equilibrium set of a game with the potential function P^* as all parties' utility function.

Theorem 4.5 *The Network Cournot Competition with linear price functions is an ordinal potential game.*

Proof: Let $P_i(D_i) = \alpha_i - \beta_i D_i$ be the linear price function for market $i \in \mathcal{M}$ where $\alpha_i \geq 0$ and $\beta_i \geq 0$ are constants determined by the properties of market i. Note that this function is decreasing with respect to D_i. Here we want to introduce a potential function P^*, and show that $\frac{\partial \pi_j}{\partial q_{ij}} = \frac{\partial P^*}{\partial q_{ij}}$ holds $\forall (i,j) \in \mathcal{E}$. The utility function of firm j is

$$\pi_j = \sum_{i \in N_{\mathcal{F}}(j)} \left(\alpha_i - \beta_i \sum_{k \in N_{\mathcal{F}}(j)} q_{kj} \right) q_{ij} - c_j(\vec{s_j}),$$

and taking partial derivative with respect to q_{ij} yields

$$\frac{\partial \pi_j}{\partial q_{ij}} = \alpha_i - \beta_i \sum_{k \in N_{\mathcal{F}}(j)} q_{kj} - \beta_i q_{ij} - \frac{\partial c_j(\vec{s_j})}{\partial q_{ij}}.$$

We define P^* to be

$$P^* = \sum_{i \in \mathcal{M}} \left[\alpha_i \sum_{j \in N_{\mathcal{M}}(i)} q_{ij} - \beta_i \sum_{j \in N_{\mathcal{M}}(i)} q_{ij}^2 - \beta_i \sum_{\substack{k \leq j \\ k,j \in N_{\mathcal{M}}(i)}} q_{ij} q_{ik} - \sum_{j \in N_{\mathcal{M}}(i)} \frac{c_j(\vec{s_j})}{|N_{\mathcal{F}}(j)|} \right],$$

whose partial derivative with respect to q_{ij} is

$$\begin{aligned}
\frac{\partial P^*}{\partial q_{ij}} &= \alpha_i - 2\beta_i q_{ij} - \frac{\partial}{\partial q_{ij}} \left(\beta_i \sum_{\substack{l \leq m \\ l,m \in N_{\mathcal{M}}(i)}} q_{il} q_{im} \right) - \frac{\partial c_j(\vec{s_j})}{\partial q_{ij}} \\
&= \alpha_i - 2\beta_i q_{ij} - \beta_i (D_i - q_{ij}) - \frac{\partial c_j(\vec{s_j})}{\partial q_{ij}} \\
&= \frac{\partial \pi_j}{\partial q_{ij}}.
\end{aligned}$$

Since this holds for each $i \in \mathcal{M}$ and each $j \in \mathcal{F}$, the Network Cournot Competition is an ordinal potential game. □

We can efficiently compute the equilibrium of the game if the potential function P^* is easy to optimize. Below we prove that this function is concave.

Theorem 4.6 *The potential function P^* from the previous theorem is concave provided that the cost functions of the firms are convex. Moreover, if the cost functions are strictly convex then the potential function is strictly concave.*

Proof: The proof goes by decomposing P^* into pieces that are concave. We first define f for one specific market i as

$$f = \sum_{j \in N_{\mathcal{M}}(i)} q_{ij}^2 + \sum_{\substack{k \leq j \\ k,j \in N_{\mathcal{M}}(i)}} q_{ij} q_{ik},$$

and prove that it is convex.

Recall that $\mathbf{q}$ is an $E \times 1$ vector of all the quantities of good produced over the existing edges of the graph. We can write $f = \mathbf{q}^T M \mathbf{q}$ where M is an $E \times E$ matrix with all elements on its diagonal equal to 1 and all other elements equal to $\frac{1}{\sqrt{2}}$:

$$M = \begin{bmatrix} 1 & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 1 & \cdots & \frac{1}{\sqrt{2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & 1 \end{bmatrix}.$$

To show that f is convex, it suffices to prove that M is positive semidefinite, by finding a matrix R such that $M = R^T R$. Consider the following $(E+1) \times E$ matrix:

$$R = \begin{bmatrix} c & c & \cdots & c \\ a & 0 & \cdots & 0 \\ 0 & a & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a \end{bmatrix},$$

where a, c are set below. Let R_i be the i-th column of R. We have $R_i \cdot R_i = a^2 + c^2$ and $R_i \cdot R_j = c^2$ for $i \neq j$.

Setting $c = 2^{-\frac{1}{4}}$ and $a = \sqrt{1-c}$ yields $M = R^T R$, showing that M is positive semidefinite, hence the convexity of f.

The following expression for a fixed market $i \in \mathcal{M}$, sum of three concave functions, is also concave.

$$\alpha_i \sum_{j \in N_{\mathcal{M}}(i)} q_{ij} - \beta_i \Bigg(\sum_{j \in N_{\mathcal{M}}(i)} q_{ij}^2 + \sum_{\substack{k \leq j \\ k,j \in N_{\mathcal{M}}(i)}} q_{ij} q_{ik} \Bigg) - \sum_{j \in N_{\mathcal{M}}(i)} \frac{c_j(\vec{s_j})}{|N_{\mathcal{F}}(j)|}.$$

Summing over all markets proves concavity of P^*. Note that if a function is the sum of a concave function and a strictly concave function, then it is strictly concave itself. Therefore, since f is concave, we can conclude strictly concavity of P^* under the assumption that the cost functions are strictly convex. $\square$

The following well-known theorem discusses the uniqueness of the solution to a convex optimization problem.

Theorem 4.7 *Let $F : \mathcal{K} \rightarrow R^n$ be a strictly concave and continuous function for some finite convex space $\mathcal{K} \in R^n$. Then the following convex optimization problem has a unique solution.*

$$\max f(x) \quad s.t. \quad x \in \mathcal{K}. \tag{4.3.2}$$

By Theorem 4.6, if the cost functions are strictly convex then the potential function is strictly concave and hence, by Theorem 4.7 the equilibrium of the game is unique.

Let $ConvexP(\mathcal{E}, (\alpha_1, \ldots, \alpha_m), (\beta_1, \ldots, \beta_m), (c_1, \ldots, c_n))$ be the following convex optimization program:

$$\min \quad -\sum_{i \in \mathcal{M}} \left[\alpha_i \sum_{j \in N_{\mathcal{M}}(i)} q_{ij} - \beta_i \sum_{j \in N_{\mathcal{M}}(i)} q_{ij}^2 - \beta_i \sum_{\substack{k \leq j \\ k,j \in N_{\mathcal{M}}(i)}} q_{ij} q_{ik} - \sum_{j \in N_{\mathcal{M}}(i)} \frac{c_j(\vec{s_j})}{|N_{\mathcal{F}}(j)|} \right] \tag{4.3.3}$$

$$\text{subject to} \qquad q_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{E}.$$

Note that in this optimization program we are trying to maximize P^* for a bipartite graph with set of edges $\mathcal{E}$, linear price functions characterized by the pair (α_i, β_i) for each market $i \in \mathcal{M}$, and cost functions c_j for each firm $j \in \mathcal{F}$.

Algorithm 5 Compute quantities at eq. for NCC with linear price functions.

```
Input: Set of edges $\mathcal{E}$, cost function $c_j$ for each firm $j$, coefficients of price function of market $i$ as $(\alpha_i, \beta_i)$.
Output: Quantities produced at equilibrium.
0: Let $C$ be this convex optimization problem
       $ConvexP(\mathcal{E}, (\alpha_1, \ldots, \alpha_m), (\beta_1, \ldots, \beta_m), (c_1, \ldots, c_n)).$
0: Solve $C$ using a convex optimization algorithm
0: Return result of convex optimization algorithm as $\vec{\mathbf{q}}$ of equilibrium quantities.
   =0
```

The above algorithm has a time complexity equal to the time complexity of a convex optimization algorithm with E variables. The best such algorithm has a running time $O(E^3)$ [86].

4.4 Finding equilibrium for cournot game with general cost and inverse demand functions

In this section, we formulate an algorithm for a much more general class of price and cost functions. Our algorithm is based on reduction of Network Cournot Competition (NCC) to a polynomial time solvable class of Non-linear Complementarity Problem (NLCP). First in Subsection 4.4.1, we introduce our marginal profit function as the vector of partial derivatives of all firms with respect to the quantities that they produce. Then in Subsection 4.4.2, we show how this marginal profit function can help us to reduce NCC to a general NLCP. We also discuss uniqueness of equilibrium in this situation which yields the fact that solving NLCP would give us the one and only equilibrium of this problem. Unfortunately, in its most general form, NLCP is computationally intractable. However, for a large class of functions, these problems are polynomial time solvable. Most of the rest of this section is dedicated to proving the fact that NCC is polytime solvable on vast and important array of price and cost functions. In Subsection 4.4.3, we rigorously define the conditions under which NLCP is polynomial time solvable. We present our algorithm in this subsection along with a theorem which shows it converges in polynomial number of steps. To show the conditions that we introduce for convergence of our algorithm in polynomial time are not restrictive, we give a discussion in Subsection 4.4.4 on the functions satisfying these conditions, and show they hold for a wide range of price functions.

Assumptions Throughout the rest of this section we assume that the price functions are decreasing and concave and the cost functions are strongly convex (*The notion of strongly convex is to be defined later*). We also assume that for each firm there is a finite quantity at which extra production ceases to be profitable even if that is the only firm operating in the market. Thus, all production quantities and consequently quantities supplied to markets by firms are finite. In addition, we assume Lipschitz continuity and finiteness of the first and the second derivatives of price and cost functions. We note that these Lipschitz continuity assumptions are very common for convergence analysis in convex optimization [86] and finiteness assumptions are implied by Lipschitz continuity. In addition, they are not very restrictive as we don't expect unbounded fluctuation in costs and prices with change in supply. For sake of brevity, we use the terms inverse demand function and price function interchangeably.

4.4.1 Marginal profit function

For the rest of this section, we assume that P_i and c_i are twice differentiable functions of quantities. We define f_{ij} for a firm j and a market i such that $(i, j) \in \mathcal{E}$ as follows.

$$f_{ij} = -\frac{\partial \pi_j}{\partial q_{ij}} = -P_i(D_i) - \frac{\partial P_i(D_i)}{\partial q_{ij}} q_{ij} + \frac{\partial c_j}{\partial q_{ij}}. \tag{4.4.4}$$

Recall that the price function of a market is only a function of the total production in that market and not the individual quantities produced by individual firms. Thus $\frac{\partial P_i(D_i)}{\partial q_{ij}} = \frac{\partial P_i(D_i)}{\partial q_{ik}} \quad \forall j, k \in N_{\mathcal{M}}(i)$. Therefore, we replace these terms by $P_i'(D_i)$.

$$f_{ij} = -P_i(D_i) - P_i'(D_i) q_{ij} + \frac{\partial c_j}{\partial q_{ij}}. \tag{4.4.5}$$

Let vector F be the vector of all f_{ij}'s corresponding to the edges of the graph in the same format that we defined the vector q. That is f_{ij} corresponding to $(i, j) \in \mathcal{E}$ appears above f_{lk} corresponding to edge $(l, k) \in \mathcal{E}$ iff $i < l$ or $i = l$ and $j < k$. Note that F is a function of q.

Moreover, we separate the part representing marginal revenue from the part representing marginal cost in function F. More formally, we split F into two functions R and S such that $F = R + S$, and the element corresponding to the edge $(i, j) \in \mathcal{E}$

in the *marginal revenue fuction* $R(q)$ is:

$$r_{ij} = -\frac{\partial \pi_j}{\partial q_{ij}} = -P_i(D_i) - P'(D_i)q_{ij},$$

whereas for the *marginal cost function* $S(q)$, it is:

$$s_{ij} = \frac{\partial c_j}{\partial q_{ij}}.$$

4.4.2 Non-linear complementarity problem

In this subsection we formally define the non-linear complementarity problem (NLCP), and prove our problem is a NLCP.

Definition 4.2 *Let $F : R^n \to R^n$ be a continuously differentiable function on R^n_+. The complementarity problem seeks a vector $x \in R^n$ that satisfies the following constraints:*

$$\begin{aligned} &x, F(x) \geq 0, \\ &x^T F(x) = 0. \end{aligned} \tag{4.4.6}$$

Theorem 4.8 *The problem of finding the vector q at equilibrium in the Cournot game is a complementarity problem.*

Proof: Let q^* be the vector of the quantities at equilibrium. All quantities must be nonnegative at all times; i.e., $q^* \geq 0$. It suffices to show $F(q^*) \geq 0$ and ${q^*}^T F(q^*) = 0$. At equilibrium, no party benefits from changing its strategy, in particular, its production quantities. For each edge $(i, j) \in \mathcal{E}$, if the corresponding quantity q^*_{ij} is positive, then q^*_{ij} is a local maxima for π_j; i.e., $f_{ij}(q^*) = -\left.\frac{\partial \pi_j}{\partial q_{ij}}\right|_{q^*} = 0$. On the other hand, if $q^*_{ij} = 0$, then $\left.\frac{\partial \pi_j}{\partial q_{ij}}\right|_{q^*}$ cannot be positive, since, if it is, firm j would benefit by increasing the quantity q_{ij} to a small amount ϵ. Therefore, $\left.\frac{\partial \pi}{\partial q_{ij}}\right|_{q^*}$ is always nonpositive or equivalently $f_{ij}(q^*) \geq 0$, i.e., $F(q^*) \geq 0$. Also, as we mentioned above, a nonzero q^*_{ij} is a local maximum for π_j; i.e., $f_{ij}(q^*) = -\left.\frac{\partial \pi}{\partial q_{ij}}\right|_{q^*} = 0$. Hence, either $q^*_{ij} = 0$ or $f_{ij}(q^*) = 0$; thus, $q^*_{ij} f_{ij}(q^*) = 0$. This yields $\sum_{(i,j)\in\mathcal{E}} q^*_{ij} f_{ij}(q^*) = {q^*}^T F(q^*) = 0$. □

Definition 4.3 *$F : \mathcal{K} \rightarrow R^n$ is said to be strictly monotone at x^* if*

$$\langle (F(x) - F(x^*))^T, x - x^* \rangle \geq 0, \forall x \in \mathcal{K}. \tag{4.4.7}$$

F is said to be strictly monotone if it is monotone at any $x^* \in \mathcal{K}$. Equivalently, F is strictly monotone if the jacobian matrix is positive definite.

The following theorem is a well known theorem for Complementarity Problems.

Theorem 4.9 *Let $F : \mathcal{K} \rightarrow R^n$ be a continuous and strictly monotone function with a point $x \in \mathcal{K}$ such that $F(x) \geq 0$ (i.e. there exists a potential solution to the CP). Then the Complementarity Problem introduced in (4.4.6) characterized by function F has a unique solution.*

Hence, the Complementarity Problem characterized by function F introduced element by element in (4.4.4) has a unique solution under the assumption that the revenue function is strongly monotone (special case of strictly monotone). Note that the marginal profit function or F in our case is non-negative in at least one point. Otherwise, no firm has any incentive to produce in any market and the equilibrium is when all production quantities are equal to zero. In the next subsection, we aim to find this unique equilibrium of the NCC problem.

4.4.3 Designing a polynomial-time algorithm

In this subsection, we introduce Algorithm 6 for finding equilibrium of NCC, and show it converges in polynomial time by Theorem 4.12. This theorem requires the marginal profit function to satisfy Scaled Lipschitz Condition(SLC) and monotonicity. We first introduce SLC, and show how the marginal profit function satisfies SLC and montonicty by Lemmas 4.1 to 4.6. We argue the conditions that the cost and price functions should have in order for the marginal profit function to satisfy SLC and monotonicity in Lemma 4.6. Finally, in Theorem 4.12, we show convergence of our algorithm in polynomial time.

The following theorem states the performance guarantee of the algorithm proposed by [80].

Theorem 4.10 (Zhao Han Convergence Theorem) *Let $F : R^n \to R^n$ be the function associated with a complementarity problem satisfying the two following conditions:*

- *∇F is a positive semidefinite matrix for a constant scalar.*

- *F satisfies SLC; i.e., for some scalar $\lambda > 0$,*

$$\|X[F(x+h) - F(x) - \nabla F(x)h]\|_\infty \leq \lambda |h^T \nabla F(x) h|$$

holds $\forall x > 0$ and $\forall h$ satisfying $\|X^{-1}h\| \leq 1$.

Then the algorithm converges in time $O\Bigg(n \max(1,\lambda)\ \log(\mu_0/\epsilon)\Bigg)$ and outputs an approximate solution $(F(x^*), x^*)$ satisfying $(x^*)^T F(x^*)/n \leq \epsilon$ where $\mu_0 = (x_0)^T F(x_0)/n$, and $(F(x_0), x_0)$ is the initial feasible point.

Before introducing the next theorem, we explain what the Jacobians ∇R, ∇S, and ∇F are for the Cournot game. First note that these are $E \times E$ matrices. Let $(i,j) \in \mathcal{E}$ and $(l,k) \in \mathcal{E}$ be two edges of the graph. Let e_1 denote the index of edge (i,j), and e_2 denote the index of edge (l,k) in the graph as we discussed in the first section. Then the element in row e_1 and column e_2 of matrix ∇R, denoted $\nabla R_{e_1 e_2}$, is equal to $\frac{\partial r_{ij}}{\partial q_{lk}}$. We name the corresponding elements in ∇F and ∇S similarly. We have $\nabla F = \nabla R + \nabla S$ as $F = R + S$.

Definition 4.4 (Scaled Lipschitz Condition (SLC)) *A function $G : D \mapsto R^n$, $D \subseteq R^n$ is said to satisfy* Scaled Lipschitz Condition (SLC) *if there exists a scalar $\lambda > 0$ such that $\forall\ h \in R^n, \forall\ x \in D$, such that $\|X^{-1}h\| \leq 1$, we have:*

$$\|X[G(x+h) - G(x) - \nabla G(x)h]\|_\infty \leq \lambda |h^T \nabla G(x) h|, \qquad (4.4.8)$$

where X is a diagonal matrix with diagonal entries equal to elements of the vector x in the same order, i.e., $X_{ii} = x_i$ for all $i \in \mathcal{M}$.

Satisfying SLC and monotonicity are essential for marginal profit function in Theorem 4.12. In Lemma 4.6 we discuss the assumptions for cost and revenue function under which these conditions hold for our marginal profit function. We use Lemmas 4.1 to 4.6 to show F satisfies SLC. More specifically, we demonstrate in Lemma 4.1, if we can derive an upperbound for LHS of SLC for R and S, then we can derive an upperbound for LHS of SLC for $F = R + S$ too. Then in Lemma 4.2 and Lemma 4.3 we show LHS of S and R in SLC definition can be upperbounded. Afterwards, we show monotonicity of S in Lemma 4.5. In Lemma 4.6 we aim to prove F satifies SLC under some assumptions for cost and revenue functions. We use the fact that LHS of SLC for F can be upperbounded using Lemma 4.3 and Lemma 4.2 combined with Lemma 4.1. Then we use the fact that RHS of SLC can be upperbounded using strong monotonicity of R and Lemma 4.5. Using these two facts, we conclude F satisfies SLC in Lemma 4.6.

Lemma 4.1 *Let F, R, S be three $R^n \to R^n$ functions such that $F(q) = R(q) + S(q), \quad \forall q \in R^n$. Let R and S satisfy the following inequalities for some $C > 0$ and $\forall$ h such that $\|X^{-1}h\| \leq 1$:*

$$\|X[R(q+h) - R(q) - \nabla R(q)h]\|_\infty \leq C\|h\|^2,$$

$$\|X[S(q+h) - S(q) - \nabla S(q)h]\|_\infty \leq C\|h\|^2,$$

where X is the diagonal matrix with $X_{ii} = q_i$. Then we have:

$$\|X[F(q+h) - F(q) - \nabla F(q)h]\|_\infty \leq 2C\|h\|^2.$$

Proof: Definition of function F implies

$$\begin{aligned}\|X[F(q+h)-F(q)-\nabla F(q)h]\|_\infty =&\|X[R(q+h)-R(q)-\nabla R(q)h]\\&+X[S(q+h)-S(q)-\nabla S(q)h]\|_\infty\end{aligned}$$

applying triangle inequality gives

$$\begin{aligned}\|X[F(q+h)-F(q)-\nabla F(q)h]\|_\infty \leq&\|X[R(q+h)-R(q)-\nabla R(q)h]\|_\infty\\&+\|X[S(q+h)-S(q)-\nabla S(q)h]\|_\infty\end{aligned}$$

Combining with assumptions of the lemma, we have the required inequality. $\square$

The following lemmas give upper bounds for LHS of the SLC for S and R respectively.

Lemma 4.2 *Assume X is the diagonal matrix with $X_{ii} = q_i$. $\forall$ h such that $\|X^{-1}h\| \leq 1$, there exists a constant $C > 0$ satisfying: $\|X[S(q+h)-S(q)-\nabla S(q)h]\|_\infty \leq C\|h\|^2$.*

Proof: Let $m_{ij} = \frac{\partial c_i}{\partial q_{ij}}$. The element of vector $X(S(q+h)-S(q)-h\nabla S)$ corresponding to edge (i,j) is given by:

$$q_{ij}(m_{ij}(q+h)+m_{ij}(q)-h\nabla c_i(q))$$

Let $2L_3$ be an upper bound of Lipschitz constants for derivates of c_i's. Then, from Theorem 4.11 and upper bound Q on production quantities, we have:

$$|q_{ij}(m_{ij}(q+h)+m_{ij}(q)-h\nabla c_i(q))| \leq QL_3\|h\|^2$$

$\square$

Lemma 4.3 *Assume X is the diagonal matrix with $X_{ii} = q_i$. $\forall\ h$ such that $\|X^{-1}h\| \leq 1$, $\exists C > 0$ such that$\|X[R(q+h) - R(q) - \nabla R(q)h]\|_\infty \leq C\|h\|^2$.*

Proof: Before we proceed, we state the following theorem from analysis and Lemma 4.4.

Theorem 4.11 *[86] Let $f : R^n \mapsto R$ be a continuously differentiable function with Lipschitz gradient, i.e., for some scaler $c > 0$,*

$$\|\nabla f(x) - \nabla f(y)\| \leq c\|x - y\| \ \ \forall\ x, y \in R^n.$$

Then, we have $\forall\ x, y \in R^n$,

$$f(y) \leq f(x) + \nabla f(x)^T(x - y) + \frac{c}{2}\|y - x\|^2 \tag{4.4.9}$$

Lemma 4.4 *For any vector $x \in R^n$ and an arbitrary $S \subseteq [n]$, let $X = \sum_{i \in S} x_i$. Then we have $\sqrt{n}\|x\| \geq X$*

Proof: Let $Y = \sum_{i \in [n]} |x_i|$. Clearly, $|Y| \geq |X|$.

$$Y^2 = \sum_{i,j \in [n]} |x_i x_j| = \sum_{i<j} 2|x_i x_j| + \|x\|^2$$

Since, $s^2 + t^2 \geq 2st\ \forall\ s, t \in R$, we have

$$X^2 \leq Y^2 \leq \sum_{i<j}(x_i^2 + x_j^2) + \|x\|^2 = n\|x\|^2$$

□

Now we are ready to prove Lemma 4.3. First note that $R(q+h) - R(q) - \nabla R(q)h$ is an $E \times 1$ vector. Let $H_i = \sum_{j \in N_{\mathcal{M}}(i)} h_{ij}$. The element corresponding to edge $(i, j) \in \mathcal{E}$

in vector $R(q+h)$ is $P_i(D_i+H_i)+P_i'(D_i+H_i)(q_{ij}+h_{ij})$. Similarly, the element corresponding to edge $(i,j)\in\mathcal{E}$ in $R(q)$ is $P_i(D_i)+P_i'(D_i)q_{ij}$ whereas the corresponding element in $\nabla R(q)h$ is $\sum_{k\in N_{\mathcal{M}}} h_{ik}\frac{\partial r_{ij}}{\partial q_{ik}} = -\sum_{k\in N_{\mathcal{M}}(i)} h_{ik}(P_i'(D_i)+P_i''(D_i)q_{ij}))+ h_{ij}P_i'(D_i)$. Therefore, the element corresponding to edge $(i,j)\in\mathcal{E}$ in vector $R(q+h)-R(q)-\nabla R(q)h$ is:

$$\begin{aligned}-P_i(D_i+H_i)-&P_i'(D_i+H_i)(q_{ij}-h_{ij})-P_i(D_i)-P_i'(D_i)q_{ij}\\&+\sum_{k\in N_{\mathcal{M}}(i)} h_{ik}(P_i'(D_i)+P_i''(D_i)q_{ij})+h_{ij}P_i'(D_i).\end{aligned}$$

Besides, X is the diagonal matrix of size $E\times E$ with diagonal entries equal to elements of q in the same order. Therefore, $X[R(q+h)-R(q)-\nabla R(q)h]$ is an $E\times 1$ vector where the element corresponding to edge $(i,j)\in\mathcal{E}$ is q_{ij} multiplied by the element corresponding to edge (i,j) in vector $R(q+h)-R(q)-\nabla R(q)h$:

$$\begin{aligned}&-q_{ij}\Bigg(P_i(D_i+H_i)+P_i'(D_i+H_i)(q_{ij}+h_{ij})-P_i(D_i)-P_i'(D_i)q_{ij}\\&\quad-\sum_{k\in N_{\mathcal{M}}(i)} h_{ik}(P_i'(D_i)+P_i''(D_i)q_{ij})-h_{ij}P_i'(D_i)\Bigg)\\=&-q_{ij}\Bigg([P_i(D_i+H_i)-P_i(D_i)-H_iP_i'(D_i)]\\&\quad+[P_i'(D_i+H_i)-P_i'(D_i)-H_iP_i''(D_i)](q_{ij}+h_{ij})+h_{ij}H_iP_i''(D_i)\Bigg)\\\le& q_{ij}\Bigg(|P_i(D_i+H_i)-P_i(D_i)-H_iP_i'(D_i)|\\&\quad+|P_i'(D_i+H_i)-P_i'(D_i)-H_iP_i''(D_i)||(q_{ij}+h_{ij})|+|h_{ij}H_iP_i''(D_i)|\Bigg).\end{aligned}$$

Let P' and P'' be Lipschitz continuous functions with Lipschitz constants $2L_1$ and $2L_2$ respectively. To bound the last expression, we use Theorem 4.11 and Lemma

4.4

$$|P_i(D_i + H_i) - P_i(D_i) - H_i P_i'(D_i)| \leq L_1 H_i^2 \leq L_1 E\|h\|^2$$

$$|P_i'(D_i + H_i) - P_i'(D_i) - H_i P_i''(D_i)| \leq L_2 H_i^2 \leq L_2 E\|h\|^2$$

$$|h_{ij} H_i P_i''(D_i)| \leq E\|h\|^2 P_i''(D_i)$$

Then, from finiteness of derivatives, we have:

$$|h_{ij} H_i P_i''(D_i)| \leq E M_2 \|h\|^2$$

Thus, the LHS is bound from above by:

$$q_{ij} E\|h\|^2 (L_1 + L_2(q_{ij} + h_{ij}) + M_2)$$

Let Q be an upper bound on maximum profitable quantity for any producer in any market. Then the LHS is bound above by $C\|h\|^2$, where:

$$C = QE(L_1 + 2QL_2 + M_2) \tag{4.4.10}$$

□

If R is assumed to be strongly monotone, we immediately have a lower bound on RHS of the SLC for R. The following lemma gives a lower bound on RHS of the SLC for S.

Lemma 4.5 *If cost functions are (strongly) convex S is (strongly) monotone*[5][6][7][8].

[5]A matrix $M \in R^{n \times n}$ is *strongly positive definite* iff $\forall\ x \in R^n$ and some $\alpha > 0$ $x^T M x \geq \alpha \|x\|^2$.

[6] A differentiable function $f : D \mapsto R^n$ is *monotone* iff its Jacobian ∇f is positive semidefinite over its domain D.

[7] A differentiable function $f : D \mapsto R^n$ is *strongly monotone* iff its Jacobian ∇f is strongly positive definite over its domain, D.

[8]A twice differentiable function $f : D \mapsto R$ is *strongly convex* iff its Hessian $\nabla^2 f$ is strongly

Proof: Let $S' = \sum_{i \in \mathcal{F}} c_i$. Then S', being a sum of strongly convex functions, is a strongly convex function. Also, $S = \nabla S'$. Thus, $h^T \nabla^2 S' h = h^T \nabla S h$ is bounded from below by $\alpha_c \|h\|^2, \forall h \in R^n$ for some $\alpha_c > 0$ if the cost functions are strongly convex and $alpha_c = 0$ is cost functions are convex. □

The following lemma combines the results of Lemma 4.2 and Lemma 4.3 using Lemma 4.1 to derive an upper bound for LHS of the SLC for F. We bound RHS of the SLC from below by using strong montonicity of R and Lemma 4.5.

Lemma 4.6 *F satisfies SLC and is monotone if:*

1. *Cost functions are convex.*
2. *Marginal revenue function is monotone.*
3. *Cost functions are strongly convex or marginal revenue function is strongly monotone.*

Proof: From lemmas 4.3, 4.2 and 4.1, RHS of SLC for F is $O(E\|h\|^2)$. If cost functions are strongly convex or marginal revenue function is strongly monotone, then from Lemma 4.5 and definition of strong monotonicity, the LHS of SLC for F is $\Omega(\|h\|^2)$. Thus, F satisfies SLC. We note that F is a sum of two monotone functions and hence is monotone. □

We wrap up with the following theorem, which summarizes the main result of this section. Lemma 4.6 guarantees that our problem satisfies the two conditions mentioned in [80]. Therefore, we can prove the following theorem.

positive definite over its domain, D.

Theorem 4.12 *Algorithm 6 converges to an equilibrium of Network Cournot Competition in time $O\big(E^2 \log(\mu_0/\epsilon)\big)$ under the following assumptions:*

1. *The cost functions are strongly convex.*
2. *The marginal revenue function is strongly monotone.*
3. *The first derivative of cost functions and price functions and the second derivative of price functions are Lipschitz continuous.*

This algorithm outputs an approximate solution $(F(q^*), q^*)$ satisfying $(q^*)^T F(q^*)/n \leq \epsilon$ where $\mu_0 = (q_0)^T F(q_0)/n$, and $(F(q_0), q_0)$ is the initial feasible point [9].

4.4.4 Price Functions for Monotone Marginal Revenue Function

This section will be incomplete without a discussion of price functions that satisfy the convergence conditions for Algorithm 6. We will prove that a wide variety of price functions preserve monotonicity of the marginal revenue function. To this end, we prove the following lemma.

Lemma 4.7 *$\nabla R(q)$ is a positive semidefinite matrix $\forall\ q \geq 0$, i.e., R is monotone, provided that for all markets $|P_i'(D_i)| \geq \frac{|P_i''(D_i)|D_i}{2}$.*

[9]Initial feasible solution can be trivially found. E.g., it can be the same production quantity along each edge, large enough to ensure losses for all firms. Such quantity can easily be found by binary search between [0, Q].

Algorithm 6 Compute quantities at equilibrium for Generalized Cournot game.

Input: The price function P_i for each market $i \in \mathcal{M}$, the cost function c_j for each firm $j \in \mathcal{F}$, and $\epsilon > 0$

Ouput: The vector of quantities produced by all firms at equilibrium.

```
0: Calculate vector $F$ of length $E$ as defined in (4.4.4).
0: Find the initial feasible[10]solution $(F(x_0), x_0)$ for the complementarity problem.
   This solution should satisfy $x_0 \geq 0$ and $F(x_0) \geq 0$.
0: Run Algorithm 3.1 from [80] to find the solution $(F(x^*), x^*)$ to the CP charac-
   terized by $F$.
0: return $x^*$ =0
```

Proof: Let e_1 be the index of the edge (i, j) and e_2 be the index of edge (l, k). The elements of ∇R are as follows.

$$\nabla R_{e_1 e_1} = \begin{cases} \frac{\partial r_{ij}}{\partial q_{ij}} & = -2P_i'(D_i) - P_i''(D_i)q_{ij} \text{ if } e_1 = e_2 \\ \frac{\partial r_{ij}}{\partial q_{ik}} & = -P_i'(D_i) - P_i''(D_i)q_{ij} \text{ if } i = l, j \neq k \\ \frac{\partial r_{ij}}{\partial q_{lk}} & = 0 \text{ if } i \neq l, j \neq k. \end{cases}$$

We note that since price functions are functions only of the total production in their corresponding markets and not the individual quantities produced by firms, $\frac{\partial P_i'(D_i)}{\partial q_{ij}} = \frac{\partial P_i'(D_i)}{\partial q_{ik}}$. Therefore, we have replaced the partial derivatives by $P_i''(D_i)$.

We must show $x^T \nabla R(D_i)x$ is nonnegative $\forall x \in R^E$ and $\forall D_i \geq 0$.

$$\begin{aligned}
x^T(\nabla R(D_i))x &= \sum_{(i,j)\in\mathcal{E}} \sum_{(k,l)\in\mathcal{E}} x_{ij}x_{lk}\frac{\partial r_{ij}}{\partial x_{lk}} = \sum_{i\in\mathcal{M}} \sum_{j,k\in N_{\mathcal{M}}(i)} x_{ij}x_{ik}\frac{\partial r_{ij}}{\partial x_{ik}} \\
&= \sum_{i\in\mathcal{M}} \Bigg(\sum_{j\in N_{\mathcal{M}}(i)} x_{ij}^2 \left[-2P_i'(D_i) - P_i''(D_i)x_{ij}\right] \\
&\qquad + \sum_{j,k\in N_{\mathcal{M}}(i), j\neq k} x_{ij}x_{ik}\left[-P_i'(D_i) - P_i''(D_i)q_{ij}\right] \Bigg) \\
&= -\sum_{i\in\mathcal{M}} \left(\sum_{j\in N_{\mathcal{M}}(i)} x_{ij}^2 P_i'(D_i) + \sum_{j,k\in N_{\mathcal{M}}(i)} x_{ij}x_{ik}(P_i'(D_i) + P_i''(D_i)q_{ij}) \right) \\
&= -\sum_{i\in\mathcal{M}} \left(P_i'(D_i) \sum_{j\in N_{\mathcal{M}}(i)} x_{ij}^2 + P_i'(D_i) \sum_{j,k\in N_{\mathcal{M}}(i)} x_{ij}x_{ik} + P_i''(D_i) \sum_{j,k\in N_{\mathcal{M}}(i)} x_{ij}q_{ij}x_{ik} \right) \\
&\geq -\sum_{i\in\mathcal{M}} \left(P_i'(D_i) \sum_{j\in N_{\mathcal{M}}(i)} x_{ij}^2 + P_i'(D_i) \sum_{j,k\in N_{\mathcal{M}}(i)} x_{ij}x_{ik} - |P_i''(D_i)||q||x|| \sum_{j\in N_{\mathcal{M}}(i)} x_{ij}| \right) \\
&\geq \sum_{i\in\mathcal{M}} \left(-P_i'(D_i)|x|^2 - P_i'(D_i) \left(\sum_{j\in N_{\mathcal{M}}(i)} x_{ij} \right)^2 + |P_i''(D_i)|D_i|x|| \sum_{j\in N_{\mathcal{M}}(i)} x_{ij}| \right)
\end{aligned}$$

Since P_i's are decreasing functions, we have , $P_i'(D_i) \leq 0, \quad \forall i \in \mathcal{M}$. Thus, over domain of P_i's ($D_i \geq 0$), the above expression is non-negative if $|P_i''(D_i)|D_i \leq 2|P_i'(D_i)|$ Hence, $x^T(\nabla R(D_i))x \geq 0$ equivalently $\nabla R(D_i)$ is positive semidefinite. □

While the above condition may seem somewhat restrictive, they allow the problem to be solved on a wide range of price functions. Intuitively, the condition implies that linear and quadratic terms dominate higher order terms. We present the following corollaries as examples of classes of functions that satisfy the above condition.

Corollary 4.1 *All decreasing concave quadratic price functions satisfy Lemma 4.7.*

Corollary 4.2 *All decreasing concave cubic price functions satisfy Lemma 4.7.*

Corollary 4.3 *Let $a_i \in R^n_{\geq 0}$ for $i \in \{1 \dots k\}$ be arbitrary positive vectors. Let $f : R^n_{\geq 0} \mapsto R$ be the following function: $f(x) = \sum_{i \in \{1 \dots k\}} (a_i^T x) \log(a_i^T x)$. Then f (and $-f$) satisfies Lemma 4.7.*

4.5 Algorithm for Cournot Oligopoly

In this section we present a new algorithm for computing equilibrium quantities produced by firms in a Cournot oligopoly, i.e., when the firms compete over a single market. Cournot Oligopoly is a standard model in Economics and computation of Cournot Equilibrium is an important problem in its own right. A considerable body of literature has been dedicated to this problem [81–85]. All of the earlier works that compute Cournot equilibrium for a general class of price and cost functions rely on solving a Linear Complementarity Problem or a Variational Inequality which in turn are set up as convex optimization problems of size $O(n)$ where n is the number of firms in oligopoly. Thus, the runtime guarantee of the earlier works is $O(n^3)$ at best. We give a novel combinatorial algorithm for this important problem when the quantities produced are integral. Our algorithm runs in time $n \log^2(Q_{max})$ where Q_{max} is an upper bound on total quantity produced at equilibrium. We note that, for two reasons, the restriction to integral quantities is practically no restriction at all. Firstly, in real-world all commodities and products are traded in integral units. Secondly, this algorithm can easily be adapted to compute approximate Cournot-Nash equilibrium for the continuous case and since the quantities at equilibrium may not be rational numbers, this is the best we can hope for.

As we have only a single market rather than a set of markets, we make a few changes to the notation. Let $[n] = \{1, \dots, n\}$ be the set of firms competing over the single market. Let $\mathbf{q} = (q_1, q_2, \dots, q_n)$ be the set of all quantities produced by the firms. Note that in this case, each firm is associated with only one quantity. Let $Q = \sum_{i \in [n]} q_i$ be the sum of the total quantity of good produced in the market. In this case, there is only a single inverse demand function $P : Z \mapsto R_{\geq 0}$, which maps total supply, Q, to market price. We assume that price decreases as the total quantity produced by the firms increases, i.e., P is a decreasing function of Q. For each firm $i \in [n]$, the function $c_i : Z \mapsto R_{\geq 0}$ denotes the cost to this firm when it produces quantity q_i of the good in the market. The profit of firm $i \in [n]$ as a function of q_i and Q, denoted $\pi_i(q_i, Q)$, is $P(Q)q_i - c_i(q_i)$. Also let $f_i(q_i, Q) = \pi_i(q_i + 1, Q + 1) - \pi_i(q_i, Q)$ be the marginal profit for firm $i \in [n]$ of producing one extra unit of product. Although the quantities are nonnegative integers, for simplicity we assume the functions c_i, P, π_i and f_i are zero whenever any of their inputs are negative. Also, we refer to the forward difference $P(Q+1) - P(Q)$ by $P'(Q)$.

4.5.1 Polynomial time algorithm

We leverage the supermodularity of price functions and Topkis' Monotonicity Theorem [87] (Theorem 4.13) to design a nested binary search algorithm which finds the equilibrium quantity vector $\mathbf{q}$ when the price function is a decreasing function of Q and the cost functions of the firms are convex. Intuitively the algorithm works as follows. At each point we guess Q' to be the total quantity of good produced by all the firms. Then we check how good this guess is by computing for each firm the set of quantites that it can produce at equilibrium if we assume the total quantity is the fixed integer Q'. We prove that the set of possible quantities for each firm at equilibrium, assuming a fixed total production, is a consecutive set of integers. Let $I_i = \{q_i^l, q_i^l + 1, \ldots, q_i^u - 1, q_i^u\}$ be the range of all possible quantities for firm $i \in [n]$ assuming Q' is the total quantity produced in the market. We can conclude Q' was too low a guess if $\sum_{i\in[n]} q_i^l > Q'$. This implies our search should continue among total quantities above Q'. Similarly, if $\sum_{i\in[n]} q_i^u < Q'$, we can conclude our guess was too high, and the search should continues among total quantities below Q'. If neither case happens, then for each firm $i \in [n]$, there exists a $q_i' \in I_i$ such that $Q' = \sum_{i\in[n]} q_i'$ and firm i has no incentive to change this quantity if the total quantity is Q'. This means that the set $\mathbf{q}' = \{q_1', \ldots, q_n'\}$ forms an equilibrium of the game and the search is over.

The pseudocode for the algorithm is provided in Algorithm 7, whose correctness we prove next. The rest of this section is dedicated to proving Theorem 4.14. Here, we present a brief outline of the proof. To help with the proof we define the functions F_i and G_i as follows. Let $F_i(q_i, Q) = P(Q+1)q_i + \frac{P'(Q)}{2}(q_i - \frac{1}{2})^2 - c(q_i)$. We note that the first difference of $F(q_i, Q)$ is the marginal profit for firm i for producing one more quantity given that the total production quantity is Q and firm i is producing q_i. Let $G_i(q_i, Q) = F_i(q_i, Q-1)$. The first difference of $G_i(q_i, Q)$ is the marginal loss for firm i for producing one less quantity given that the total production quantity

Algorithm 7 Compute quantities at Equilibrium in a Cournot oligopoly.

Input: Price function P and all cost functions c_i.

Output: $\vec{q}$, vector of quantities produced by firms at the game's equilibrium.

```
0: Let $Q_{\min} := 1$
0: Let $Q_i^*$ be the monopoly optimum quantity of firm $i$.
0: Let $Q_{\max} := \sum_{i\in[n]} Q_i^*$
0: while $Q_{\min} \leq Q_{\max}$ do
0:    $Q' := \lfloor \frac{Q_{\min}+Q_{\max}}{2} \rfloor$
0:    for all $i \in [n]$ do
0:       Binary search to find the minimum nonnegative integer $q_i^l$ satisfying
0:       $f_i(q_i^l, Q') = \pi_i(q_i^l + 1, Q' + 1) - \pi_i(q_i^l, Q') \leq 0$
0:       Binary search to find the maximum integer $q_i^u \leq Q' + 1$ satisfying
0:       $f_i(q_i^u - 1, Q' - 1) = \pi_i(q_i^u, Q') - \pi_i(q_i^u - 1, Q' - 1) \geq 0$
0:       Let $I_i = \{q_i^l, \dots, q_i^u\}$ be the set of all integers between $q_i^l$ and $q_i^u$.
0:    end for
0:    if $\Sigma_{i\in[n]} q_i^l > Q'$ then
0:       $Q_{\min} := Q' + 1$
0:    else if $\Sigma_{i\in[n]} q_i^u < Q'$ then
0:       $Q_{\max} := Q' - 1$
0:    else
0:       Find $\mathbf{q} = (q_1, \dots, q_n)$ such that $q_i \in I_i$ and $\sum_{i\in[n]} q_i = Q'$
0:       return $\mathbf{q}$
0:    end if
0: end while=0
```

is Q and firm i is producing q_i. Maximizers of these functions are closely related to equilibrium quantities a firm can produce given that the total quantity in market is Q. We make this connection precise and prove the validity of binary search in Lines 8-12 of Algorithm 7 in Lemma 4.8. In Lemma 4.9, we prove that F_i and G_i are supermodular functions of q_i and $-Q$. In lemmas 4.10 and 4.11, we use Topkis' Monotonicity Theorem to prove the monotonicity of maximizers of F_i and G_i. This, along with lemmas 4.12 and 4.13 leads to the conclusion that the outer loop for finding total quatity at equilibrium is valid as well and hence the algorithm is correct.

4.5.2 Proof of correctness

Throughout this section *we assume that the price function is decreasing and concave and the cost functions are convex.*

Lemma 4.8 *Let $q_i^*(Q) = \{q_i^l \dots q_i^u\}$, where $q_i^l = \min argmax_{q_i \in \{0 \dots Q_{max}\}} F_i(q_i, Q)$ and $q_i^u = \max argmax_{q_i \in \{0 \dots Q_{max}\}} G_i(q_i, Q)$. Then $q_i^*(Q)$ is the set of consecutive integers I_i given by binary search in lines 8-12 of Algorithm 7. This is the set of quantities firm i can produce at equilibrium given that the total quantity produced is Q.*

Proof: Again let $P'(Q) = P(Q+1) - P(Q)$ be the forward difference of the price function, and let $c_i'(q_i) = c_i(q_i + 1) - c_i(q_i)$. From definition of profit function π_i and f_i, we have $f_i(q_i, Q) = P(Q+1) + P'(Q)q_i - c_i'(q_i)$. Assume Q is fixed.

Suppose we have $q_i < \tilde{q}_i$. The marginal profit of firm at production quantity q_i is $P(Q+1) + P'(Q)q_i - c_i'(q_i)$ whereas the marginal profit at production quantity $\tilde{q}_i$ is $P(Q+1) + P'(Q)\tilde{q}_i - c_i'(\tilde{q}_i)$. Thus, $P(Q+1) + P'(Q)q_i > P(Q+1) + P'(Q)\tilde{q}_i$ since $P'(Q)$ is negative (from concavity of P) and $q_i < \tilde{q}_i$. As the discrete cost functions are convex, we have $c_i'(q_i) < c_i'(\tilde{q}_i)$. This implies $f_i(q_i, Q) > f_i(\tilde{q}_i, Q)$ when $q_i < \tilde{q}_i$. Thus, for a fixed Q, $f_i(q_i, Q)$ is a non-increasing function of q_i. Similarly, we can see that $f_i(q_i, Q)$ is a non-increasing function of Q. From definitions of F_i and G_i, we have:

$$F_i(q_i + 1, Q) - F_i(q_i, Q) = f_i(q_i, Q) \tag{4.5.11}$$

$$G_i(q_i + 1, Q) - G_i(q_i, Q) = F_i(q_i + 1, Q - 1) - F_i(q_i, Q - 1) = f_i(q_i, Q - 1) \tag{4.5.12}$$

For a fixed Q, Let q_l be the minimum maximizer of $F_i(q_i, Q)$. Then $f_i(q_l - 1, Q) > 0$. Let q_u be the maximum maximizer of $G_i(q_i, Q)$. Because f_i is non-increasing, we have $f_i(q_l - 1, Q - 1) \geq f_i(q_l - 1, Q) > 0$. Thus, any number smaller than q_l cannot be a maximizer of G_i and we have $q_l \leq q_u$. Let $q \in \{q_l \ldots q_u\}$. Then, because $q \geq q_l$ we have $f_i(q, Q) \leq 0$ and from $q \leq q_u$, we have $f_i(q - 1, Q - 1) \geq 0$. Thus, q is an equilibrium quantity when total production quantity is Q. If $q < q_l$, then $f_i(q, Q) > 0$ and if $q > q_u$ then $f_i(q - 1, Q - 1) > 0$. Thus $\{q_l \ldots q_u\}$ is the set of equilibrium quantities. In Line 4.5.1 of Algorithm 7, we are searching for the minimum maximizer of F_i and in Line 4.5.1 we are searching for maximum maximizer of G_i. Binary search for these quantities is valid because first differences for both functions (equations 4.5.11 and 4.5.12) are decreasing. $\square$

Lemma 4.9 *Let $F_i^-(q_i, -Q) = F_i(q_i, Q)$ and $G_i^- = G_i(q_i, -Q)$. Then F_i^- and G_i^- are supermodular functions.*

Proof: We use the following definition from submodular optimization in the lemma.

Definition 4.5 *Given lattices $(X_1, \geq)$ and $(X_2, \geq)$, $f : X_1 \times X_2 \mapsto R$ is supermodular iff for any $x_1, y_1 \in X_1; x_2, y_2 \in X_2$ such that $x_1 \geq y_1$ and $x_2 \geq y_2$, the following holds:*

$$f(x_1, y_2) - f(x_1, x_2) \geq f(y_1, y_2) - f(y_1, x_2)$$

We have, $F_i(q_i, Q) = P(Q+1)q_i + \frac{P'(Q)}{2}(q_i - \frac{1}{2})^2 - c_i(q_i)$. Let $-Q_1 > -Q_2$. Let $q_i' > q_i$. Then, we have:

$$F_i(q_i, Q_1) - F_i(q_i, Q_2) = (P(Q_1+1) - P(Q_2+1))q_i + \frac{P'(Q_1) - P'(Q_2)}{2}(q_i - \frac{1}{2})^2$$
$$F_i(q_i', Q_1) - F_i(q_i', Q_2) = (P(Q_1+1) - P(Q_2+1))q_i' + \frac{P'(Q_1) - P'(Q_2)}{2}(q_i' - \frac{1}{2})^2$$

Since P and P' are a decreasing functions, we have $P(Q_1) \geq P(Q_2)$ and $P'(Q_1) \geq P'(Q_2)$. From this and the fact that $q_i' > q_i$, we have:

$$F_i(q_i', Q_1) - F_i(q_i', Q_2) \geq F_i(q_i, Q_1) - F_i(q_i, Q_2)$$

Therefore F_i^- is a supermodular function. Since $G_i(q_i, Q) = F_i(q_i, Q-1)$, may a similar argument we can conclude that G_i^- is supermodular. □

Lemma 4.10 *Let $I = \{q_i^l, \ldots, q_i^u\} = q_i^F(Q) = argmax_{q_i \in \{1 \ldots Q_{max}\}} F_i(q_i, Q)$ and $I' = \{q_i'^l, \ldots, q_i'^u\} = q_i^F(Q') = argmax_{q_i \in \{1 \ldots Q_{max}\}} F_i(q_i, Q')$. Let $Q > Q'$. Then $q_i'^l \geq q_i^l$ and $q_i'^u \geq q_i^u$.*

Proof: We need the following definition and Topkis' Monotonicity Theorem for proving the lemma.

Definition 4.6 *Given a lattice* $(X, \geq)$*, we define* Strong Set Ordering *over* $A, B \subseteq X$*. We say* $A \geq_s B$ *iff* $\forall a \in A, \forall b \in B, \max\{a, b\} \in A \wedge \min\{a, b\} \in B$*.*

We note that the strong set ordering induces a natural ordering on sets of consecutive integers. Let $I_1 = \{l_1, \ldots, u_1\}$. Let $I_2 = \{l_2, \ldots, u_2\}$. Then $I_1 \geq_s I_2$ iff $l_1 \geq l_2$ and $u_1 \geq u_2$.

Theorem 4.13 (Topkis' Monotonicity Theorem [87]) *For any lattices* $(X, \geq)$ *and* $(T, \geq)$*, let* $f : X \times T \mapsto R$ *be a supermodular function and let* $x^*(t) = argmax_{x \in X} f(x, t)$*. If* $t \geq t'$*, then* $x^*(t) \geq_s x^*(t')$*, i.e.,* $x^*(t)$ *is non-decreasing in* t*.*

We note that in the theorem above, strong set ordering is used over x^* because *argmax* returns a set of values from lattice X.

Now we are ready to prove Lemma 4.10. From Lemma 4.9, $F_i^-(q_i, -Q)$ is a supermodular function. Thus, from Theorem 4.13, $q_i^F(Q) = argmax F_i(q_i, Q)$ is a non-decreasing function of $-Q$, i.e., q_i^F is a non-increasing function of Q. Thus $Q > Q' \implies I' \geq_s I$. As noted above, strong set ordering on a set of consecutive integers implies that $q_i'^l \geq q_i^l$ and $q_i'^u \geq q_i^u$. □

Lemma 4.11 *Let* $I = \{q_i^l, \ldots, q_i^u\} = q_i^G(Q) = argmax_{q_i \in \{1 \ldots Q_{max}\}} G_i(q_i, Q)$ *and* $I' = \{q_i'^l, \ldots, q_i'^u\} = q_i^G(Q') = argmax_{q_i \in \{1 \ldots Q_{max}\}} F_i(q_i, Q')$*. Let* $Q > Q'$*. Then* $q_i'^l \geq q_i^l$ *and* $q_i'^u \geq q_i^u$*.*

Proof: From Lemma 4.9, $G_i^-(q_i, -Q)$ is a supermodular function. Thus, from Theorem 4.13, $q_i^G(Q) = argmax F_i(q_i, Q)$ is a non-decreasing function of $-Q$, i.e., q_i^G is a non-increasing function of Q. Thus $Q > Q' \implies I' \geq_s I$. As noted above, strong set ordering on a set of consecutive integers implies that $q_i'^l \geq q_i^l$ and $q_i'^u \geq q_i^u$.

□

Lemma 4.12 *Let Q be total production quantity guessed by Algorithm 7 at a step of outer binary search. Let $I = (I_1, \ldots, I_n)$, where $I_i = \{q_i^l, \ldots, q_i^u\}$, be the set of best reponse ranges of all firms if the total quantity is a fixed integer Q. Then, if $\sum_{i=1}^n q_i^u < Q$, there does not exist any equilibrium for which the total produced quantity is greater than or equal to Q.*

Proof: Assume for contradiction that such an equilibrium exists for total quantity $Q' > Q$. From Lemma 4.11, we have $q_i^G(Q) \geq_s q_i^G(Q') = \{q_i'^l \ldots q_i'^u\}$. Thus, we have $q_i^u \geq q_i'^u$. Since Q' is an equilibrium quantity, $\sum_{i=1}^n q_i'^u \geq Q'$. Thus, we have $Q' < Q$ and this is a contradiction. □

Lemma 4.13 *Let Q be total production quantity guessed by Algorithm 7 at a step of outer binary search. Let $I = (I_1, \ldots, I_n)$, where $I_i = \{q_i^l, \ldots, q_i^u\}$, be the set of best reponse ranges of all firms if the total quantity is a fixed integer Q. Then, if $\sum_{i=1}^n q_i^l > Q$, there does not exist any equilibrium for which the total produced quantity is less than or equal to Q.*

Proof: Assume for contradiction that such an equilibrium exists for total quantity $Q' < Q$. From Lemma 4.10, we have $q_i^F(Q) \leq_s q_i^F(Q') = \{q_i'^l \ldots q_i'^u\}$. Thus, we have

$q_i^l \leq q_i'^l$. Since Q' is an equilibrium quantity, $\sum_{i=1}^{n} q_i'^l leq Q'$. Thus, we have $Q' > Q$ and this is a contradiction. □

Finally, the results of this section culiminate in the following theorem.

Theorem 4.14 *Algorithm 7 successfully computes the vector* $\mathbf{q} = (q_1, q_2, \ldots, q_n)$ *of quantities at one equilibrium of the Cournot oligopoly if the price function is decreasing and concave and the cost function is convex. In addition, the algorithm runs in time* $O(n \log^2(Q_{\max}))$ *where* $Q_{\max}$ *is the maximum possible total quantity in the oligopoly network at any equilibrium.*

Proof: Lemma 4.8 guarantees that the inner binary search successfully finds the best response range for all firms. Lemmas 4.12 and 4.13 ensure that the algorithm always continues its search for the total quantity at equilibrium in the segment where all the equilibria are. Thus, when the search is over, it must be at an equilibrium of the game if one exists. If an equilibrium does not exist, then the algorithm will stop when it has eliminated all quantities in $\{1 \ldots Q_{max}\}$ as possible total equilibrium production quantities. Let $Q_{\max}$ be the maximum total quantity possible at any equilibrium of the oligopoly network. Our algorithm performs a binary search over all possible quantities in $[1, Q_{\max}]$, and at each step finds a range of quantities for each firm $i \in [n]$ using another binary search. This means the algorithm runs in time $O(n \log^2(Q_{\max}))$. We can find an upper bound for $Q_{\max}$, noting that $Q_{\max}$ is at most the sum of the production quantites of the firms when they are the only producer in the market; i.e, $Q_{\max} \leq \sum_{i \in [n]} Q_i^*$ where $Q_i^* = q_i^*(q_i)$ is the optimal quantity to be produced by firm i when there is no other firms to compete with. □

4.6 Conclusion

In this chapter, a generalized version of the Cournot Competiton was introduced and studied. Instead of assuming firms produce a single product and compete in a single market, we take into account a more realistic model where each firm can produce any subset of goods corresponding to different markets. As a result, the cost function for each firm can be any complex function of different quantities of goods produced in different markets. We provide both optimal and approximation algorithms which find the quantities produced by each firm (in each of the markets it has access to) in the equilibria of the game. We try to generalize the firms' cost functions as well as markets' price functions as much as possible. Finally, for the traditional version of Cournot Competition, we provide a combinatorial algorithm for finding quantities at equilibria which is significantly faster than the previous algorithms before this work.

Chapter 5: Market Pricing for Data Streams

5.1 Introduction

Modern, Internet-enabled marketplaces have the potential to serve an extremely large volume of transactions. Giant online markets such as eBay and Amazon work with massive datasets that record the exchange of goods between many different buyers and sellers. Such datasets present an opportunity: it is natural to analyze the history of transactions to estimate demand and solve central pricing problems. Indeed, a recent and exciting line of literature in the algorithmic game theory community has set out to understand how the availability of data, in the form of samples from a transaction history, can be employed to tune prices and design mechanisms [88–92]. Big-data environments are a boon for such tasks. However, as datasets grow ever larger, data-analysis algorithms must become ever more efficient. An algorithm that runs in polynomial (or even linear) time may not have a reasonable running time in practice.

In this paper, we study the development of pricing algorithms that are appropriate for use with massive datasets. We will adopt the model of streaming algorithms, a standard model for massive dataset analysis. In the streaming algorithm model, a stream of data arrives sequentially and must be analyzed by an algorithm with limited memory. These streams can only be read once (or a limited number of times), and hence streaming algorithms must be frugal in the amount and nature of data that they choose to store.

Streaming algorithms were first theoretically introduced in fields such as data mining and machine learning over 20 years ago in order to model problems in which the data cannot be accessed all at once. Over the past decade, there has been a significant demand for algorithms to process and handle dynamic data coming from huge and growing graphs such as social networks, webpages and their links and citations of academic work. These algorithmic techniques are also relevant to market design problems. For instance, one might imagine that there is a set of items (e.g., goods for sale) and a set of potential buyers (e.g., individual consumers) to which they should be matched. Such markets are essentially bipartite matching problems, which have themselves been the subject of study in the context of streaming algorithms. This begs the question: to what extent can market design and pricing problems be resolved adequately in the streaming model?

We have in mind two main applications of solving this pricing problem on a massive collection of static data, especially on the data of previous sales. First, by computing optimal prices on past transactions, one can subsequently employ those prices as a guideline for setting future prices; this is a key step in many recent pricing methodologies based on stastical learning theory. Also, the optimal welfare or revenue in hindsight is a useful benchmark for the online pricing mechanism being employed by the platform, and can therefore be used to evaluate and tune.

5.1.1 Our Results

In this paper, we instantiate our high-level question by focusing on the *envy-free pricing problem* with big data. Our model is as follows. Suppose there is a bipartite graph G with a set of n unit-demand buyers $b_1, \ldots, b_n$ on one side, and a set of k distinct types of items $v_1, \ldots, v_k$, with l copies of each on the other side. The utility of buyer b_j for item v_i is denoted by u_{v_i,b_j}, and is shown by a weighted edge between the corresponding two vertices. The price assigned to item i is denoted by p_i. The goal is to assign prices to items, and then items to buyers, such that the assignment is *envy-free*[1], i.e., each buyer prefers the item assigned to her rather than an item assigned to another buyer. Subject to the envy-freeness condition, the designer wishes to maximize either the social welfare or revenue of the corresponding allocation. This is precisely the envy-free pricing problem introduced by Guruswami et al. [93]. We ask: how well can envy-free prices be computed in the *streaming setting*?

We note that there are many possible representations of the input as a data stream. We will perform our analysis under a model in which the utility values u_{v_i,b_j} arrive in a data stream in an arbitrary order. We note that there are other potential

[1]If we assign item v_{i_1} to buyer b_{j_1} and item v_{i_2} to buyer b_{j_2}, then we have $u_{v_{i_1},b_{j_1}} - p(i_1) \geq u_{v_{i_2},b_{j_1}} - p(i_2)$.

options, such as assuming that all values associated with a certain agent arrive simultaneously, or that the values u_{v_i,b_j} are not provided directly but rather the input contains only the revealed preference of a buyer in response to prices. We leave the exploration of these alternative models as an avenue for future work.

We consider both social-welfare maximization and revenue maximization versions of the envy-free pricing problem. First, we provide streaming mechanisms that compute both allocation and prices of the items using $O(k^2 l)$ space. Later, we present streaming mechanisms that only compute the prices using space $\tilde{O}(k^3)$, approximating social welfare (or revenue) within a factor of $1-\epsilon$, where poly-logarithmic factors are hidden in the notation of $\tilde{O}$. At the end, we present lower bounds on the required space of any mechanism that computes optimum prices for either social-welfare maximization or revenue maximization.

In Theorem 5.1 we provide an envy-free streaming mechanism for the social-welfare maximization problem using $O(k^2 l)$ space.

Theorem 5.1 *There exists an envy-free mechanism for the social-welfare maximization problem in the streaming setting using $O(k^2 l)$ space. This mechanism remembers the allocation as well as the prices.*

Indeed, finding the maximum matching in a bipartite graph with $O(k)$ vertices in the streaming setting requires $\Omega(k^2)$ space [94]. Thus, for $l = 1$, the space required by our mechanisms in Theorem 5.2 and Theorem 5.1 are tight.

The following theorem extends our result to the revenue maximization problem.

Even in the static (i.e., non-streaming) environment, this problem has resisted constant-factor approximation factors for simple versions, including the case of unit-demand bidders studied here. We frame our result as a reduction: given an algorithm for computing envy-free prices in the static setting, we show how to construct a streaming algorithm with the same approximation guarantee. As with the welfare maximization problem, the required space is $O(k^2 l)$.

Theorem 5.2 *Given an envy-free α-approximation mechanism for the revenue maximization problem, there exists an envy-free α-approximation mechanism for the revenue maximization problem in the streaming setting using $O(k^2 l)$ space. This mechanism remembers the allocation as well as the prices.*

Each of the above results are with respect to algorithms that return not only a profile of envy-free prices, but also the corresponding allocation. We note that the size of the allocation is $O(kl)$, and thus any mechanism that remembers the allocation requires at least $\Omega(kl)$ space. This space may be quite large when l is large, which is not desirable. What if we are only interested in determining the envy-free prices, and just being within an approximation of the maximum social welfare(or revenue)? As it turns out, this variation of the problem allows significant improvement when l is large. We provide an almost optimal streaming mechanism using $\tilde{O}(k^3)$ space that computes prices. That is, the required space here is poly-logarithmic in the number of buyers and the number of copies of each item type. The following theorem states our result for the social-welfare maximization problem.

Theorem 5.3 *Let ϵ be an arbitrary small constant. There exists a streaming mechanism for the social-welfare maximization problem which with high probability gives an envy-free $(1-\epsilon)$-approximate solution using $\tilde{O}(k^3)$ space. This mechanism only remembers the prices.*

The following theorem extends our results to the revenue maximization problem as well. Note that, again here the required space is poly-logarithmic in the number of buyers and the number of copies of each item type.

Theorem 5.4 *Given an envy-free α-approximation mechanism for the revenue maximization problem, and any small constant ϵ, there exists a streaming mechanism for the revenue maximization problem which with high probability gives an envy-free $(1-\epsilon)\alpha$-approximate solution using $\tilde{O}(k^3)$ space. This mechanism only remembers the prices.*

To show that the approximation in Theorem 5.3 is necessary, we prove there is no streaming mechanism to find the prices that maintain the optimal social-welfare using space sublinear in l.

Theorem 5.5 *There is no envy-free streaming mechanism that finds the welfare-optimal envy-free prices using space $o(l)$. This bound holds even for $k = 2$.*

As with welfare maximization, any algorithm that computes revenue-optimal envy-free prices would require space that is at least linear in l.

Theorem 5.6 *There is no envy-free streaming mechanism which finds the set of prices that maximize the revenue using a space sublinear in l. This bound holds even for $k = 2$.*

5.1.2 Related Work

In this paper we focus our attention on the problem of finding envy-free prices for unit-demand bidders in the streaming setting, a problem that has received much attention in the static setting. The revenue-maximizing envy-free pricing problem was introduced by Guruswami et al. [93]. There has since been a significant line of work attacking variants of this problem [95–97], and mounting evidence suggests that it is computationally hard to obtain better than a polylogarithmic approximation for general unit-demand bidders [98, 99]. For welfare maximization, it is well-known that a Walrasian equilibrium corresponds to a set of envy-free prices that optimizes welfare, and such an equilibrium always exists for unit-demand bidders. Moreover, in the static setting such prices can be found in polynomial time [100, 101]. Our focus is on developing streaming algorithms for these problems.

Our motivation of determining prices from sampled data relates to a recent line of literature on the sample complexity of pricing problems and applications of statistical learning theory. Much of this work has focused on the problem of learning an approximately revenue-optimal reserve price in a single-item auction [89,90,102,103]. More generally, statistical learning methods have been used to quantify the sampling complexity of learning approximately optimal auctions, in the prior-free context by Balcan et al. [88] and in a prior-independent setting by Morgenstern and Roughgarden [92].

Hsu et al. [91] study the genericity of market-clearing prices learned from sampled data, and demonstrate that under some conditions on buyer preferences (including the unit-demand case studied here) prices computed from a large dataset will approximately clear a "similar" market; that is, one where buyer preferences are drawn from a the same underlying distribution.

Our technical results build upon recent work in the streaming algorithms literature on maximum matching. Chitnis et al. [94] consider the matching problem in the streaming setting and provide optimum solutions to both vertex cover and matching in $\tilde{O}(k^2)$ space, where k is the size of the solution. In addition, they show that any streaming algorithm for the maximum matching problem requires $\Omega(k^2)$ space. Later, they extend this result to dynamic streams in which we have both addition and deletion of edges [104].

McGregor [105] considered the matching problem in the streaming setting with several passes. He provides a $(1-\epsilon)$-approximation algorithm for unweighted graphs and a $(0.5 - \epsilon)$-approximation algorithm for weighted graphs, both with constant number of passes and using $\tilde{O}(n)$ space.

Kapralov et al. [106] provide an streaming algorithm that estimates the size of a maximum matching in the random order setting (i.e., the graph is chosen by an adversary, but the order of arrival of edges is chosen uniformly at random among all permutations). They provide a ploylogarithmic approximation of the size of maximum matching, using a polylogarithmic space.

Later, Esfandiari et al. [107] consider the maximum matching problem in planer graphs and bounded arboricity graphs. They provide a constant approximation of the size of a maximum matching in these graphs using $\tilde{O}(n^{2/3})$ space in the streaming setting. Later, simultaneously Bury et al. [108] and Chitnis et al. [104] extend this algorithm to work for both addition and deletion of edges using a larger space of $\tilde{O}(n^{4/5})$.

5.2 Pricing problem: Maximizing Social Welfare

In this section, we consider the problem of assigning prices to items, and items to buyers in a streaming setting such that the assignment would be envy-free, and the *social welfare* is maximized. The social welfare would be sum of the weights (or utilities) of the assigned edges. As we explained earlier, we only use $O(k^2 l)$ memory for storage of the stream of edges. Our approach is to store the $kl + 1$ edges with maximum weight for each item, and to run the optimum algorithm to find the social welfare maximizing envy-free assignment in offline setting when the stream ends. We call the optimum streaming algorithm of this section SWM to use it in Section 5.4. Let G be a weighted bipartite graph with k vertices corresponding to the k item types on one side, and n vertices corresponding to the buyers on the other side. The weight of the edges denote the utilities of buyers for item types.

Each item type, can be sold to at most l buyers. In other words, there are l copies of each item type available for sale. Let G' be the graph constructed from graph G, such that for each item we only keep kl edges with maximum weight (breaking ties arbitrarily), and remove the rest of edges from the graph. Note that a feasible solution matches each item to at most l buyers. Here, we slightly abuse the notation and simply call this structure a matching.

The next lemma shows that removing these edges does not affect the weight of the maximum matching.

Lemma 5.1 *The value of the maximum weighted matching in G is equal to the value of the maximum weighted matching in G'.*

Proof: Let M' be a maximum weighted matching in G', and M be the maximum weighted matching in G which has the maximum number of intersecting edges with G'. Since removal of edges from a graph does not increase the weight of the maximum matching, it is clear that the weight of M is not less than the weight of M'. It remains to show that the reverse is also true. If M only contains edges present in G', then clearly the weight of M and M' would be equal. Suppose in contrast that M contains an edge $e = (v_i, b_j)$ (between buyer b_j and item v_i) which does not belong to G'. Note that M has exactly kl edges and therefore, one of the neighbors of item v_i in G' must be unassigned by matching M. Let $b_{j'}$ be this neighbor. Remove edge e from M and replace it with edge $e' = (v_i, b_{j'})$. We know that the weight of edge e' is bigger than or equal to the weight of e. In case the weight of e is bigger than weight of e', the new matching is bigger than M. On the other hand, if the weights are equal, we have another maximum weighted matching in G which shares more edges with the edges of G'. Both cases lead to a contradiction with the assumption about M. Therefore, the weight of maximum matchings in both graphs should be equal. $\square$

In the next step, we build another graph H from graphs G and G' as follows. Starting from graph G', we build H by adding a dummy buyer vertex d_i for each vertex v_i corresponding to an item type in G. The weight of the edge between d_i and v_i denoted by u_{v_i,d_i} would be equal to the weight of $(kl+1)$-th maximum weight edge connected to item v_i in graph G.

Lemma 5.2 *There is a maximum weighted matching in H which does not include any dummy buyer vertices.*

Proof: Note that G' can be viewed as a subgraph of graph H in which for each item type we only keep kl edges connected to it with maximum weight. The rest follows from Lemma 5.1. $\square$

Finally, we present our pricing and allocation rule. Recall that we keep graph G' and graph H using $O(kl^2)$ available memory while the edges are being streamed. By Lemma 5.2, we know that there exists a maximum weighted matching M in graph H which is a subgraph of graph G' (does not contain any of dummy vertices in graph H). Consider a minimum weight vertex cover C corresponding to this matching such that the values assigned to all dummy vertices would be zero (Since dummy vertices are not matched by M, such a vertex cover can be found). For a vertex z, let c_z denote the value assigned to z by vertex cover C. We set the price of item v_i to its value in our vertex cover c_{v_i}, and assign it to its matched buyer in M. Since the social welfare in this case would be the weight of M which is a maximum weighted matching, it is clear that our algorithm maximizes social welfare. It is left to show that our algorithm produces an envy-free assignment.

Lemma 5.3 *Our assignment algorithm is envy-free.*

Proof: Suppose for contradiction our assignment algorithm is not envy-free, and there exists a buyer b_j who prefers item v_q over item v_i that our algorithm assigned to her. Consider the dummy buyer vertex d_q that we added and connected to item v_q in graph H. As we argued, our chosen vertex cover C, sets $c_{d_k} = 0$ for every dummy vertex d_k. Therefore, the value assigned to item v_q by our vertex cover must be at least the weight of the edge between v_q and d_q, i.e, $c_{v_q} \geq u_{v_q,d_q}$. Also, recall that our algorithm sets the price of item v_q to c_{v_q}. Hence, since buyer b_j prefers item v_q, her profit for this item must be non-negative, i.e, $u_{v_q,b_j} - c_{v_q} \geq 0$. From the above two inequalities we have $u_{v_q,b_j} \geq u_{v_q,d_q}$ which means the weight of the edge between buyer b_j and item v_q is bigger than the weight of the $kl+1$-th biggest edge connected to item v_q. From this statement we can argue that graph G' contains the edge (b_j, v_q), and thus $c_{b_j} + c_{v_q} \geq u_{v_q,b_j}$. Furthermore, since the edge (b_j, v_i) belongs to the maximum weighted matching, we must have

$$c_{b_j} + c_{v_i} = u_{v_i,b_j} \Rightarrow u_{v_i,b_j} - c_{v_i} = c_{b_j} \geq u_{v_q,b_j} - c_{v_q}$$

This is a contradiction to the fact that buyer b_j prefers item v_q to item v_i assigned to her by our algorithm. □

The following theorem is the main result of this section.

Theorem 5.7 *Our streaming assignment algorithm which assigns prices to items and items to buyers in the aforementioned market is an envy-free social welfare maximizing assignment, and it uses $O(kl^2)$ memory.*

Proof: As we discussed above, we only keep graphs G' and H which have $O(k^2l)$ edges. As we explained in this section, our algorithm suggest an assignment that maximizes social welfare. Furthermore, by Lemma 5.3 this assignment is also envy-free. □

5.3 Pricing problem: Maximizing Revenue

Just like the previous section, we try to find a pricing for items, and an envy-free assignment of items to buyers in our described market when the input is revealed in a streaming fashion. However, in this section, we aim to maximize *revenue* instead of social welfare. Here we show that if we are given an envy-free α-approximation mechanism for the revenue maximization problem then we can have an envy-free α-approximation mechanism for the revenue maximization problem in the streaming setting with $O(k^2l)$ available memory. We call this mechanism designed for the streaming setting RM for use in the later sections.

First, we construct graph G' from graph G by keeping only the $kl + 1$ largest edges connected to each item while the edges are being streamed.

Let M be the envy-free assignment in G which yields the maximum revenue and has the maximum number of intersecting edges with the edges of G'. If we prove that G' includes all the edges in M, then we can say M is a feasible envy-free assignment in G' and conclude that the optimum solution in G' is at least as good as the optimum solution in G. The following lemma shows this property.

Lemma 5.4 *G' includes all the edges of M.*

Proof: Suppose for contradiction M has at least one edge $e = (v_i, b_j)$ which is not present in G'. Since item v_i has $kl + 1$ neighbors in G', and we have k item types with l available copies of each item, according to the pigeonhole principle v_i has at least one neighbour like buyer b_q who has not bought any items. Let e' be the edge between b_q and v_i. Since G' has the $kl + 1$ largest edges of G for each item type, the weight of e is no more than the weight of e'. If the weight of e is equal to the weight of e', buyer b_j can be replaced by buyer b_q in M for an envy free revenue maximizing assignment in G that has more intersecting edges with G' than M resulting in a contradiction with the choice of M. On the other hand, if the weight of e is less than the weight of e', buyer b_q is envious of the outcome for buyer b_j which is a contradiction to envy-freeness of M. Therefore, G' includes all the edges in M. □

Now, we know that the optimum solution in G is a feasible solution in G'. It is left to show that the optimum solution in G' is feasible in G, to be able to conclude that the optimum solution in G' is also optimum in G. Let M' be the optimum (revenue maximizing) envy-free assignment in G'. We prove that M' is an envy-free assignment in G.

Lemma 5.5 *The optimum envy-free assignment in G' denoted by M' is an envy-free assignment in G.*

Proof: Suppose by contradiction that M' is not an envy-free assignment in G. Then, G has a buyer like b_j who is envious of item v_i bought by buyer b_q. Since M' is an envy-free assignment in G', the edge $e = (v_i, b_j)$ does not belong to G'. Item v_i has $kl + 1$ neighbors in G', and there are kl available items for purchase. Hence, according to the pigeonhole principle, v_i has at least one neighbor like b_p who has not bought any items in M'. Since G' has the $kl + 1$ largest edges for each item node, the weight of the edge between v_i and b_p is not less than the weight of e which means b_p would be envious of the item that b_j has bought which is a contradiction to the envy-freeness of M' in G'. Thus, M' should also be an envy-free assignment in G. □

The following theorem is a corollary of the above lemmas and is the main result of this section.

Theorem 5.8 *Given an envy-free α-approximation mechanism for the offline revenue maximization problem, there exists an envy-free α-approximation mechanism for the revenue maximization problem in the streaming setting using $O(k^2 l)$ space.*

Proof: We construct graph G' in the $O(k^2l)$ available memory while the edges of graph G are being streamed. According to Lemma 5.5, the optimum assignment in G' is a feasible assignment in G. Furthermore, due to lemma 5.4 we know that the optimum solution in G' is at least as good as the optimum solution in G. Hence, keeping the edges of graph G' while the edges of G are being streamed is enough for finding the optimum assignment in G. It is clear that given an envy-free mechanism that approximates maximum revenue in offline graph G' within a factor α we can use the exact mechanism as an α-approximation for the revenue maximization problem in graph G. $\square$

5.4 Improving Space Efficiency While Approximating Social Welfare

In this section, we try to improve space efficiency in the problem solved in Section 5.2, when we relax the goal of achieving maximum social welfare to obtaining an approximation of it. More specifically, suppose we have k item types, l available items of each type, and n buyers, and the utilities of buyers for item types are revealed in a streaming fashion. Recall that we can find the social welfare maximizing prices for the items and an assignment of items to buyers in $O(k^2l)$ available memory. In this section, our goal is to find prices for item types when the amount of available memory is independent of l (number of available items of each item type). We prove when each buyer picks the most profitable item based on the prices that our algorithm suggests and his own utilities, there would be no more than l requests for any of the item types with high probability, and the social welfare would be a good approximation of the optimum social welfare. Thus, we can conclude this self selection of items by customers is envy-free and valid with high probability, and we do not have to deal with item to buyer allocations after setting the prices. Our approach here is to collect a sample of buyers while the data is being streamed, decide the prices based on this sample, and prove that these prices would yield a good approximation of the social welfare and an envy-free assignment of items to buyers in the original graph while the assignment of items is done by the buyers themselves and not by us. This algorithm is especially favorable over previous ones when the number of different item types is relatively small compared to the total number of items. In other words, when k is small compared to l.

Let B be the set of our n buyers, and V the set of k item types. Assume we have l available items of each type. Let G be the weighted bipartite graph of buyers and item types showing utilities of buyers for items. For arbitrary constants $\delta, \epsilon > 0$, our algorithm finds prices of items in V such that the greedy item picking strategy by buyers would yield a valid envy-free assignment and achieves a social welfare that is $(1-2\epsilon)$-approximation of the maximum possible social welfare with probability $1-\delta$. We define a new parameter $t = 3\ \frac{-log(\delta)+log(2k)+klog(n)}{\epsilon^2}$, and sample every buyer in B with probability $\frac{t}{l}$. Let B' be the set of buyers chosen in our sampling, and G' be the induced subgraph of G when we remove all the vertices that are not in B'. We assume there are $(1-\epsilon)t$ available copies from each item type in graph G' which can be sold to the buyers in B'. As we discussed in previous sections, we can find the optimal prices for items in B' to achieve maximum Social welfare in graph G'

Algorithm 8

Input: Weighted bipartite graph G with $B \cup V$ as set of vertices, l number of available items from each item type, and constants $\epsilon, \delta > 0$.

Output: Price vector $\vec{p}$ which yields a $(1-2\epsilon)$-approximation of opt SW and an envy-free assignment with probability $1-\delta$.

```
0: $t \leftarrow 3\ \frac{-log(\delta)+log(2k)+klog(n)}{\epsilon^2}$
0: $B' \leftarrow \emptyset$
0: for $b \in B$ do
0:     Add $b$ to $B'$ with probability $\frac{t}{l}$
0: end for
0: Let $G'$ be the subgraph of $G$ induced by $B' \cup V$
0: $l' \leftarrow t(1-\epsilon)$ be the number of available items of each item type in $G'$
0: Upon stream of edges in $G$, ignore any edge $e \notin G'$
0: Find optimal $\vec{p}$ using SWM Algorithm (section 5.2) on edges of $G'$
0: Return $\vec{p}$ =0
```

in $O(k^2t)$ available memory. After this step, we use the same prices for the general case, and prove that these prices along with the greedy item selection by buyers satisfies the aforementioned criteria.

Once the prices are determined by our algorithm, we let each buyer pick his own profit maximizing item to accomplish an envy-free assignment. We use a combination of Chernoff and union bounds in the following two lemmas to prove the probability that this assignment is invalid is small. More formally, we show that when buyers choose items greedily, the probability that the number of requests for each item is within $(1-2\epsilon)l)$ and l is at least $1-\delta$.

Lemma 5.6 *Fix the prices of all available item types in V. Sample the buyers by choosing each buyer with probability q. Let x_i be the number of buyers in our sample who prefer item i over any other item. Let y_i be the number of buyers who prefer item i in the original setting with all buyers. Then assuming $x_i \leq ql$ we have*

$$Pr(|y_i - \frac{x_i}{q}| > \epsilon l) \leq 2exp(-\frac{\epsilon^2 ql}{3})$$

Proof:

$$Pr(|y_i - \frac{x_i}{q}| > \epsilon l) = Pr(|y_i q - x_i| > \epsilon ql)$$

We need to apply Chernoff bound to find an upperbound on the above probability. Here $y_i q$ can be thought of as a guess for x_i. Therefore, by letting $\zeta = \frac{\epsilon ql}{x_i}$ and $\mu = x_i$ in the Chernoff bound given by the inequality $Pr(|\hat{x} - \mu| > \zeta\mu) \leq 2exp(-\frac{\zeta^2}{3}\mu)$ we have

$$Pr(|y_i q - x_i| > \zeta\mu) \leq 2exp(-\mu\frac{\zeta^2}{3}) = 2exp(\frac{-\epsilon^2 q^2 l^2}{3x_i}) \leq 2exp(-\frac{\epsilon^2 ql}{3}).$$

Where the last inequality follows from the assumption that $x_i \leq ql$. □

Lemma 5.7 *Let y_i be the number of buyers who prefer item i over any other item in the original setting with prices set by Algorithm 8. Then we have*

$$Pr(|y_i - (1-\epsilon)l| > \epsilon l) \leq 2exp(-\frac{\epsilon^2 t}{3})$$

Proof: Algorithm 8 chooses each item with probability $\frac{t}{l}$ and sells $(1-\epsilon)t$ of each item to the sampled buyers for an envy-free assignment in G'. Therefore, for a fixed pricing and a fixed item $i \in [k]$, we can use Lemma 5.6 with parameters $q = \frac{t}{l}$ and $x_i = (1-\epsilon)t$.

$$Pr(|y_i - \frac{(1-\epsilon)t}{\frac{t}{l}}| > \epsilon l) \leq 2exp(-\frac{\epsilon^2 \frac{t}{l} l}{3}) \Rightarrow Pr(|y_i - (1-\epsilon)l| > \epsilon l) \leq 2exp(-\frac{\epsilon^2 t}{3})$$

□

Lemma 5.8 *When the prices are set by Algorithm 8 and the buyers greedily pick the best item for themselves, the probability that the number of requests for each item is between* $(1-2\epsilon)l$ *and* l *is at least* $1-\delta$*. Note that this also means the greedy selection of items by the buyers would yield a valid assignment with probability at least* $1-\delta$*.*

Proof:

From Lemma 5.7 we have

$$Pr(y_i > l) + Pr(y_i < (1-2\epsilon)l) \leq 2exp(-\frac{\epsilon^2 t}{3})$$

Let $\mathcal{P}$ denote the set of all possible price vectors for the item types in V that Algorithm 8 might return. Since there are k item types and n reasonable price values for each item type $|\mathcal{P}| < n^k$. Let $A_{\vec{p},i}$ be the event that with price vector $\vec{p}$, the number of requests for item i is more than l or less than $(1-2\epsilon)l$. Then

$$Pr(A_{\vec{p},i}) = Pr(y_i > l) + Pr(y_i < (1-2\epsilon)l) \leq 2exp(-\frac{\epsilon^2 t}{3})$$

Furthermore, using Union Bound we show that the probability of any of the events $A_{\vec{p},i}$ happening for all possible price vectors $\vec{p} \in \mathcal{P}$ and all item types $i \in [k]$ is

$$\bigcup_{\vec{p}\in\mathcal{P}} \bigcup_{i\in[k]} A_{\vec{p},i} \leq 2\ k\ n^k\ exp(-\frac{\epsilon^2 t}{3}) \leq \delta$$

Where the last inequality follows from the choice of t. □

Lemma 5.9 *Consider the price vector $\vec{p}$ returned by algorithm 8 combined with the greedy selection of items by the individual buyers. If every item type is requested by the buyers between $(1-2\epsilon)l$ and l times then the social welfare of this assignment is within $(1-2\epsilon)$-approximation of the maximum possible social welfare.*

Proof: Let H be a bipartite graph with the set of buyers B on the left side and l vertices for each item type in V on the right side. The weight of the edges between a buyer vertex and an item would be the utility of the buyer for that item type. Let H' be a similar bipartite graph with set of buyers on the left side and $(1-2\epsilon)l$ vertices for each item type in V on the right side. Then the weight of the maximum weighted matching in H' is at least $(1-2\epsilon)$ of the weight of the maximum weighted matching in H.

Consider the greedy selection of items by the buyers after we run Algorithm 8 and set prices for item types. Suppose every item type is sold between $(1-2\epsilon)l$ and l times. Remove some buyers so that every item is picked by exactly $(1-2\epsilon)l$

buyers. The remaining edges form a matching in graph H'. It is left to show that this matching is a maximum weighted matching in H'. For this purpose, we build a vertex cover C of same weight for the matching.

For every buyer remaining in the graph, let its vertex cover value be the profit (utility minus price) that he makes from his selected item. For each item, let its vertex cover value be the suggested price by Algorithm 8 for its item type. These values in the vertex cover would clearly cover every edge in the matching. Let b_i be a buyer in B. Let v_j be the item type that b_i chooses and v_h be another item type. We want to show that C covers the weight of the edge between b_i and an vertex of the item type v_h. Note that $u_{b_i,v_j} - p_{v_j} > u_{b_i,v_h} - p_{v_h}$. The value of C for buyer b_i is $u_{b_i,v_j} - p_{v_j}$ and the value of C for any vertex of item type v_j is p_{v_j}. Therefore, $u_{b_i,v_j} - p_{v_j} + p_{v_h} > u_{b_i,v_h}$. Thus, our algorithm produces a maximum weighted matching in H'. □

Theorem 5.9 *The pricing suggested by Algorithm 8 along with the greedy selection of items by the buyers yields a valid envy-free assignment and a* $(1 - 2\epsilon)$*-approximation of the maximum possible social welfare with probability* $1 - \delta$*. The space needed by Algorithm 8 is independent of* l*, the number of available items of each item type.*

Proof: Since Algorithm 8 runs the social welfare maximizing algorithm of section 5.2 on a sample of buyers assuming there are $(1 - \epsilon)t$ available item of each type, only $O(k^2t)$ memory is needed for finding the optimal price vector $\vec{p}$ of graph G'.

Here t is a function of k, n, δ and ϵ and does not depend on l. Lemma 5.8 guarantees the number of requests for each item type does not exceed the number of available items of that type with probability at least $1 - \delta$ and Lemma 5.9 proves the social welfare in this case would be within $(1-2\epsilon)$-approximation of the maximum possible social welfare. □

5.5 Improving Space Efficiency While Approximating Optimum Revenue

In section 5.2, we introduced a simple streaming mechanism (RM) that finds the price vector and an envy-free assignment of items to buyers to α-approximate maximum *revenue* given a mechanism that α-approximate the maximum revenue in the offline case. In this section, we are concerned with reducing the amount of space used by our streaming algorithm. As we mentioned earlier, $O(k^2l)$ available space is needed for any streaming algorithm that finds a revenue maximizing assignment in our setting. Just like the previous section, we are interested in a streaming algorithm for which the amount of space used is independent of l, the number of copies of each available item type. Algorithm 9 is our algorithm for this purpose. As a result of reduction in the required memory, the revenue of the assignment given by our algorithm loses another $(1 - 2\epsilon)$ approximation factor compared to the maximum possible revenue. When l is small compared to k, this algorithm would be beneficial since it dramatically improves the amount of space used. The algorithm and some of the proofs are similar to the ones in the previous section.

Algorithm 9

Input: Weighted bipartite graph G with $B \cup V$ as set of vertices, l number of available items from each item type, and constants $\epsilon, \delta > 0$.

Output: Price vector $\vec{p}$ which yields a $(1-2\epsilon)$-approximation of opt revenue and an envy-free assignment with probability $1-\delta$.

```
0: $t \leftarrow 3\ \frac{-log(\delta)+log(2k)+klog(n)}{\epsilon^2}$
0: $B' \leftarrow \emptyset$
0: for $b \in B$ do
0:     Add $b$ to $B'$ with probability $\frac{t}{l}$
0: end for
0: Let $G'$ be the subgraph of $G$ induced by $B' \cup V$
0: $l' \leftarrow t(1-\epsilon)$ be the number of available items of each item type in $G'$
0: Upon stream of edges in $G$, ignore any edge $e \notin G'$
0: Find optimal $\vec{p}$ using RM Algorithm (section 5.3) on edges of $G'$
0: Return $\vec{p}$ =0
```

Lemma 5.10 *Let y_i be the number of buyers who prefer item i over any other item type in the original setting with prices set by Algorithm 9. Then we have*

$$Pr(|y_i - (1-\epsilon)l| > \epsilon l) \leq 2exp(-\frac{\epsilon^2 t}{3})$$

Proof: This lemma is exactly the same as Lemma 5.7 that we proved in the previous section. $\square$

Lemma 5.11 *When the prices are set by Algorithm 9 and the buyers greedily pick the best item for themselves, the probability that the number of requests for each item type is between $(1-2\epsilon)l$ and l is at least $1-\delta$. Note that this also means the greedy selection of items by the buyers would yield a valid assignment with probability at least $1-\delta$.*

Proof: Again this lemma is the same as Lemma 5.8 proved in the previous section and since our parameters in sampling is the same in both algorithms the proofs are exactly the same. $\square$

So far in this section, we used exactly similar lemmas as the previous section to show our sampling approach results in an envy-free valid assignment with high probability. Next we want to show why our algorithm also yields a revenue that is a good approximation of the optimum revenue. Note that Algorithm 9 uses the revenue maximizing Algorithm introduced in Section 5.2 as opposed to the social welfare maximizing Algorithm of Section 5.3 that Algorithm 8 uses, and our goal here is to maximize the revenue. Therefore, showing that our algorithm's revenue is $(1-2\epsilon)$-approximation of the maximum possible revenue is different from the previous section.

Lemma 5.12 *Fix a price vector $\vec{p}$ for all available item types in V. If $\vec{p}$ produces revenue R in G, it produces a revenue bigger than $(1-\epsilon)\frac{t}{l}R$ in G' with probability at least $1-\delta$.*

Proof: Let x_i be the number of requests for item type i in graph G' and y_i be the number of requests for the same item type in graph G when the price vector is fixed. Lemma 5.6 with parameters $q = \frac{t}{l}$ and $y_i = l$ shows $Pr(x_i < (1-\epsilon)t) < 2exp(-\frac{\epsilon^2 t}{3})$. Again, using Union Bound the same way we used in Lemma 5.7, we can show the probability that any of the x_i's is less than $(1-\epsilon)t$ is less than δ. Thus, with probability at least $1-\delta$ each item type is sold to at least $(1-\epsilon)t$ buyers in G'. Since each item type is sold to at most l buyers in G, our revenue in G' must be at least $(1-\epsilon)\frac{t}{l}$ of the revenue in G with probability $1-\delta$. $\square$

Lemma 5.13 *Consider the price vector $\vec{p}$ returned by algorithm 9 along with the greedy selection of items by the individual buyers. If the number of requests for every item type is at least $(1-2\epsilon)l$ and at most l, then the revenue of this assignment is within $(1-2\epsilon)$-approximation of the maximum possible revenue with probability at least $1-\delta$.*

Proof: Let $Rev_A(G)$ denote the revenue of the price vector given by algorithm A on graph G and $Rev_A(G')$ be the revenue that the same price vector produces on the sampled graph G'. Note that in case every item type is requested by the buyers at least $(1-2\epsilon)l$ times, we have

$$Rev_{Alg9}(G') \leq \frac{l'}{(1-2\epsilon)l} Rev_{Alg9}(G) = \frac{(1-\epsilon)t}{(1-2\epsilon)l} Rev_{Alg9}(G)$$

. Let OPT be the optimum offline revenue maximizing assignment for the specific graph G. Recall that the price vector chosen by Algorithm 9 maximizes the revenue in G' and thus

$$Rev_{OPT}(G') \leq Rev_{Alg9}(G')$$

. Furthermore, due to Lemma 5.12, we have

$$\frac{(1-\epsilon)t}{l} Rev_{OPT}(G) \leq Rev_{OPT}(G')$$

with probability at least $1-\delta$. Combining the above three inequalities proves $(1-2\epsilon)Rev_{OPT}(G) \leq Rev_{Alg9}(G)$ with probability at least $1-\delta$.

$\square$

Theorem 5.10 *The pricing suggested by Algorithm 9 along with the greedy selection of items by the buyers yields a valid envy-free assignment and a* $(1-2\epsilon)\alpha$*-approximation of the maximum possible revenue with probability at least* $1-2\delta$ *given an envy-free mechanism that* α*-approximates maximum revenue in the offline case. The space needed by Algorithm 9 is independent of* l*, the number of available items of each item type.*

Proof: Lemma 5.11 shows the greedy selection of items by buyers after running Algorithm 9 results in a valid and envy-free assignment with probability at least

$1 - \delta$. Lemma 5.13 shows in case the assignment is valid, the revenue is within $(1 - 2\epsilon)$-approximation of the maximum revenue with probability $1 - \delta$. Thus, with probability $1 - 2\delta$ the Algorithm 9 results in a valid envy-free assignment and approximates the maximum revenue within a $(1 - 2\epsilon)\alpha$ factor. Here α is added to the approximation factor since RM gives an α-approximation for maximum revenue in the streaming setting, given an α-approximation mechanism for the offline case.

□

5.6 A Hardness Proof for Social Welfare Maximization Problem

In Section 5.2, we presented a streaming algorithm which finds an envy-free social welfare maximizing assignment of prices to items and items to buyers using $O(k^2 l)$ memory, where k is the number of item types and l is the number of available items of each type. This result raises the following interesting question. Is it necessary to have $\Omega(l)$ available memory if we want to only determine the prices? In other words, if the number of item types is small compared to the total number of items (or k is small compared to l, can we find prices in space independent of l? In this section we prove for any constant $\epsilon > 0$, no streaming algorithm can ϵ-approximate the envy-free social welfare maximizing *prices* in $o(l)$ space. The proof is done via a reduction from *Disjointness*, a well-known communication complexity problem .

Definition 5.1 Disjointness Problem *is a communication complexity problem in which Alice is given a string $x \in \{0,1\}^n$ and Bob is given a string $y \in \{0,1\}^n$. Their goal is to decide whether there is an index i, such that $x_i = y_i = 1$. Index i*

in this case is called an intersection*. It is known that the minimum number of bits required to be exchanged between Alice and Bob to find an intersection is $\Omega(n)$ bits even with multi-passes allowed.*

Theorem 5.11 *For any arbitrary small constant $\epsilon > 0$, there is no streaming algorithm which uses $o(l)$ space and ϵ-approximates all the item prices of the social welfare maximizing price vector.*

Proof: For an arbitrary ϵ, assume for the sake of contradiction there exists an algorithm A which can find an ϵ-approximation of an optimal pricing in $o(l)$ space. We show a reduction from any instance of Disjointness problem to an instance of our market design problem such that if Algorithm A exists, Disjointness problem can be solved using $o(l)$ space.

Let $\mathcal{I}_1$ be an instance of Disjointness problem with $x \in \{0,1\}^l$ as Alice's string and $y \in \{0,1\}^l$ as Bob's string. A corresponding instance of our market design problem $\mathcal{I}_2$ can be built as follows. Consider two item types in $\mathcal{I}_2$, one corresponding to Alice and one corresponding to Bob. Suppose each of these two item types have $2l$ copies available. Let $G = (V_1, V_2, E)$ be the bipartite graph of item types and buyers in instance $\mathcal{I}_2$, with $V_1 = \{v_{Alice}, v_{Bob}\}$ as the item type vertices. We start with $2l$ buyer vertices $V_a = \{a_1, a_2, \ldots, a_l\}$ and $V_b = \{b_1, b_2, \ldots, b_l\}$ in V_2. For any index i, if $x_i = 1$, we connect v_{Alice} to both vertices a_i and b_i. Similarly, for any index i such that $y_i = 1$, we connect v_{Bob} to both vertices a_i and b_i. Let I_x be the set of all indices j such that $x_j = 1$ in string x. We add a set U_{Alice} with $2l - |I_x|$ buyer vertices to V_2, and connect v_{Alice} to all vertices in U_{Alice} so that the degree of v_{Alice}

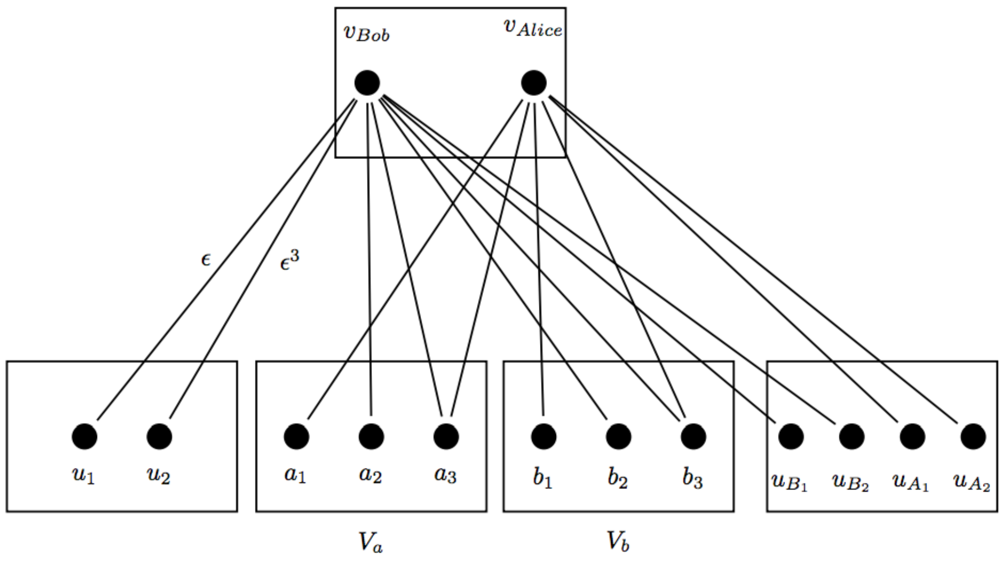

Figure 5.6.1: An example for Theorem 5.11

would be exactly $2l$. Similarly, we add a set of vertices U_{Bob} with $2l - |I_y|$ buyer vertices to V_2 and connect v_{Bob} to all vertices in U_{Bob}. Finally, we add two buyer vertices u_1, u_2 to V_2 and connect v_{Bob} to both of them. Note that the set of buyer vertices V_2 would be $\{u1, u2\} \cup V_a \cup V_b \cup U_{Alice} \cup U_{Bob}$. The utility of buyers u_1 and u_2 for Bob's item are ϵ and ϵ^3 respectively. The utility of any other buyer for any other item connected to it (v_{Alice} or v_{Bob}) would be 1. In other words, the weight of the edge (v_{Bob}, u_1) is ϵ, the weight of the edge (v_{Bob}, u_2) is ϵ^3, and the weight of all the other edges in E is 1. (See Figure 5.6.1 as an example.) [2]

Now suppose both Alice and Bob know about algorithm A. Let E_{Alice} be the set of edges connected to v_{Alice}, and E_{Bob} the set of edges connected to v_{Bob} in graph G. Note that Alice only knows about E_{Alice} and buyer vertices $V_2 \backslash (\{u_1, u_2\} \cup$

[2]Here, $x = \{1, 0, 1\}$ and $y = \{0, 1, 1\}$. The weight of the first and the second edges are ϵ and ϵ^3 as they are labeled, and the weight of all other edges are one.

U_{Bob}). Similarly, Bob only knows about E_{Bob} and buyer vertices $V_2 \backslash U_{Alice}$. Alice starts streaming her edges and running algorithm A on it. She would send the information that algorithm A stores in $o(l)$ available space to Bob. Bob at the other end receives all the information stored by algorithm A and sent by Alice, and continues running algorithm A by streaming his own edges. Algorithm A can find social welfare maximizing prices for both Alice and Bob items in $o(l)$ space. The algorithm finishes at Bob's end after he streams all of his edges. At this point we claim that Bob can decide whether the strings have intersection or not based on the following two case. If the price suggested by algorithm A for his item type is less than ϵ, Bob should declare no intersections exist and if the price is above ϵ^2 he should declare existence of at least one intersection. Furthermore, algorithm A would never set a price between ϵ and ϵ^2 for v_{Bob}. Next we prove why this claim is valid.

Suppose Alice and Bob's strings have no intersections. Then in graph G, no buyer is connected to both Alice and Bob. That is v_{Bob} is connected to $2l$ buyer vertices with utility 1 for his item and none of his buyers want Alice's item. Alice's item is also connected to $2l$ buyer vertices that do not want Bob's item. The optimal prices for both Alice and Bob's items to maximize social welfare would be 1 in this case, and no item would be sold to buyers u_1 and u_2.

On the other hand, if the strings have at least one intersection, say at index i, both v_{Alice} and v_{Bob} would be connected to a_i and b_i. To maximize social welfare in this case, Alice would sell all of her items to the $2l$ buyers who want her item at price

1, and Bob can sell at most $2l - 2$ items to those who want his item at price 1 and has to sell two items to buyers u_1 and u_2 who want to pay ϵ and ϵ^3 for his item respectively. Therefore, the price for Bob's item should be ϵ^3. Since the goal is to maximize social welfare, Bob cannot decide to leave out u_1 and u_2 and sell $2l - 2$ items at price 1 to the buyers whose utility for his item is 1.

Due to our assumption, algorithm A can ϵ-approximate all optimal prices for social welfare maximization while Alice and Bob stream the edges using only $o(l)$ available space. Specifically, if the optimal price for v_{Bob} is 1, i.e, there is no intersection in the two strings, algorithm A would set a price higher than ϵ for v_{Bob}. Otherwise, in case the optimal price for v_{Bob} is ϵ^3, algorithm A would set a price lower than ϵ^2 for Bob's item. These two prices are the only optimal prices for Bob's item and thus, only these two cases exist. Hence, Bob can distinguish between these two cases by checking the price set for his item once the algorithm ends. Any price less than ϵ^2 corresponds to an intersection, and any price higher than ϵ signals no intersection between the strings. □

5.7 A Hardness Proof for Revenue Maximization Problem

In Section 5.6, we presented a hardness proof to show no streaming algorithm exists to approximate the optimal prices using $o(l)$ available space in our market design problem with the goal of *social welfare maximization*. In this section, we establish a hardness result for the case that our goal is to find *prices* that maximize the revenue, however, the result of this section does not involve any approximation.

That is we only guarantee there exists no algorithm which finds the exact optimal prices for revenue maximization market design problem in $o(l)$ space. Just like previous section, our approach is based on a reduction from *Disjointness problem.*

Theorem 5.12 *There is no streaming algorithm which uses $o(l)$ memory, and finds the revenue maximizing price vector for an envy free assignment.*

Proof:

Suppose for contradiction, we have an algorithm A which finds an envy-free assignment and the revenue maximizing price vector using $o(l)$ available space. Let $x \in \{0,1\}^l$ be Alice's string and $y \in \{0,1\}^l$ be Bob's string. Consider two item types one corresponding to Alice and one corresponding to Bob. Suppose each of these two item types have l copies available. Let $G = (V_1, V_2, E)$ be the bipartite graph of item types and buyers, with $V_1 = \{v_{Alice}, v_{Bob}\}$ as the item type vertices. Consider three sets of buyers $V_a = a_1, a_2, \ldots, a_l$, $V_b = b_1, b_2, \ldots b_{l-|I_x|}$ and $V_c = c_1, c_2, \ldots, c_{l-|I_y|}$. For any index i, if $x_i = 1$, buyer a_i has utility one for Alice's item. Similarly, for each index i such that $y_i = 1$, buyer a_i has utility one for Bob's item. Moreover, suppose all buyers in V_b have utility 1 for Alice's item and all buyers in V_c have utility 1 for Bob's item. Finally, we add buyer v_1 with utility $1 - \frac{1}{2l}$ for Bob's item.

Alice knows all the buyers connected to her item with their utilities but she is not aware of the utilities of edges connected to Bob's item. She starts running algorithm A while streaming her edges. Once the stream of Alice's edges ends, she sends the information that algorithm A stores in $o(l)$ available space to Bob. Bob at the other end only knows about the buyers and the utility of edges connected to his own item. He also receives all the information stored by algorithm A and sent by Alice. Bob continues running algorithm A by streaming his own edges as A's input. Algorithm A finds the revenue maximizing prices for both Alice and Bob's items in $o(l)$ space. The algorithm finishes at Bob's end after he streams all of his edges. At this point we claim that Bob can decide whether the strings have an intersection or not based on the following two cases. If the price suggested by algorithm A for his item type is 1, Bob should declare no intersections exists and if the price is $1 - \frac{1}{2l}$ he should declare existence of at least one intersection. Next we prove validity of this claim. In case x and y have no intersections, both Alice and Bob's items are connected to exactly l disjoint vertices. The optimal price for both item types is 1 and the optimal revenue would be $2l$. On the other hand, if Alice and Bob's strings have an intersection, then Bob can either sell at most $l - 1$ items at price 1 which with Alice's revenue would result in a total revenue of $2l - 1$, or he can sell all his l items at price $1 - \frac{1}{2l}$ which would give an overall revenue of $2l - \frac{1}{2}$ after adding Alice's revenue. Since the second strategy would result in a higher revenue, the optimal revenue maximizing price for Bob's item would be $1 - \frac{1}{2l}$. Thus, the price given by Algorithm A for Bob's item can help distinguish between the two cases in Disjointness problem using only $o(l)$ available space, which is a contradiction. $\square$

5.8 Conclusion

In this chapter, we studied another version of online bipartite matching problem with an additional difficulty faced by online marketplaces such as Amazon and ebay: huge size of datasets. Massive datasets and limited space gives a whole new aspect to the problems which were previously solved efficiently, and this work would not be complete without studying a matching problem in the *streaming setting*. We designed algorithms which decide about the assignment of prices to items and items to buyers in order to achieve goals such as social welfare or revenue maximization while minimizing the space needed. Our assignments also satisfy envy-free constraints which ensure fairness in the mechanisms. Assuming k items and at most l copies of each of the items, we solved both problems efficiently with space $O(k^2 l)$. Furthermore, we provided algorithms which $(1 - \epsilon)$-approximate the maximum possible social welfare or revenue in space independent of l for any small $\epsilon > 0$. This is a great improvement over the previous algorithm since these algorithms significantly reduce the amount of memory used in case number of item types is small compared to the total number of items. Finally, we showed approximation of all optimal *prices* that maximize social welfare (or revenue) in the streaming setting needs at least $\Omega(l)$ space, and thus if we want to get close to all the optimal prices, the memory usage will not be very efficient.

Bibliography

[1] Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *Symposium on Discrete Algorithms (SODA)*, pages 1098–1116, 2011.

[2] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. In *Foundations of Computer Science (FOCS)*, pages 461–471, 2007.

[3] Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. Nonmonotone submodular maximization via a structural continuous greedy algorithm. In *ICALP*, pages 342–353, 2011.

[4] David S. Johnson, Maria Minkoff, and Steven Phillips. The prize collecting steiner tree problem: theory and practice. In *Symposium on Discrete Algorithms (SODA)*, pages 760–769, 2000.

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

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

[7] Aaron Archer, MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Howard Karloff. Improved approximation algorithms for prize-collecting steiner tree and tsp. *SIAM J. Comput.*, 40(2):309–332, March 2011.

[8] Kedar Dhamdhere, Vineet Goyal, R. Ravi, and Mohit Singh. How to pay, come what may: Approximation algorithms for demand-robust covering problems. In *Foundations of Computer Science (FOCS)*, pages 367–378, 2005.

[9] Abraham D. Flaxman, Alan Frieze, and Michael Krivelevich. On the random 2-stage minimum spanning tree. In *Symposium on Discrete Algorithms (SODA)*, pages 919–926, 2005.

[10] Nicole Immorlica, David Karger, Maria Minkoff, and Vahab S. Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In *Symposium on Discrete Algorithms (SODA)*, pages 691–700, 2004.

[11] Anupam Gupta, Martin Pál, R. Ravi, and Amitabh Sinha. Sampling and cost-sharing: Approximation algorithms for stochastic optimization problems. *SIAM J. Comput.*, 40(5):1361–1401, September 2011.

[12] David B. Shmoys and Chaitanya Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *Foundations of Computer Science (FOCS)*, pages 228–237, 2004.

[13] Chaitanya Swamy and David B Shmoys. Sampling-based approximation algorithms for multi-stage stochastic optimization. In *Foundations of Computer Science (FOCS)*, pages 357–366, 2005.

[14] Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM J. on Optimization*, 12(2):479–502, February 2002.

[15] Moses Charikar, Chandra Chekuri, and Martin Pál. Sampling bounds for stochastic optimization. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Technique (APPROX/RANDOM)*, pages 610–610, 2005.

[16] Irit Katriel, Claire Kenyon-Mathieu, and Eli Upfal. Commitment under uncertainty: Two-stage stochastic matching problems. *Theoretical Computer Science*, 408(2):213–223, 2008.

[17] Nan Kong and Andrew J. Schaefer. A factor $\frac{1}{2}$approximation algorithm for two-stage stochastic matching problems. *European Journal of Operational Research*, 172:740–746, 2004.

[18] Yuval Emek, Michal Feldman, Iftah Gamzu, Renato Paes Leme, and Moshe Tennenholtz. Signaling schemes for revenue maximization. In *ACM Conference on Electronic Commerce (EC)*, pages 514–531, 2012.

[19] Patrick Briest, Shuchi Chawla, Robert Kleinberg, and S Matthew Weinberg. Pricing randomized allocations. In *Symposium on Discrete Algorithms (SODA)*, pages 585–597, 2010.

[20] Patrick Briest and Heiko Röglin. The power of uncertainty: Bundle-pricing for unit-demand customers. In *Workshop on Approximation and Online Algorithms (WAOA)*, pages 47–58, 2010.

[21] Robert Kleinberg and Seth Matthew Weinberg. Matroid prophet inequalities. In *Symposium on Theory of Computing (STOC)*, pages 123–136, 2012.

[22] Saeed Alaei. Bayesian combinatorial auctions: Expanding single buyer mechanisms to many buyers. In *Foundations of Computer Science (FOCS)*, pages 512–521. IEEE, 2011.

[23] Mohammad Mahdian and Amin Saberi. Multi-unit auctions with unknown supply. In *ACM Conference on Electronic commerce (EC)*, pages 243–249, 2006.

[24] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Technique (APPROX/RANDOM)*, pages 39–52, 2010.

[25] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *Workshop on Internet and Network Economics (WINE)*, pages 246–257, 2010.

[26] Siddharth Barman, Seeun Umboh, Shuchi Chawla, and David Malec. Secretary problems with convex costs. In *International colloquium conference on Automata, Languages, and Programming (ICALP)*, pages 75–87, 2012.

[27] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM*, 53:324–360, 2006.

[28] Rajendra Bhatia and Chandler Davis. A better bound on the variance. *The American Mathematical Monthly*, 107(4):353–357, 2000.

[29] Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, pages 253–264, 2007.

[30] Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *SODA*, 2013.

[31] Jonathan Aronson, Martin Dyer, Alan Frieze, and Stephen Suen. Randomized greedy matching. ii. *Random Struct. Algorithms*, 6(1):55–73, 1995.

[32] T-H. Hubert Chan, Fei Chen, Xiaowei Wu, and Zhichao Zhao. Ranking on arbitrary graphs: Rematch via continuous LP with monotone and boundary condition constraints. In *SODA*, 2014 (To appear).

[33] Jon Feldman, Nitish Korula, Vahab Mirrokni, S. Muthukrishnan, and Martin Pál. Online ad assignment with free disposal. In *WINE*, pages 374–385, 2009.

[34] Alvin E. Roth, Tayfun Sonmez, and M. Utku Unver. Pairwise kidney exchange. Working Paper 10698, National Bureau of Economic Research, August 2004.

[35] Matthias Poloczek and Mario Szegedy. Randomized greedy algorithms for the maximum matching problem with new analysis. In *FOCS*, pages 708–717, 2012.

[36] Gagan Goel and Pushkar Tripathi. Matching with our eyes closed. In *FOCS*, pages 718–727, 2012.

[37] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, pages 352–358, 1990.

[38] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, pages 982–991, 2008.

[39] Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *SIGACT News*, 39(1):80–87, 2008.

[40] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *SODA*, pages 1253–1264, 2011.

[41] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized on-line matching. In *FOCS*, pages 264–273, 2005.

[42] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM*, 50(6):795–824, 2003.

[43] Martin E. Dyer and Alan M. Frieze. Randomized greedy matching. *Random Struct. Algorithms*, 2(1):29–46, 1991.

[44] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *STOC*, pages 587–596, 2011.

[45] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *STOC*, pages 597–606, 2011.

[46] Gagan Goel and Pushkar Tripathi. Matching with our eyes closed. *CoRR*, abs/1306.2988, 2013.

[47] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating $1 - 1/e$. In *FOCS*, pages 117–126, 2009.

[48] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *ESA*, pages 170–181, 2010.

[49] Vahideh H. Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: online actions based on offline statistics. In *SODA*, pages 1285–1294, 2011.

[50] Bernhard Haeupler, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching: improved approximation algorithms. In *WINE*, pages 170–181, 2011.

[51] Karsten Neuhoff, Julian Barquin, Maroeska G Boots, Andreas Ehrenmann, Benjamin F Hobbs, Fieke AM Rijkers, and Miguel Vazquez. Network-constrained cournot models of liberalized electricity markets: the devil is in the details. *Energy Economics*, 27(3):495–525, 2005.

[52] Wei Jing-Yuan and Yves Smeers. Spatial oligopolistic electricity models with cournot generators and regulated transmission prices. *Operations Research*, 47(1):102–112, 1999.

[53] Christopher J Day, Benjamin F Hobbs, and Jong-Shi Pang. Oligopolistic competition in power networks: a conjectured supply function approach. *Power Systems, IEEE Transactions on*, 17(3):597–607, 2002.

[54] Mariano Ventosa, Alvaro Baıllo, Andrés Ramos, and Michel Rivier. Electricity market modeling trends. *Energy policy*, 33(7):897–913, 2005.

[55] Andrew F Daughety. *Cournot oligopoly: characterization and applications.* Cambridge University Press, 2005.

[56] Steven A Gabriel, Supat Kiet, and Jifang Zhuang. A mixed complementarity-based equilibrium model of natural gas markets. *Operations Research*, 53(5):799–818, 2005.

[57] Annex to the green paper: A European strategy for sustainable, competitive and secure energy. Brussels, 2006.

[58] David M Kreps and Jose A Scheinkman. Quantity precommitment and Bertrand competition yield Cournot outcomes. *The Bell Journal of Economics*, pages 326–337, 1983.

[59] Nirvikar Singh and Xavier Vives. Price and quantity competition in a differentiated duopoly. *The RAND Journal of Economics*, pages 546–554, 1984.

[60] Martin J Osborne and Carolyn Pitchik. Price competition in a capacity-constrained duopoly. *Journal of Economic Theory*, 38(2):238–260, 1986.

[61] Harold Hotelling. *Stability in competition.* Springer, 1990.

[62] Jonas Häckner. A note on price and quantity competition in differentiated oligopolies. *Journal of Economic Theory*, 93(2):233–239, 2000.

[63] Xavier Vives. *Oligopoly pricing: old ideas and new tools.* The MIT press, 2001.

[64] Rahmi Ilkılıç. Cournot competition on a network of markets and firms. Technical report, Fondazione Eni Enrico Mattei, 2009.

[65] Moshe Babaioff, Brendan Lucier, and Noam Nisan. Bertrand networks. In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce*, EC '13, pages 33–34, New York, NY, USA, 2013. ACM.

[66] Edmund Eisenberg and David Gale. Consensus of subjective probabilities: The pari-mutuel method. *The Annals of Mathematical Statistics*, 30(1):165–168, 1959.

[67] Nikhil R Devanur, Christos H Papadimitriou, Amin Saberi, and Vijay V Vazirani. Market equilibrium via a primal-dual-type algorithm. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 389–395. IEEE, 2002.

[68] Nikhil R Devanur and Vijay V Vazirani. An improved approximation scheme for computing arrow-debreu prices for the linear case. In *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science*, pages 149–155. 2003.

[69] Kamal Jain. A polynomial time algorithm for computing an arrow-debreu market equilibrium for linear utilities. *SIAM Journal on Computing*, 37(1):303–318, 2007.

[70] Nikhil R Devanur and Ravi Kannan. Market equilibria in polynomial time for fixed number of goods or agents. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 45–53. IEEE, 2008.

[71] James B Orlin. Improved algorithms for computing fisher's market clearing prices: computing fisher's market clearing prices. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 291–300. ACM, 2010.

[72] Mehdi Ghiyasvand and James B. Orlin. A simple approximation algorithm for computing arrow-debreu prices. *Oper. Res.*, 60(5):1245–1248, 2012.

[73] Nicole Immorlica, Evangelos Markakis, and Georgios Piliouras. Coalition formation and price of anarchy in cournot oligopolies. In *Internet and Network Economics*, pages 270–281. Springer, 2010.

[74] Amos Fiat, Elias Koutsoupias, Katrina Ligett, Yishay Mansour, and Svetlana Olonetsky. Beyond myopic best response (in cournot competition). In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 993–1005. SIAM, 2012.

[75] Simon P Anderson and Régis Renault. Efficiency and surplus bounds in cournot competition. *Journal of Economic Theory*, 2003.

[76] Rabah Amir. Cournot oligopoly and the theory of supermodular games. *Games and Economic Behavior*, 1996.

[77] Paul Milgrom and John Roberts. Rationalizability, learning, and equilibrium in games with strategic complementarities. *Econometrica: Journal of the Econometric Society*, 1990.

[78] Nikolai S Kukushkin. *Cournot Oligopoly with" almost" Identical Convex Costs.* Instituto Valenciano de Investigaciones Económicas, 1993.

[79] Jörgen W Weibull. Price competition and convex costs. Technical report, SSE/EFI Working Paper Series in Economics and Finance, 2006.

[80] YB Zhao and JY Han. Two interior-point methods for nonlinear $p_*(\tau)$-complementarity problems. *Journal of optimization theory and applications*, 102(3):659–679, 1999.

[81] Lars Thorlund-Petersen. Iterative computation of cournot equilibrium. *Games and Economic Behavior*, 2(1):61–75, 1990.

[82] Charles D Kolstad and Lars Mathiesen. Computing cournot-nash equilibria. *Operations Research*, 39(5):739–748, 1991.

[83] Koji Okuguchi and Ferenc Szidarovszky. On the existence and computation of equilibrium points for an oligopoly game with multi-product firms. *Annales, Univ. Sci. bud. Roi. Fotvos Nom*, 1985.

[84] Lars Mathiesen. Computation of economic equilibria by a sequence of linear complementarity problems. In *Economic equilibrium: model formulation and solution*, pages 144–162. Springer, 1985.

[85] FA Campos, J Villar, and J Barquín. Solving cournot equilibriums with variational inequalities algorithms. *Generation, Transmission & Distribution, IET*, 4(2):268–280, 2010.

[86] Stephen Poythress Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[87] Donald M Topkis. Minimizing a submodular function on a lattice. *Operations research*, 1978.

[88] Maria-Florina Balcan, Avrim Blum, Jason D. Hartline, and Yishay Mansour. Reducing mechanism design to algorithm design via machine learning. *Journal of Computer and System Sciences*, 74(8):1245 – 1270, 2008. Learning Theory 2005.

[89] Richard Cole and Tim Roughgarden. The sample complexity of revenue maximization. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 243–252, New York, NY, USA, 2014. ACM.

[90] Peerapong Dhangwatnotai, Tim Roughgarden, and Qiqi Yan. Revenue maximization with a single sample. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, EC '10, pages 129–138, New York, NY, USA, 2010. ACM.

[91] Justin Hsu, Jamie Morgenstern, Ryan M. Rogers, Aaron Roth, and Rakesh Vohra. Do prices coordinate markets? *CoRR*, abs/1511.00925, 2015.

[92] Jamie Morgenstern and Tim Roughgarden. The pseudo-dimension of nearly-optimal auctions. In *NIPS*, page Forthcoming, 12 2015.

[93] Venkatesan Guruswami, Jason D Hartline, Anna R Karlin, David Kempe, Claire Kenyon, and Frank McSherry. On profit-maximizing envy-free pricing. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1164–1173. SIAM, 2005.

[94] Rajesh Chitnis, Graham Cormode, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: maximal matching and vertex cover. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1234–1251. SIAM, 2015.

[95] M. Cheung and C. Swamy. Approximation algorithms for single-minded envy-free profit-maximization problems with limited supply. In *Foundations of Computer Science, 2008. FOCS '08. IEEE 49th Annual IEEE Symposium on*, pages 35–44, Oct 2008.

[96] Ning Chen, Arpita Ghosh, and Sergei Vassilvitskii. Optimal envy-free pricing with metric substitutability. *SIAM Journal on Computing*, 40(3):623–645, 2011.

[97] Ning Chen and Xiaotie Deng. Envy-free pricing in multi-item markets. *ACM Trans. Algorithms*, 10(2):7:1–7:15, February 2014.

[98] Patrick Briest and Piotr Krysta. Buying cheap is expensive: Approximability of combinatorial pricing problems. *SIAM J. Comput.*, 40(6):1554–1586, December 2011.

[99] Parinya Chalermsook, Julia Chuzhoy, Sampath Kannan, and Sanjeev Khanna. Improved hardness results for profit maximizations pricing problems with unlimited supply. In *Proceedings of APPROX*, 2012.

[100] L. S. Shapley and M. Shubik. The assignment game i: The core. *International Journal of Game Theory*, 1:111–130, 1971. 10.1007/BF01753437.

[101] Sushil Bikhchandani and John W. Mamer. Competitive Equilibrium in an Exchange Economy with Indivisibilities. *Journal of Economic Theory*, 74(2):385–413, June 1997.

[102] Hu Fu, Nicole Immorlica, Brendan Lucier, and Philipp Strack. Randomization beats second price as a prior-independent auction. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 323–323, New York, NY, USA, 2015. ACM.

[103] Zhiyi Huang, Yishay Mansour, and Tim Roughgarden. Making the most of your samples. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 45–60, New York, NY, USA, 2015. ACM.

[104] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to dynamic graph streams. pages 1326–1344, 2016.

[105] Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 170–181. Springer, 2005.

[106] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 734–751. SIAM, 2014.

[107] Hossein Esfandiari, Mohammad T Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1217–1233. SIAM, 2015.

[108] Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. *arXiv preprint arXiv:1505.02019*, 2015.