PRICING PROBLEMS IN ONLINE MARKETS

by

CHAOLUN XIA

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

S. Muthukrishnan

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2018

ABSTRACT OF THE DISSERTATION

Pricing Problems in Online Markets

By CHAOLUN XIA

Dissertation Director:

S. Muthukrishnan

The Internet Economy includes various online markets with billions of transactions. In this dissertation, we study pricing-related problems in both advertising markets and labor markets.

In an online advertising market, the advertiser pays for showing the ads to his target users. Discovered by previous works [1, 2], the prices of showing ads to users with different attributes vary a lot. Motivated by this observation, we develop two targeting algorithms to help an advertiser reach more target users when he has a budget. The first algorithm with provable guarantee is for the case when all the user information is completely revealed, and the second one is for the case when the user information is partially present. We also crawled LinkedIn and Facebook price to verify our algorithm. We further point out that these two algorithms are feasible only if the pricing has arbitrage. As price arbitrage may hurt the market revenue, we introduce arbitrage-free pricing for such markets, and finally propose three arbitrage-free pricing algorithms with provable revenue guarantee for the market.

In a labor market with multiple workers and tasks, a worker possibly has several skills and a task requires a worker with a certain skill. To fairly match workers and tasks, we introduce the stable pricing mechanism by extending stable matching. We propose three truthful stable pricing mechanisms with revenue guarantee to ensure the fairness for both workers and tasks.

Acknowledgements

First, I need to give my simple, straightforward but truthful, accumulated and deep-seated gratitude to my adviser Prof. Shan Muthukrishnan. Without the luck to be Muthu's student, it would be impossible for me to complete this research. During the years in working with him, I am impressed and influenced by his wisdom and kindness. I am sincerely grateful for all kinds of supports, academic, mental, financial, etc., that I received from him. Even in our first hangout conversation, I received much of encouragement from him. All of these constitute my confidence and determination to finish this research.

I would like to thank Prof. Badri Nath, Prof. Desheng Zhang and Dr. Nitish Korula for serving on my pre-defense and defense committee and providing me with insightful feedback on my thesis work and presentation. I would like to thank Prof. Martin Farach-Colton and Prof. William Steiger for serving on my qualifying committee and providing me with insightful feedback on my research and presentation.

I would also like to thank Prof. Tina Eliassi-Rad. As my first-year adviser at Rutgers, she introduced me to Rutgers PhD program, and provided me with valuable guidance at my early stage of PhD research. Moreover, she also helped me build the connection with industry, i.e. her recommendation to my first internship. Another warm thank you should go to Prof. Mor Naaman who provided me, together with advisory, a variety of valuable guidance and support, at my early stage of PhD research.

I feel honored and thankful to work with all my talented co-authors, including Ke Xie, Raz Schwartz, Jeremy Ting, Daniela Vianna, Jun Hu, Yan Zhu, Saikat Guhua, etc. Although we did not publish a paper together, but the discussion with Meng Li and Yan Wang directly inspired a part of my research. Besides academic discussion, Qiang Ma and Jinyun Yan provided me with valuable suggestions and help, constitute the success of this research.

I would also like to express my gratitude to my intern mentors, Yongkang Zhu (Google), Xu

Ling (Google) and Jin Ou (Facebook). They not only provided me with professional mentorship during my internship, but also gave me valuable and unique advice and help after the internship. The experience of working with these talents indirectly contributes to the success of this thesis.

In the long journey, I am grateful for those who intentionally or unintentionally helped me out of the hardship that I went through, chronologically including Eddie Xie, Tinglin Liu, Sai Lv, Hao Shen, Yulong Yang, Lin Zhong, Jiayan Jiang, etc. I am also grateful for those who provided me with honest opinions and helpful suggestions when I made tough decisions, chronologically including Ying Zhan, Sai Lv, Jingjing Liu, Darja Kruševskaja, Priya Govindan, etc. Special thanks should be given to a group of lovely, humorous, sincere and tolerant friends – LangRenSha in New Jersey, who always cheered me up.

I would also give my gratitude to all the school staff who helped me during my study in Rutgers.

Finally, I want to give my utmost thanks to my parents, Zhelei Xia and Hong Zhao, for their unconditional support, and my wife, Yifei Yao, for her unconditional support and persistent company across these years.

Dedication

To my parents and wife

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A variety of blooming Internet-based markets constitute the great success of Internet Economy, which has been witnessed worldwide during the past few decades. Individuals, companies, governments and other organizations or groups are frequently interacting with each other and completing numerous transactions in such markets. For instance, in online advertising markets, e.g. Google and Facebook, an advertiser pays for promoting its products or brand to the potential consumers. In online labor markets, e.g. MTurk and TaskRabbit, an employer pays for a task that needs to be done by freelancing workers.

The pricing issue is unavoidable for a market. Although plenty of pricing research has been done in Economics and other fields, these Internet-based markets, which appeared recently, still face unprecedented challenges to price the transactions or products. One of the most important reasons is that unlike traditional markets, billions of users with large diversity are involved in a single Internet-based market, and the products or services provided in such a market inherit the diversity of users. Under this situation, the market has to tailor its business model to the diversity of users. For example, in a data market that sells user-authorized information, users with different privacy leakage should be compensated accordingly [3], otherwise no user would be willing to share her information. In a crowdsourcing market, a worker, who has many skills thus capable of many tasks, should be paid and matched fairly, otherwise she might leave the market [4]. In an advertising market, the same ads may be charged differently to be shown to different users because different user attributes are valuated differently by a single advertiser. These business models imply that the products or services provided by different users are different, thus their prices should depend on involved users. Consequently, although these Internet-based markets with a variety of users run different business models, most of them face a similar but distinct research challenge in pricing their transactions or products.

In this thesis, we focus on solving novel pricing-related problems in the Internet-based advertising market and labor market. In the rest of this chapter, we first provide the overview of online markets. Then we introduce the advertising market and the labor market, followed by our contributions on the pricing-related problems in these two markets, in Section 1.2 and 1.3 respectively. Then we summarize the potential pricing-related research directions and the connections between our works with others in Section 1.4.

1.1 Overview of Online Markets

Online market is a term that is used widely, sometimes known as electronic market or digital market. This term has a broad meaning in its relatively short history, usually meaning differently to different people, which could be partially explained due to the lack of well-established research. In this thesis, we refer an online market as a whole market where buyers and sellers get connected through the Internet to complete online transactions for a specific type of products or services. Same as traditional markets, online markets can be categorized with the economic concepts, for instance, existence of competition (e.g. perfect completion and monopoly), the market size, suppliers and consumers. Based on the provided products or services, online markets can be also classified into these specific but many categories, e.g. the online exchange market (where, for example, stocks and Bitcoins are traded for profit), the online labor market, the online data market, the online advertising market, the cloud market, the e-commerce market (e.g. Amazon, eBay and Alibaba) and etc.

The major difference between traditional markets and online markets are the marketplaces. While a traditional market consists of a set of traditional marketplaces, an online market consists of one or more online marketplaces. Different from a traditional marketplace which is often a location or area, an online marketplace is a system or platform based on the Internet technologies where individuals and organizations execute commercial transactions. In such online marketplaces, products and services that are direct or indirect are provided for sale. Online marketplaces have their own advantages over traditional marketplaces. For purchasers, these online marketplaces provide a variety of efficient and economic purchasing options to numerous products or services, e.g. from buying a book to hiring a worker. For suppliers, such

online marketplaces provide easily accessible platforms and opportunities to reach potential customers, to establish new brands and to reduce sales risks. Besides the efficient access and worldwide connection, as the techniques of artificial intelligence and big data grow quickly, online marketplaces usually have a better understanding of their users, therefore can provide techniques to facilitate the completion of transactions. For example, in Netflix, a well-designed recommendation system will help users find their preferred movies. In online advertising markets, precise-targeting techniques are used to help advertisers to reach their potential customers and improve user experience in watching ads. All these revolutionary techniques cannot be applied without the support of online marketplaces.

Although online marketplaces (OMs) share many common properties, they are different from each other in term of, for example, business models. In order to characterize and compare OMs, we here discuss the classification methods of OMs, aiming to provide a clear view of the distinctions among these OMs, especially the distinction between the online markets we studied in this thesis and other online markets. However, as new OMs with novel business models emerge quickly, the taxonomy of OMs also involes. The methods to build the taxonomy are studied as a research problem, e.g. [5, 6, 7, 8, 9]. The general criteria to put OMs into different but potentially overlapped categories include the three major components of an OM: (1) involved sellers and buyers, (2) products or services in trade and (3) adopted pricing mechanisms. Note that a single OM that runs multiple businesses may belong to multiple categories. In this part, we summarize the mainstream methods to classify OMs. We first summarize three ways to classify OMs based on the involved sellers and buyers:

- **Number**. Based on the numbers of involved sellers and buyers, we can classify OMs into three categories: *1:M*, *M:1* and *M:M*. An OM that belongs to the 1:M category consists only one (or very few) seller but many buyers. In such an OM, the single seller usually is the platform. For example, in Amazon Cloud, the platform is the only seller to provide virtual machines to many customers. Similarly, M:1 denotes many sellers and one (or very few) buyer, and the only buyer is usually the platform. For example, DataCup pays for collecting personal data from many users. M:M represents many buyers and sellers. Most online labor marketplaces belong to this category, e.g. Amazon Mechanic Turk and TaskRabbit. Compared with 1:M and M:1, the M:M marketplaces usually are closer to

perfect competition.

- **Role**. Either a seller or a buyer can be an individual or a business, so based on the roles of sellers and buyers, we can classify OMs into B2B, B2C and C2C by extending these concepts from e-commerce. In this thesis, B2B refers to the case where one business makes a commercial transaction with another. For example, in Alibaba, factories are trading intermediate goods with each other. B2C denotes the OMs where businesses sell final products to customers, e.g. Amazon. In C2C marketplaces, customers are trading with each other, e.g. eBay. In B2B, the buying decisions are more rational and the relationship between buyers and sellers are usually long-term.

- **Distinction**. In many markets, there is a clear distinction between sellers and buyers. For example, in Amazon Mechanical Turk, workers seldom become an employer to publish tasks. However, a different category is the OMs where sellers and buyers can switch frequently, e.g. the stock market and the Bitcoin market.

We next summarize the taxonomy of OMs based on what are in trade.

- **Goods, Information or Services**. In economics, a common distinction between goods and services are that goods are tangible while services are non-physical. As information can be also traded, we classify the OMs into three categories based on what are traded: *Normal Goods*, *Information Goods* and *Services*. Information goods are traded in data markets, e.g. Infochimps and DataCup. Services are provided by, for example, Uber and Upwork. There are many OMs selling normal goods, e.g. Amazon and eBay.

- **Vertical or Horizontal**. [8] classifies OMs based on the type of products. Vertical OMs, sometimes called *industry-specific* OMs, focus on aggregating the supply and demand of products/services in a specific industry, aiming to optimizing the relationship between buyers and sellers. Horizontal OMs, also known as *functional* OMs, aim to optimize specific functions in organizations through facilitating cross-industry transactions. Usually, products/services that are common among multiple industries are provided in horizontal OMs.

We finally summarize the methods to classify OMs based on how the prices of transactions are determined as follows:

- **Price Control**. We can classify the OMs based on the ability to control the prices of transactions:
  - *No Control*. In online exchange marketplaces, the OM owner has no control over the prices of transactions. Instead, the prices are fully determined by user bids thus the supply and demand equilibrium. Usually, these OMs neither produce nor consume products in trade, and they only provide exchange platforms to manage transactions. Typical examples include the stock market and the Bitcoin market.
  - *Full Control*. The other extreme category is the OMs which can set the prices for all the products independent of current user bids which represent supply and demand. For example, Amazon Cloud (excluding spot business) sets the prices for various types of reserved virtual machines without asking customers for their bids, and Uber charges the cost for a ride without asking users for their bids. In this category, the prices are usually determined based on prior knowledge.
  - *Partial Control*. There are also many OMs in between. For example, many online advertising marketplaces use different auctions with reserved prices, thus the prices of transactions, as the outcomes of auctions, are determined by three factors, i.e. the user bids, the choice of auctions and the choice of reserved prices. Although prices heavily depend on the user bids, the online advertising marketplaces have control over the choice of auctions and reserved prices, so they have partial control over the prices.
- **Pricing Mechanism**. OMs can be categorized by the pricing mechanisms that are used.
  - *Fixed Pricing*. A fixed pricing mechanism does not adjust prices frequently over time. This pricing mechanism is ideally used for OMs where the supply and demand relationship is relatively steady, or there is long-term cooperation between buyers and sellers. Using fixed pricing mechanisms can sometimes help buyers and sellers to manage the budget.

- *Auction*. As variable pricing, auctions determine the market value of the products based on supply and/or demand, usually in real time. There are various types of auctions adopted by different markets, e.g. double auctions in the stock market, sealed-bid auctions in the online advertising market. Different auctions are designed for different purposes, e.g. maximizing revenue or social efficiency, thus auctions bring some flexibility to the OMs.

With these classification methods, we are able to compare different OMs. In this thesis, we focus on the online advertising market and the online labor market, so we compare current online advertising marketplaces and online labor marketplaces with other OMs, e.g. online exchange marketplaces and cloud marketplaces.

In term of *involved buyers and sellers*, there is a clear difference between online labor marketplaces and online advertising marketplaces. A large proportion of online labor marketplaces belong to the M:M and C2C categories because current online labor marketplace usually have many workers and employers, and a significant proportion of employers are individuals or non-business organizations (e.g. in TabskRabbit, many tasks are like home-moving). By contrast, most closed-site online advertising marketplaces (e.g. sponsored search advertising and social network advertising) belong to the 1:M and B2B categories because advertisers as the buyers are usually companies and the platform (e.g. a search engine or a social network) is the only seller. From this perspective, many online exchange marketplaces (e.g. stock and Bitcoin exchanges) are similar to online labor markets, i.e. M:M and C2C because a large proportion of stock and Bitcoin accounts are personal. However, in online exchange markets, there is no clear distinction between buyers and sellers because a buyer (or a seller) can switch to a seller (or a buyer) easily, which makes online exchange markets significantly different from other online markets. Cloud marketplaces are similar with online advertising marketplaces, i.e. 1:M and B2B.

From the perspective of *products or services in trade*, online labor markets provide labor services to employers, e.g. tagging an image and moving a home. Online advertising markets provide the services that lead to successful sales of products for advertisers, and cloud markets provide virtual machines as the products. Despite of differences, the services or products provided by these three markets bring a specific and practical function to the buyers, thus directly

increase the utility of buyers. However, in online exchange markets, buying any product which is specific for exchange cannot directly increase buyers' utility, i.e. a unit of stock or Bitcoin cannot be used or consumed directly.

When compared through *pricing mechanisms*, online markets are different from each other. In most exchange markets, the marketplaces have no control over the prices. For example, any Bitcoin exchange has no power to set the price of Bitcoin without becoming a buyer or seller. Therefore, exchange markets have no need to solve pricing problems similar with what are discussed in this thesis. But many other online markets are different from exchange markets. In labor markets, the marketplaces have different pricing policy. For example, in Upwork, the price of a task is usually decided by the negotiation between the worker and the employer, thus the marketplace has no control over the prices either. Although the price of a task is determined by the employer, Amazon Mechanical Turk uses fixed pricing to additionally charge a task requiring workers with specific attributes. For example, the employer needs to pay 25 cents in addition to hire a worker from age 30 to 35. Thus, AMT has partial control over the task prices. In TaskRabbit, prices are automatically set for some tasks through a function named QuickAssign. With this function, the marketplace has full control over the labor prices. In online advertising markets, many marketplaces have partial control over ads prices by using auctions with reserved prices. This is similar to the spot business in cloud markets, however, many cloud marketplaces use fixed pricing to fully control the prices of virtual machines.

1.2 Online Advertising Market

Online advertising is one of the most important industries to the Internet Economy. In 2016, the global online advertising markets have generated 178 billions of US dollars[1], and this number is estimated to be 204 billions in 2017, i.e. with 14.6% annual growth rate. Even in the US where the markets and techniques of online advertising are highly developed, the annual growth rate from 2016 to 2017 is still steady and significant – 15.9%, much larger than the annual GDP growth rate of the US. Moreover, online advertising markets are still expected to grow quickly in future.

[1] https://www.statista.com/statistics/246567/global-online-advertising-revenue/

1.2.1 Business Models

As the techniques and business are both developing, online advertising markets are now providing various forms of ads to their customers with different needs. For example, a movie trailer can be shown on YouTube; a job post may appear on LinkedIn, and a sponsored search may come up when a user is using a search engine. In general, online advertising can be categorized into different, but often overlapping businesses:

- Display Advertising. On a website as a publisher, display ads often appear in obvious sections that are intentionally designed and reserved for paid ads, e.g. a banner, text box or video box. The advertising companies whose business includes display ads are Google, Facebook, mMedia, etc. For example, Fig 1.1 shows the display ad placed in the banner on the website of New York Times.

Figure 1.1: The banner in the red box is a display ad on New Your Times

- Search Advertising. Ads of this type are specifically run by search engines, e.g. Google, Microsoft and Yahoo. Usually, search ads are placed on the web page that demonstrates search results for a given search query. Search ads are targeted for those search engine users who have matched input search terms, e.g. "buy car" in Fig 1.2. One of the reasons to the success of this business model is that many users often use a search engine to identify and compare purchasing options immediately before making a purchasing decision. As search terms often reflect search engine users' interests and intentions, sponsored search ads are able to offer highly relevant information and purchasing options to a search engine user based on her search query.

- Social Advertising. In social networks, social ads are often placed between two posts in

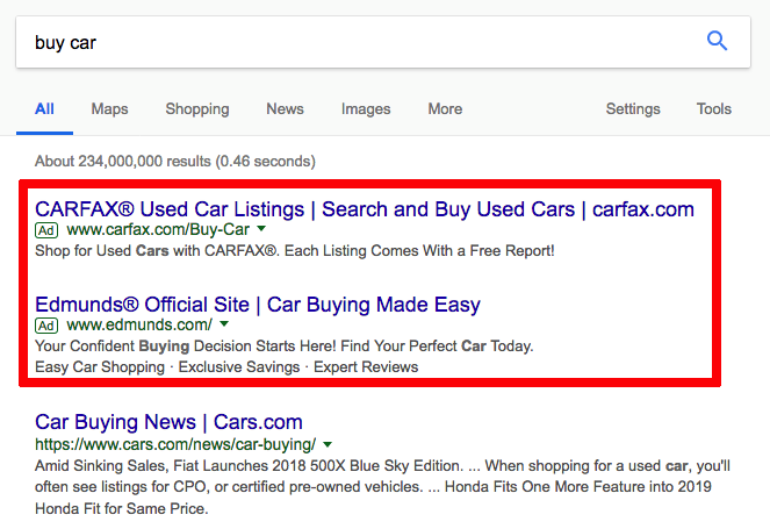


Figure 1.2: The two search results in the red box are the search ads on Google

the timeline, or other positions, e.g. sidebars. Fig 1.3 shows the ad – a job description – in the timeline of LinkedIn. Social platforms providing social advertising are able to collect more user information from user input. For example, LinkedIn knows the past working experience of its users, and Facebook knows a user's social relationship. With such collected or inferred user information, social platforms provide a unique targeting system for advertisers to reach their audience in a more accurate way.

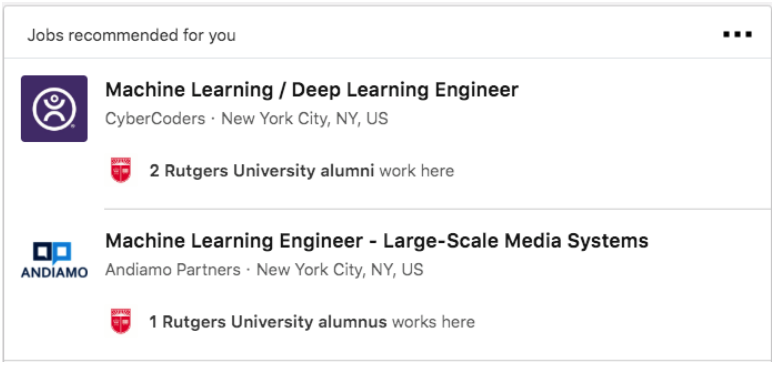


Figure 1.3: The job ads on LinkedIn

- Video Advertising. A video ad is generally considered as a short video containing sponsored information which appears before, during and/or after a video stream. One typical example of video ads is a movie trailer appearing within a YouTube video.

1.2.2 Pricing Models

The major pricing models used by major advertising platforms include:

- CPM (cost per mille) is the cost for every one thousand impressions. An impression, sometimes known as an ad view, is a term that refers to the case that an ad is viewed once by a user, or displayed once on a web page.
- CPC (cost per click), as a performance-based metric, is the cost for a user click on the ad, no matter how many impressions are served trying to get the click.
- CPA (cost per action), as another performance-based metric, is the cost for a customized user action, no matter how many impressions/clicks are served trying to get the action. An action could be, for example, the purchase of a product, installment of an app and the membership registration.

1.2.3 User Targeting

Online advertising platforms provide targeting systems to grant an advertiser with the precise control over the users who will see his ads. Their targeting languages contain detailed information shared directly by users, inferred from user daily activities and social relationship, or purchased from third-part data providers. This includes detailed educational records about the user, past and present employment experience, significant life events like changes in marital status or birth of a baby. For example, in LinkedIn, an Internet company can accurately deliver its job ads to fresh graduates with machine learning skills; in Facebook, a luxury car dealer can restrict that only the local and rich users could be exposed to its ads. Fig 1.4 shows the interface of Facebook targeting system, and the audience are restricted to New Jersey, with 30+ years old, and with annual income over $\$500,000$.

The ad market, which provides a variety of user attributes as the targeting options, , faces a pricing problem to set the optimal price for each user attribute – it is natural to allow that the costs of showing an ad to users with different attributes should not be always the same. The price of a targeting option in many advertising markets are usually decided by auctions, e.g. GSP [10] and VCG [11, 12, 13], or fixed prices. In our work [14], we propose arbitrage-free

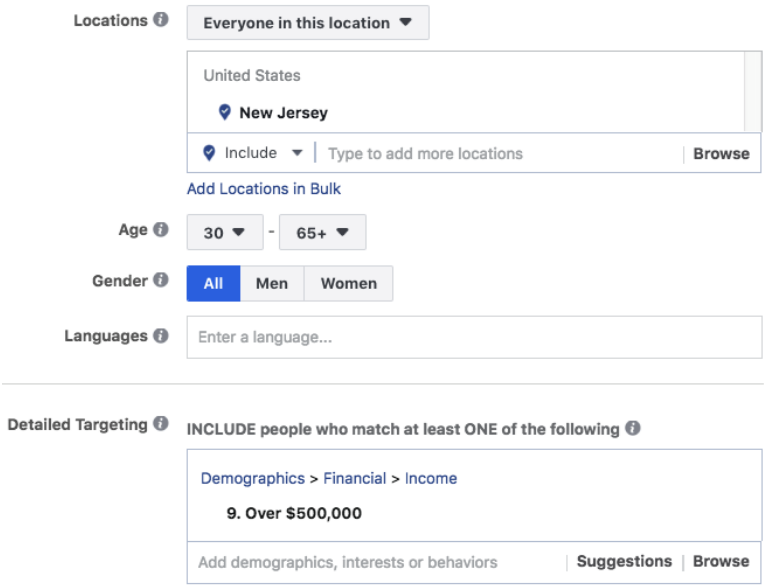


Figure 1.4: The interface of Facebook targeting system

pricing algorithms to set the ideal price for each user attribute as an targeting option provided by advertising markets, summarized in Chapter 3.

Provided with a plenty targeting options and their prices, advertisers not only have fine control over their audience but also face a targeting problem – given a set of users with specific attributes to target, the optimal strategies to split the budget over targeting options to reach as many target users as possible. In our work [15], we propose two targeting algorithms to benefit advertisers, summarized in Chapter 2.

1.3 Online Labor Market

As online labor markets have been continuously growing, we witness a promising alternative, or even revolution, to traditional employment model for the global economy. These markets provide a novel method to connect various workers and employers all around the world. Moreover, where, when and how the work is performed are redefined in such markets. For employers, besides the more flexible and less time-consuming hiring process, they have a more economic access to workers with a variety of specialized skills. For workers, as long as they have professional skills, they can start to make money at any time at any location. There are also all sorts of tasks, simple or complex, short-term or long-term, ranging from recognizing birds from images

to building a large business intelligence system.

According to [16], there are two different but sometimes overlapping types of tasks:

- *Microwork*, where a task can be completed by a single worker within a short time, e.g. a few minutes. To finish a microwork, usually the workers only need some basic numeracy and literacy skills, e.g. image tagging, data entry and text transcription. Many unemployed individuals without advanced professional skills are involved in the market of microworks, e.g. Amazon Mechanical Turk (AMT) and CrowdFlower.

- *Macrowork*, where the task tends to be a larger project that requires longer time to perform – days or months. For example, Upwork and Freelancer are providing such macrowork. A macrowork usually prefers workers with a higher level of professional skills, e.g. translation, software development and data analysis. For example, the Fig 1.5 shows the description of a web development task in Upwork. The estimated duration and compensation are in the blue box, and the required skills are in the red box.

Figure 1.5: The description of a web development task in Upwork

The major difference between microwork and macrowork is the complexity and compensation of the task. As the professional barriers of taking a microwork are very low, the compensation of a microwork is usually small, making that a worker has to complete as many microworks as possible to sustain a reasonable salary. Therefore, the quality of microworks are the major concerns of the employers. There are many research to improve the quality, e.g. to select the

workers with some attributes that fit for a given task [17], to learn the true label from multiple noisy labels [18], and to financially incentivize workers to produce higher quality [19]. In contrast, since specific skills are required by a macrowork, the hourly payment is usually higher, making that macrowork becomes a good choice for many professionals who need part time jobs. Through data analysis, [20] shows that the hiring probability is highly influenced by the skillset of the worker. [21] models the expertise level of a worker's skill.

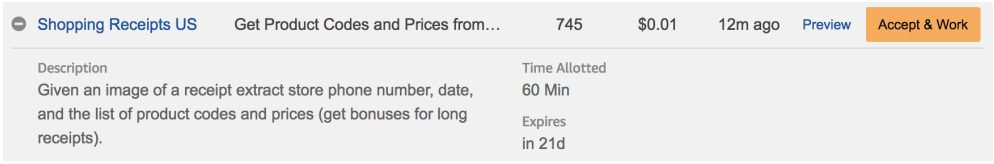


Figure 1.6: The description of a work in MTurk

In most microwork markets, the payment of a task is set by the employer, e.g. 1 cent to identify required information from image in AMT as shown in Fig 1.6. The workers can freely choose to work on microworks of which they satisfy the basic qualifications. This business model implies that a worker has to spend much time to identify the most profitable and suitable microworks by herself [22]. Meanwhile, the microwork has very limited control of the workers who works on it. In contrast, a macrowork often needs interviews before making hiring decisions as shown in the yellow box in Fig 1.5, and the payment is sometimes negotiable rather than solely decided by the employer.

Between the microwork and macrowork, many tasks belong to the overlap, e.g. translating a single page of text. For such a task, taking interview might be too costly for the task owner, but publishing it as a microwork, quality might be a concern. Therefore, the market needs a pricing and matching system that is able to automatically do the followings:

- Determine the payment of a task
- Assign a proper worker to the task
- Treat both the worker and the task fairly

With such a system, a worker no longer needs to spend time in interviewing for tasks or identifying profitable tasks by herself because the system is guaranteed to treat her fairly, and similarly, a task no longer needs to struggle in interviewing candidates or determining (or

negotiating) the payment because the system will do all of these automatically. In our work [4] which summarized in Chapter 4, we address this problem in the following steps. We first identify the cases that automatic pricing and matching will possibly treat workers and/or tasks unfairly if the market simply aims to maximize the revenue. After that, we formulate the novel research problem – revenue-maximizing stable pricing problem. To tackle this problem, we first propose a truthful, stable mechanism with randomized uniform pricing with revenue guarantee, then we extend this mechanism with non-uniform pricing. Finally, we design an online stable mechanism for the case when tasks come online.

1.4 Research Directions

In this thesis, we contribute to the answers to the following three pricing-related research directions that are interested by the Internet-based markets.

The first research direction is analyzing the current prices. According to both theories and experience, the strategies of entities in the market are highly influenced by the prices. Therefore, to understand the current prices of a market is one of the most important ways to understand the status quo of the market. In advertising markets, [2] analyzes the auction prices of Facebook advertising market by reverse engineering its suggested bids, and [23] explores the ads prices considering privacy. Beyond advertising, [24] analyzes the spot prices in the Amazon cloud market, [25] investigates the costs in the Amazon crowdsourcing market, and [26] studies the patterns of surge prices in Uber ridesharing market. In our work [1], we crawled the prices data from LinkedIn and Facebook advertising markets, and through analysis, we find that the prices of showing ads to users with different attributes vary a lot. Our analysis results are used for verifying the hypotheses, which provides the core insights for the data-driven algorithm in Chapter 2.4.

The second research direction is the optimal buying strategies when buyers are given some knowledge of the current prices. Rationally, a smart buyer would explore the optimal buying strategies to benefit his own business by exploiting the knowledge of the prices in the market. For example, [27] and [28] study the optimal bidding strategies in the sponsored search ads market and the cloud market respectively. In our work [15], we formulate and solve a novel

targeting problem to find the optimal targeting strategy for advertisers (as the buyers in the advertising market) given the analysis results of the current prices, which is summarized in Chapter 2.

The last but not least research direction is the optimal pricing strategy for the market or sellers. A market owner or sellers are seeking to price the transactions or products properly to maximize their revenue or other measurable benefits, sometimes in long term. Rich pricing research has been provided for this issue. [29] studies the revenue maximization problem for envy-free pricing, which is one representative equilibrium pricing in a variety of economic settings [30]. In advertising markets, various auction methods are used, e.g. GSP [10] and VCG [11, 12, 13]. In data markets, price arbitrage occurs among queries when one query can be determined by a group of other queries, and [31] proposes an arbitrage-free pricing for queries. In our work [14], we find that a market with a variety of users has similar but distinct arbitrage problems because a single user may have many attributes, and we propose a revenue-maximizing arbitrage-free pricing for the market owner, which is summarized in Chapter 3. Moving to the labor market, we find that the revenue-maximizing pricing may treat workers and/or employers unfairly. Therefore in our work [4], we propose stable pricing mechanisms with guarantee of both revenue and fairness for the labor market, which is summarized in Chapter 4.

Chapter 2

Targeting Algorithms in Online Advertising Markets

2.1 Introduction

Online advertising is one of the pillars in the Internet industry. In 2013, the online advertising markets generated 42.8 billion dollars in revenue in the US alone[1]. An online advertising market allows advertisers to pay for targeting (by showing ads to) specific audience through its targeting language. Google AdWords, the largest online ad network, for instance, allows advertisers to target users based on search terms from user input, the website (publisher) that the user is browsing, and simple user demographics (gender, age, location). The cost of reaching a user from a specific user set is set by auction mechanisms, e.g. second-price auction [32] or by contracts.

Other online advertising markets, specifically that run by Facebook, LinkedIn and other OSNs, offer much finer targeting controls. Their targeting languages contain detailed information shared directly by users, inferred from user daily activities [33] or purchased from third-part data providers. This includes detailed educational records about the user, past and present employment experience, significant life events like changes in marital status or birth of a baby. LinkedIn, for instance, allows advertisers to target a software engineer in Microsoft, or a college student whose major is nursing. Obviously, the price to advertise users varies with their characteristics [2]. For example, in a certain time period, the cost to target any software engineer in Microsoft could be twice high as the cost to target a nursing student.

This motivates a natural *targeting problem*. Each user x has a set $C(x)$ of characteristics. An advertiser has set S_T of characteristics of his interest and hence the set $U(S_T)$ of users (for any $x \in U(S_T)$, $S_T \subseteq C(x)$) is the advertisers' *preferred set* of users to reach with

[1] http://www.statista.com/statistics/275883/online-advertising-revenue-in-the-us-by-half-year/

ads. The advertiser has some budget b_0 and can split it to reach users with any combination of characteristics. The optimal strategy is a way to split the budget so that the set T of users reached has as much of $U(S_T)$ as possible, i.e., $|T \cap U(S_T)|$ is maximized. We consider two settings and our contributions are:

- `OSN-perspective.` An OSN can take in an advertiser's preferred set and solve his targeting problem. This is called proxy-bidding and OSNs (and other ad platforms like AdWords) provide such a service. In this case, the OSN knows the precise mapping from x to $C(x)$ for any x. We present a polynomial algorithm that has $1 - 1/e$ approximation guarantee. In this algorithm, we first define the marginal increment ratio, and then iteratively allocate budget to the characteristic sets with the largest ratio.

- `Advertiser-perspective.` If an advertiser were to solve the targeting problem on his own with the price estimates provided by the OSNs, then the advertiser does not know $C(x)$ precisely for each x and only knows the number of users, i.e. $|U(S)|$, with a given set S of characteristics. In this case, the advertiser cannot infer the overlap between $U(S_1)$ and $U(S_2)$ for two different characteristic sets S_1 and S_2. We focus on *subset targeting*, that is, given S_T of interest to the advertiser, he splits budget between S_i's such that $U(S_i) \subseteq U(S_T)$ and $U(S_i)$'s are pairwise disjoint. We propose a fast greedy algorithm using subsets and study its performance empirically using a unique dataset consisting of more than one million suggested bids from Facebook and LinkedIn that we crawl.

The rest of this chapter is organized as follows. Section 2.2 introduces the background of OSN advertising and formulates the problem. In Section 2.3, we describe the approximation algorithm for OSN-perspective. In Section 2.4, we first propose two heuristics and based on them we propose the greedy algorithm for Advertiser-perspective. We introduce the non-trivial work of crawling the datasets in Section 2.5, and evaluate the greedy algorithm for Advertiser-perspective in Section 2.6. We summarize related works in Section 2.7, and conclude in Section 2.8.

2.2 Problem Formulation

In OSN advertising markets considered in this chapter, e.g. Facebook and LinkedIn, an OSN user x has one or more characteristics,and every characteristic contains exactly one attribute $a \in A$ and one value $v \in D(a)$ of the attribute. For example, if a user has the characteristic (Location:CA), it means that the OSN knows that the user lives (or works) in California. Let A be the set of all attributes, e.g. $A = \{\text{Location, Age, ...}\}$, and $D(a)$ be the domain of the attribute a, e.g. $D(\text{Location}) = \{\text{CA, NY, NJ,...}\}$. Let S denote a set of characteristics and $U(S)$ denote the set of all the users who have **all** the characteristics in S. For example, if $c_1 =$ (Location:CA) and $c_2 =$ (Gender:Female) and $S_1 = \{c_1, c_2\}$, we say that $U(S_1)$ is the set of all the users who are female and live in CA. But any user in $U(S)$ could possibly have other characteristics, e.g. (Age:18-21). Let $A(S)$ denote the set of attributes involved in S, e.g. $A(S_1) = \{\text{Location, Gender}\}$.

When an advertiser wants to promote his campaign, we assume that he has a limited budget b_0 and a single preferred characteristic set S_T, which means that the advertiser only wants to target (by showing an ad to) users in $U(S_T)$. For example, a luxury car dealer in CA may only want to target rich people in CA. To be clear, if the characteristic set of a user is a superset of S_T, the user is equivalently preferred by the advertiser, but a user who lacks any characteristic in S_T is not preferred. For example, a rich software engineer in CA, a rich banker in CA and a rich woman in CA are assumed to be equivalently preferred by this car dealer, but a software engineer in CA (without knowing s/he is rich) is not preferred by the dealer. For any S, the number of its users (i.e. $|U(S)|$) and the price $p(S)$ are public to advertisers. More specifically, $p(S)$ is the unit cost to target any user in $U(S)$.

It is trivial to see that if an advertiser allocates budget b ($\leq |U(S)| \cdot p(S)$) to target the user set $U(S)$, the number of targeted users is $\frac{b}{p(S)}$. We assume that these $\frac{b}{p(S)}$ users are uniformly sampled from $U(S)$, i.e. each user $x \in U(S)$ has the equal probability $\frac{b}{|U(S)|p(S)}$ to be targeted. Now we formulate the targeting as a maximization problem in Eq (2.1): given the preferred characteristic set S_T and the budget b_0 from the advertiser and the price function $p(S)$ for any S from the OSN, we want to find an optimal allocation of budget, i.e. a vector $\mathbf{B} = (B_1, ..., B_N)$ where B_i is the budget allocated to target the user set that corresponds to

the i-th characteristic set[2] (i.e. $U(S_i)$), to

$$\begin{aligned} \underset{B_1,...,B_N}{\text{maximize}} \quad & \sum_{x \in U(S_T)} \min\{1, f(x)\} \\ \text{subject to} \quad & 0 \leq B_i, \forall i \in \{1, ..., N\} \\ & \sum_{i=1}^{N} B_i \leq b_0 \end{aligned} \tag{2.1}$$

$f(x)$ is the expected (total) number of times that the user x is targeted. In other words, if x is preferred (i.e. $x \in U(S_T)$), we say that the advertiser has reached $f(x)$ unique preferred user in expectation (by ignoring others). We use $\min\{1, f(x)\}$ to formulate the constraint that even if any user is targeted more than once, we only count it as one targeted user. Thus, the expected total number of reached preferred users is $\sum_{x \in U(S_T)} \min\{1, f(x)\}$. According to the uniform assumption, $f(x)$ is defined in Eq (2.2) where $\mathbb{1}_{x \in U(S_i)}$ is an indicator function, returning 1 if $x \in U(S_i)$ and 0 otherwise.

$$f(x) = \sum_{i=1}^{N} \mathbb{1}_{x \in U(S_i)} \frac{B_i}{|U(S_i)| p(S_i)} \tag{2.2}$$

However, in many OSN markets, e.g. Facebook, LinkedIn and Twitter, advertisers are unable to evaluate the indicator function $\mathbb{1}_{x \in U(S_i)}$ for any x and S_i, because they are not allowed to know which specific users have which specific characteristics for privacy concerns. Thus, we consider two settings of the problem. For the first setting where $\mathbb{1}_{x \in U(S_i)}$ is computable, we name it as the *OSN-perspective* setting since OSNs has the complete knowledge. For the second setting where $\mathbb{1}_{x \in U(S_i)}$ is not computable, we name it as the *Advertiser-perspective* setting. For the OSN-perspective setting, we propose a polynomial algorithm with performance guarantee, and for the Advertiser-perspective setting, we propose a greedy algorithm based on reasonable heuristics.

[2] Assuming that we have an ordered indexing, from 1 to N, for every characteristic set, and we will analyze the size of N in Section 2.4.2.

2.3 The Algorithm for OSN-perspective

To present the algorithm, we first formulate R_i as the ratio of *marginal increment* (of the objective function in Eq (2.1)) to a small amount of budget allocated to the i-th characteristic set, i.e. S_i. We define that a user set $U(S)$ is *full* if it is true that $\forall x \in U(S) \cap U(S_T)$, $f(x) \geq 1$. In other words, being full means that every user belonging to the intersection $U(S) \cap U(S_T)$ has already been targeted for at least once in expectation. On the other hand, if $U(S)$ is not full, there always exists at least one user x such that $x \in U(S) \cap U(S_T)$ and $f(x) < 1$. Thus, for any $U(S_i)$ that is not full, we can always find a small $\epsilon_i \in (0, 1]$ such that $f(x) + \epsilon_i \leq 1$, $\forall x \in U(S_i) \cap U(S_T)$ and $f(x) < 1$. Let $h_i = \max\{f(x) | x \in U(S_i) \cap U(S_T), f(x) < 1\}$, then the largest value of ϵ_i is $1 - h_i$. It is straightforward to see that if the advertiser allocates $\epsilon_i \cdot |U(S_i)| \cdot p(S_i)$ dollars to target $U(S_i)$, the objective function will be increased by $\epsilon_i \sum_{x \in U(S_i) \cap U(S_T)} \mathbb{1}_{f(x)<1}$. Dividing the increment by the budget, we have R_i in Eq (2.3).

$$R_i = \frac{\sum_{x \in U(S_i) \cap U(S_T)} \mathbb{1}_{f(x)<1}}{|U(S_i)| p(S_i)} \tag{2.3}$$

The essential idea of the algorithm is that for each iteration, we first identify the characteristic set S_{i^*} with the largest marginal increment, i.e. $i^* = \text{argmax}_i R_i$, and then allocate budget to $U(S_{i^*})$ till the budget is exhausted or $U(S_{i^*})$ is full.

Algorithm 2.1 Approximation Algorithm

Input S_T and b_0 from the advertiser, pricing and mapping from the OSN.
Output $(B_1, ..., B_N)$.

$\forall x, f(x) = 0; \forall i \in [N], B_i = 0$
repeat
  $i^* \leftarrow \text{argmax}_i R_i$
  $\Delta b \leftarrow \min\{b_0, |U(S_{i^*})| p(S_{i^*})(1 - h_{i^*})\}$
  $B_{i^*} \leftarrow B_{i^*} + \Delta b$
  for $x \in U(S_{i^*}) \cap U(S_T)$ **do**
    $f(x) \leftarrow f(x) + \frac{\Delta b}{|U(S_{i^*})| p(S_{i^*})}$
  end for
until $b_0 = \sum_{i=1}^{N} B_i$ or $\forall x \in S_T$, $f(x) \geq 1$
return B

Proposition 2.1. *The algorithm will stop after at most m iterations. With a proper preprocessing, the running time for each iteration is $O(m + N)$. Thus the overall time complexity is*

$O(m^2 + mN)$ where[3] $m = |U(S_T)|$ and N is the total number of characteristic sets.

Proof Sketch: In each iteration, there is at least one user $x \in U(S_T)$ whose $f(x)$ is increased to 1 (if there is no such a user, it means that the budget runs out and the algorithm will stop immediately). Thus the algorithm will stop after at most m iterations. We can do an $O(mN)$ reprocessing (only once before "repeat" in line 2) to build the mapping between all the users in $U(S_T)$ and all the characteristic sets. Moreover, we can maintain the values of $\sum_{x \in U(S_i)} \mathbb{1}_{f(x)<1}$ for all S_i and update them after we update $f(x)$ between line 7 and 8, thus the running time of each iteration becomes $O(m + N)$.

We show that the greedy that maximizes the marginal value in Eq (2.3) yields $1 - 1/e$ guarantee in Theorem 2.2. Here we extend the techniques in [34] (for budgeted maximum coverage problem) to prove Theorem 2.2.

Theorem 2.2. *Algorithm 2.1 is a $1 - 1/e$ approximation to the targeting problem in the OSN-perspective setting.*

Proof. W.l.o.g. we assume that by using Algorithm 2.1, b_0 is exhausted after exactly $T > 0$ iterations. Let $\mathbf{B}^t$ denote the budget allocation vector after t-th iteration and the allocated budget is denoted by $|\mathbf{B}^t| = \sum_{i=1}^{N} B_i^t$. Let c^t denote the budget allocated in the t-th iteration, i.e. $c_t = |\mathbf{B}^t| - |\mathbf{B}^{t-1}|$; $g(\mathbf{B})$ is the value of objective function in Eq (2.1) given the allocation $\mathbf{B}$. It is easy to see that $\mathbf{B}^0 = \mathbf{0}$ and $g(\mathbf{B}^0) = 0$. We also assume that there exists an optimal allocation $\mathbf{B}^*$ (not necessarily unique) which maximizes the objective function (note that the number of iterations to compute the optimal allocation is not necessarily T). Let $j(t) \in [N]$ denote the index of the characteristic set chosen by the algorithm at the t-th iteration (i.e. $S_{j(t)}$).

Lemma 2.3. *After each iteration $t \in [T]$, the following holds:*

$$g(\mathbf{B}_t) \geq \frac{c_t}{b_0} g(\mathbf{B}^*) + (1 - \frac{c_t}{b_0}) g(\mathbf{B}^{t-1}) \tag{2.4}$$

Proof. Let $\hat{\mathbf{B}}^{t-1}$ denote an allocation vector such that $\hat{\mathbf{B}}^{t-1} = (\max\{B_i^{t-1}, B_i^*\})_{i=1}^{N}$. It is trivial to see that $|\hat{\mathbf{B}}^{t-1}| - |\mathbf{B}^{t-1}| \leq b_0$. For any i such that $B_i^{t-1} < \hat{B}_i^{t-1}$, we can find $R_i \leq R_j(t)$ since the marginal increment ratio $R_{j(t)}$ chosen by Algorithm 2.1 is the largest at

[3] *This is polynomial because the number of bits to represent the full membership of a user is N.*

iteration t. Thus we have $g(\hat{\mathbf{B}}^{t-1}) - g(\mathbf{B}^{t-1}) \leq b_0 R_{j(t)}$. Noting that $g(\mathbf{B}^*) \leq g(\hat{\mathbf{B}}^{t-1})$ and $R_{j(t)} = \frac{g(\mathbf{B}^t) - g(\mathbf{B}^{t-1})}{c_t}$, we prove the lemma. □

Lemma 2.4. *After each iteration $t \in [T]$, the following holds:*

$$g(\mathbf{B}_t) \geq \Big(1 - \prod_{j=1}^{t}(1 - \frac{c_j}{b_0})\Big) g(\mathbf{B}^*) \tag{2.5}$$

Proof. We prove the lemma by induction on the number of iterations in which the allocation $\mathbf{B}_t$, $t = 1, ..., T$ are considered. For $t = 1$, it is true by directly applying Lemma 2.3. Suppose the statement of the lemma holds for iterations from 1 to $t-1$, we show that it is also holds for the t-th iteration.

$$\begin{aligned} g(\mathbf{B}_t) \quad &\geq \frac{c_t}{b_0} g(\mathbf{B}^*) + (1 - \frac{c_t}{b_0})\Big(1 - \prod_{j=1}^{t-1}(1 - \frac{c_j}{b_0})\Big) g(\mathbf{B}^*) \\ &= \Big(1 - \prod_{j=1}^{t}(1 - \frac{c_j}{b_0})\Big) g(\mathbf{B}^*) \end{aligned} \tag{2.6}$$

□

Note that $\prod_{t=1}^{T}(1 - \frac{c_t}{b_0}) \leq (1 - \frac{1}{T})^T < \frac{1}{e}$, thus we have $g(\mathbf{B}^T) \geq (1 - \frac{1}{e}) g(\mathbf{B}^*)$, proving the theorem. □

2.4 The Algorithm for Advertiser-perspective

Although the proposed algorithm in Section 2.3 has a desirable performance guarantee, it is not practical to individual advertisers due to the two challenges as follows.

Huge Search Space. Algorithm 2.1 has a polynomial time complexity w.r.t. the number of all the characteristic sets, i.e. N, however, in the mainstream OSN advertising markets, N could be very large since the advertiser can compose a characteristic set by arbitrarily choosing compatible[4] characteristics. Even if we only allow that all the characteristics in a characteristic set have different attributes (thus they are compatible to each other), there

[4]For example, the two characteristics (Location:CA) and (Age:18-21) are compatible to each other but (Age:18-21) and (Age:22-24) are not because any user has only one number for her age. Whether any two characteristics are compatible is decided by the OSNs. It is mostly likely safe to say that if any two characteristics with different attributes, they are compatible.

are up to $\prod_{a \in A} \Big(|D(a)| + 1 \Big)$ distinct characteristic sets (in our crawled LinkedIn dataset, $N \approx 2.4 \times 10^{12}$)! This means that any practical algorithm cannot traverse all the characteristic set even for once.

Incomplete Information. In the Advertiser-perspective setting, advertisers do not know the value of the indicator function $\mathbb{1}_{x \in U(S_i)}$ for any x and S_i. To our best knowledge, in this setting advertisers cannot even compute the objective function in Eq (2.1) in polynomial time for a given allocation of budget.

2.4.1 Data-driven Heuristics

We propose two heuristics to address the two challenges respectively. For the huge search space, we proposed the heuristic of subset targeting and verify it through a dataset of suggested bids crawled from Facebook and LinkedIn. The detailed description of the dataset is in Section 2.5. In short, we have 29420 distinct characteristic sets from Facebook and 8056 from LinkedIn. For each S, we know $p(S)$ and $|U(S)|$. To present the heuristics, we first define cheap characteristic set in Definition 2.1.

Definition 2.1. *We say a characteristic set S is* cheap *to S_T iff $|U(S)| \cdot p(S) < |U(S) \cap U(S_T)| \cdot p(S_T)$.*

It implies that $|U(S) \cap U(S_T)| > 0$ is a necessary condition for any S to be cheap to S_T. If S is cheap to S_T, according to the definition, it means that if the advertiser wants to target only $|U(S) \cap U(S_T)|$ preferred users, the cost to target $U(S)$ is less than the cost to directly target $U(S_T)$. Finding out cheap characteristic sets to S_T is the essential idea to address the targeting problem. We present the first heuristic as follows.

Heuristic 2.5 (Subset Targeting)**.** *For any preferred characteristic set S_T, the algorithm only needs to consider allocating budget to a set $\mathbf{S}$ of characteristic sets such that $U(S) \subseteq U(S_T)$, $\forall S \in \mathbf{S}$.*

Instead of directly targeting $U(S_T)$, in general, the advertiser could consider three types of targeting strategies, namely *superset* targeting (i.e. targeting some S such that $U(S) \supseteq U(S_T)$), *subset* targeting (i.e. $U(S) \subseteq U(S_T)$) and *overlap* targeting (i.e. $U(S) \cap U(S_T) \neq \emptyset$ but neither

superset nor subset). It is trivial to see that targeting any S such that $U(S) \cap U(S_T) = \emptyset$ cannot increase the objective function, thus the algorithm will never target disjoint characteristic sets. To be clear, when we mention subset targeting for S_T, it means that we find a characteristic set S such that $U(S) \subseteq U(S_T)$ other than $S \subseteq S_T$, similarly for superset and overlap targeting.

Heuristic 2.5 would be reasonable if for any S_T and S, when $U(S)$ is the superset of $U(S_T)$ or is partially overlapped with $U(S_T)$, S has very low probability to be cheap to S_T. If it is true, we lose only a few cheap characteristic sets by ignoring superset and overlap targeting. To verify this, we test the following three hypotheses through data analysis. Note that, for each hypothesis we test it over all the data snapshots and report the average results.

Hypothesis 2.6 (Infeasibility of Superset Targeting)**.** *For any S and S_T such that $U(S) \supseteq U(S_T)$, it is true that $p(S_T) \leq p(S)\frac{|U(S)|}{|U(S_T)|}$.*

The superset targeting strategy is that, instead of directly targeting $U(S_T)$ with the unit cost $p(S_T)$, the advertiser targets a superset $U(S) \supseteq U(S_T)$. For example, instead of targeting any software engineer in Microsoft, we target any employee in Microsoft. One necessary condition for this strategy to be feasible is that we can reject Hypothesis 2.6. By examining all the 175392 pairs of characteristic sets (one's user set is the superset of the other's user set) from Facebook, there is only 1.3 ($< 0.01\%$) pair violating the hypothesis. Similarly in LinkedIn, only 17 ($\approx 0.07\%$) out of the 24420 pairs violate the hypothesis. These observations support that if $U(S) \supseteq U(S_T)$, S is unlikely to be cheap to S_T for any S and S_T. Thus the hypothesis is verified.

Hypothesis 2.7 (Infeasibility of Overlap Targeting)**.** *For any S and S_T such that $U(S_T) \not\subseteq U(S)$, $U(S) \not\subseteq U(S_T)$ and $U(S_T) \cap U(S) \neq \emptyset$, it is true that $p(S_T) \leq p(S)\frac{|U(S)|}{|U(S_T) \cap U(S)|}$.*

With this strategy, for instance, instead of targeting software engineers in Microsoft, we target male employees in Microsoft. One necessary condition for this strategy to be feasible is that we can reject Hypothesis 2.7. However, as we verify all the 326758 pairs of characteristic sets (one's user set is partially overlapped with the other's user set) from Facebook, there are only 56 ($\approx 0.02\%$) pairs violating the hypothesis. Similarly, out of all the 16112 pairs of characteristic sets from LinkedIn, there are only 15 ($\approx 0.09\%$) violating pairs. These observations confirm Hypothesis 2.7, thus the overlap targeting is infeasible.

Hypothesis 2.8 (Infeasibility of Subset Targeting)**.** *For any S and S_T such that $U(S) \subseteq U(S_T)$, it is true that $p(S_T) \leq p(S)$.*

With this strategy, the advertiser can alternatively target a subset of $U(S_T)$. For example, instead of targeting any software engineer in Microsoft, the advertiser only targets entry-level software engineers in Microsoft. One necessary condition for this strategy to be feasible is that we can reject Hypothesis 2.8. Surprisingly, we find that among all the 175392 pairs (one's user set is the subset of the other's user set) of characteristic sets from Facebook, there are 93165 ($\approx 53.1\%$) pairs violating the hypothesis. Similarly, out of all the 24420 pairs from LinkedIn, we find 9430 ($\approx 38.6\%$) violating pairs. These observations show that, both in Facebook and LinkedIn, subset targeting are potentially feasible strategies to solve the targeting problem.

Through hypotheses testing, we show that for more than 99.9% cases, there is no cheap characteristic set for superset or overlap targeting. Thus we conclude that the Heuristic 2.5 is reasonable. Note that, any combination of the three targeting strategies still belongs to one of them. For example, assuming $U(S)$ is a subset of $U(S')$ and $U(S')$ is partially overlapped with $U(S_T)$, it is easy to see that $U(S)$ must be either a subset or an overlapped set of $U(S_T)$. This means that although we only verify the three "simple" strategies, the verification indeed covers all possible "paths", i.e. any combination of the three strategies.

Next, the critical reason for the second challenge is that a user with multiple characteristics appear in multiple (say k) user sets. If an allocation targets all the k user sets, we do not find any polynomial method (w.r.t. N) to compute $f(x)$ because we do not know the indicator function $\mathbb{1}_{x \in U(S)}$ for any x and S. However, if an allocation only targets k pairwise-disjoint subsets of $U(S_T)$, this would not be a problem because once we allocate budget B_i to $U(S_i)$, the value of objective function will be increased by $|U(S_i) \cap U(S_T)| \cdot \min\{1, \frac{B_i}{p(S_i)|U(S_i)|}\}$. Thus, we propose Heuristic 2.9.

Heuristic 2.9 (Disjoint Targeting)**.** *We only consider to allocate budget to a set of characteristic sets whose corresponding user sets are pairwise disjoint.*

Remark 2.10. *Applying Heuristic 2.5 and 2.9 together, we consider to allocate budget to only a set* $\mathbf{S}$ *of characteristic sets such that:*

- $\forall S \in \mathbf{S}$, $U(S) \subseteq U(S_T)$.

- $\forall S \neq S' \in \mathbf{S}, U(S) \cap U(S') = \emptyset$;

2.4.2 Revised Greedy Strategies

Based on the two heuristics, we propose a top-down greedy algorithm presented in 3 subroutines in Procedure 2.2, 2.3 and 2.4. Given S_T and b_0, by calling FINDSUBSETS(S_T, b_0) it outputs the budget allocation in sparse representation. The result is a set L of pairs, i.e. $L = \{(S, b)|b > 0\}$. Each pair (S, b) stands for the algorithmic decision that the advertiser should allocate b dollars to target $U(S)$, and the advertiser is expected to reach unique $\frac{b}{p(S)}$ users in $U(S_T)$. By following the allocation L, the advertiser is expected to target $\sum_{(S,b)\in L} \frac{b}{p(S)}$ users in $U(S_T)$.

Starting with the initial preferred characteristic set S_T and initial budget b_0, the algorithm iteratively selects and adds the locally optimal characteristic to the current preferred characteristic set (denoted by $S^{(t)}$; note that t, the iteration number, starts from 0 and $S^{(0)} = S_T$) to construct a new preferred characteristic set $S^{(t+1)}$. It is easy to see that $U(S^{(t+1)}) \subseteq U(S^{(t)}) \subseteq ... \subseteq U(S_T)$ since the algorithm only adds characteristics. If the remaining budget $b^{(t)}$ (note that $b^{(0)} = b_0$) is not enough to target all the users in $U(S^{(t+1)})$, then we recursively call the algorithm to solve another targeting problem with $S^{(t+1)}$ as the initial preferred characteristic and $b^{(t)}$ as the total budget. Otherwise, by calling Procedure 2.4, the algorithm allocates $p(S^{(t+1)}) \cdot |U(S^{(t+1)})|$ budget to fully target $U(S^{(t+1)})$, and after this, the algorithm will select and add another characteristic to $S^{(t)}$ to form a new current preferred characteristic set $S'^{(t+1)}$, and repeats this process until all the users in $U(S_T)$ are targeted or the budget is exhausted.

Algorithm 2.2 Find Subsets

```
1: procedure FINDSUBSETS(S, b)
2:     n ← min{|U(S)|, b/p(S)}
3:     a* ← argmax_{a∈A\A(s)} EVALUATEATTRIBUTE(S, b, a)
4:     if EVALUATEATTRIBUTE(S, b, a*) > n then
5:         return ALLOCATEBUDGET(S, b, a*)
6:     else
7:         return {(S, b)}
8:     end if
9: end procedure
```

Proposition 2.11. *The time complexity of the algorithm is* $O(|A|^2 + |A| \sum_{a\in A} |D(a)|^2)$.

Algorithm 2.3 Evaluate Attribute

procedure EVALUATEATTRIBUTE(S, b, a)
  $D' \leftarrow D(a)$, $n \leftarrow 0$, $V \leftarrow \emptyset$
  while $b > 0$ and $D' \neq \emptyset$ **do**
    $v^* \leftarrow \underset{v \in D'}{argmin}\ p(S \cup \{(a : v)\})$
    $D' \leftarrow D' - \{v^*\}$
    if $\forall v \in V, u(\{(a : v^*)\}) \cap u(\{(a : v)\}) = \emptyset$ **then**
      $S^* \leftarrow S \cup \{(a : v^*)\}$
      $\Delta n \leftarrow \min\{|U(S^*)|, \frac{b}{p(S^*)}\}$
      $n \leftarrow n + \Delta n$
      $b \leftarrow b - p(S^*)\Delta n$
      $V \leftarrow V \cup \{v^*\}$
    end if
  end while
  return n
end procedure

Proof Sketch: for any input pair (S_T, b_0), Procedure 2.2 and 2.4 will be called at most $|A|$ times each in total, and Procedure 2.3 will be called at most $|A|^2$ times. The amortized running time of the non-recursive operations in Procedure 2.2, 2.3 and 2.4 are $O(|A|)$, $O(\frac{1}{|A|}\sum_{a\in A}|D(a)|^2)$ and $O(\frac{1}{|A|}\sum_{a\in A}|D(a)|^2)$ respectively.

Note that, the time complexity is significantly lower than that of Algorithm 2.1 because this algorithm does not enumerate all the characteristic sets as Algorithm 2.1 does, instead, it goes through all the characteristics. Incorporating data-driven heuristics, it is not expected to have performance guarantee. Thus we evaluate its effectiveness through experiments in Section 2.6.

2.5 Price Data Acquisition

In this section, we introduce the dataset we use to test the hypotheses in Section 2.4.1 and evaluate the algorithm in Section 2.4.2, and the method that we crawl it. We need real price data from OSN advertising markets. However, to our best knowledge, there is no such large dataset available yet.

Fortunately, in order to guide advertisers who face a variety of targeting characteristic sets, both Facebook and LinkedIn advertising markets provide *Suggested Bid* which is a function that, for each characteristic set S, shows the suggested bid to win an action to show an ad to

Algorithm 2.4 Allocate Budget

procedure ALLOCATEBUDGET(S, b, a)
  $D' \leftarrow D(a), n \leftarrow 0, V \leftarrow \emptyset$
  while $b > 0$ and $D' \neq \emptyset$ **do**
    $v^* \leftarrow \underset{v \in D'}{argmin}\ p(S \wedge \langle a : v \rangle)$
    $D' \leftarrow D' - \{v^*\}$
    if $\forall v \in V, u(\{(a : v^*)\}) \cap u(\{(a : v)\}) = \emptyset$ **then**
      $S^* \leftarrow S \cup \{(a : v^*)\}$
      if $b \geq |U(S^*)|p(S^*)$ **then**
        $R \leftarrow R \cup \{(S^*, |U(S^*)| \cdot p(S^*))\}$
        $b \leftarrow b - |U(S^*)| \cdot p(S^*)$
      else if $A(s^*) \neq A$ **then**
        return $R \cup$ FINDSUBSETS(S^*, b)
      end if
      $V \leftarrow V + \{v^*\}$
    end if
  end while
  return R
end procedure

one user in $U(S)$ and the number of users in $U(S)$, i.e. $|U(S)|$. Moreover, through reverse-engineering, the recent work in [2] finds that in Facebook, the suggested bids are sampled from recent winning bids. This means that the suggested bid of a characteristic set S is the best estimate, that we are able to get from real OSN advertising markets so far, of the cost, i.e. $p(S)$, of showing an ad to one user in $U(S)$. Therefore, for any S, we use the suggested bid $\hat{p}(S)$ of S to estimate $p(S)$ for all the experiments in this chapter. Although there is no literature about how LinkedIn generates suggested bids, we also include them for experiments.

2.5.1 Suggested Bid

We first introduce suggested bids. In the advertising systems of Facebook and LinkedIn, once an advertiser creates an ad with a characteristic set S at time t, a suggested bid $\hat{p}(S)$ and the number of users in $U(S)$ are provided. It can be formulated as the function SB in Eq (2.7).

$$SB : (S, t) \rightarrow (\hat{p}(S), |U(S)|) \tag{2.7}$$

Factor	Impact on $\hat{p}(S)$?	
	LinkedIn	Facebook
Advertising History	✗	✓
OSN Activity	✗	✗
Profile	✓	✗
Budget	✗	✗
Ad Content	✗	✗

Table 2.1: Impact of different factors on $\hat{p}(S)$.

Verification. Besides S and t, there are many other factors, for example, ad content, advertising history of the advertiser, social activities[5] of the advertiser, profiles of the advertiser, and total amount of budget, that we speculate that they might influence the suggested bid $\hat{p}(S)$. To test existence of their influence, for each factor we conduct a simple A/B testing. Due to the limited space, we omit the detail and summarize the testing results in Table 2.1. In LinkedIn, if the advertiser's profile shows that he is not in the US, $\hat{p}(S)$ will be slightly different. If an advertiser has advertising records, Facebook might slightly adjust the suggested bid shown to him. Since we want to use $\hat{p}(S)$ to approximate $p(S)$ as close as possible, when we crawl suggested bids, we try to fix all other factors as follows.

Take A Snapshot. We follow the method in [35] to take a snapshot of suggested bids. Since suggested bids (and true costs) in Facebook are sensitive [2] to time, for a large number of characteristic sets, we have to crawl their suggested bids simultaneously.

New Advertiser Accounts. We follow the setting in [1]. For crawling, we create and use new advertiser accounts without any adverting history or social interaction. Each account's location is set to US. We choose Feed Ads (text ads shown in Timeline) which is a common ads type in both Facebook and LinkedIn.

We create two crawlers interacting with Facebook [36] and LinkedIn [37] advertising systems respectively. Each interaction consists of two steps. First, a crawler logs in an advertiser account, composes a characteristic set S and sends it to the OSN. The OSN returns the query result $(\hat{p}(S), |U(S)|)$ to the crawler. Since the crawlers do not create or run real campaigns, the advertising history of accounts remains empty during the entire crawling period. Besides, the

[5]The advertiser account of Facebook (or LinkedIn) is based on the normal Facebook (or LinkedIn) account.

Attribute	Domain Size	Explanation or Examples
Location	51	50 states and D.C.
Gender	2	Male, Famale
Age	8	Grouped in 5, e.g. 25-29
Income	10	Annual income interval
Education Level	13	Bachelor, Master
Ethnic Affinity	3	Hispanic, Asian, Africa

Table 2.2: Crawled Facebook User Attributes

crawling process neither costs money nor participates real auctions in the markets.

2.5.2 Crawling Suggested Bids

Facebook Ads Market. We first manually select 6 common targeting attributes and their 87 frequent values listed in Table 2.2 from Facebook advertising system. Thus we have 87 characteristics. We enumerate all the characteristic sets with up to 3 distinct characteristics. As a result, we get 29420 characteristic sets, among which there are 1, 87, 2311 and 27021 characteristic sets with exactly 0[6], 1, 2 and 3 characteristics, respectively. The crawler simultaneously issues the 29420 queries, each of which is for a characteristic set, as a snapshot. We take one snapshot per day from July 2015 to Aug 2015. As a result, we have 35 such snapshots, i.e. 1029700 data in total. We test hypotheses or evaluate algorithms in Sec 2.4 on each snapshot and report the average results over all the 35 snapshots. We use Facebook Ads APIs[7] to build the crawler.

LinkedIn Ads Market. Similarly for LinkedIn, we consider 8 common attributes and 449 of their frequent values in Table 2.3, thus we have 449 characteristics. We enumerate all the characteristic sets with up to 2 distinct characteristics. As a result, we have 8311 characteristic sets, among which there are 1, 449 and 8056 characteristic sets with exactly 0, 1 and 2 characteristics, respectively. The crawler simultaneously issues the 8311 queries as a snapshot. We harvest 4 snapshots, i.e. 33244 data in total from LinkedIn. Since LinkedIn does not release

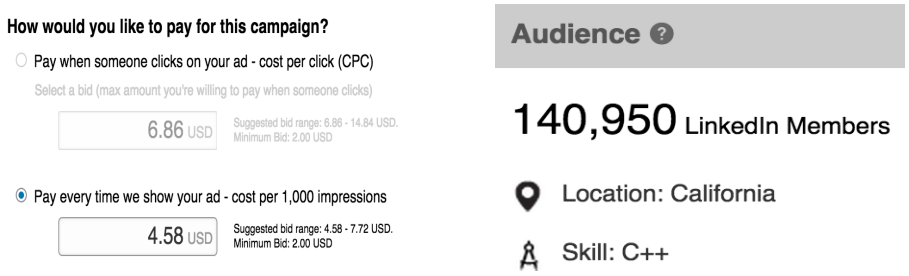
[6]For any characteristic set, if we do not specify the location, it is the US by default. Thus, the empty characteristic set corresponds to all the users in the US. It is similar for crawling LinkedIn data.

[7]https://developers.facebook.com/docs/graph-api

Attribute	Domain Size	Explanation or Examples
Location	51	50 states and D.C.
Industry	17	Agriculture, Medical
Skill	39	Programming, Cooking
Company Size	9	Number of employees
Company Name	171	Top US companies
Job Title	126	Sampled Job positions
Job Seniority	10	CXO, Director, Entry-level
Job Function	26	Sales, Support, Research

Table 2.3: Crawled LinkedIn User Attributes

open APIs for suggested bids, we use Python and Selenium (a web automation package) to mimic real advertisers to harvest data from LinkedIn advertising interface. Fig 2.1 is an example of the web page showing the suggested bid as $4.58\$$ and the number of qualifying users as 140950 for the characteristic set $\{$(Location:CA),(Skill:C++)$\}$.



(a) (b)

Figure 2.1: An example of the suggested bids provided by LinkedIn.

2.6 Experiments

In this section, we evaluate the greedy algorithm proposed for Advertiser-perspective setting in Sec 2.4.2 with both Facebook and LinkedIn datasets that are introduced respectively in Sec 2.5.2. In short, we have 4 snapshots of suggested bids from LinkedIn and 35 snapshots from Facebook, i.e. more than 1 million data in total. Since the algorithm follows subset targeting, we do not consider any characteristic set already with 2 characteristics in LinkedIn or with 3 characteristics in Facebook as S_T (because their user sets have no subset in the crawled data). We also do not consider any characteristic set whose user set size is too small. For each of

the remaining characteristic sets, we consider it as S_T in turn, i.e. the input for Procedure 2.2 (together with a budget). For each snapshot, we calculate the average results over all S_T, and then report the final average results over all snapshots.

2.6.1 Budget Variation

We first evaluate the algorithm for different budgets. We define the measurement *budget-increment* curve as follows. The baseline algorithm that we consider is to target $U(S_T)$ directly, which means the baseline can reach $\frac{b_0}{p(S_T)}$ preferred users with the budget b_0. A single evaluation point (x, y) on the budget-increment curve, shown in Fig 2.2 and 2.3, denotes that if the advertiser is given the budget enough to exactly target x ($\in (0, 1]$) proportion of preferred users by the baseline, our greedy algorithm can increase the number of reached preferred users by y proportion, i.e. reach $x(1 + y)$ proportion. In Facebook, we compute the curve for each of the 2399 characteristic sets (as S_T). The overall budget-increment curve is presented by the blue solid line in Fig 2.2. Similarly, we plot the overall budget-increment curve for 253 characteristic sets from LinkedIn with the red solid line. The green dashed line is for the baseline.

We have the following observations. First, the less budget the advertiser has, the more effective the algorithm is. As shown in Fig 2.2, when the advertiser has very limited budget, e.g. the budget is enough to target only a small portion $x \in (0, 0.05]$ of the preferred users, by our algorithm, he is able to target 40.8% more users on Facebook and 37.6% more on LinkedIn. When the advertiser has enough budget to target 20% of the total preferred users by the baseline, he is still able to target 24.9% more users in Facebook and 19.3% more in LinkedIn. Second, when the budget is approaching to monopoly the entire user set, the effectiveness of the algorithm is closed to 0, but it is guaranteed that the advertiser can target at least as many users as the baseline does. The reason that the effectiveness appears to be decreasing is that the user supplies of cheap characteristic sets are limited, and as the budget gradually increases, the user supplies of those cheap characteristic sets gradually run out. Then the algorithm has to allocate remaining budget to characteristic sets that are not cheap to S_T. Third, although Facebook and LinkedIn run different advertising markets, we find that our algorithm produces budget-increment curves in a similar shape, which shows its generalizability.

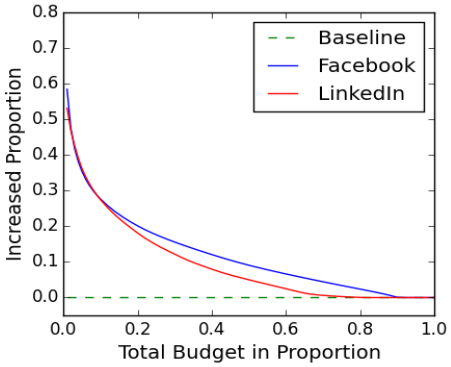


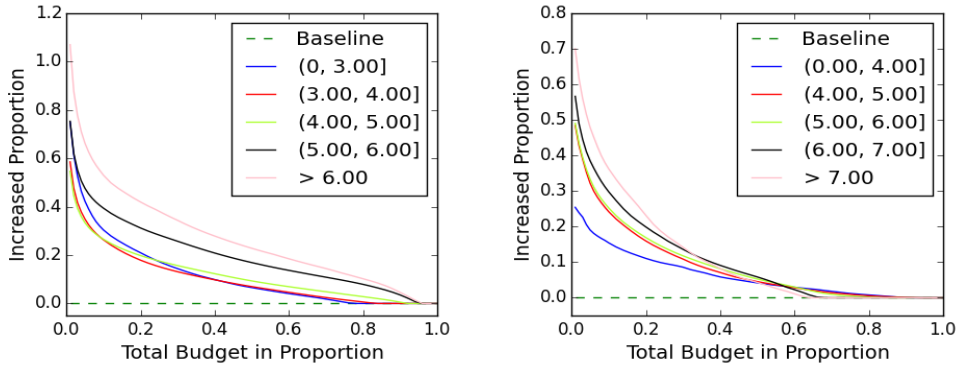
Figure 2.2: Increment with different amount of budget.

2.6.2 Price Variation

We then evaluate the algorithm's effectiveness when the cost of S_T, i.e. $p(S_T)$, varies. For LinkedIn, we create 5 price intervals, namely $(0, 4.00]$, $(4.00, 5.00]$, $(5.00, 6.00]$, $(6.00, 7.00]$ and $(7.00, +\infty]$ (all prices are in the US dollar). Then we map all the 253 characteristic sets into one of the 5 price intervals based on their prices. As a result, the 5 intervals get 5.4%, 27.2%, 28.5%, 17.5% and 21.4% of the total characteristic sets respectively. For each price interval, we compute its individual budget-increment curve for each characteristic set assigned to this interval, and then average them to get the interval-average curve. We process 2399 characteristic sets from Facebook in the similar way. The results are shown in Fig 2.3(a) and 2.3(b). We find that no matter how large $p(S_T)$ is, the proposed algorithm consistently outperforms the baseline. As shown in Fig 2.3(b), the area under each budget-increment curve decreases as the interval price decreases. This indicates that the larger $p(S_T)$ is, the larger effectiveness the algorithm has, both on LinkedIn and Facebook.

2.7 Related Work

The knowledge of the price distributions can help advertisers make better decisions. If prices are not known, advertisers can learn them with potential penalty during exploration [38]. Instead we focus on the case when the prices are known to advertisers. In sponsored search markets, budget optimization (BO) is a well studied problem in [27][39][40][41]. When given the landscapes of keywords, as pointed out by [42][43], the degenerated BO problem is an



(a) Facebook (b) LinkedIn

Figure 2.3: Performance with preferred characteristic sets in different prices.

instance of the Multiple-Choice Knapsack problem [44]. [39] proves that a simple random strategy has the $1 - 1/e$ guarantee.

However, instead of maximizing the profit, our problem has new challenges when the advertisers want to target a specific user segment in OSNs and to maximize the number of reached *unique* preferred users. [45] proposes algorithms to identify a set of alternative topics [46] that have (approximately) the same audience on a micro-blogging platforms for a targeting topic. We point out that their algorithm is not applicable in our setting for the two following reasons. First, they assume that advertisers have to purchase all the users in a set which is not true in practice. Second, they only consider the setting where the complete user information is available (similar to the OSN-perspective setting in this chapter).

Although our problem is related to the *Allocation* problem [47][48][49], they are essentially different, since our problem aims to benefit individual advertisers rather than the market owners.

2.8 Conclusions

We study the targeting problem for advertisers. For the OSN's perspective, we present a polynomial time algorithm and prove that it has $1-1/e$ guarantee. For the advertiser's perspective, we show through data analysis that the strategy of targeting subsets of audience sets is viable and propose a greedy algorithm based on subset targeting. For evaluation, we crawl a large unique dataset which contains more than one million suggested bids from Facebook and LinkedIn,

and we show that the proposed algorithm makes advertisers reach more target audience than directly targeting the users.

Our work suggests that smart advertisers can utilize the "arbitrage" to target more users for the given budget, but we do not know how these strategies influence the revenue of OSNs. Therefore, it will be of interest to study pricing mechanisms for OSNs that prevent any potential arbitrage available for advertisers. Moreover, with the large dataset of suggested bids crawled from Facebook and LinkedIn, an open problem is to predict the prices of various characteristic sets.

Chapter 3

Arbitrage-free Pricing in Online Advertising Markets

3.1 Introduction

User-based markets are a central part of the Internet Economy. In user data markets, many companies are selling opt-in email addresses. In online advertising markets, advertisers want to buy the impressions of users with specific sets of attributes, e.g. a luxury car company may prefer to show ads to rich users.

In such user-based markets, the core value of a user arises from her attributes. In TowerData, buyers can purchase the emails of users with specific demographics. Google AdWords, for example, the largest online ad network, allows advertisers to target users based on demographics and search terms. Ad markets run by online social networks, including Facebook, LinkedIn and Twitter, offer much finer targeting controls over user attributes with detailed information which is shared directly by users, inferred from user daily activities or purchased from third parties. This includes users' educational records, past and present employment experience, significant life events like changes in marital status or birth of a baby, etc. Twitter allows advertisers to target users by topics that the users are interested in.

In such markets, buyers can purchase their *target* users[1] through the query system provided by the market. Let q_i denote a simple selection query with conditions over user attributes, e.g. $q_i =$"Gender:1" returns all the male users. Let U_i be the set of all the users satisfying query q_i. A buyer can specify the users with a query and purchase them. Therefore, the market owner needs to solve a pricing problem – how to price all the queries from buyers that return users with different attributes? In this chapter, we consider the following posted pricing model. Let p_i denote the price of query q_i, i.e. the price of any user $\mathbf{u}$ satisfying query q_i (i.e. $\mathbf{u} \in U_i$).

[1] "Buying a user" is short for buying, for example, the impression of a user in advertising markets.

A buyer needs to pay $n \cdot p_i$ if he purchases $n \in \{1, \ldots, |U_i|\}$ users in U_i. The practical need behind this pricing model is that a single target user can provide positive utility to the buyer. Moreover, we assume that it would not cause much trouble to the buyer if he gets additional users who do not satisfy his target query[2].

The above pricing model benefits from the *versioning* theory for pricing information goods [50]. This theory proposes that different buyers may use an information product in different ways, and the market should provide different versions for such a product at different prices. In user-based markets, a user with multiple attributes can potentially have different versions. For example, a user who is a programmer interested in cars, can be priced and sold as at least two versions, including as a user interested in cars as one version for car dealers and as a programmer as another version for IT companies on hiring. Versioning theory is needed in such user-based markets because a user may be retrieved by multiple queries if she has more than one attribute.

We point out that such a pricing model, however, may suffer **version-arbitrage** (see Definition 3.6). In user-based markets, version-arbitrage occurs if two queries q_i and $q_{i'}$ return similar user sets but p_i and $p_{i'}$ differ a lot. If version-arbitrage exists, a buyer who really wants q_i (or $q_{i'}$) might purchase $q_{i'}$ (or q_i) instead. Version-arbitrage is caused by the fact that a user with multiple attributes potentially satisfies many queries. We use an example to illustrate the version-arbitrage and its difference from *determinancy* arbitrage in [51].

Example 3.1. *Let q_1 ="Income >100" and q_2 ="Income >101" be two queries, i.e. q_1 (or q_2) returns all the users with income higher than* 100 *(or* 101*). If $p_2 < p_1$, the version-arbitrage exists: a savvy buyer who wants to buy a user satisfying q_1 will buy a user satisfying q_2 instead (assuming $U_2 \neq \emptyset$) because any user satisfying q_2 satisfies q_1 with probability 1. However, in non-trivial databases, q_1 (or q_2) does not determine q_2 (or q_1) [51].* □

Motivated by the discussion above, we study pricing queries with conditions over user attributes, and seek revenue-maximizing pricing for a given demand. In particular, we formulate *arbitrage-free* pricing problem in user-based markets and our contributions are:

[2]This assumption is obviously true in advertising markets, i.e. showing an ad to a non-target user will not decrease the sale. In data markets, it is true if the buyer does not want aggregate results.

- Any uniform pricing where all the queries have identical price is arbitrage-free. We show that the optimal uniform pricing can be computed in polynomial time. We also show that this is an $O(\log min\{\mathcal{U}, \sum_{j=1}^{\mathcal{B}} d_j\})$ approximation to the optimal, possibly non-uniform arbitrage-free pricing, where $\mathcal{U}$ is the total number of users and $\sum_{j=1}^{\mathcal{B}} d_j$ is total number of users requested by the buyers. Besides, we show that this approximation bound is tight for uniform pricing solutions.

- We design a different, efficient greedy algorithm to compute the arbitrage-free non-uniform pricing with the same $O(\log \min\{\mathcal{U}, \sum_{j=1}^{\mathcal{B}} d_j\})$ guarantee. But by experiments, we show that its revenue is significantly larger than that of the optimal uniform pricing.

- We consider a generalized setting where a buyer has a *minimal* demand on his target users. In previous setting, the allocation problem (given a pricing) is polynomial time solvable. In this setting, we prove that both the allocation problem and pricing problem are not only NP-hard, but also hard to approximate. We present an $O(D)$ approximate allocation algorithm where D is the largest minimal demand. Turning to the pricing problem, we present a polynomial algorithm, that – based on the approximate allocation – computes a uniform pricing and we show that it is an $O(D \log \min\{\mathcal{U}, \sum_{j=1}^{\mathcal{B}} d_j\})$ approximation to the optimal arbitrage-free pricing.

Our research begins the study of the pricing that is free of version-arbitrage in user-based markets, which is new (despite many versions of envy-free, truthful and other variants of pricing in the literature) and highly needed (online marketing crucially relying on buying users based on multiple attributes). Our algorithmic results use various maxflow optimization methods.

3.2 Preliminary

3.2.1 Pricing Model

The market provides N queries for buyers. Let q_i be a *query* where $i \in \{1, \ldots, N\}$. For example, q_i="Income>100k $\wedge$ Gender:Female" returns all the female users with income higher than 100k. Different from a general database query, the notion *query* in this chapter can be viewed as a simplified selection rule over user attributes. A user is said to satisfy q_i if she can

be retrieved by q_i. For a buyer, we assume that there exists a query that represents his true target. The pricing is over queries, and the price p_i of query q_i is the **unit** price, i.e. the price per user retrieved by q_i. For example, if $p_i = \$2$, a buyer needs to pay $\$10$ for 5 users satisfying q_i.

3.2.2 Problem Formulation

$[n]$ denotes the integer set $\{1, \ldots, n\}$. Let U be the set of all the users, and we define that $\mathcal{U} \triangleq |U|$. Let $\mathbf{u} \in U$ be a user. $u^i = 1$ if $\mathbf{u}$ satisfies q_i where $i \in [N]$, otherwise $u^i = 0$. Let M be the quantity that $M \triangleq \sum_{i=1}^{N} |\{\mathbf{u}|u^i = 1, \mathbf{u} \in U\}|$. Let B be the set of buyers, and we define that $\mathcal{B} \triangleq |B|$. Buyer $j \in [\mathcal{B}]$ is denoted by a triplet (t_j, d_j, c_j) indicating that he wants to buy at most $d_j \in \mathbb{Z}_+$ users (as *demand*) satisfying q_{t_j} (as *target*) where $t_j \in [N]$, and $c_j \in \mathbb{R}_+$ is the *maximum cost* that he is willing to pay for each target user. $c_j \cdot d_j$ can be viewed as the budget constraint of buyer j.

Let $A = (A_1, \ldots, A_{\mathcal{B}})$ be an allocation of (indivisible) users to buyers where A_j is the set of users allocated to buyer j. We assume that any user can be either sold to one[3] buyer or unsold.

Definition 3.2. *An allocation A is feasible if the three constraints are all satisfied $\forall j \in [\mathcal{B}]$:*

- *Target Constraint:* $\forall \mathbf{u} \in A_j$, $u^{t_j} = 1$;
- *Demand Constraint:* $|A_j| \leq d_j$;
- *Uniqueness Constraint:* $\forall j' \neq j$, $A_{j'} \cap A_j = \emptyset$.

Let $\mathbf{P} = (p_1, \ldots, p_N)$ be the pricing function over the N queries.

Definition 3.3. *$(\mathbf{P}, A)$ is feasible if A is feasible and $\forall A_j \neq \emptyset$, $p_{t_j} \leq c_j$.*

Given $(\mathbf{P}, A)$[4], the revenue is $R(\mathbf{P}, A) \triangleq \sum_{j=1}^{\mathcal{B}} |A_j| \cdot p_{t_j}$. In this chapter, we define and solve the following *pricing problem*.

---

[3] In Section 3.5, we will discuss a more general setting where a user can be sold to multiple buyers with limited times. The algorithms for that setting are almost the same, so are the corresponding analyses.

[4] By default, we require $(\mathbf{P}, A)$ to be feasible and we will not explicitly mention this requirement later.

Definition 3.4 (Pricing Problem)**.** *Given* (N, U, B) *as the input, compute* $\mathbf{P}$ *and* A *such that* $\mathbf{P}$ *is arbitrage-free (see Definition 3.7) and* $R(\mathbf{P}, A)$ *is maximized.*

To solve the pricing problem, we need to define and solve the *allocation problem*.

Definition 3.5 (Allocation Problem)**.** *Given* $(N, U, B, \mathbf{P})$ *as the input, compute* $A = \operatorname{argmax}_{A'} R(\mathbf{P}, A')$.

Let $R(\mathbf{P}) \triangleq \max_A R(\mathbf{P}, A)$ be the optimal revenue of $\mathbf{P}$ (when the allocation is optimal).

3.2.3 Arbitrage-Free Pricing

In this part, we first formally introduce version-arbitrage and then define the arbitrage-free pricing. Note that, our arbitrage-free pricing is free of version-arbitrage, different from the arbitrage-free pricing in Koutris et al. [51] which is free of determinancy-arbitrage.

In user markets, version-arbitrage is the opportunity that a buyer is able to get a lower unit price (in expectation) of his target users by strategically choosing a substitute target (i.e. query) other than his true target. Assuming that any query is satisfied by at least one user, let $\pi(i|i')$ be the conditional probability that a user satisfies q_i if she satisfies $q_{i'}$:

$$\pi(i|i') = \frac{|\{\mathbf{u}|u^i = 1, u^{i'} = 1, \mathbf{u} \in U\}|}{|\{\mathbf{u}|u^{i'} = 1, \mathbf{u} \in U\}|}$$

We assume that a buyer, whose true target is q_i, has the *prior belief* that buying a user satisfying $q_{i'}$ is equivalent (in expectation) to buying a fraction $\pi(i|i')$ of a user satisfying q_i. Although this assumption is not necessarily practical in all the user-based markets due to unpredictable allocation rules, strategies based on this assumption (or similar ones) were studied for advertisers in online advertising markets [52, 53, 15]. Based on this assumption, we define the version-arbitrage as follows.

Definition 3.6 (Version-Arbitrage)**.** *In a market with* U*, the pricing* $\mathbf{P}$ *contains* version-arbitrage *if* $\exists i, i' \in [N]$, $p_{i'} < \pi(i|i') \cdot p_i$.

It is easy to see that Example 3.1 is a special case where $\pi(1|2) = 1$ and $p_2 < p_1$, so arbitrage exists. Next, we define the arbitrage-free pricing.

Definition 3.7 (Arbitrage-free)**.** *In a market with* U*, the pricing* $\mathbf{P}$ *is said to be* arbitrage-free *if* $\forall i, i' \in [N]$, $p_{i'} \geq \pi(i|i') \cdot p_i$.

Notably, the arbitrage-free constraints are independent of buyers B, but only depends on U in the market. This is a desirable property that no matter whether buyers report their parameters truthfully [54] or not, the market can always make the pricing arbitrage-free.

3.3 Theoretical Results

3.3.1 Uniform Pricing Algorithm

In this section, we first show that any uniform pricing is arbitrage-free. Then we show the optimal uniform pricing can be computed in polynomial time. Finally we prove that the optimal uniform pricing provides an $O(\log min\{\mathcal{U}, \sum_{j=1}^{\mathcal{B}} d_j\})$ guarantee to the optimal arbitrage-free pricing. A pricing is *uniform* if all its entries are identical, otherwise *non-uniform*. Let $\mathbf{p}_\xi$ denote the uniform pricing that $\forall i \in [N], p_i = \xi$.

Proposition 3.1. *Any uniform pricing is arbitrage-free.*

It is easy to verify Proposition 3.1. Although we do not know whether to find the optimal arbitrage-free pricing is NP-hard or has polynomial time solutions, Proposition 3.1 provides a class of feasible solutions.

Theorem 3.2. *The optimal uniform pricing $\mathbf{p}_{\xi^*}$ can be computed in polynomial time $O(\mathcal{U}\mathcal{B}M + \mathcal{U}\mathcal{B}^2)$.*

To prove this theorem, it is enough to prove Lemma 3.3 and Corollary 3.5. Basically, Lemma 3.3 states that given any uniform pricing $\mathbf{p}_\xi$, the optimal allocation can be computed in polynomial time. Following Lemma 3.4, Corollary 3.5 shows that the optimal uniform price ξ^* must be one of the maximum costs of buyers.

Lemma 3.3. *The allocation problem for uniform pricing can be solved in polynomial time $O(\mathcal{U}M + \mathcal{U}\mathcal{B})$.*

Proof. We model the allocation problem when the pricing is $\mathbf{p}_\xi$ as a maxflow problem as follows. We introduce s and t as the source and sink respectively. We then introduce N nodes $y_1, \ldots, y_N$. Assuming all the users are indexed from 1 to $\mathcal{U}$ and $\mathbf{u}_k$ denotes the k-th user. For each $k \in [\mathcal{U}]$: (1) we introduce a node x_k and a directed edge from s to x_k with capacity 1;

and (2) $\forall i \in [N]$, we introduce a directed edge from x_k to y_i with capacity 1 if $u_k^i = 1$. For each buyer j that $c_j \geq \xi$, introduce a node z_j, a directed edge from y_{t_j} to z_j and a directed edge from z_j to t with capacity d_j. It is easy to verify that the amount of the maximum flow f^* is the maximum number of sold users when the pricing is $\mathbf{p}_\xi$, thus producing the revenue $R(\mathbf{p}_\xi) = \xi \cdot f^*$. The allocation can also be easily inferred from the residual graph after the maxflow algorithm completes. We use the Ford-Fulkerson algorithm that runs in $O(|E| \cdot f^*)$. Since $f^* \leq \mathcal{U}$ and $|E| \leq M + 2\mathcal{B}$, the time complexity is $O(\mathcal{U}M + \mathcal{U}\mathcal{B})$. □

Next we show that it only needs to solve at most $\mathcal{B}$ allocation problems to compute the optimal uniform price ξ^* in Corollary 3.5. Before showing Corollary 3.5 that is specific to uniform pricing, we show a general result for any pricing in Lemma 3.4, which immediately implies Corollary 3.5 and will be used for proving Lemma 3.8 later.

Lemma 3.4. $C \triangleq \{c_j | j \in [\mathcal{B}]\}$. *For any* $\mathbf{P}$ *that* $\exists p_i \notin C$, *there exists* $\mathbf{P'}$ *(not necessarily arbitrage-free) that* $R(\mathbf{P'}) \geq R(\mathbf{P})$ *and* $\forall i \in [N], p'_i \in C$.

Proof. W.l.o.g., we assume that the distinct values $\theta_1, \ldots, \theta_\mathcal{C}$ in C are: $\theta_0 < \theta_1 < \ldots < \theta_\mathcal{C}$ where $\theta_0 = 0$ is a dummy variable. Given such a $\mathbf{P}$, we construct the corresponding $\mathbf{P'}$ as follows. $\forall p_i \in C$, we still set $p'_i = p_i$; $\forall p_i > \theta_\mathcal{C}$, we set p'_i to $\theta_\mathcal{C}$, and clearly no revenue is lost since no user can be sold at the price higher than $\theta_\mathcal{C}$; $\forall p_i \in (\theta_{k-1}, \theta_k)$, we set p'_i to θ_k, and no revenue is lost because any user who can be sold at p_i can be sold at θ_k. It is easy to verify that $R(\mathbf{P'}) \geq R(\mathbf{P})$ and $\forall i \in [N]$, $p'_i \in C$. □

Lemma 3.4 implies the fact that there exists a pricing (not necessarily arbitrage-free) with the optimal revenue and every entry in C. With similar proof (omitted), we have Corollary 3.5.

Corollary 3.5. *The optimal uniform price* $\xi^* \in \{c_j | j \in [\mathcal{B}]\}$.

Based on Lemma 3.3 and Corollary 3.5, the optimal uniform price ξ^* can be computed by Algorithm 3.1 that solves at most $\mathcal{B}$ allocation problems, which proves Theorem 3.2.

Theorem 3.6. *The optimal uniform pricing computed by Algorithm 3.1 is an* $O(\log min\{\mathcal{U}, \sum_{j=1}^{\mathcal{B}} d_j\})$ *approximation to the optimal arbitrage-free pricing.*

Algorithm 3.1 Optimal Uniform Pricing

Input: N, U and B
Output: ξ^*
1: $C \leftarrow \{c_j | j \in [\mathcal{B}]\}$
2: $\xi^* \leftarrow \text{argmax}_{\xi \in C} R(\mathbf{p}_\xi)$
3: **return** ξ^*

Proof. Let $C \triangleq \{c_j | j \in [\mathcal{B}]\}$. W.l.o.g., we assume that the distinct values $\theta_1, \ldots, \theta_{\mathcal{C}}$ in C are $\theta_1 < \ldots < \theta_{\mathcal{C}}$. Let O_k denote the maximum number of sold users when the uniform price is θ_k. It is true that $\forall k \in [\mathcal{C}]$, $R(\mathbf{p}_{\theta_k}) = \theta_k O_k$. Besides, it is true that $O_1 \geq \ldots \geq O_{\mathcal{C}}$, but the sequence of revenues $\theta_1 O_1, \ldots, \theta_{\mathcal{C}} O_{\mathcal{C}}$ is not necessarily monotone. Let $\mathbf{P}^*$ be the optimal pricing without the arbitrage-free constraint. $R(\mathbf{P}^*)$ is a trivial upper bound of the revenue of the optimal arbitrage-free pricing. To prove the theorem, we first prove Lemma 3.7 and Lemma 3.8.

Lemma 3.7. *In any feasible allocation for a pricing, the number of users sold at the price no less than θ_k is bounded by O_k, $\forall k \in [\mathcal{C}]$.*

Proof. Assume there exists $\mathbf{P}$ for which we can find a feasible allocation A such that $\exists k \in [\mathcal{C}]$, the number of sold users at prices no less than θ_k is larger than O_k. We can create an allocation A' from A as follows. For the users sold at the price no less than θ_k in A, we allocate them to the same buyers in A' at the price θ_k. For other users, we discard them. It is easy to verify that A' is feasible for the uniform price θ_k and $|A'| > O_k$, contradicting with that O_k is the maximum number of sold users for the uniform price θ_k. □

Lemma 3.8. $R(\mathbf{P}^*) \leq \theta_{\mathcal{C}} O_{\mathcal{C}} + \sum_{k=1}^{\mathcal{C}-1} \theta_k (O_k - O_{k+1})$.

Proof. Let o_k denote the number of users sold at the price no less than θ_k in the optimal allocation for $\mathbf{P}^*$. Thus, $o_k - o_{k+1}$ is the number of users sold at the exact price θ_k. From Lemma 3.4, we know that every entry of $\mathbf{P}^*$ is in C, so its revenue can be formulated as:

$$\begin{aligned} R(\mathbf{P}^*) &= \theta_{\mathcal{C}} o_{\mathcal{C}} + \sum_{k=1}^{\mathcal{C}-1} \theta_k (o_k - o_{k+1}) = \sum_{k=2}^{\mathcal{C}} o_k (\theta_k - \theta_{k-1}) + o_1 \theta_1 \\ &\leq \sum_{k=2}^{\mathcal{C}} O_k (\theta_k - \theta_{k-1}) + O_1 \theta_1 \qquad \text{(a)} \end{aligned}$$

$$= \theta_{\mathcal{C}} O_{\mathcal{C}} + \sum_{k=1}^{\mathcal{C}-1} \theta_k (O_k - O_{k+1})$$

Inequality (a) is because (1) from Lemma 3.7, $\forall l \in [N]$, $o_k \leq O_k$ and (2) $\forall k \in \{2, \ldots, \mathcal{C}\}$, $\theta_k \geq \theta_{k-1}$. □

Now we can prove Theorem 3.6. Since $R(\mathbf{p}_{\xi^*}) = \max\{\theta_1 O_1, \ldots, \theta_{\mathcal{C}} O_{\mathcal{C}}\}$, it is true that $\forall k \in [\mathcal{C}]$, $\theta_k \leq \frac{R(\mathbf{p}_{\xi^*})}{O_k}$. Replacing θ_k by $\frac{R(\mathbf{p}_{\xi^*})}{O_k}$ in Lemma 3.8, we complete the proof as follows:

$$\begin{aligned}
R(\mathbf{P}^*) &\leq R(\mathbf{p}_{\xi^*})\Big(1 + \sum_{k=1}^{\mathcal{C}-1} \frac{O_k - O_{k+1}}{O_k}\Big) \\
&\leq R(\mathbf{p}_{\xi^*})\Big(1 + \sum_{k=1}^{\mathcal{C}-1} (\frac{1}{1+O_{k+1}} + \ldots + \frac{1}{O_k})\Big) = R(\mathbf{p}_{\xi^*})\Big(1 + \sum_{i=O_{\mathcal{C}}+1}^{O_1} \frac{1}{i}\Big) \\
&\leq R(\mathbf{p}_{\xi^*})\Big(\sum_{i=1}^{O_{\mathcal{C}}} \frac{1}{i} + \sum_{i=O_{\mathcal{C}}+1}^{O_1} \frac{1}{i}\Big) \qquad\qquad = R(\mathbf{p}_{\xi^*}) \cdot H_{O_1} \\
&\leq R(\mathbf{p}_{\xi^*})(\ln O_1 + 1) \\
&\leq R(\mathbf{p}_{\xi^*})(\ln min\{\mathcal{U}, \textstyle\sum_{j=1}^{\mathcal{B}} d_j\} + 1)
\end{aligned}$$

□

Proposition 3.9. *The $O(\log min\{\mathcal{U}, \sum_{j=1}^{\mathcal{B}} d_j\})$ revenue guarantee is tight for uniform pricing solutions.*

Proof. We show one of the worst cases. $\forall i \in [N]$: (1) there are 2^i users that only satisfy q_i, and (2) exists buyer i represented as $(i, 2^i, 2^{-i})$. Thus $\mathcal{U} = \sum_{j=1}^{\mathcal{B}} d_j = 2^{N+1} - 1$. It is easy to see that the revenue of any uniform pricing is less than 2, but the revenue of the optimal arbitrage-free pricing is N (any pricing is arbitrage-free in this case). □

3.3.2 Non-uniform Pricing Algorithm

Based on uniform pricing, in this section we study arbitrage-free *non-uniform* pricing, which is more practical for real markets. We first devise a greedy algorithm to produce an arbitrage-free non-uniform pricing of which the revenue is guaranteed to be no less than the revenue

of the optimal uniform pricing. In order to speed up the algorithm for large markets, we propose an approximate algorithm to solve the allocation problem efficiently for any pricing while preserving the same performance guarantee.

Definition 3.8. *Given* $\mathbf{P}$*,* $\alpha_i \triangleq \max\{\pi(i'|i) \cdot p_{i'} | i' \in [N], i' \neq i\}$ *and* $\beta_i \triangleq \min\{\frac{p_{i'}}{\pi(i|i')} | i' \in [N], i' \neq i\}$*. We call* $[\alpha_i, \beta_i]$ *the* arbitrage-free interval[5] *of* p_i*.*

Proposition 3.10. *If* $\mathbf{P}$ *is arbitrage-free and only one entry is varied within its arbitrage-free interval, i.e. updating* p_i *into any value in* $[\alpha_i, \beta_i]$*, the resulting pricing is still arbitrage-free.*

Although Proposition 3.10 is straightforward to verify, it connects uniform pricing which is naturally arbitrage-free with arbitrage-free non-uniform pricing. Most importantly, Proposition 3.10 provides valid operations to update the pricing while keeping it arbitrage-free. It is easy to see that with proper preprocessing, α_i and β_i can be computed in $O(N)$ time. Based on Lemma 3.4 and Proposition 3.10, we have Corollary 3.11.

Corollary 3.11. $C_i(\mathbf{P}) \triangleq \{\alpha_i, \beta_i\} \cup \{c_j | j \in [\mathcal{B}], t_j = i, c_j \in [\alpha_i, \beta_i]\}$*. For any arbitrage-free pricing* $\mathbf{P}$ *that* $\exists i \in [N]$*,* $p_i \notin C_i(\mathbf{P})$*, there exists an arbitrage-free pricing* $\mathbf{P'}$ *that* $R(\mathbf{P'}) \geq R(\mathbf{P})$ *and* $\forall i \in [N]$*,* $p'_i \in C_i(\mathbf{P'})$*.*

With Proposition 3.10, the proof of Corollary 3.11 is similar to the proof of Lemma 3.4, thus omitted. Corollary 3.11 reveals the desirable property of the optimal arbitrage-free non-uniform pricing, which implies that any algorithm only needs to search over $O(\mathcal{B})$ values (because $|C_i(\mathbf{P})| \leq \mathcal{B} + 2$) other than the entire real interval $[\alpha_i, \beta_i]$ for p_i. Based on this, we propose Algorithm 3.2 that iteratively updates the optimal uniform pricing to arbitrage-free non-uniform pricing. First, we show a polynomial subroutine to solve the allocation problem for a non-uniform pricing as follows.

Lemma 3.12. *For any non-uniform pricing, the optimal allocation can be computed in polynomial time* $O(\mathcal{U}M + \mathcal{U}\mathcal{B} + \mathcal{B}\log\mathcal{B})$*.*

Proof. We model the allocation problem for non-uniform pricing as a minimum cost maximum flow problem. The network is constructed as follows. We introduce s and t as the source and

[5] *W.l.o.g. we assume that* β_i *always exists, i.e.* $\{i' | i' \in [N], i' \neq i, \pi(i|i') > 0\} \neq \emptyset$ *, otherwise, the arbitrage-free interval of* p_i *becomes* $[\alpha_i, \infty)$*.*

sink respectively. We introduce N nodes $y_1, \ldots, y_N$. Assuming all the users are indexed from 1 to $\mathcal{U}$ and $\mathbf{u}_k$ denotes the k-th user. For each $k \in [\mathcal{U}]$: (1) we introduce a node x_k and a directed edge from s to x_k with capacity 1 and cost 0; and (2) $\forall i \in [N]$, we introduce a directed edge from x_k to y_i with capacity 1 and cost 0 if $u_k^i = 1$. For each buyer j that $c_j \geq p_{t_j}$, introduce a node z_j, a directed edge from y_{t_j} to z_j with cost 0 and a directed edge from z_j to t with capacity d_j and a negative cost $-p_{t_j}$. We can verify that there is no directed cycle with negative cost.

Let $w(f) > 0$ be the absolute value of the minimum cost when the amount of the required s-t flow is f. Clearly, it is true that $w(f)$ is the maximum revenue when exactly f users are sold. Since the costs are all negative, $w(f)$ is maximized when $f = f^*$ where f^* is the amount of the maximum s-t flow.

Since only the edges from z_j to t are associated with non-zero costs, to find the augmenting path with the smallest cost is $O(|E|)$ (after sorting buyers by c_j) where $|E| \leq M + 2\mathcal{B}$. Since $f^* \leq \mathcal{U}$, the overall complexity is $O(\mathcal{U}M + \mathcal{U}\mathcal{B} + \mathcal{B}\log\mathcal{B})$. □

Now we present Algorithm 3.2. It starts with the optimal uniform pricing, then iteratively and greedily updates the pricing and finally outputs an arbitrage-free non-uniform pricing. In each iteration, the algorithm greedily updates the price of a query if the update increases the revenue. To update p_i, it picks up the new price from $C_i(\mathbf{P})$ that greedily maximizes the revenue. Let $\mathbf{P}_{-i,p}$ denote the resulting pricing when we only change the i-th entry of $\mathbf{P}$ to p.

Algorithm 3.2 Arbitrage-free Non-uniform Pricing

Input: N, U, B and $\mathbf{p}_{\xi^*}$
Output: An arbitrage-free non-uniform pricing $\mathbf{P}$

$\mathbf{P} \leftarrow \mathbf{p}_{\xi^*}$
$\forall i \in [N]$, compute α_i and β_i
repeat
  for $i \in [N]$ **do**
    $p^* = \text{argmax}_{p \in C_i(\mathbf{P})} R(\mathbf{P}_{-i,p})$
    if $R(\mathbf{P}_{-i,p^*}) > R(\mathbf{P})$ **then**
      $\mathbf{P} \leftarrow \mathbf{P}_{-i,p}$
      $\forall i' \in [N]$, re-compute $\alpha_{i'}$ and $\beta_{i'}$
    end if
  end for
until no price changes
return $\mathbf{P}$.

Proposition 3.13. *Algorithm 3.2 always converges; in each iteration of Repeat-Loop (lines* 4-10*), it solves at most* $\mathcal{B} + 2N$ *allocation problems for non-uniform pricing.*

Proof. Since the revenue of any pricing is trivially bounded by $\sum_{j=1}^{\mathcal{B}} c_j d_j$ and after each iteration except the last one the revenue always increases, the algorithm will always converge. Since $|\bigcup_{i=1}^{N} C_i(\mathbf{P})| \leq \mathcal{B} + 2N$, there will be at most $\mathcal{B} + 2N$ allocation problems of non-uniform pricing in each iteration. □

Let T be the number of iterations, the overall complexity is $O(T(\mathcal{B}+N)(\mathcal{U}M + \mathcal{U}\mathcal{B} + \mathcal{B}\log\mathcal{B}))$. We leave the theoretical analysis of T for future work, however, we will see in experiments that Algorithm 3.2 converges very quickly, i.e. $T < 4$ on average. We next analyze its performance in Proposition 3.14. Experiments show that the arbitrage-free non-uniform pricing has significantly larger revenue in practice, however, the fact that the arbitrage-free interval of any entry is dynamic in each iteration makes it difficult to analyze the theoretical improvement of the arbitrage-free non-uniform pricing.

Proposition 3.14. *Assume that Algorithm 3.2 converges after* $T \geq 1$ *iterations (of Repeat-Loop, i.e. lines 4-10). Let* $\mathbf{P}^t$ *be the pricing computed after* $t \in [T]$ *iterations. The following statements are true:*

(a) $\forall t \in [T]$, $\mathbf{P}^t$ *is arbitrage-free.*

(b) $\forall t \in [T]$, $R(\mathbf{P}^t)$ *has* $O(\log min\{\mathcal{U}, \sum_{j=1}^{\mathcal{B}} d_j\})$ *performance guarantee to the optimal arbitrage-free pricing;*

(c) If $T > 1$, $\mathbf{P}^t$ *is non-uniform,* $\forall t \in [T]$.

(d) Unless breaking the arbitrage-free constraints, changing any entry of $\mathbf{P}^T$ *alone cannot increase the revenue.*

Proof. Since the update of any price is within its arbitrage-free interval, according to Proposition 3.10, (a) is true. The algorithm starts with the optimal uniform pricing $\mathbf{p}_{\xi^*}$, and every price update increases the revenue, thus we have $\forall t \in [T]$, $R(\mathbf{P}^t) \geq R(\mathbf{p}_{\xi^*})$, which immediately implies (b). To prove (c), we only need to show that any update (in line 7) never makes

the resulting pricing $\mathbf{P}$ back to uniform. Since $R(\mathbf{P}) > R(\mathbf{p}_{\xi^*})$ and $\mathbf{p}_{\xi^*}$ is the optimal uniform pricing, $\mathbf{P}$ cannot be uniform, proving (c). According to the convergence criteria, (d) is true. □

In Algorithm 3.2, the minimum cost maximum flow solver (described in the proof for Lemma 3.12) that computes the optimal allocation for a non-uniform pricing is called frequently, i.e. up to $\mathcal{B} + 2N$ times in each iteration. Although to our best knowledge, it is faster than most of general mincost maxflow solvers for integral flows and non-unit capacity (see details in the survey [55]), it still does not scale for large inputs. By generalizing the online bipartite matching [56], we propose an approximation as Algorithm 3.3 to the allocation problem. Algorithm 3.3 greedily sells users satisfying queries with the highest price.

Algorithm 3.3 Efficient Approximate Allocation

Input: N, U, B and $\mathbf{P}$
Output: $\hat{A}$

$\forall j \in [\mathcal{B}]$, $\hat{A}_j \leftarrow \emptyset$
Re-order all the buyers such that $p_{t_1} \geq \ldots \geq p_{t_\mathcal{B}}$
for $j \in [\mathcal{B}]$ **do**
    if $c_j \geq p_{t_j}$ **then**
        while $|\hat{A}_j| < d_j$ **and** $\exists \mathbf{u} \in U$, $u^{t_j} = 1$ **do**
            $\hat{A}_j \leftarrow \hat{A}_j \cup \{\mathbf{u}\}$
            $U \leftarrow U - \{\mathbf{u}\}$
        end while
    end if
end for
return $\hat{A}$.

Proposition 3.15. *For any pricing, Algorithm 3.3 produces an allocation in polynomial time* $O(M + \mathcal{B} \log \mathcal{B})$.

Algorithm 3.3 is much faster than the mincost maxflow solver in Lemma 3.12, i.e. by at least a factor $\frac{\mathcal{U}}{\log \mathcal{B}}$. In real user markets, the number of users to sell is significantly larger than the number of buyers, i.e. $\mathcal{U} \gg \mathcal{B}$, and $\mathcal{U}$ could be billions, so the improvement is significant. We next show the performance guarantee of Algorithm 3.3 in Lemma 3.16, which will be also used for proving Theorem 3.21 later.

Lemma 3.16. *Algorithm 3.3 is a* 2*-approximation to the allocation problem for any pricing.*

Proof. Let A^* be the optimal allocation for $\mathbf{P}$. Consider some $\mathbf{u}$ who is allocated to buyer j in A^* (i.e. $\mathbf{u} \in A^*_j$) by the mincost maxflow solver, but is allocated to a different buyer j' in $\hat{A}$ (i.e. $\mathbf{u} \in \hat{A}_{j'}$) or not allocated to any buyer in $\hat{A}$. There are three cases. Case 1: if $\mathbf{u}$ is finally unallocated in $\hat{A}$, it is true that $|\hat{A}_j| = d_j \geq |A^*_j|$ because if $|\hat{A}_j|$ was less than d_j, $\mathbf{u}$ must have been allocated to buyer j. Consider that $\mathbf{u}$ is allocated to buyer j' in $\hat{A}$. Case 2: if $p_{t_j} > p_{t_{j'}}$, it must be true that $|\hat{A}_j| = d_j \geq |A^*_j|$ for the same reason as Case 1. In Case 1 and 2, the revenue contributed by $\hat{A}_j$ is no less than A^*_j. Case 3: if $p_{t_j} \leq p_{t_{j'}}$, it is true that we might lose the revenue p_{t_j} because $\mathbf{u}$ is not allocated to j. In this case, however an equal or higher revenue $p_{t_{j'}}$ is produced. This implies that the total revenue loss $R(\mathbf{P}, A^*) - R(\mathbf{P}, \hat{A})$ is bounded by the total revenue produced $R(\mathbf{P}, \hat{A})$. Therefore we prove the lemma that $2R(\mathbf{P}, \hat{A}) \geq R(\mathbf{P}, A^*) = R(\mathbf{P})$. □

We next show that the above analysis for Algorithm 3.3 is tight. Consider the following example. $N = 2$. There are 2 users, $\mathbf{u}_1 = (1, 1)$ and $\mathbf{u}_2 = (1, 0)$. There are 2 buyers $(1, 1, w + \epsilon)$ and $(2, 1, w)$. The pricing $\mathbf{P} = (w + \epsilon, w)$ where $\epsilon > 0$. The optimal allocation is $A_1 = \{\mathbf{u}_2\}$ and $A_2 = \{\mathbf{u}_1\}$, thus the total revenue is $2w + \epsilon$. However Algorithm 3.3 generates $\hat{A}_1 = \{\mathbf{u}_1\}$ and $\hat{A}_2 = \emptyset$, thus the total revenue is $w + \epsilon$ only.

We construct Algorithm 2.1 with the faster allocation in Algorithm 3.3, to compute an arbitrage-free non-uniform pricing. Algorithm 2.1 is the same with Algorithm 3.2 except that we replace the optimal allocation (mincost maxflow solver $R(\mathbf{P}_{-i,p})$ in line 5) with the approximate allocation computed by Algorithm 3.3.

Theorem 3.17. *Let T be the number of iterations that Algorithm 2.1 needs to converge. Algorithm 2.1 runs in $O(T(\mathcal{B} + N)(M + \mathcal{B} \log \mathcal{B}))$ and has all the properties (a)-(d) claimed for Algorithm 3.2 in Proposition 3.14.*

3.3.3 A Generalized Setting: Minimal Demand

Previously, we assumed that any buyer only restricts the maximum number d_j of target users he will buy. In this section, we consider a general setting where a buyer also has a minimum demand. That is, buyer j now becomes $(t_j, \underline{d}_j, d_j, c_j)$ where $\underline{d}_j$ and d_j ($\underline{d}_j \leq d_j$) are the minimum and maximum number of target users that buyer j will buy respectively. In this

setting, the *Demand Constraint* in Definition 3.2 for a feasible allocation A becomes: $|A_j| \in \{0, \underline{d}_j, \ldots, d_j\}$. This means that, for buyer j, we either allocate 0 or at least $\underline{d}_j$ users to him. Note that, the pricing problem in previous setting is indeed a special case that $\forall j$, $\underline{d}_j = 1$ of this setting.

This generalized setting is also practical in many user markets. For example, a business wants to trigger a cascade of promotion for its product in an online community. If the initial seed size is too small, the growth of cascade would be very slow or even not triggered [57, 58] at all. Therefore the business needs some guarantee of the seed size by specifying a large enough $\underline{d}_j$. Another example is that, a buyer wants to purchase the contacts of users with certain attributes for survey. If he is not able to reach enough number of target users, the survey results lack statistical significance. Therefore, he is not willing to buy any data set with size less than his minimum demand.

However, the minimum demand constraint makes the allocation problem and the pricing problem harder to solve, which can be seen in Theorem 3.18 and Corollary 3.19.

Theorem 3.18. *In the generalized setting, the allocation problem (even if the pricing is uniform at* 1*) is (1) NP-hard and (2) hard to approximate (unless P=NP) within* $\mathcal{U}^{\frac{1}{2}-\epsilon}$ *or* $\mathcal{B}^{1-\epsilon}$, $\forall \epsilon > 0$.

Proof. For (1), we reduce Maximum Independent Set (MIS) to the allocation problem. For an undirected graph $G = \{V, E\}$. MIS finds a subset $V' \subseteq V$ with the maximum cardinality that $\forall v, v' \in V'$, the edge $(v, v') \notin E$. The reduction is as follows. Given an MIS instance (V, E), we first introduce $|V|$ queries. Then we introduce $|V|^2$ users indexed from 1 to $|V|^2$, i.e. $(\mathbf{u}_1, \ldots, \mathbf{u}_{|V|^2})$, and each of these users satisfies no query. For each node $v_j \in V$: (1) we introduce a buyer j as $(j, |V|, |V|, 1)$; and (2) $\forall k \in \{|V| \cdot (j-1) + 1, \ldots, |V| \cdot j\}$, we set $u^j_k = 1$. For each edge $(v_j, v_{j'}) \in E$ of the MIS instance (assuming $j < j'$), we make the user with index $|V| \cdot (j' - 1) + j$ no longer satisfy $q_{j'}$, i.e. set $u^{j'}_{|V| \cdot (j'-1)+j} = 0$, but make the user with index $|V| \cdot (j-1) + j'$ satisfy $q_{j'}$, i.e. set $u^{j'}_{|V| \cdot (j-1)+j'} = 1$. After these operations, we can allocate $|V|$ users to either buyer j or j', but not to both of them since they have a common user with index $|V| \cdot (j-1) + j'$. It is true that (1) for any q_j where $j \in [|V|]$, there are exactly $|V|$ users satisfying it; (2) let U be the set of users who satisfy at least one query, $\mathcal{U} = |V|^2 - |E|$. It is easy to verify that the solution for the MIS instance is n if and only if the number of sold

users in the allocation problem is $n \cdot |V|$ when the pricing is uniform at 1.

For (2), we have shown that an MIS with $|V|$ nodes corresponds to the allocation problem where $\frac{|V|^2}{2} \leq \mathcal{U} \leq |V|^2$ and $\mathcal{B} = |V|$, and if the number of sold users in the allocation problem is $n \cdot |V|$, the solution of the corresponding MIS is n. Since MIS is hard to approximate within $|V|^{1-\epsilon}$, $\forall \epsilon > 0$, the allocation problem is hard to approximate within $\mathcal{U}^{\frac{1}{2}-\epsilon}$ or $\mathcal{B}^{1-\epsilon}$, $\forall \epsilon > 0$. □

Corollary 3.19. *In the generalized setting, the pricing problem is both NP-hard and hard to approximate (unless P=NP) within $\mathcal{U}^{\frac{1}{2}-\epsilon}$ or $\mathcal{B}^{1-\epsilon}$, $\forall \epsilon > 0$.*

Proof. Theorem 3.18 shows that in the generalized setting, the allocation problem is NP-hard even when the pricing is uniform. We reduce the allocation problem with uniform pricing to the pricing problem. For each instance of the allocation problem with input B', U' and the uniform price ξ, we create an instance of pricing problem with input B and U as follows. For each buyer j in B', if $c_j \geq \xi$, we create a buyer (t_j, d_j, ξ) in B. We simply set $U = U'$. It is easy to see that the optimal arbitrage-free pricing of the pricing problem is $(\xi, \ldots, \xi)$, and the optimal revenue of the pricing problem is $\xi \cdot k$ if and only if the allocation problem has the optimal allocation of size k. □

Next, we first propose an approximate algorithm for the allocation problem, and then based on this approximation, we propose another approximate algorithm for the pricing problem.

Algorithm 3.4 computes the approximate allocation in two steps, each of which produces a partial allocation. The *first partial allocation* process (lines 1-9), first re-orders all the buyers so that $p_{t_1}\underline{d}_1 \geq \ldots \geq p_{t_\mathcal{B}}\underline{d}_\mathcal{B}$. Then starting from $j = 1$, it allocates $\underline{d}_j$ target users (if enough) to buyer j, in sequence. After that, the *second partial allocation* process (lines 10-16) discards buyers who received no user in the first partial allocation. For each of the remaining buyers, it creates a dummy buyer without the minimum demand as $(t_j, d_j - \underline{d}_j, c_j)$. Then it calls Algorithm 3.3 with the remaining users, the dummy buyers and the same pricing as the input. Finally, the two partial allocations are merged as the whole approximate allocation.

Proposition 3.20. *In the generalized setting, Algorithm 3.4 computes an approximate allocation $\hat{A}$ for any pricing in polynomial time $O(M + \mathcal{B}\log\mathcal{B})$.*

Algorithm 3.4 Approximate Allocation With Minimal Demand

Input: N, U, B and $\mathbf{P}$
Output: An approximate allocation $\hat{A}$

1: Re-order $\mathcal{B}$ buyers so that $p_{t_1}\underline{d}_1 \geq \ldots \geq p_{t_\mathcal{B}}\underline{d}_\mathcal{B}$
2: Let $\hat{A}^1$ and $\hat{A}^2$ be two empty allocations
3: **for** $j \in [\mathcal{B}]$ **do**
4: $\quad U_j \leftarrow \{\mathbf{u}|u^{t_j} = 1, \mathbf{u} \in U\}$
5: $\quad$ **if** $c_j \geq p_{t_j}$ **and** $|U_j| \geq \underline{d}_j$ **then**
6: $\quad\quad \hat{A}^1_j \leftarrow$ arbitrary $\underline{d}_j$ users from U_j
7: $\quad\quad U \leftarrow U - \hat{A}^1_j$
8: $\quad$ **end if**
9: **end for**
10: $B' \leftarrow \emptyset$
11: **for** $j \in [\mathcal{B}]$ **do**
12: $\quad$ **if** $|\hat{A}^1_j| = \underline{d}_j$ **and** $d_j \neq \underline{d}_j$ **then**
13: $\quad\quad B' \leftarrow B' \cup \{(t_j, d_j - \underline{d}_j, c_j)\}$
14: $\quad$ **end if**
15: **end for**
16: $\hat{A}^2 \leftarrow$ Call Algorithm 3.3 with N, U, B' and $\mathbf{P}$ as input
17: $\forall j \in [\mathcal{B}]$, $\hat{A}_j \leftarrow \hat{A}^1_j \cup \hat{A}^2_j$
18: **return** $\hat{A}$

Theorem 3.21. *Let* $D \triangleq \max\{\underline{d}_j | j \in [\mathcal{B}]\}$. *The approximate allocation* $\hat{A}$ *produced by Algorithm 3.4 is an* $O(D)$*-approximation to the optimal allocation.*

Proof. Let A^* be the optimal allocation. W.l.o.g. we assume that $\forall j \in [\mathcal{B}]$, the number of users satisfying q_{t_j} is no less than $\underline{d}_j$, otherwise we can remove buyer j. This theorem is implied by Lemma 3.22 and Lemma 3.16. In short, Lemma 3.22 implies that for every unit of revenue generated by the first partial allocation (lines 1-9), A^* can generate at most $D + 1$ units of revenue. According to Lemma 3.16, it is true that for every unit of revenue generated by the second partial allocation (lines 10-16), A^* can generate at most 2 units of revenue. Therefore, we have $R(\mathbf{P}, \hat{A}) \geq \frac{R(\mathbf{P}, A^*)}{\max\{D+1,2\}}$, which implies the theorem. We only need to prove Lemma 3.22. □

Lemma 3.22. *If buyer* j *is allocated with* $\underline{d}_j$ *target users in the first partial allocation, the loss of revenue (compared to* A^**) is bounded by* $p_{t_j}(\underline{d}_j)^2$.

Proof. If buyer j is allocated with $\underline{d}_j$ target users in the first partial allocation, the revenue $p_{t_j}\underline{d}_j$ is produced. It is true that the removal of $\underline{d}_j$ users satisfying q_{t_j} will prevent at most

$\underline{d}_j$ of buyers (with index larger than j) from being allocated with any target user, because the numbers of available target users become smaller than their minimum demands. Let $n \in [\underline{d}_j]$ be the number of buyers who cannot be allocated with users due to allocating $\underline{d}_j$ users to buyer j. Clearly, there is no loss if $n = 0$. W.lo.g., we assume that these n buyers are indexed from $j+1$ to $j+n$. It is true that $\forall j' \in \{j+1, \ldots, j+n\}$, the largest possible revenue loss caused by losing buyer j' is bounded by $p_{t_{j'}}(\underline{d}_{j'} - 1 + k_{j'})$. $k_{j'}$ is the number of such users: (1) they satisfy both q_{t_j} and $q_{t_{j'}}$; and (2) if we did not allocate any of the $k_{j'}$ users to buyer j, buyer j' could have been allocated with $\underline{d}_{j'}$ users – but we actually have allocated all the $k_{j'}$ users to buyer j. Thus the total loss L_j of allocating $\underline{d}_j$ users to buyer j is bounded by $\sum_{j'=j+1}^{j+n} p_{t_{j'}}(\underline{d}_{j'} - 1 + k_{j'})$. It is obviously true that $\sum_{j'=j+1}^{j+n} k_{j'} \leq \underline{d}_j$ and $\forall j' \in \{j+1, .., j+n\}$, $k_{j'} \geq 1$. Therefore, $\forall n \in [\underline{d}_j]$, we have:

$$
\begin{aligned}
L_j &\leq \sum_{j'=j+1}^{j+n} p_{t_{j'}} \underline{d}_{j'} + \sum_{j'=j+1}^{j+n} p_{t_{j'}}(k_{j'} - 1) \\
&\leq n \max_{j' \in \{j+1, \ldots, j+n\}} \{p_{t_{j'}} \underline{d}_{j'}\} + \sum_{j'=j+1}^{j+n} p_{t_{j'}}(k_{j'} - 1) \\
&\leq n p_{t_j} \underline{d}_j + \sum_{j'=j+1}^{j+n} p_{t_{j'}}(k_{j'} - 1) \\
&\leq n p_{t_j} \underline{d}_j + \max_{j' \in \{j+1, \ldots, j+n\}} \{p_{t_{j'}}\} \sum_{j'=j+1}^{j+n} (k_{j'} - 1) \\
&\leq n p_{t_j} \underline{d}_j + \max_{j' \in \{j+1, \ldots, j+n\}} \{p_{t_{j'}}\} (\underline{d}_j - n) \\
&\leq n p_{t_j} \underline{d}_j + p_{t_j} \underline{d}_j (\underline{d}_j - n) \\
&= p_{t_j} (\underline{d}_j)^2
\end{aligned}
$$

□

We next show that the $O(D)$-approximation is tight for Algorithm 3.4. Consider the following example. There are $d + 1$ queries and d^2 users. $\forall i \in [d]$, all the $d - 1$ users indexed between $(d-1) \cdot (i-1) + 1$ and $(d-1) \cdot i$ (both inclusive) only satisfy q_i. The last d users, indexed between $d^2 - d + 1$ and d^2, satisfy all the queries. There are $d + 1$ buyers, and buyer j is (j, d, d, ∞). Let the pricing be: $\forall i \in [d]$, $p_i = 1$ and $p_{d+1} = 1 + \epsilon$. Algorithm 3.4 will

only allocate d users to buyer $d+1$ while all the other buyers are allocated with 0 user, thus the revenue is $d \cdot (1+\epsilon)$. However, the optimal solution that $\forall i \in [d]$, allocates buyer i with d users has revenue d^2.

With Theorem 3.21, we propose Algorithm 5 to produce a uniform pricing in this setting. Let $\hat{A}_\xi$ be the approximate allocation output by Algorithm 3.3 when the input uniform price is ξ. Similar to Algorithm 3.1, Algorithm 5 outputs the uniform price $\hat{\xi} = \text{argmax}_{\xi \in C} R(\mathbf{p}_\xi, \hat{A}_\xi)$ where $C = \{c_j | j \in [\mathcal{B}]\}$. We analyze it as follows.

Theorem 3.23. *Algorithm 5 runs in $O(\mathcal{B}M + \mathcal{B}^2 \log \mathcal{B})$ to compute a uniform pricing which is an $O(D \log \min\{\mathcal{U}, \sum_{j=1}^{\mathcal{B}} d_j\})$ approximation to the optimal arbitrage-free pricing in the generalized setting.*

Proof. According to Proposition 3.20, it is easy to see that the time complexity of Algorithm 5 is $O(\mathcal{B}M + \mathcal{B}^2 \log \mathcal{B})$ since $|C| \leq \mathcal{B}$. Similarly with the proof for Theorem 3.6, we can derive the $O(D \log \min\{\mathcal{U}, \sum_{j=1}^{\mathcal{B}} d_j\})$ guarantee for Algorithm 5 based on Theorem 3.21, since $\hat{A}_\xi$ is an $O(D)$ approximation to the allocation problem in the generalized setting. □

Compared with the previous setting where buyers have no minimum demand, we do not find a polynomial time solution to compute the optimal uniform pricing in this setting. Because the allocation problem is NP-hard, the guarantee of the uniform pricing drops. However, as Algorithm 5 does not call any maxflow optimization, its time complexity is lower than that of Algorithm 3.1. We leave the arbitrage-free non-uniform pricing in this setting as future work.

3.4 Experiments

In this section, we use synthetic data to evaluate Algorithms 3.1, 3.2, 2.1 and 3.3. To randomly generate instances of the pricing problem, we first set the values of $\mathcal{U}$, $\mathcal{B}$, N, m and c where m is the maximal number of queries that any user can satisfy and c is the upper bound of buyers' maximum costs. For each buyer $j \in [\mathcal{B}]$, t_j, d_j and c_j are independently and uniformly sampled from the integer sets $[N]$, $[\lfloor \frac{4\mathcal{U}}{\mathcal{B}} \rfloor]$ and $[c]$ respectively. For each user $\mathbf{u}_i \in U$, m_i is independently and uniformly sampled from the integer set $[m]$, and we randomly make her satisfy m_i distinct queries. We generate instances of the pricing problem of three different sizes: *small* size where

$(\mathcal{U}, \mathcal{B}, N, m, c) = (100, 20, 10, 4, 5)$, *medium* size where $(\mathcal{U}, \mathcal{B}, N, m, c) = (1000, 100, 50, 20, 1000)$ and *large* size where $(\mathcal{U}, \mathcal{B}, N, m, c) = (10^6, 1000, 500, 200, 1000)$.

3.4.1 Optimal Arbitrage-free Pricing

In this part, we compare the optimal arbitrage-free pricing, the optimal uniform pricing computed by Algorithm 3.1 and the arbitrage-free non-uniform pricing by Algorithm 3.2. Since we haven't found any (even pseudo) polynomial algorithm to compute the optimal arbitrage-free pricing, we use an exponential algorithm with grid search and backtracking to compute the numerically optimal arbitrage-free pricing. Thus the experiment can be only conducted on instances of small size, and we randomly generate 1000 such instances.

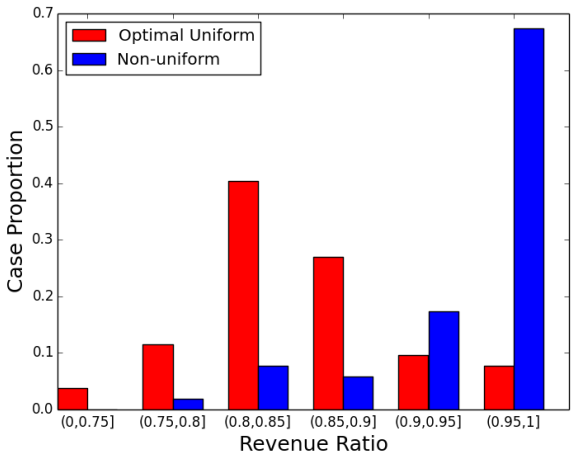


Figure 3.1: Uniform pricing vs. Non-uniform pricing (compared with the "optimal" baseline)

Let $\mathbf{P}^*$ be the optimal arbitrage-free pricing. For each test case, we record two revenue ratios $r_1 = \frac{R(\mathbf{p}_{\xi^*})}{R(\mathbf{P}^*)}$ and $r_2 = \frac{R(\mathbf{P})}{R(\mathbf{P}^*)}$ where $\mathbf{p}_{\xi^*}$ is the optimal uniform pricing computed by Algorithm 3.1 and $\mathbf{P}$ is the arbitrage-free non-uniform pricing by Algorithm 3.2. The results are shown in Fig 3.1 where the x-axis denotes the interval of revenue ratios and y-axis denotes the proportion of the test cases of which the revenue ratios fall into the interval denoted by x. The red bar is for r_1 and the blue bar is for r_2. Among all the cases, $r_1 \in [0.732, 0.983]$ and $r_2 \in [0.796, 0.997]$. The mean values of r_1 and r_2 are 0.849 and 0.948 respectively. We observe that the actual revenue ratios of the optimal arbitrage-free uniform pricing and the non-uniform pricing are both significantly larger than the theoretical guarantee which is

$1/(1+\log\min\{\mathcal{U}, \sum_{j=1}^{\mathcal{B}} d_j\}) \approx 0.18$. In particular, the arbitrage-free non-uniform pricing is remarkably close to the optimal arbitrage-free pricing.

3.4.2 Approximate Allocation

In this part, we compare the approximation in Algorithm 3.3 with the optimal one (i.e. the min-cost maxflow solver in Lemma 3.12) for the allocation problem. Note that in line 6, Algorithm 3.3 selects any user $\mathbf{u}$ satisfying q_{t_j}. For experiments, we use a heuristic to select such a user: among all the users satisfying q_{t_j}, we select the one with the least number of other queries that she satisfies. With proper preprocessing, this heuristic does not increase the asymptotic time complexity of Algorithm 3.3.

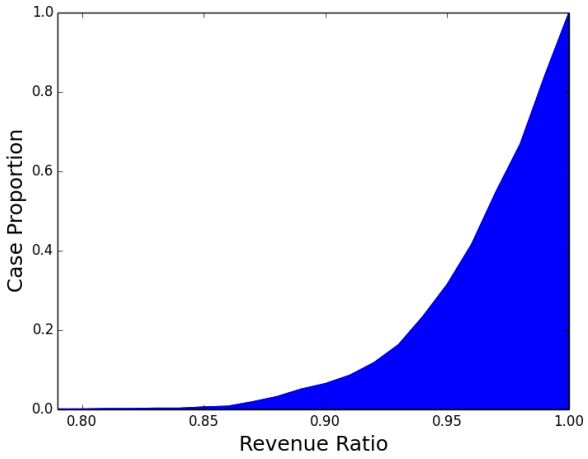


Figure 3.2: Compare the approximate allocation (by Algorithm 3.3) with the optimal allocation (by mincost maxflow)

The experiments are conducted on the instances of medium size. We first randomly generate 1000 cases of medium size and for each case we randomly generate a pricing vector $\mathbf{P}$ where $\forall i \in [N]$, p_i is independently and uniformly sampled from $[c]$. We define the revenue ratio as $\frac{R(\mathbf{P},\hat{A})}{R(\mathbf{P},A^*)}$ where $\hat{A}$ is the approximate allocation computed by Algorithm 3.3 and A^* is the optimal allocation computed by the mincost maxflow solver in Lemma 3.12. We plot the results in Fig 3.2 where x-axis denotes the ratio and y-axis denotes the accumulated proportion of cases of which the revenue ratios are less than or equal to x. Among all the 1000 cases, the least revenue ratio is 0.79; the ratios of 76.6% cases are at least 0.95; 16% cases reach the optimum, i.e. with ratio as 1. The average ratio is 0.968, much larger than the theoretical

guarantee 0.5.

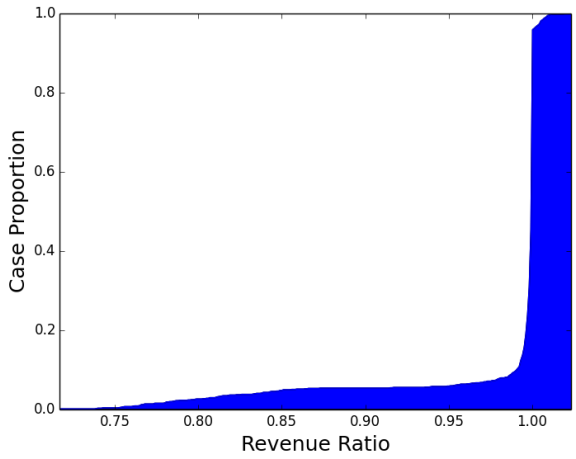


Figure 3.3: Compare the non-uniform pricings by Algorithm 3.2 and Algorithm 2.1

Next, we randomly generate 1000 cases of medium size, for each of which, we compute its arbitrage-free non-uniform pricing. Let $\mathbf{P}$ and $\mathbf{P}'$ be the pricing output by Algorithm 3.2 and 2.1 respectively. In order to compare $R(\mathbf{P})$ and $R(\mathbf{P}')$, we still call the mincost maxflow solver after $\mathbf{P}$ and $\mathbf{P}'$ are produced. We plot the results in Fig 3.3 where x-axis denotes the revenue ratio $\frac{R(\mathbf{P}')}{R(\mathbf{P})}$ and y-axis denotes the accumulated proportion of cases of which the revenue ratios are less than or equal to x. With the mean value 0.988, the ratios are in $[0.717, 1.023]$. Among the 1000 cases, we find that in 94.1% cases, the ratio is at least 0.95 and in 4.1% cases, the ratio is larger than 1 (because Algorithm 3.2 does not guarantee global optimum). Notably, Algorithm 2.1 is faster than Algorithm 3.2 by at least a factor $\frac{\mathcal{U}}{\log \mathcal{B}}$.

3.4.3 Uniform and Non-uniform Pricing

In this part, we first measure the convergence of Algorithm 2.1, and then compare the arbitrage-free non-uniform pricing by Algorithm 2.1 with the optimal uniform pricing by Algorithm 3.1. We randomly generate two datasets, 1000 cases of large size and 5000 cases of medium size.

Convergence. We observe that Algorithm 2.1 converges very quickly, as shown in Fig 3.4. Among all the all the 1000 instances of large size, it converges after 3.77 iterations on average, and in the worst case, 14 iterations. Among all the 5000 instances of medium size, it converges after 2.96 iterations on average, and in the worst case, 7 iterations.

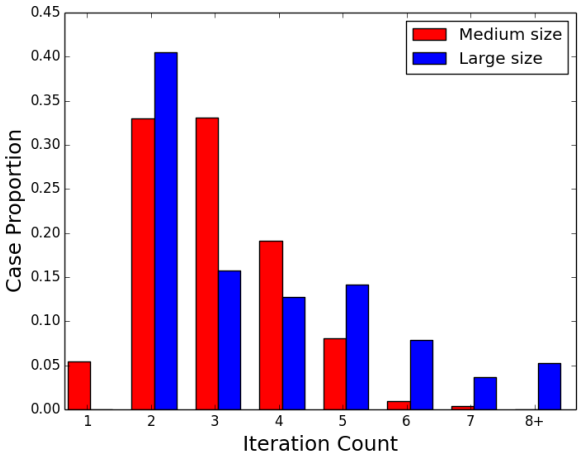


Figure 3.4: Convergence for non-uniform pricing (Algorithm 2.1)

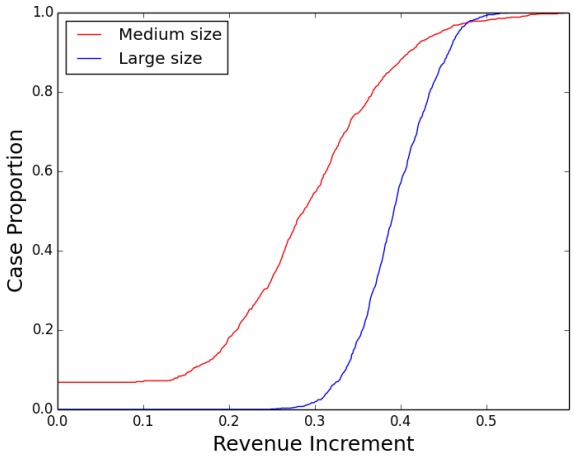


Figure 3.5: Compare uniform pricing with non-uniform pricing

Let R be the (approximate) revenue of the arbitrage-free non-uniform pricing output by Algorithm 2.1, and the relative revenue increment is calculated as $\frac{R}{R(\mathbf{p}_{\xi^*})} - 1$. We observe that R is significantly larger than $R(\mathbf{p}_{\xi^*})$, shown in Fig 3.5. For each curve, x-axis is the relative revenue increment and y-axis is the accumulated case proportion of which the relative revenue increment is less than or equal to x. For all the 1000 instances of large size, the increment is in $[0.25, 0.43]$, with 0.394 as the mean and 0.046 as the standard deviation. For all the 5000 instances of medium size, the increment is in $[0, 0.596]$, with 0.281 as the mean and 0.112 as the standard deviation. We conclude that the arbitrage-free non-uniform pricing significantly outperforms the optimal uniform pricing, typically in large markets.

3.5 Discussion

In the previous sections, we constrain that each user can be sold at most once. In this part, we discuss when the constraint becomes that a user $\mathbf{u}_k$ where $k \in [\mathcal{U}]$ can be sold up to w_k different buyers. Note that, we require that any user cannot be sold to one buyer more than once because we assume that the buyers need to buy distinct users. Moreover, the definition of arbitrage and arbitrage-free pricing remain the same. Next, we brief the differences of algorithms and analyses.

The optimal uniform pricing becomes an $O(\log \min\{\sum_{k=1}^{\mathcal{U}} w_k, \sum_{j=1}^{\mathcal{B}} d_j\})$ approximation, which was $O(\log \min\{\mathcal{U}, \sum_{j=1}^{\mathcal{B}} d_j\})$ in Section 3.3.1. The algorithm to compute the optimal uniform pricing is the same as Algorithm 3.1 except two differences in solving the allocation problem (in Lemma 3.3). First, when we construct the network flow problem (in the proof of Lemma 3.3), the step that "For each user $\mathbf{u}_k \in U$: (1) we introduce a node x_k and a directed edge from s to x_k with capacity 1; and (2) ..." becomes "For each user $\mathbf{u}_k \in U$: (1) we introduce a node x_k and a directed edge from s to x_k with capacity w_k; and (2) ...". Second, to compute the maximum flow, we cannot use Ford-Fulkerson algorithm, of which the time complexity depends on the amount of flow. Instead, we use the max flow solver in [59] with time complexity $O(|V||E|)$. Thus, we now need $O((\mathcal{U} + N + \mathcal{B})(M + \mathcal{B}))$ time to solve the allocation problem for uniform pricing, which was $O(\mathcal{U}(M + \mathcal{B}))$ previously in Lemma 3.3. The modification for non-uniform pricing algorithm is similar, thus omitted.

3.6 Related Work

Revenue maximizing and envy-free pricing. Pricing is a well-studied area in Economics. In particular, the envy-free pricing [29, 60, 61] in Walrasian Equilibrium [54] is relevant to our work. In recent years, envy-free pricing is studied in various settings [62, 63, 64, 65, 66, 67]. [29] first addresses the computational issue of envy-free pricing. They show that the problem is NP-hard even for the two special cases where the buyers are either unit-demand or single-minded. For the latter case, the uniform pricing provides a logarithmic approximation in terms of the number of buyers and the number of items. In the unlimited supply setting, [68] uses a randomized single price to achieve expected revenue within a logarithmic factor of the total

social welfare for buyers with general valuation functions. [69] proves that the envy-free pricing problem in a graph where items are edges is NP-hard, and provides a better approximation algorithm than [29] for sparse instances. [70] claims some equivalency between envy-free pricing and a pricing free of determinancy-arbitrage in data markets. However, envy-free pricing is essentially different from the arbitrage-free pricing in the user markets we considered, because not every (revenue-maximizing) envy-free pricing is a pricing free of version-arbitrage, and it is unclear how to convert an envy-free pricing to an arbitrage-free pricing while maximizing the revenue.

Arbitrage-free pricing in data markets. The issue of query-based price arbitrage has gained much attention in data markets [51, 71, 72, 73, 74, 70]. [51] introduces the notion *arbitrage-freeness* to query pricing. They define that a query q is *determined* by a set Q of queries on database instance $\mathcal{D}$ if the answer of q can be inferred from the answer of Q on $\mathcal{D}$. Determinancy-arbitrage occurs if q is determined by Q and the price of q is less than the total price of all the queries in Q. However, determinancy-arbitrage is foundamentally different from version-arbitrage, which can be easily seen in Example 3.1 and Definition 3.6.

Price arbitrage in OSN advertising markets. Price arbitrage has been discovered and exploited in OSN advertising markets. [52] first exploits price arbitrage in topic targeting, and [53] proposes a more complex arbitrage strategy through path combination. [15] shows through data analysis that arbitrage exists in both Facebook and LinkedIn ad markets, and proposes strategies to exploit arbitrage to benefit advertisers. Our work is partially motivated by these arbitrage strategies, and our arbitrage-free pricing can make these strategies infeasible in online advertising markets.

3.7 Conclusion

In this chapter, we addressed the pricing problem, in particular, revenue maximizing arbitrage-free pricing in user-based markets. We presented a variety of efficient algorithms for arbitrage-free pricing with provable approximation guarantees on their revenue, and hardness results for certain variations. We believe that there is a real need to study mechanisms for allocation and pricing of users based on multiple attributes as much of online user-based markets rely on such

systems.

Chapter 4

Stable Pricing in Online Labor Markets

4.1 Introduction

Online labor markets contribute significantly to Internet Economy. For example, in Amazon Mechanical Turk (AMT)[1] and CrowdFlower[2], agents are employed for human intelligence tasks. Workers in TaskRabbit[3] perform daily-life tasks, e.g. moving, cleaning. In Upwork[4], specialists are hired for more professional tasks, e.g. software development, translation.

For such markets, it is essential to price transactions and match tasks with workers suitably. In AMT, tasks are priced by owners. In Upwork, besides pricing the task, an employer needs to interview applicants which is burdensome. Recently, *automatic* pricing and matching is adopted in some online labor markets. For example, TaskRabbit provides an option called *Quick Assign*[5]. With this option, once the employer finishes defining the task by specifying requirements (e.g. in Fig 4.1), date and location, the system automatically charges a price [6] and allocates a qualified worker to the task.

We study this automatic pricing and allocation problem. The natural goal is to optimize the revenue, but we require additional properties. In online labor markets, no worker or task should be treated unfairly by the pricing and allocation. We formulate this as a form of *stability* (similar to [75]), which is illustrated as follows:

---

[1] `https://www.mturk.com`

[2] `https://www.crowdflower.com`

[3] `https://www.taskrabbit.com`

[4] `https://www.upwork.com`

[5] `https://support.taskrabbit.com/hc/en-us/articles/205313120-What-is-Quick-Assign-`

[6] To our best knowledge, the price only depends on the requirements and other facts like time and location. In other words, the task owner needs to pay the given price for any qualified worker. For example, $33/hr for any worker with a car, and $71/hr for any worker with a car.

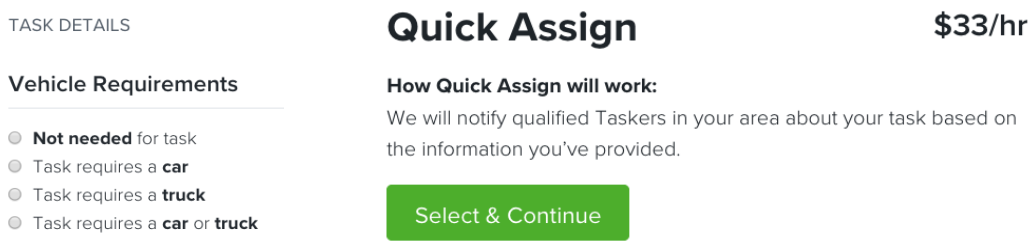


Figure 4.1: Specifying the requirement for a moving task

Example 4.1. *We assume that any worker prefers a task with a higher price* [7]*. There are two workers,* w_1 *and* w_2*, and two tasks,* t_1 *and* t_2*. Only* w_1 *is qualified for* t_1*.* w_1 *and* w_2 *are both qualified for* t_2*, but the owner of* t_2 *prefers* w_1 *to* w_2*. Regardless of the pricing, the matching that* $\{(w_1, t_1), (w_2, t_2)\}$ *maximizes the revenue. However, this matching is unfair to* w_1 *when* t_1 *is priced less than* t_2*: if we announce the pricing and allow workers and tasks to match independently at their will, the matching will be* $\{(w_1,t_2)\}$*.*

In this chapter, we solve the *stable pricing problem*. Our contributions are:

- We design a truthful, stable mechanism with randomized *uniform* pricing (SMUP), and prove that it has $(1 + \log h)$ factor approximation guarantee on revenue in expectation, where h is the maximum price for a task. In uniform pricing, all the tasks are priced equally. Given any uniform pricing, we show that one can compute the optimal allocation in polynomial time while preserving the truthfulness.

- We design a truthful stable mechanism with randomized *non-uniform* pricing (SMNP). Given an arbitrary non-uniform pricing, the allocation problem is NP-hard. We propose a $\frac{3}{2}$-approximation for the allocation problem by generalizing the (unweighted) maximum stable matching in [76, 77]. Therefore we show that SMUP is $(3 + 3\log h)$ factor appoximation of the optimal revenue, which is slightly worse than SMUP above, but our experiments show that they have similar performance in practice, and SMNP is more robust.

- We consider *online* scenario where tasks arrive over time, and present a truthful online

[7] *For example, the price of a task is the hourly payment from the task owner, and a worker's hourly wage is a fixed fraction of the price of the task assigned to her.*

stable mechanism with randomized uniform pricing. Combined with a greedy matching strategy, we show that this mechanism is $(2 + 2\log h)$ factor approximation for the optimal revenue.

Our results are obtained by using combinatorial optimization techniques suitably with random choice of prices. The rest of the chapter is organized as follows. We first formulate the pricing problem. We then analyze the stable mechanism with uniform pricing and with non-uniform pricing respectively. Furthermore, we analyze the online stable mechanism. Finally we compare different mechanisms by experiments, followed by the related work and conclusion.

4.2 Preliminary

4.2.1 Problem Formulation

$[k]$ denotes the integer set $\{1, \ldots, k\}$. There are $\{1, \ldots, S\}$ skills, a set W of workers and a set T of tasks. We define $\mathcal{W} \triangleq |W|$ and $\mathcal{T} \triangleq |T|$. A worker is denoted by a vector $\mathbf{w} = (w_1, \ldots, w_S)$ where $w_s \in \mathbb{N}$ is her level of skill $s \in [S]$. For example, when $S = 3$, $\mathbf{w} = (4, 5, 0)$ denotes that this worker has skill 1 at level 4 and skill 2 at level 5. We use superscript to denote the index of a worker, e.g. $\mathbf{w}^i$ is the i-th worker where $i \in [\mathcal{W}]$. Task j is denoted by a triplet (s_j, l_j, c_j) where $s_j \in [S]$ is the required skill, $l_j \in \mathbb{Z}_+$ is the minimum level of the required skill, and $c_j \in \mathbb{R}_+$ is the maximum cost that the task is willing to pay. For example, (1, 2, \$5) denotes a task that is willing to pay at most \$5 for hiring a worker whose level of skill 1 is at least 2. We assume that 0 is the lowest level and a larger integer denotes a higher level, so worker $\mathbf{w}$ is said to be *qualified* for task j if $w_{s_j} \geq l_j$.

An allocation (or matching) A is the set of worker-task pairs. For any $(\mathbf{w}, j) \in W \times T$, $\mathbf{w}$ and j[8] are said to be matched with each other if $(\mathbf{w}, j) \in A$. $|A|$ denotes the matching size.

Definition 4.2. *A is* feasible *if (1) $\forall(\mathbf{w}, j) \in A$, $\mathbf{w}$ is qualified for j, i.e. $w_{s_j} \geq l_j$ and (2) any task (or worker) can be matched up to one worker (or task).*

$\mathbf{P}$ is the pricing function where $p(s, l) \in \mathbb{R}_+$ is the price of **any** task that requires skill s with minimum level l. Given s, it is natural to require the pricing to be non-decreasing over

[8] When the context is clear, we use j to denote task j.

levels, i.e. $p(s,l) \geq p(s,l'), \forall l \geq l'$. Note that, the payment of a task only depends on what the task reports, i.e. task j pays $p(s_j, l_j)$ if it is matched with **any** qualified worker, otherwise 0. When the context is clear, we use p_j to denote $p(s_j, l_j)$. By default, the pricing-allocation pair ($\mathbf{P}$, A) is required to be *feasible* that A is feasible and $\forall(\mathbf{w}, j) \in A$, $p_j \leq c_j$.

We assume that any worker prefers a task with the higher price, and any task prefers a qualified worker with the higher level of the required skill. Based on this assumption, we can formulate *unfairness* as the existence of *blocking pairs*.

Definition 4.3 (Blocking Pair)**.** *Given W, T, $\mathbf{P}$ and A, a pair $(\mathbf{w}, j) \in W \times T$ is* blocking *if both are true:*

- $\mathbf{w}$ *is unmatched or matched with j' that $p_{j'} < p_j$*
- *j is unmatched or matched with $\mathbf{w}'$ that $w'_{s_j} < w_{s_j}$.*

In other words, ($\mathbf{w}$, j) is blocking if $\mathbf{w}$ and j both prefer each other. We next define stability in our market.

Definition 4.4 (Stability)**.** *($\mathbf{P}$, A) is* stable[9] *w.r.t. W and T if $\forall(\mathbf{w}, j) \in W \times T$, $(\mathbf{w}, j)$ is not blocking.*

Let $R(\mathbf{P}, A) \triangleq \sum_{(\mathbf{w},j)\in A} p_j$ be the revenue. We next define the pricing problem and allocation problem.

Definition 4.5 (Stable Pricing Problem)**.** *Given W and T, to compute the revenue-maximizing pricing-allocation pair ($\mathbf{P}$, A) subject to that ($\mathbf{P}$, A) is stable.*

Definition 4.6 (Allocation Problem)**.** *Given W, T and $\mathbf{P}$, to compute the revenue-maximizing allocation A subject to that ($\mathbf{P}$, A) is stable.*

4.2.2 Stable Pricing Mechanism

We focus on the stable pricing problem, and in particular, we consider mechanism design approaches. A mechanism consists of a pricing and an allocation, but will have additional truthfulness as we describe below.

[9] *The stability in this thesis is equivalent to the weak stability in [78].*

Definition 4.7 (Stable Mechanism)**.** *Let* $\Phi(\mathbf{P}, A)$ *be a mechanism with* $\mathbf{P}$ *and* A*.* $\Phi(\mathbf{P}, A)$ *is a* stable mechanism *if* $(\mathbf{P}, A)$ *is stable.*

A randomized (offline or online) mechanism $\Phi(\mathbf{P}, A)$ is said to have α-*guarantee on revenue* if $\alpha \cdot \mathbf{E}[R(\mathbf{P}, A)] \geq R(\mathbf{P}^*, A^*)$. $(\mathbf{P}^*, A^*)$ is the optimal solution to the offline stable pricing problem in Definition 4.5 where the true information of all the workers and tasks is assumed to be known by the market. In mechanism design, any task may not necessarily report its true information to the market if misreporting can increase its utility. Therefore, we consider *truthful* mechanisms, in which the dominant strategy of any task is to report all its private parameters truthfully[10] regardless whether other tasks report their parameters truthfully [54]. A mechanism is *individual-rational* (IR) if the utility of every task is non-negative. Next, we define the utility of a task.

Let $\phi = (s_j, l_j, c_j)$ be the parameters reported by j, while $\underline{\phi} = (\underline{s}_j, \underline{l}_j, v_j,)$ be the private parameters owned by j, where v_j is the true valuation if j is matched with any qualified worker $\mathbf{w}$ that $w_{\underline{s}_j} \geq \underline{l}_j$. Given A and $\mathbf{P}$, we consider the following utility function u_j for j:

$$u_j = \begin{cases} 0 & j \text{ is unmatched} \\ v_j \cdot \mathbb{1}(w_{\underline{s}_j} \geq \underline{l}_j) - p(s_j, l_j) & j \text{ is matched with } \mathbf{w} \end{cases}$$

$\mathbb{1}(w_{\underline{s}_j} \geq \underline{l}_j)$ is the indicator function returning 1 if $w_{\underline{s}_j} \geq \underline{l}_j$, otherwise 0.

4.3 Theoretical Results

In this section, we design two offline truthful stable mechanisms where the information of all the workers and tasks are known to the market in advance. Then we design an online truthful stable mechanisms where the tasks come online and only the information of all the users are known to the market in advance.

[10] We do not consider truthfulness from the side of workers, i.e. whether workers report their skill/levels truthfully, because in many online labor markets, workers' proficiency in skills is publicly accessible, e.g. in term of ratings, reviews, certificates or experience.

4.3.1 Existence of Stable Pricing Mechanism

To warm up, we show the existence of feasible stable mechanisms. We first define *monotone allocation*.

Definition 4.8 (Monotone Allocation)**.** *A is* monotone *if: $\nexists(\mathbf{w}', \mathbf{w}, j) \in W \times W \times T$ that $(\mathbf{w}, j) \in A$, $\mathbf{w}'$ is unmatched and $w'_{s_j} > w_{s_j}$. A monotone allocation is* maximal *if $\nexists(\mathbf{w}, j) \in W \times T$ that $\mathbf{w}$ and j could be matched but not.*

A pricing is called *uniform* if all its entries are equal, otherwise *non-uniform*. When the context is clear, we use $\mathbf{p}$ to denote a uniform pricing.

Proposition 4.1. *If A is not monotone, $\nexists\mathbf{P}$ that $(\mathbf{P}, A)$ is stable. For any uniform pricing $\mathbf{p}$, $(\mathbf{p}, A)$ is stable if A is a maximal monotone allocation.*

Proof. The first claim is obvious, so we only prove the second one. Given a uniform pricing $\mathbf{p}$ and a maximal monotone allocation A, suppose $(\mathbf{w}, j)$ is a blocking pair. First, it is impossible that $\mathbf{w}$ is matched because of the uniform pricing. $\mathbf{w}$ and j cannot be both unmatched because A is maximal. The remaining case is that $\mathbf{w}$ is unmatched but j is matched with some $\mathbf{w}'$ that $w'_{s_j} < w_{s_j}$, contradicting to Definition 4.8. □

Since it is straightforward to construct a maximal monotone allocation and generate a uniform pricing, we omit the proof for Corollary 4.2.

Corollary 4.2. *A stable mechanism always exists.*

4.3.2 Stable Mechanism with Uniform Pricing

In this part, we present a truthful stable mechanism with $(1+\log h)$-guarantee on revenue based on randomized uniform pricing where h is the maximum price we can set for a task. We first solve the allocation problem for uniform pricing.

Theorem 4.3. *Given a uniform pricing, the allocation problem can be solved in polynomial time $O(S^2\mathcal{W} + \min\{\mathcal{W}, \mathcal{T}\}(S\mathcal{W} + \mathcal{T}))$.*

To prove Theorem 4.3, we first prove Lemma 4.4 and 4.5. Lemma 4.4 shows that for any uniform pricing $\mathbf{p}$, we can compute its revenue-maximizing allocation $\tilde{A}^*_{\mathbf{p}}$ (but $(\mathbf{p}, \tilde{A}^*_{\mathbf{p}})$ is not

necessarily stable) in polynomial time. Lemma 4.5 shows that given $\tilde{A}^*_{\mathbf{p}}$, we can construct the optimal allocation $A^*_{\mathbf{p}}$ in polynomial time that $(\mathbf{p}, A^*_{\mathbf{p}})$ is stable and $|\tilde{A}^*_{\mathbf{p}}| = |A^*_{\mathbf{p}}|$.

Lemma 4.4. *Given uniform pricing* $\mathbf{p}$*, the revenue maximizing allocation* $\tilde{A}^*_{\mathbf{p}}$ *(but* $(\mathbf{p}, \tilde{A}^*_{\mathbf{p}})$ *is not necessarily stable) can be computed in polynomial time* $O(\min\{\mathcal{W}, \mathcal{T}\}(S\mathcal{W} + \mathcal{T}))$.

Proof. Let p be the uniform price. After removing any task j that $c_j < p$, we model this problem as a maxflow instance. Besides the source and sink, we create three types of nodes, corresponding to workers (x-type), skills/levels (y-type) and tasks (z-type) respectively. For each remaining task j, we introduce two nodes z_j and y_{s_j,l_j} (do not introduce y_{s_j,l_j} again if it exists); then introduce two directed edges with capacity 1, from y_{s_j,l_j} to z_j and from z_j to the sink. For each pair $y_{s,l}$ and $y_{s,l'}$, if $l' < l$ and $\nexists\, y_{s,\hat{l}}$ such that $l' < \hat{l} < l$, introduce a directed edge from $y_{s,l}$ to $y_{s,l'}$ with infinite capacity. This means that a skill at level l can be used as the same skill at a lower level l'. Finally, for each worker $\mathbf{w}^i$, introduce a node x_i and a directed edge with capacity 1 from the source to x_i. For every skill $w^i_s > 0$ that she has, introduce a directed edge with capacity 1 from x_i to $y_{s,l}$ if all the three conditions are true: (1) $l \leq w^i_s$, (2) $y_{s,l}$ exists and (3) any node $y_{s,l'}$ such that $l < l' \leq w^i_s$ does not exist. We remove any x_i from the network if she has no edge incident to y-type nodes.

Let f^* denote the maximum flow from the source to sink. It is easy to verify that $|\tilde{A}^*_{\mathbf{p}}| = f^*$. Thus the optimal revenue is $f^* \cdot p$. The number of nodes corresponding to workers is $O(\mathcal{W})$, to skills/levels is $O(\mathcal{T})$, and to tasks is $O(\mathcal{T})$. Therefore, the total number of nodes is $O(\mathcal{W} + \mathcal{T})$. The number of edges from x-type nodes to y-type nodes is $O(S\mathcal{W})$, among y-type nodes is $O(\mathcal{T})$, and from y-type nodes to z-type nodes is $O(\mathcal{T})$. Thus, the total number of edges is $O(S\mathcal{W} + \mathcal{T})$. Since Ford-Fulkerson algorithm is $O(|E| \cdot f^*)$ and we have $f^* = O(\min\{\mathcal{W}, \mathcal{T}\})$, the time complexity to run Ford-Fulkerson on this network is $O(\min\{\mathcal{W}, \mathcal{T}\}(S\mathcal{W} + \mathcal{T}))$. □

If we directly compute $\tilde{A}^*_{\mathbf{p}}$ as maximum bipartite matching, i.e. without the y-type nodes, the time complexity increases to $O(\mathcal{W}^2\mathcal{T} + \mathcal{W}\mathcal{T}^2)$ because the total number of nodes is still $O(\mathcal{W} + \mathcal{T})$ while the total number of edges is up to $O(\mathcal{W}\mathcal{T})$. It is much worse because in practice, $S \ll \min\{\mathcal{W}, \mathcal{T}\}$.

However, $\tilde{A}^*_{\mathbf{p}}$ is not necessarily monotone. For example, it is possible that the maxflow algorithm assigns a worker with skill s at level 1 to the task requiring s at level 1 but leaves a worker with s at level 2 unmatched. Thus, according to Proposition 4.1, $(\mathbf{p}, \tilde{A}^*_{\mathbf{p}})$ is not necessarily stable. In order to guarantee stability, intuitively, one may model the allocation problem as a minimum cost maxflow rather than the maxflow by additionally introducing costs on the edges from x-type nodes to y-type nodes. However, solving a mincost maxflow is much more expensive (see the survey [55]) than the efficient maxflow we proposed in Lemma 4.4.

Algorithm 4.1 Stability Adjustment

Input: W, T and $\tilde{A}$
Output: a monotone allocation A

1: $W' \leftarrow$ the set of all the unmatched workers
2: $A \leftarrow \tilde{A}$
3: Let $g(A, s)$ be the index of any s-marginal worker in A
4: **while** $W' \neq \emptyset$ **do**
5: Arbitrarily select and remove $\mathbf{w}$ from W'
6: **for** $s \in [S]$ **do**
7: **if** $g(A, s)$ exists **and** $w_s > w_s^{g(A,s)}$ **then**
8: Update A by matching $\mathbf{w}$ with the task assigned to $\mathbf{w}^{g(A,s)}$
9: $W' \leftarrow W' \cup \{\mathbf{w}^{g(A,s)}\}$
10: **break**
11: **end if**
12: **end for**
13: **end while**
14: **return** A.

Therefore, we propose *stable-adjustment* in Algorithm 4.1, and prove in Lemma 4.5 that it can construct $A^*_{\mathbf{p}}$ from $\tilde{A}^*_{\mathbf{p}}$ efficiently. To present it, we first define *skill-marginal*.

Definition 4.9 (s-Marginal)**.** *In an allocation, we call a worker* s-marginal *if she is matched with a task requiring skill s and her level of s is no higher than all the other workers matched with tasks requiring s. If $\mathbf{w}$ is s-marginal, we call w_s the* marginal-level *of s.*

Lemma 4.5. *Given $\tilde{A}$ which is not necessarily monotone, Algorithm 4.1 computes a monotone allocation A in polynomial time $O(S^2\mathcal{W} + S\mathcal{W}\log\min\{\mathcal{W}, \mathcal{T}\})$ that $|A| = |\tilde{A}|$.*

Proof. It is obvious that the adjustment will finally converge. W.l.o.g., we assume that after $M > 0$ iterations, the while-loop (lines 4-13) stops. Let A_m be the allocation at iteration $m \in \{0, \ldots, M\}$, e.g. $A_0 = \tilde{A}$ and $A_M = A$. Let $g(A_m, s)$ be the index of any s-marginal

worker in A_m. Let $\mathbf{L}_s = (w_s^{g(A_0,s)}, \ldots, w_s^{g(A_M,s)})$ be the sequence of the marginal-levels of skill s during the while-loop (lines 4-13). It true that $\forall s$, $\mathbf{L}_s$ is non-decreasing because the while-loop keeps replacing the s-marginal worker $\mathbf{w}^{g(A,s)}$ with a worker $\mathbf{w}$ that $w_s > w_s^{g(A,s)}$.

So once an s-marginal worker becomes unmatched, she will never be matched with any task requiring skill s because her level of s cannot exceed the marginal level of s. This means, for any skill s, every worker can be matched and unmatched with a task requiring s at most once respectively. So a worker will get matched and unmatched for at most S times, implying $M = O(S\mathcal{W})$. In each iteration (lines 5-12), the algorithm sends at most S queries to get the marginal workers of all the skills, and makes at most one update in $g(A, s)$ for some s. If we implement $g(A, s)$ by a min-heap, a single query is $O(1)$ and a single update is $O(\log |A|) = O(\log \min\{\mathcal{W}, \mathcal{T}\})$. Therefore, the overall time complexity is $O(S^2\mathcal{W}+S\mathcal{W} \log \min\{\mathcal{W}, \mathcal{T}\})$. Note that, the time complexity for preprocessing S min-heaps is dominated.

After the algorithm stops, it is true that for any unmatched $\mathbf{w}$ and skill s, $w_s \leq w_s^{g(A,s)}$. Therefore, A is monotone according to Definition 4.8. From the algorithm, it is true that $|A|$=$|\tilde{A}|$, thus proving the Lemma. □

To compute $A^*_{\mathbf{p}}$, we run the stable-adjustment with $\tilde{A}^*_{\mathbf{p}}$ as the input allocation, which is output by the maxflow in Lemma 4.4. The stable-adjustment only introduces an extra additive term $O(S^2\mathcal{W})$ to the time complexity of the maxflow. In practice, $S \ll min(\mathcal{W}, \mathcal{T})$, so it is efficient.

We next present the **S**table **M**echanism with **U**niform **P**ricing (SMUP) as follows.

Stable Mechanism with Uniform Pricing (SMUP)

Input T, W and h

1. Randomly pick up an integer k from $\{0, \ldots, \lfloor \log h \rfloor\}$, and set every entry of $\mathbf{p}$ to 2^k;
2. Run the maxflow algorithm in Lemma 4.4 to get $\tilde{A}^*_{\mathbf{p}}$;
3. Run the stable-adjustment in Algorithm 4.1 to get $A^*_{\mathbf{p}}$.

Output: $\mathbf{p}$ and $A^*_{\mathbf{p}}$;

Theorem 4.6. *SMUP is polynomial, truthful and IR. SMUP has* $(1 + \log h)$*-guarantee on revenue if* $v_j \in [1, h]$, $\forall j \in [\mathcal{T}]$.

Proof. It is trivial to verify individual rationality (IR) because any task j that $c_j < 2^k$ is removed according to Lemma 4.4. From Theorem 4.3, we know that the mechanism is polynomial time, and ($\mathbf{p}$, $A^*_{\mathbf{p}}$) is stable. We will prove truthfulness in Lemma 4.7 and the $(1 + \log h)$-guarantee on revenue in Lemma 4.8. □

Lemma 4.7. *Reporting* ($\underline{s}_j$, $\underline{l}_j$, v_j) *truthfully is the dominant strategy of task* j.

Proof. We will show that j can never improve its utility by reporting some parameters (s_j, l_j, c_j) other than its private parameters ($\underline{s}_j$, $\underline{l}_j$, v_j). First, it is obvious that reporting $s_j \neq \underline{s}_j$ cannot help, so the task will always report $\underline{s}_j$. We assume that j will only consider to report $l_j > \underline{l}_j$ because matching with $\mathbf{w}$ that $w_{s_j} < \underline{l}_j$ yields negative utility.

We next show that, for any $v > 0$, reporting ($\underline{s}_j$, l_j, v) cannot derive more utility than reporting ($\underline{s}_j$, $\underline{l}_j$, v) when $l_j > \underline{l}_j$. There are two cases. First, if reporting l_j and $\underline{l}_j$ both make j matched, it is obvious that reporting l_j derives no more utility than reporting $\underline{l}_j$ because $p(\underline{s}_j, \underline{l}_j) = p(\underline{s}_j, l_j)$. Next, we only need to prove that if reporting $\underline{l}_j$ fails to make j matched, reporting l_j also fails. There are two cases when reporting ($\underline{s}_j$, $\underline{l}_j$, v) fails. First, $v < 2^k$, so reporting ($\underline{s}_j$, l_j, v) fails too. Second, $v \geq 2^k$ but there is no more qualified worker $\mathbf{w}$ that $w_{\underline{s}_j} \geq \underline{l}_j$. So there is no $\mathbf{w}$ that $w_{\underline{s}_j} \geq l_j$, and reporting ($\underline{s}_j$, l_j, v) also fails in the second case.

We finally show that reporting ($\underline{s}_j$, $\underline{l}_j$, c_j) derives no more utility than reporting ($\underline{s}_j$, $\underline{l}_j$, v_j) for any $c_j \neq v_j$. There are two cases. First, if reporting c_j and v_j both make j matched, it is

true that these two strategies derive the same utility for j because $p(\underline{s}_j, \underline{l}_j) = 2^k$ is independent of c_j. Next, we only need to prove that if reporting v_j fails to make j matched, reporting c_j also fails. There are two cases when reporting v_j fails. First, $v_j < 2^k$. In such cases, if reporting $c_j \in (0, 2^k)$, j still fails to match; if reporting $c_j \geq 2^k$, j might be matched, however, $u_j = v_j - 2^k < 0$. The second case is when $v_j \geq 2^k$. In such cases, j participates the allocation algorithm without being precluded. However, the allocation algorithm, i.e. the maxflow described in Lemma 4.4 and the stable-adjustment in Algorithm 4.1, is independent of c_j, so reporting any c_j will not change the final allocation. □

Lemma 4.8. *SMUP has* $(1 + \log h)$*-guarantee on revenue if* $v_j \in [1, h]$, $\forall j \in [\mathcal{T}]$.

Proof. Let $\mathbf{P}^*$ and A^* be the optimal solution to the pricing problem, and $n(s, l)$ be the number of matched tasks requiring skill s at minimum level l in A^*. Thus, the optimal revenue $R(\mathbf{P}^*, A^*) = \sum_{s=1}^{S} \sum_{l=1}^{L} p^*(s, l) \cdot n(s, l)$ where $L \triangleq \max\{l_j | j \in [\mathcal{T}]\}$. Let $\mathbf{p}$ be the uniform pricing with every entry as 2^k where k is sampled from $\{0, \ldots, \lfloor \log h \rfloor\}$. From $\mathbf{P}^*$ and A^* (but in fact we do not know $\mathbf{P}^*$ or A^*), we could construct an allocation $\tilde{A}$ such that $(\mathbf{p}, \tilde{A})$ is feasible as follows: (1) discard any task j that $c_j < 2^k$; (2) for all the remaining matched tasks in A^*, match them with the same workers as A^* does. Next, we show the lower bound of the expectation of $R(\mathbf{p}, \tilde{A})$:

$$
\begin{aligned}
\mathbf{E}[R(\mathbf{p}, \tilde{A})] &= \sum_{k=0}^{\lfloor \log h \rfloor} \frac{1}{1 + \lfloor \log h \rfloor} \sum_{s=1}^{S} 2^k \cdot m_s(2^k) \\
&= \frac{1}{1 + \lfloor \log h \rfloor} \sum_{s=1}^{S} \sum_{k=0}^{\lfloor \log h \rfloor} 2^k \cdot \sum_{l=l(s,k)}^{L} n(s, l) \\
&= \frac{1}{1 + \lfloor \log h \rfloor} \sum_{s=1}^{S} \sum_{l=1}^{L} n(s, l) \cdot (2^{k(s,l)+1} - 1) \\
&\geq \frac{1}{1 + \lfloor \log h \rfloor} \sum_{s=1}^{S} \sum_{l=1}^{L} n(s, l) \cdot p^*(s, l) \qquad = \frac{R(\mathbf{P}^*, A^*)}{1 + \lfloor \log h \rfloor}
\end{aligned}
$$

$m_s(2^k)$ is the total number of matched tasks (in A^*) that requires skill s and is priced no less than 2^k. $l(s, k)$ is the lowest level satisfying $p^*(s, l(s, k)) \geq 2^k$. Similarly, $k(s, l)$ is the integer satisfying $2^{k(s,l)} \leq p^*(s, l) < 2^{k(s,l)+1}$.

However, ($\mathbf{p}$, $\tilde{A}$) is not necessarily stable. We observe that if $(\mathbf{w}, j)$ is a blocking pair, $\mathbf{w}$ must be unmatched due to the uniform pricing. Therefore, there exists A' such that $|A'| \geq |\tilde{A}|$ and ($\mathbf{p}$, A') is stable and feasible. Let $A^*_{\mathbf{p}}$ be the optimal allocation given uniform pricing $\mathbf{p}$. It is obvious that $\mathbf{E}[R(\mathbf{p}, A^*_{\mathbf{p}})] \geq \mathbf{E}[R(\mathbf{p}, A')]$. Although we cannot compute A' because we do not know $\mathbf{P}^*$ or A^*, we can compute $A^*_{\mathbf{p}}$ in polynomial time according to Theorem 4.3, thus completing the proof. □

4.3.3 Stable Mechanism with Non-uniform Pricing

Although SMUP provides an efficient solution with revenue guarantee, non-uniform pricing is usually more desirable to real markets. Therefore in this part, we consider a truthful stable mechanism with non-uniform pricing. For truthfulness, we still use randomized techniques to generate a non-uniform pricing. The allocation problem for non-uniform pricing has a close connection to the generalized stable matching where the preference lists not only contain ties but are also incomplete [79]. Although the existence of a generalized stable matching is NP-complete [80], later we will show that it is polynomial to find an allocation A for any non-uniform pricing $\mathbf{P}$ that ($\mathbf{P}$, A) is stable. Maximum generalized stable matching (Max SMTI) is NP-hard [79], and our allocation problem can be viewed as a weighted version of a special case (where the preferences lists of all the workers are the same) of Max SMTI. So the NP-hardness of Max SMTI cannot imply the NP-hardness of our allocation problem for non-uniform pricing, which we independently prove in Theorem 4.9.

Theorem 4.9. *The allocation problem is NP-hard even if the pricing only contains two distinct values.*

Proof. Because Minimum Maximal Matching for subdivision graph is NP-complete [81], Exact Maximal Matching (EMM) for subdivision graph is NP-complete. k-EMM is to find a maximal matching of size k. Let $G = (V, E)$ be the subdivision graph of some graph $G' = (V', E')$ that $V = V' \cup E'$ and $E = \{(e, v) : v \in V' \wedge e \in E' \wedge v \text{ is incident to } e\}$. G also has a bipartition (V', E'). W.l.o.g., we assume $|V'| = |E'| = n$. Given G and $k \in \{1, \ldots, n-1\}$, we reduce k-EMM of G to our allocation problem as follows.

Introduce $n+1$ skills, indexed from 1 to $n+1$. For each node $v_j \in V'$, introduce a task j as

$(j, 1, \infty)$. Introduce $n-k$ tasks indexed from $n+1$ to $2n-k$, and $\forall j \in \{n+1, \ldots, 2n-k\}$, task j is $(n+1, 1, \infty)$. For each node $e_i \in E'$, introduce a worker $\mathbf{w}^i$ such that (1) $\forall (e_i, v_j) \in E$, $w^i_j = 2$; (2) $w^i_{n+1} = 1$; and (3) for any other j, $w^i_j = 0$. Note that, $\forall i \in [n]$, since e_i has degree 2 in G, $\mathbf{w}^i$ has exactly 3 skills. Introduce $n-k$ workers indexed from $n+1$ to $2n-k$, and $\forall i \in \{n+1, \ldots, 2n-k\}$, set $\mathbf{w}^i$ such that (1) $w^i_{n+1} = 0$ and (2) $w^i_j = 1, \forall j \in [n]$. Overall, we have $2n-k$ tasks and workers respectively.

Next, pick up any $\epsilon \in (0, \frac{1}{n})$, and create a pricing $\mathbf{P}$ such that (1) $\forall j \in [n]$, $p(j, 1) = 1 + \epsilon$ and (2) $p(n+1, 1) = 1$. Clearly, for any feasible A, the revenue $R(\mathbf{P}, A) \leq 2n - k + \epsilon n$. We claim that $R(\mathbf{P}, A) = 2n - k + \epsilon n$ if and only if G has a maximal matching M of size k.

First, we assume that M is a maximal matching of size k. We construct A from M as follows. For $(e_i, v_j) \in M$, we match worker $\mathbf{w}^i$ and task j in A. For the remaining $n-k$ unmatched tasks, each of which is indexed no more than n, we arbitrarily match them with $\mathbf{w}^{n+1}, \ldots, \mathbf{w}^{2n-k}$ respectively. For the rest $n-k$ unmatched workers, each of which is indexed no more than n, we arbitrarily match them with tasks $n+1, \ldots, 2n-k$ respectively. Thus, all the tasks and workers are matched in A, implying $R(\mathbf{P}, A) = 2n - k + \epsilon n$. We then show that A is stable. For this, there is only one non-trivial case requiring a proof: $\forall i, j \in [n]$ and $(e_i, v_j) \notin M$, $\mathbf{w}^i$ and task j cannot be a blocking pair. If they form a blocking pair in A, it is true that e_i and v_j could be matched with each other in M but in fact they are both unmatched in M, contradicting with that M is a maximal matching.

Next, if $R(\mathbf{P}, A) = 2n - k + \epsilon n$, it is true that all the tasks and workers are matched in A. This means that there are exactly k tasks, each of which is indexed no more than n and matched with a worker indexed no more than n in A. We can construct M with all these k pairs. Next, we show that M is a maximal matching of G. If $M \cup \{(e_i, v_j)\}$ is also a feasible matching, it is true that (1) $w^i_j = 2$ and $p_j = 1 + \epsilon$; (2) task j is matched with some $\mathbf{w}^{i'}$ where $i' > n$ and $w^{i'}_j = 1$; and (3) $\mathbf{w}^i$ is matched with some task j' where $j' > n$ and $p_{j'} = 1$. These together indicate that ($\mathbf{w}^i$, j) is a blocking pair, contradicting with the stability of ($\mathbf{P}$, A). □

We then propropose a $\frac{3}{2}$-approximation, in Algorithm 4.2, to the allocation problem by generalizing the state-of-arts [76, 77] which both provide a $\frac{3}{2}$-approximation for (unweighted) maximum general stable matching. We first define the notion of *substitutable*.

Definition 4.10 (Substitutable)**.** *A matched (e.g. with task j) worker* $\mathbf{w}$ *is* substitutable *if there exists an unmatched worker* $\mathbf{w}'$ *that* $w_{s_j} = w'_{s_j}$.

Then we define *select*(j, Ψ), a function returning the worker in the worker set Ψ who is most preferred by task j. Let $\mathbf{w}$ be the worker returned by *select*(j, Ψ), so $\mathbf{w} = \operatorname{argmax}_{\mathbf{w}' \in \Psi} w'_{s_j}$. We define how *select*$(j, \Psi)$ deals with ties as follows: (1) it prefers the worker with the highest level of s_j; (2) among workers with the same level, it prefers unmatched to matched workers; (3) among matched workers with the same level, it prefers substitutable workers. In other cases, *select*(j, Ψ) breaks the tie arbitrarily.

Ψ_j is initialized as the set of all the workers qualified for task j. We present the approximation in Algorithm 4.2. In each iteration, it picks an unmatched task j^* with the highest price (breaking ties arbitrarily), and tries to match j^* with workers in Ψ_{j^*}. For every selected worker $\mathbf{w}$ returned by *select*(j^*, Ψ_{j^*}), it will match $\mathbf{w}$ with j^* if one of the three cases is true: (1) $\mathbf{w}$ is unmatched; or (2) $\mathbf{w}$ is substitutable; or (3) j^* has the same price with j (which is now matched with $\mathbf{w}$), and j^* has selected all its qualified workers once (i.e. $r_{j^*} = 1$) but j has not (i.e. $r_j = 0$). Otherwise the algorithm will not match $\mathbf{w}$ with j^*. If a task has selected all its qualified workers but remains unmatched, we give it one more chance, i.e. set r_j from 0 to 1. If the task fails to match again, i.e. $r_j = 1$, it will never be considered. We will show that if we combine all these design together, this approximation will have $\frac{3}{2}$-guarantee, and with any one missing, the guarantee drops to 2.

We first prove the stability and analyze time complexity in Lemma 4.10, and then show its performance guarantee in Theorem 4.12.

Lemma 4.10. *Given any* $\mathbf{P}$*, Algorithm 4.2 outputs* $\hat{A}$ *in polynomial time* $O(\mathcal{W}^2\mathcal{T} + \mathcal{T}\log\mathcal{T})$ *that (*$\mathbf{P}$*,* $\hat{A}$*) is stable.*

Proof. First we prove the time complexity. Either $|T'|$ or some $|\Psi_j|$ decreases by 1 during each iteration of the inner while-loop (lines 9-26), except when j selects a substitutable worker (lines 15-17). According to Claim 4.11, the total number of such exception during the entire algorithm is $O(\mathcal{W})$. Since Ψ_j will be reset at most once (lines 29-30), the total number of any operation within the outer while-loop (lines 6-35) is bounded by $\mathcal{T} + \sum_{j=1}^{\mathcal{T}} |\Psi_j| = O(\mathcal{W}\mathcal{T})$. If we implement *select*() naively, it is $O(\mathcal{W})$, so the total complexity of the outer while-loop is

Algorithm 4.2 Allocation for Non-uniform Pricing

Input: T, W and $\mathbf{P}$
Output: $\hat{A}$

$T' \leftarrow \{j | c_j \geq p(s_j, l_j), j \in [\mathcal{T}]\}$
$\forall j \in T'$, $\Psi_j \leftarrow \{\mathbf{w} | w_{s_j} \geq l_j, \mathbf{w} \in W\}$
$\forall j \in T', r_j \leftarrow 0$
$\hat{A} \leftarrow \emptyset$
$t \leftarrow 0$ ▷ Only for indexing iterations
while $T' \neq \emptyset$ **do**
    $t \leftarrow t + 1$
    $j^* \leftarrow \text{argmax}_{j \in T'} p(s_j, l_j)$
    while $\Psi_{j^*} \neq \emptyset$ **and** task j^* is unmatched **do**
        $\mathbf{w} \leftarrow$ *select*(j^*, Ψ_{j^*})
        if $\mathbf{w}$ is unmatched **then**
            Let $\mathbf{w}$ match with task j^* in $\hat{A}$
            $T' \leftarrow T' - \{j^*\}$
        else ▷ Assuming $\mathbf{w}$ is matched with task j
            if $\mathbf{w}$ is substitutable **then**
                Let $\mathbf{w}$ re-match with task j^* in $\hat{A}$
                $T' \leftarrow T' \cup \{j\} - \{j^*\}$
            else if $p_{j^*} = p_j$ **and** $r_{j^*} > r_j$ **then**
                Let $\mathbf{w}$ re-match with task j^* in $\hat{A}$
                $T' \leftarrow T' \cup \{j\} - \{j^*\}$
                $\Psi_j \leftarrow \Psi_j - \{\mathbf{w}\}$
            else
                $\Psi_{j^*} \leftarrow \Psi_{j^*} - \{\mathbf{w}\}$
            end if
        end if
    end while
    if task j^* is unmatched **then**
        if $r_{j^*} = 0$ **then**
            $r_{j^*} = 1$
            $\Psi_{j^*} \leftarrow \{\mathbf{w} | w_{s_{j^*}} \geq l_{j^*}, \mathbf{w} \in W\}$
        else
            $T' \leftarrow T' - \{j^*\}$
        end if
    end if
end while
return $\hat{A}$.

$O(\mathcal{W}^2\mathcal{T})$. Additionally, we need $O(\mathcal{T}\log\mathcal{T})$ to maintain a priority queue for all the tasks. So the overall time complexity is $O(\mathcal{W}^2\mathcal{T}+\mathcal{T}\log\mathcal{T})$.

Next we prove the stability. Assume that $\mathbf{w}$ and task j form a blocking pair. We first prove that $\mathbf{w}$ cannot be unmatched when the algorithm finishes. According to the algorithm, once $\mathbf{w}$ is selected (line 10) by any j at some iteration, she is matched till the end of the algorithm. No matter whether task j is unmatched or matched with a less preferred worker $\mathbf{w}'$ that $w_{s_j} > w'_{s_j}$, it must be true that j once selected $\mathbf{w}$, which is a contradiction.

Claim 4.11. *A worker is not substitutable after she is selected (line 10) twice by the algorithm.*

Proof of Claim. $\mathbf{w}$ *is always matched after the first selection. If she is substitutable after some later selection (e.g. by task j), it is true that $\exists\mathbf{w}'$ such that $w_{s_j} = w'_{s_j}$ and $\mathbf{w}'$ is unmatched. However, this contradicts with the fact that j selected* $\mathbf{w}$ *other than* $\mathbf{w}'$ *according to the* select() *function.*

Next, we prove that it is impossible for task j to be finally unmatched. Assuming j is finally unmatched. It is obvious that j fails to match with $\mathbf{w}$ twice, and according to Claim 4.11, $\mathbf{w}$ cannot be substitutable after the second selection from j. j fails to match with $\mathbf{w}$ for the second time because at that moment, $\mathbf{w}$ is matched with some different task j_1 that $p_{j_1} \geq p_j$. So she cannot finally match with some task j_2 that $p_{j_2} < p_j$, contradicting with that $\mathbf{w}$ prefers j_2 to j.

Finally, we show that $(\mathbf{w}, j)$ cannot be a blocking pair when j and $\mathbf{w}$ are both finally matched. If $\mathbf{w} \in \Psi_j$ after the algorithm ends, j must be matched with some $\mathbf{w}'$ that $w'_{s_j} \geq w_{s_j}$, which is a contradiction. If $\mathbf{w} \notin \Psi_j$ at last, it is true that $\mathbf{w}$ has been selected at least twice (at least once by j). So after the algorithm removes $\mathbf{w}$ from Ψ_j for the first time, $\mathbf{w}$ is never substitutable and $\mathbf{w}$ is matched with some task j' that $p_{j'} \geq p_j$. So it is impossible that $\mathbf{w}$ is finally matched with a task priced less than p_j. □

Theorem 4.12. *Algorithm 4.2 is a $\frac{3}{2}$-approximation to the allocation problem.*

Proof. The proof contains two parts, Lemma 4.13 as the first part and the remaining part based on Lemma 4.13.

Lemma 4.13. *For any pricing* $\mathbf{P}$*, Algorithm 4.2 outputs $\hat{A}$ such that $\frac{3}{2}|\hat{A}| \geq |A^*|$ where A^* is the optimal allocation.*

Proof. Assuming we know A^*. Construct a graph as follows. For each task that is matched in either A^* or $\hat{A}$, create a task node, and similarly for each worker who is matched in either A^* or $\hat{A}$, create a worker node. For any pair of a task node and a worker node, if they are matched in A^*, introduce an undirected edge between them, and similarly, if they are matched in $\hat{A}$, also introduce an undirected edge between them. In the graph, there are at most two edges between two nodes. If we consider a component with exactly two nodes and two edges as a 2-cycle, each connected component in the graph is either a cycle or path.

It is obvious that in any cycle, any task matched in A^* is also matched in $\hat{A}$, so there is no revenue loss in cycles. Also, in any path with even number of edges, the number of tasks matched in A^* is the same with the number of tasks matched in A (but there might be some revenue loss). We call a path with odd number of edges as an *augmenting* path. It is obvious that in any augmenting path, there is at least one task matched in A^* but not matched in $\hat{A}$. Clearly, in an augmenting path with $2k+1$ edges ($k \in \mathbb{N}$), the number of task nodes and worker nodes are both $k+1$, and there are exactly k tasks matched in $\hat{A}$ and $k+1$ tasks matched in A^*. It is enough to prove this lemma by showing that the graph does not contain any augmenting path with 1 or 3 edges.

First, it is obvious that an augmenting path with 1 edge cannot exist. If there was, the worker node and task node would have formed a blocking pair in $\hat{A}$, contradicting with that $(\mathbf{P}, \hat{A})$ is stable.

Suppose that there is an augmenting path with 3 edges. As shown in Fig 4.2, the solid line denotes that $\mathbf{w}$ and j are matched in $\hat{A}$, while $(\mathbf{w}, j')$ and $(\mathbf{w}', j)$, represented by dashed lines, are matched respectively in A^* (but neither $\mathbf{w}'$ nor j' is matched in $\hat{A}$).

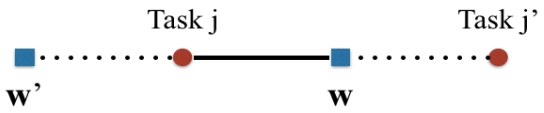


Figure 4.2: An augmenting path with 3 edges

We first show $w_{s_j} > w'_{s_j}$. If $w_{s_j} < w'_{s_j}$, $(\mathbf{w}', j)$ is a blocking pair in $\hat{A}$, which is a contradiction. If $w_{s_j} = w'_{s_j}$, $\mathbf{w}$ is finally substitutable. However, j' has selected $\mathbf{w}$ twice, according to Claim 4.11, which is a contradiction. We next show $p_j > p_{j'}$. It is trivial to see that $p_j < p_{j'}$ is impossible. If $p_j = p'_j$, it is true that $r_j = 1$, otherwise j' would have been

matched with $\mathbf{w}$. However, this contradicts with Claim 4.14 because $\mathbf{w}'$ is finally unmatched. Since we have $w_{s_j} > w'_{s_j}$ and $p_j > p_{j'}$, ($\mathbf{w}$, j) is a blocking pair in A^*, which is a contradiction. Therefore, any augmenting path with 3 edges cannot exist.

Claim 4.14. *$\forall j$, if $\exists \mathbf{w}$ such that (1) $\mathbf{w}$ is substitutable or unmatched and (2) $w_{s_j} \geq l_j$, we have $r_j = 0$.*

□

Now we show the second part based on Lemma 4.13. Suppose there are $K \leq \mathcal{T}$ unique values (i.e. prices) in $\mathbf{P}$, namely $\pi_1 \geq \ldots \geq \pi_K$. Let $\hat{A}_k$ (or A^*_k) be the partial approximate (or optimal) allocation which only includes all the matched worker-task pairs ($\mathbf{w}$, j) that $p_j \geq \pi_k$. Let $\hat{A}(k)$ (or $A^*(k)$) be the approximate (or optimal) allocation if the input task set only includes all the tasks priced no less than π_k in T. Since the order for each task being chosen (line 8) for the first time depends on its price, it is true that $|\hat{A}_k| = |\hat{A}(k)|$, according to Claim 4.15.

Claim 4.15. *$\forall p_{j'} > p_j$, if task j' is matched (or unmatched) at the moment when task j is chosen by the algorithm (line 8) for the first time, task j' will still be matched (or unmatched) when the algorithm finishes.*

Proof of Claim. *It is easy to see that task j cannot affect j' if $p_{j'} > p_j$, except that $\mathbf{w}$ who is matched with j' is substitutable. In such a case, $\mathbf{w}$ will be re-matched with j, however, there is at least one unmatched worker available for j', so j' will still be matched.*

With Lemma 4.13, it is true that $\forall k \in [K]$, $\frac{3}{2}|\hat{A}_k| = \frac{3}{2}|\hat{A}(k)| \geq |A^*(k)| \geq |A^*_k|$. Introducing a dummy price $\pi_{K+1} = 0$, we have that $\forall k \in [K]$,

$$\frac{3}{2}(\pi_k - \pi_{k+1}) \cdot |\hat{A}_k| \geq (\pi_k - \pi_{k+1}) \cdot |A^*_k|$$

Summing over all the K inequalities, we have:

$$\begin{aligned} R(\mathbf{P}, \hat{A}) &= \pi_1 \cdot |\hat{A}_1| + \sum_{k=2}^{K} \pi_k \cdot (|\hat{A}_k| - |\hat{A}_{k-1}|) \\ &\geq \frac{2}{3}\Big(\pi_1 \cdot |A^*_1| + \sum_{k=2}^{K} \pi_k \cdot (|A^*_k| - |A^*_{k-1}|)\Big) \qquad = \frac{2}{3} R(\mathbf{P}, A^*) \end{aligned}$$

□

Proposition 4.16. *The ratio $\frac{3}{2}$ is tight for Algorithm 4.2.*

Proof. We prove this by an example: there are 3 tasks, $(1, 1, +\infty)$, $(2, 1, +\infty)$ and $(3, 1, +\infty)$, and they are priced at $p + 2\epsilon$, $p + \epsilon$ and p respectively where $\epsilon > 0$; 3 workers, $\mathbf{w}^1 = (1, 0, 0)$, $\mathbf{w}^2 = (1, 1, 0)$ and $\mathbf{w}^3 = (0, 1, 1)$. A feasible approximate allocation by Algorithm 4.2 is ($\mathbf{w}^2$, 1) and ($\mathbf{w}^3$, 2) but $\mathbf{w}^1$ and task 3 are both left unmatched (the algorithm first matches task 1 with $\mathbf{w}^2$, then task 2 with $\mathbf{w}^3$), with revenue $2p + 3\epsilon$. However, the optimal allocation is ($\mathbf{w}^1$, 1), ($\mathbf{w}^2$, 2) and ($\mathbf{w}^3$, 3), with revenue $3p + 3\epsilon$. So the $\frac{3}{2}$-approximation is tight for Algorithm 4.2. □

Based on the approximation, we present the **S**table **M**echanism with **N**non-uniform **P**ricing (SMNP) as follows.

Stable Mechanism with Non-uniform Pricing (SMNP)

Input T, W and h

1. $\forall s \in [S]$: (1) pick up an integer k_s uniformly and independently sampled from $\{0, \ldots, \lfloor \log h \rfloor\}$, and (2) $\forall l$: $p(s, l) \leftarrow 2^{k_s}$;
2. Run Algorithm 4.2 to compute $\hat{A}$;

Output: $\mathbf{P}$ and $\hat{A}$.

Theorem 4.17. *SMNP is polynomial, truthful and IR. SMNP has $(3 + 3 \log h)$-guarantee on revenue if $v_j \in [1, h]$, $\forall j \in [\mathcal{T}]$.*

Proof. According to Lemma 4.10, ($\mathbf{P}$, $\hat{A}$) is stable, and $\hat{A}$ can be computed in polynomial time. SMNP is IR since task j will never be charged more than c_j (line 1 in Algorithm 4.2). The proof of truthfulness is similar to the proof that SMUP is truthful in Lemma 4.7, thus we omit it.

We next prove the $(3 + 3 \log h)$-guarantee on revenue in two steps. In the first step, we show that if we knew (but in fact we do not know) the optimal stable pricing-allocation pair ($\mathbf{P}^*$, A^*), we could have constructed $\tilde{A}$ satisfying that: given any $\mathbf{P}$ chosen at random (as

SMNP does), (1) $(\mathbf{P}, \tilde{A}\ )$ is feasible but not necessarily stable and (2) $\mathbf{E}[R(\mathbf{P}, \tilde{A})] \geq \frac{R(\mathbf{P}^*, A^*)}{2+2\log h}$. In the second step, we show that we could construct A' from $\tilde{A}$ such that $(\mathbf{P}, A')$ is stable and $R(\mathbf{P}, A') \geq \frac{R(\mathbf{P}, \tilde{A})}{2}$. Let $A^*_{\mathbf{P}}$ be the optimal solution to the allocation problem given $\mathbf{P}$, so we have $R(\mathbf{P}, A^*_{\mathbf{P}}) \geq R(\mathbf{P}, A')$. According to Theorem 4.12, we have $\frac{3}{2}R(\mathbf{P}, \hat{A}) \geq R(\mathbf{P}, A^*_{\mathbf{P}})$, so we have $\mathbf{E}[R(\mathbf{P}, \hat{A})] \geq \frac{R(\mathbf{P}^*, A^*)}{3+3\log h}$. Since according to Lemma 4.10, we can compute $\hat{A}$ in polynomial time, the proof is completed. Next, we show the proof for the two steps.

In the first step, we show that given $\mathbf{P}$, we could construct $\tilde{A}$ from A^* as follows: (1) discard any task j that $c_j < p_j$; (2) for all the remaining matched tasks in A^*, match them with the same workers as A^* does.

Claim 4.18. $\mathbf{E}[R(\mathbf{P}, \tilde{A})] \geq \frac{R(\mathbf{P}^*, A^*)}{1+\log h}$.

Proof of Claim. *Let $R_s(\mathbf{P}, \tilde{A})$ denote the revenue contributed by all the tasks requiring skill s, i.e. $R(\mathbf{P}, \tilde{A}) = \sum_{s=1}^{S} R_s(\mathbf{P}, \tilde{A})$. Since for any s, k_s is sampled independently, $\mathbf{E}[R(\mathbf{P}, \tilde{A})] = \sum_{s=1}^{S} \mathbf{E}[R_s(\mathbf{P}, \tilde{A})]$. The proof that $\forall s$, $\mathbf{E}[R_s(\mathbf{P}, \tilde{A})] \geq \frac{R_s(\mathbf{P}^*, A^*)}{1+\log h}$ is similar to the proof for the lower bound of expected revenue of the randomized uniform pricing in Lemma 4.8, thus omitted.*

However, $(\mathbf{P}, \tilde{A})$ is not necessarily stable because $\tilde{A}$ is not necessarily monotone. In the second step, we show that Algorithm 4.3 uses a simple greedy method to construct A' from $\tilde{A}$ and $\mathbf{P}$ such that (1) $(\mathbf{P}, A')$ is stable; and (2) $R(\mathbf{P}, A') \geq \frac{R(\mathbf{P}, \tilde{A})}{2}$. Note that, this method is merely for proving the existence of A', so instead of analyzing time complexity, we only need to prove that it will always halt in Claim 4.19.

Algorithm 4.3 Stability Adjustment 2

Input: T, W, $\mathbf{P}$ and $\tilde{A}$
Output: A'
1: $A' \leftarrow \tilde{A}$
2: **while** there is a blocking pair $(\mathbf{w}, j)$ in $(\mathbf{P}, A')$ **do**
3: Update A' by matching $\mathbf{w}$ with j
4: **end while**
5: **return** A'.

Claim 4.19. *Algorithm 4.3 always halts in finite iterations.*

Proof of Claim. *If ($\mathbf{w}$, j) is a blocking pair, through matching them with each other, $\mathbf{w}$ and j*

are not blocking, and they will never be blocking. It is because (1) if $\mathbf{w}$ *is later re-matched with* j'*, it must be true that* $p_{j'} > p_j$*; and (2) if* j *is later re-matched with* $\mathbf{w}'$*, it must be true that* $w'_{s_j} > w_{s_j}$*. Since the total number of blocking pairs is bounded by* $\mathcal{WT}$*, Algorithm 4.3 always halts in finite iterations.*

Claim 4.20. $R(\mathbf{P}, A') \geq \frac{R(\mathbf{P},\tilde{A})}{2}$*, and (*$\mathbf{P}$*,* A'*) is stable*

Proof of Claim. *Since the algorithm always halts according to Claim 4.19, (*$\mathbf{P}$*,* A'*) is stable. We next show that* $R(\mathbf{P}, A') > \frac{R(\mathbf{P},\tilde{A}))}{2}$*. The loss of revenue might occur only when we re-match* $\mathbf{w}$ *and* j *(line 3) in* A' *if* $\mathbf{w}$ *and* j *were both matched (with others) before this re-match. In the worst case,* $\mathbf{w}$ *was matched with* j' *and* j *was matched with* $\mathbf{w}'$ *in* $\tilde{A}$*, but after the re-match,* j' *is finally matched in* A'*. In such a case, we lose the revenue* $p_{j'}$ *but guarantee the revenue* p_j*. Since* $p_{j'} < p_j$*, the total loss of revenue is less than* $\frac{R(\mathbf{P},\tilde{A}))}{2}$*, thus proving the claim.*

□

In this section, we presented SMUP and SMNP. Due to the NP-hardness of allocation problem for non-uniform pricing, the revenue guarantee of SMNP is slightly worse than SMUP. As we show through experiments, SMNP and SMUP generate nearly the same revenue but SMNP is more robust. We also tried to assign different prices over different levels of a skill, however, this strategy makes the guarantee worse, which is omitted here.

4.3.4 Online Stable Mechanism with Uniform Pricing

In this section, we present an online truthful stable mechanism. We assume that the market runs in M rounds, and the pricing and the allocation are static. That is, if we fix the price $p(s, l)$ in round m, we cannot change it in later rounds. Similarly, if we match a worker and a task in round m, we cannot re-match or unmatch them in later rounds. We require global stability. Let $\mathbf{P}$ be the pricing. Let A_m be the allocation in round m and A be the global allocation during all the M rounds, i.e. $A = \bigcup_{m=1}^{M} A_m$. We require not only ($\mathbf{P}$, A_m) to be stable $\forall m \in [M]$, but also ($\mathbf{P}$, A) to be stable.

Under these constraints, it is easy to see that if workers arrive online, there is no stable mechanism. So we study the case where all the workers are present in the first round but tasks

arrive online. In this setting, task j needs to report two additional parameters a_j and d_j where $a_j \leq d_j \in [M]$ to the market. They denote the arrival and departure time respectively, i.e. the market can match j in any round $m \in \{a_j, \ldots, d_j\}$. Let m_j be the round that j gets matched, $\underline{a}_j$ and $\underline{d}_j$ be the private time parameters owned by j. The utility function becomes: $u_j = v_j \cdot \mathbb{1}(w_{\underline{s}_j} \geq \underline{l}_j \wedge m_j \in \{\underline{a}_j, \ldots, \underline{d}_j\}) - p(s_j, l_j)$ if j is matched with $\mathbf{w}$; otherwise $u_j = 0$. Besides misreporting $\underline{s}_j$, $\underline{l}_j$ and v_j, task j can strategically choose the time to appear in the market. As it is natural in online mechanism design, we adopt a restricted misreporting model where we assume no early arrival but unrestricted departure, i.e. j may report any $a_j \geq \underline{a}_j$. This assumption is practical because $\underline{a}_j$ can be viewed as the earliest time that the employer realizes he needs to solve a task.

We have not found any mechanism with non-uniform pricing that is simultaneously stable, truthful and with non-trivial revenue guarantee. So we present the **O**nline **S**table **M**echanism with **U**niform **P**ricing (OSMUP) as follows. Let *MaxFlow*(W, T, $\mathbf{p}$) denote the maxflow algorithm in Lemma 4.4 with inputs W as the worker set, T as the task set and $\mathbf{p}$ as the uniform pricing respectively. Let *StableAdjustment*(W, T, $\tilde{A}$) denote the stable-adjustment in Algorithm 4.1 with W, T and $\tilde{A}$ as input. Let T_m be the set of tasks that arrive in round m (i.e. $a_j = m$).

Algorithm 4.4 OSMUP

Input: W and $T_1, \ldots, T_M$ in sequence
Output: $\mathbf{p}$ and $A_1, \ldots, A_M$

$W_1 \leftarrow W$
Sample k from $\{0, \ldots, \lfloor \log h \rfloor\}$ and $\forall s, l, p(s, l) \leftarrow 2^k$
for $m \in [M]$ **do**
    $\tilde{A}_m \leftarrow$ *MaxFlow*(W_m, T_m, $\mathbf{p}$)
    $A_m \leftarrow$ *StableAdjustment*(W_m, T_m, $\tilde{A}_m$)
    $W_{m+1} \leftarrow W_m - \{\mathbf{w} | \mathbf{w}$ is matched in $A_m\}$
end for
return $\mathbf{p}$ and $A_1, \ldots, A_M$

Theorem 4.21. *OSMUP is polynomial, truthful and IR. OSMUP has* $(2 + 2\log h)$*-guarantee on revenue if* $\forall j \in [\mathcal{T}]$, $v_j \in [1, h]$.

OSMUP is obviously polynomial and IR. We prove the stability in Lemma 4.22, the $(2 + 2\log h)$-guarantee on revenue in Lemma 4.23 and the truthfulness in Lemma 4.25 respectively.

Lemma 4.22. $(\mathbf{p}, A)$ *is stable where* $\mathbf{p}$ *is the uniform pricing of* 2^k *where* k *is sampled from* $\{0, \ldots, \lfloor \log h \rfloor\}$.

Proof. Let $T(m)$ be the set of all the tasks that $a_j \in [m]$, i.e. $T(m) = \bigcup_{i=1}^{m} T_i$. Let $A(m)$ be the union of allocations for the first m rounds, i.e. $A(m) = \bigcup_{i=1}^{m} A_i$.

To prove the lemma, it is enough to prove the claim that $\forall m \in [M]$, $(\mathbf{p}, A(m))$ is stable w.r.t. W and $T(m)$. According to Theorem 4.3, it is true that $\forall m \in [M]$, $(\mathbf{p}, A_m)$ is stable w.r.t. W_m and T_m, which immediately implies the case when $m = 1$, i.e. $(\mathbf{p}, A(1))$ is stable w.r.t. W and $T(1)$. We next prove that for any $m = 1, \ldots, M-1$, if $(\mathbf{p}, A(m))$ is stable w.r.t. W and $T(m)$, $(\mathbf{p}, A(m+1))$ is also stable w.r.t. W and $T(m+1)$. For any $(\mathbf{w}, j)$ that $\mathbf{w} \in W$ and $a_j \leq m$, it cannot be a blocking pair. For any $(\mathbf{w}, j)$ that $\mathbf{w} \in W_{m+1}$ and $a_j = m+1$, it cannot be a blocking pair. It is only possible that there exists a blocking pair $(\mathbf{w}, j)$ that $\mathbf{w} \in W \setminus W_{m+1}$ and $a_j = m+1$. However, $\forall \mathbf{w} \in W \setminus W_{m+1}$, $\mathbf{w}$ is matched in A_m, so she cannot prefer a different task because of the uniform pricing, thus completing the proof. □

Lemma 4.23. *OSMUP has* $(2 + 2\log h)$*-guarantee on revenue if* $\forall j \in [\mathcal{T}]$, $v_j \in [1, h]$.

Proof. Assuming that we have randomly generated the uniform pricing $\mathbf{p}$. Let $A^*_{\mathbf{p}}$ be the optimal allocatio given $\mathbf{p}$ if all the tasks are present in the first round. So $(\mathbf{p}, A^*_{\mathbf{p}})$ is exactly the offline mechanism (SMUP), which has $(1 + \log h)$-gurantee on revenue according to Lemma 4.8. It is enough to prove this lemma by showing that $|A| \geq \frac{1}{2}|A^*_{\mathbf{p}}|$ where $A = \bigcup_{m=1}^{M} A_m$, which is implied by the following claim (which is proved as Theorem 2.5 in [79]).

Claim 4.24. *[79] For any stable matching problem, the size of the largest matching is at most twice the size of the smallest.*

Since $(\mathbf{p}, A)$ and $(\mathbf{p}, A^*_{\mathbf{p}})$ are both stable w.r.t. W and T, according to Claim 4.24, we have $|A| \geq \frac{1}{2}|A^*_{\mathbf{p}}|$. □

Lemma 4.25. *Reporting* $(\underline{a}_j, \underline{d}_j, \underline{s}_j, \underline{l}_j, v_j)$ *truthfully is the dominant strategy for task* j.

Proof. The proof of the truthfulness of $\underline{s}_j$, $\underline{l}_j$ and v_j is similar to the proof for Lemma 4.7, thus omitted here. We show the proof of the truthfulness of $\underline{a}_j$ and $\underline{d}_j$. First, reporting $\underline{d}_j$ truthfully cannot make the task worse off because the pricing and allocation are independent of d_j. Due to

the assumption of no early arrival, we only consider reporting $a_j \in \{\underline{a}_j + 1, \ldots, \underline{d}_j\}$. Since the pricing is uniform, we only need to show that if reporting $\underline{a}_j$ cannot make j matched, reporting a_j cannot either, which is proved by the following arguments. Since j is not matched in round $\underline{a}_j$, it is true that in $W_{\underline{a}_j}$, there is no worker qualified for j. Since W_m is non-increasing over all the M rounds, there is no qualified worker in $W_{\underline{a}_j+1}, \ldots, W_M$ either. So reporting a_j cannot make j matched if reporting $\underline{a}_j$ cannot. □

4.4 Experiments

In this section, we use real price data from Amazon Mechanical Turk and synthetic skill data to evaluate the two offline stable mechanisms SMUP and SMNP, and the online stable mechanism OSMUP.

4.4.1 Crawled AMT Dataset

We use the API[82] to crawl 46309 tasks from Amazon Mechanical Turk (HIT) and their prices. We discard the long tail ($< 5\%$), i.e. any task priced no less than 1000 cents. Let $\mathcal{D}$ be the *price pool* containing all the remaining 44043 tasks priced between 1 and 999 cents. We plot the price distribution of $\mathcal{D}$ in Fig 4.3 where (x, y) denotes that y is the accumulated percentage of tasks priced no more than x cents.

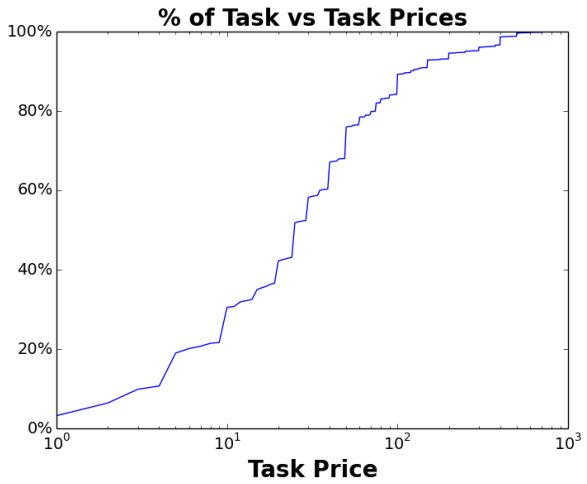


Figure 4.3: Distribution of task prices

However, the tasks from AMT do not contain skill or level information. So we create

synthetic skills and levels. To randomly generate input instances of the pricing problem, we set h to 999, S to 100, and L to 10 where L is the maximum level of a skill. For each task j, we uniformly and independently sample s_j from $[S]$, l_j from $[L]$ and c_j from the price pool $\mathcal{D}$ with replacement. For each worker, we randomly assign her $k \in \{1, \ldots, 5\}$ skills, each of which is at the level sampled independently from $[L]$. We generate two types of instances, the instance of *small size* where $\mathcal{W} = \mathcal{T} = 1000$, and the instance of *large size* where $\mathcal{W} = \mathcal{T} = 100000$.

4.4.2 Evaluation on SMUP and SMNP

In this part, we evaluate how the choice of pricing, i.e. uniform and non-uniform, impacts the revenue. In fact, we have not found any polynomial time solution to the pricing problem (i.e. the offline optimization problem without truthfulness). But we find that computing the optimal uniform pricing and its corresponding allocation is an $O(\log \min\{\mathcal{W}, \mathcal{T}\})$-approximation. Clearly, it is not truthful, but produces a revenue no less than the revenue of SMUP which uses randomized uniform pricing. However, this method, as an approximately optimal baseline, needs to run the maxflow in Lemma 4.4 and the stable-adjustment in Algorithm 4.1 by $O(\mathcal{T})$ times. So we generate 10000 instances of small size for evaluations.

For each case, we use R^*, R_u and R_n to denote the revenue generated by the baseline, SMUP and SMNP respectively. Let $r_u = \frac{R_u}{R^*}$, and $r_n = \frac{R_n}{R^*}$ be the *revenue ratios* of SMUP and SMNP respectively. We plot the CDFs of the two revenue ratios in Fig 4.4 where (x, y) denotes that there are y percentages of instances that $r_u \leq x$ (or $r_n \leq x$). The mean values of r_u and r_n over the 10000 cases are very close, i.e. $\mu(r_u) = 0.435$ and $\mu(r_n) = 0.431$, while their standard deviations differ significantly, i.e. $\sigma(r_u) = 0.31$ and $\sigma(r_n) = 0.06$. This means that, although theoretically SMUP has $(1 + \log h)$-guarantee on revenue while SMNP only has $(3 + 3\log h)$-guarantee, in practice, they produce nearly the same revenue. However, SMNP is more robust than SMUP. So we recommend SMNP.

Besides revenues, we also analyze matching size, i.e. the number of matched worker-tasks pairs. We have highly similar observations. Let m_u and m_n be the number of matched worker-task pairs in SMUP and SMNP respectively. We have $\mu(m_u) = 457$, $\mu(m_n) = 445$, $\sigma(m_u) = 382.3$ and $\sigma(m_n) = 39.2$, which again shows that they have similar performance in practice, but SMNP is more robust.

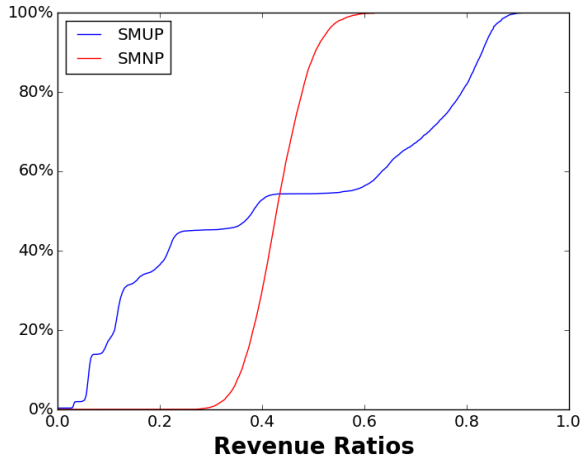


Figure 4.4: Distributions of revenue ratios

We also analyze the revenue and matching size over 1000 input instances of large size. In such cases, R^* is not able to compute, so we directly compare R_u and R_n. We observe nearly the same patterns: (1) the mean of revenue of SMNP, i.e. $\mu(R_n)$, is slightly ($\approx 0.8\%$) higher than that of SMUP, i.e. $\mu(R_u)$; and (2) the standard deviation of R_u is 9.7 times as the standard deviation of R_n.

4.4.3 Evaluation on OSMUP

In this part, we evaluate the online stable mechanism OSMUP. We use SMUP as the baseline. We consider 1000 instances of large size. Within each instance: we randomly generate the same uniform pricing for both OSMUP and SMUP; we assume that the market runs in 100 rounds, and randomly assign the arrival time $a_j \in [100]$ to each task j. Let R_o denote the revenue produced by OSMUP, and we are particularly interested in the relative ratio $\frac{R_o}{R_u}$ where R_u is revenue of SMUP. Among all the 1000 cases, the least revenue ratio is 0.733; there are 75.0% cases with the ratio at least 0.95; the average ratio is 0.948, much larger than the theoretical guarantee which is 0.5 according to Claim 4.24.

4.5 Related Work

Stable Matching. The stable matching problem was first introduced in the seminal paper [75], and has received much attention, e.g. [83, 78]. Among many variants [79, 84], finding

a maximum stable matching with ties and incomplete preference lists (MAX SMTI) [80] is closely related to our work. This problem is known to be APX-hard [85]. [86] gives a 1.875-approximation, and [76, 77] improve the ratio to 1.5. Our allocation problem, which can be viewed as a weighted version of a special case of MAX SMTI, has not been addressed yet, despite many weighted variants were studied [83, 84].

In two-sided markets, [87] shows the equivalence of stable matching and core. In heterogeneous and competitive job markets, [88, 89, 90] consider the matching between workers and firms. In these pricing (and matching) problems, the wages are determined by equilibrium [91]. Their results apply for labor markets where the prices and matching are determined freely by employers and workers, but not for the market, considered in this chapter, where a monopoly pricing and compelling matching are needed to maximize the revenue.

Envy-free Pricing Mechanisms. Pricing is a well-studied area in Economics. In particular, the envy-free pricing [29, 60, 61] in Walrasian Equilibrium [54] is relevant to our work. In recent years, envy-free pricing is studied in various settings [68, 62, 64, 67]. [29] first addresses the computational issue of envy-free pricing. They show that the problem is NP-hard even for the two special cases where the buyers are either unit-demand or single-minded. However, the envy-free pricing they discussed is different from our stable pricing because in our market, not only the buyer, i.e. the task, will envy, but the item, i.e. the worker, will also envy.

Crowdsourcing Markets. In crowdsourcing markets, [92, 93, 94] consider online budget feasible mechanisms for a single employer with a budget and multiple workers. When the skill levels of workers are unknown, [95, 96] provide learning methods for online task assignment. [97] discusses that during a work session, additional rewards lead to higher effort of workers. [19] presents an algorithm to decide when to offer bonuses to workers to improve the overall utility of the employer.

4.6 Conclusions

We addressed the pricing problem, in particular, revenue maximizing stable pricing in online labor markets. We presented three efficient truthful stable mechanism with provable guarantees on revenue. We believe that there is a real need to further study and improve mechanisms for

pricing and allocation of workers and tasks based on fairness since many online labor markets rely on such mechanisms.

Chapter 5

Summary and Future Work

In this thesis, we solve three pricing-related research problems in online markets. The first problem we study is the targeting problem for advertisers. For the OSN's perspective setting where the user information is completely accessible, we present a polynomial time algorithm and prove that it has $1 - 1/e$ guarantee. For the advertiser's perspective setting where the user information is only partially accessible, we show through data analysis that the strategy of targeting subsets of audience sets is viable and propose a greedy algorithm based on subset targeting. For evaluation, we crawl a large unique dataset which contains more than one million suggested bids from Facebook and LinkedIn, and we show that the proposed algorithm makes advertisers reach more target audience than directly targeting the users.

The second pricing problem we address is the revenue maximizing arbitrage-free pricing in user-based markets, e.g. advertising markets. We first point out that the version-arbitrage potentially exists in the pricing for a set of user attributes, and define the arbitrage-free pricing. Then we presented a variety of efficient algorithms for arbitrage-free pricing with provable approximation guarantees on the revenue, and hardness results for certain variations. We believe that there is a real need to study mechanisms for allocation and pricing of users based on multiple attributes as much of online advertising markets rely on such systems.

The last pricing problem we address is revenue maximizing stable pricing problem in online labor markets. As online markets start to use automatic pricing and allocation system, we point out such a system may treat workers or tasks unfairly. To prevent this, we define the stable pricing mechanism, in which every worker and task will be treated fairly. We presented two efficient truthful stable mechanisms with provable guarantees on revenue for the offline case. For the case where tasks arrive online, we also present an online efficient truthful stable mechanism with provable guarantees on revenue.

The following pricing-related problems/directions would be interesting to markets:

- **Fairness in Pricing**. Our stable pricing mechanism is guaranteed that every entity on the market, i.e. all the workers and tasks, will be treated fairly. This constraint is very strong, sometimes hard to implement for a real market – for example, we cannot find a stable pricing mechanism when we allow workers arrive online as we discussed in Chapter 4. One interesting direction would be that, if we slack the absolute stability to ϵ-stability where ϵ is a parameter to control the balance between fairness and unfairness – in other words, if we allow that some proportion of entities are treated unfairly, whether the revenue guarantee will be improved correspondingly? It is interesting to mathematically analyze the tradeoff between revenue and fairness. Another direction of designing fair/stable pricing mechanism in ride-sharing markets. In some ride-sharing markets, the matching between drivers and passengers are automatically determined by algorithms as well as the charge, so either a driver or a passenger would have the risk to be treated unfairly. Besides, the pool pricing also has the potential to unfairly split the fees over all the pool passengers. As the pricing model and business model in ride-sharing markets are quite different from those in labor markets, we cannot directly apply our stable pricing mechanisms, thus we have new challenges.

- **Group Pricing**. We previously assume that a task only requires one worker. However in practice, a task may require a group of workers to cooperate, thus pricing the task, forming a group and distributing the income over the groups become a new pricing problem. Given our stable pricing mechanisms, one straightforward solution is that, the employer manually breaks down the tasks into a set of sub-tasks with corresponding skills and utilities, and submit these sub-tasks independently to the labor market running stable pricing mechanisms. Although this is a feasible solution, it does require the employer to be quite familiar with the task decomposition and the utility of each sub-task – which contradicts with the fact that many customers only know how much they are willing to pay and what they need. Therefore, a fair pricing and grouping mechanism is urgently needed by a labor market which aims to provide advanced freelancing solutions to customers.

- **Price Prediction**. As the targeting problem relies on the knowledge of prices which

change over time, accurately predicting the future prices of different but highly correlated user attributes, will significantly help advertisers make decisions.

References

[1] C. Xia, S. Guha, and S. Muthukrishnan, "How much is your attention worth? analysis of prices in linkedin advertising network," in *NetEcon*, 2016.

[2] Y. Liu, C. Kliman-Silver, R. Bell, B. Krishnamurthy, and A. Mislove, "Measurement and analysis of osn ad auctions," in *COSN*, 2014.

[3] C. Li, D. Y. Li, G. Miklau, and D. Suciu, "A theory of pricing private data," *ACM Trans. Database Syst.*, 2014.

[4] C. Xia and S. Muthukrishnan, "Revenue-maximizing stable pricing in online labor markets," in *HCOMP*, 2017.

[5] G. Hodge and C. Cagle, "Business-to-business e-business models: classification and textile industry implications," *AUTEX Research Journal*, 2004.

[6] R. Clarke, "Towards a taxonomy of b2b e-commerce schemes," *BLED*, 2001.

[7] S. Kaplan and M. Sawhney, "B2b e-commerce hubs: towards a taxonomy of business models," *Harvard Business Review*, 2000.

[8] B. M. Movahedi, K. M. Lavassani, and V. Kumar, "E-marketplace emergence: Evolution, developments and classification," *Journal of Electronic Commerce in Organizations*, 2012.

[9] S. Wang and N. P. Archer, "Electronic marketplace definition and classification: Literature review and clarifications," *Enterp. Inf. Syst.*, 2007.

[10] B. Edelman, M. Ostrovsky, and M. Schwarz, "Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords," *American economic review*, 2007.

[11] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of Finance*, 1961.

[12] E. H. Clarke, "Multipart pricing of public goods," *Public Choice*, 1971.

[13] T. Groves, "Incentives in teams," *Econometrica: Journal of the Econometric Society*, 1973.

[14] C. Xia and S. Muthukrishnan, "Arbitrage-free pricing in user-based markets," in *AAMAS*, 2018.

[15] C. Xia, S. Guha, and S. Muthukrishnan, "Targeting algorithms for online social advertising markets," in *ASONAM*, 2016.

[16] S. C. Kuek, C. Paradi-Guilford, T. Fayomi, S. Imaizumi, P. Ipeirotis, P. Pina, M. Singh *et al.*, "The global opportunity in online outsourcing," The World Bank, Tech. Rep., 2015.

[17] H. Li, B. Zhao, and A. Fuxman, "The wisdom of minority: Discovering and targeting the right group of workers for crowdsourcing," in *WWW*, 2014.

[18] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *Journal of Machine Learning Research*, 2010.

[19] M. Yin and Y. Chen, "Bonus or not? learn to reward in crowdsourcing." in *IJCAI*, 2015.

[20] M. Kokkodis, P. Papadimitriou, and P. G. Ipeirotis, "Hiring behavior models for online labor markets," in *WSDM*, 2015.

[21] M. Kokkodis and P. Ipeirotis, "The utility of skills in online labor markets," 2014.

[22] D. Martin, B. V. Hanrahan, J. O'Neill, and N. Gupta, "Being a turker," in *CSCW*, 2014.

[23] L. Olejnik, T. Minh-Dung, C. Castelluccia *et al.*, "Selling off privacy at auction," in *NDSS*, 2014.

[24] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing amazon ec2 spot instance pricing," *ACM Trans. Econ. Comput.*, 2013.

[25] P. G. Ipeirotis, "Analyzing the amazon mechanical turk marketplace," *XRDS*, 2010.

[26] L. Chen, A. Mislove, and C. Wilson, "Peeking beneath the hood of uber," in *IMC*, 2015.

[27] P. Rusmevichientong and D. P. Williamson, "An adaptive algorithm for selecting profitable keywords for search-based advertising services," in *EC*, 2006.

[28] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang, "How to bid the cloud," in *SIGCOMM*, 2015.

[29] V. Guruswami, J. D. Hartline, A. R. Karlin, D. Kempe, C. Kenyon, and F. McSherry, "On profit-maximizing envy-free pricing," in *SODA*, 2005.

[30] L. Walras, *Elements of Pure Economics*, 1954.

[31] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu, "Query-based data pricing," *J. ACM*, 2015.

[32] B. Edelman, M. Ostrovsky, and M. Schwarz, "Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords," 2005.

[33] B. Bi, M. Shokouhi, M. Kosinski, and T. Graepel, "Inferring the demographics of search users: Social data meets search queries," in *WWW*, 2013.

[34] S. Khuller, A. Moss, and J. S. Naor, "The budgeted maximum coverage problem," *Information Processing Letters*, 1999.

[35] S. Guha, B. Cheng, and P. Francis, "Challenges in measuring online advertising systems," in *IMC*, 2010.

[36] "Facebook Online Advertising System," https://www.facebook.com/advertising.

[37] "LinkedIn Online Advertising System," https://www.linkedin.com/ads/.

[38] A. Ghosh, B. I. Rubinstein, S. Vassilvitskii, and M. Zinkevich, "Adaptive bidding for display advertising," in *WWW*, 2009.

[39] J. Feldman, S. Muthukrishnan, M. Pal, and C. Stein, "Budget optimization in search-based advertising auctions," in *EC*, 2007.

[40] E. Even Dar, V. S. Mirrokni, S. Muthukrishnan, Y. Mansour, and U. Nadav, "Bid optimization for broad match ad auctions," in *WWW*, 2009.

[41] S. Muthukrishnan, M. Pál, and Z. Svitkina, "Stochastic models for budget optimization in search-based advertising," *Algorithmica*, 2010.

[42] C. Borgs, J. Chayes, N. Immorlica, K. Jain, O. Etesami, and M. Mahdian, "Dynamics of bid optimization in online advertisement auctions," in *WWW*, 2007.

[43] Y. Zhou, D. Chakrabarty, and R. Lukose, "Budget constrained bidding in keyword auctions and online knapsack problems," in *WINE*, 2008.

[44] P. Sinha and A. A. Zoltners, "The multiple-choice knapsack problem," *Operations Research*, 1979.

[45] M. Eftekhar, S. Thirumuruganathan, G. Das, and N. Koudas, "Price trade-offs in social media advertising," in *COSN*, 2014.

[46] M. Eftekhar, N. Koudas, and Y. Ganjali, "Reaching a desired set of users via different paths: an online advertising technique on micro-blogging platforms." in *EDBT*, 2015.

[47] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani, "Adwords and generalized online matching," *Journal of the ACM*, 2007.

[48] G. Goel and A. Mehta, "Online budgeted matching in random input models with applications to adwords," in *SODA*, 2008.

[49] N. R. Devanur and T. P. Hayes, "The adwords problem: online keyword matching with budgeted bidders under random permutations," in *EC*, 2009.

[50] C. Shapiro and H. R. Varian, "Versioning: the smart way to sell information," *Harvard Business Review*, 1998.

[51] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu, "Query-based data pricing," in *PODS*, 2012.

[52] M. Eftekhar, S. Thirumuruganathan, G. Das, and N. Koudas, "Price trade-offs in social media advertising," in *COSN*, 2014.

[53] M. Eftekhar, N. Koudas, and Y. Ganjali, "Reaching a desired set of users via different paths: an online advertising technique on micro-blogging platforms." in *EDBT*, 2015.

[54] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press, 2007.

[55] P. Kovács, "Minimum-cost flow algorithms: an experimental evaluation," *Optimization Methods and Software*, 2015.

[56] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *STOC*, 1990.

[57] M. Cha, A. Mislove, B. Adams, and K. P. Gummadi, "Characterizing social cascades in flickr," in *WOSN*, 2008.

[58] A. Guille, H. Hacid, C. Favre, and D. A. Zighed, "Information diffusion in online social networks: A survey," *SIGMOD Rec.*, 2013.

[59] J. B. Orlin, "Max flows in O(nm) time, or better," in *STOC*, 2013.

[60] F. Gul and E. Stacchetti, "Walrasian equilibrium with gross substitutes," *Journal of Economic theory*, 1999.

[61] H. B. Leonard, "Elicitation of honest preferences for the assignment of individuals to positions," *The Journal of Political Economy*, 1983.

[62] M. Cheung and C. Swamy, "Approximation algorithms for single-minded envy-free profit-maximization problems with limited supply," in *FOCS*, 2008.

[63] S. Im, P. Lu, and Y. Wang, "Envy-free pricing with general supply constraints," in *WINE*, 2010.

[64] M. Feldman, A. Fiat, S. Leonardi, and P. Sankowski, "Revenue maximizing envy-free multi-unit auctions with budgets," in *EC*, 2012.

[65] A. Fiat and A. Wingarten, "Envy, multi envy, and revenue maximization," in *WINE*, 2009.

[66] N. Chen, A. Ghosh, and S. Vassilvitskii, "Optimal envy-free pricing with metric substitutability," in *EC*, 2008.

[67] J. Hartline and Q. Yan, "Envy, truth, and profit," in *EC*, 2011.

[68] M.-F. Balcan, A. Blum, and Y. Mansour, "Item pricing for revenue maximization," in *EC*, 2008.

[69] P. Briest and P. Krysta, "Single-minded unlimited supply pricing on sparse instances," in *SODA*, 2006.

[70] V. Syrgkanis and J. Gehrke, "Pricing queries approximately optimally," in *CoRR*, 2015.

[71] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu, "Toward practical query pricing with querymarket," in *SIGMOD*, 2013.

[72] S. Deep and P. Koutris, "The design of arbitrage-free data pricing schemes," in *ICDT*, 2017.

[73] B.-R. Lin and D. Kifer, "On arbitrage-free pricing for general data queries," *VLDB Endow.*, 2014.

[74] C. Li and G. Miklau, "Pricing aggregate queries in a data marketplace." in *WebDB*, 2012.

[75] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, 1962.

[76] E. Mcdermid, "A 3/2-approximation algorithm for general stable marriage," in *ICALP*, 2009.

[77] Z. Király, "Linear time local approximation algorithm for maximum stable marriage," *Algorithms*, 2013.

[78] R. W. Irving, "Stable marriage and indifference," *Discrete Applied Mathematics*, 1994.

[79] D. F. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, and Y. Morita, "Hard variants of stable marriage," *Theoretical Computer Science*, 2002.

[80] K. Iwama, S. Miyazaki, Y. Morita, and D. Manlove, "Stable marriage with incomplete lists and ties," in *ICALP*, 1999.

[81] J. D. Horton and K. Kilakos, "Minimum edge dominating sets," *SIAM Journal on Discrete Mathematics*, 1993.

[82] P. G. Ipeirotis, "Analyzing the amazon mechanical turk marketplace," *XRDS: Crossroads, The ACM Magazine for Students*, 2010.

[83] D. Gusfield and R. W. Irving, *The Stable Marriage Problem: Structure and Algorithms*, 1989.

[84] K. Iwama and S. Miyazaki, "A survey of the stable marriage problem and its variants," in *ICKS*, 2008.

[85] M. M. Halldórsson, R. W. Irving, K. Iwama, D. F. Manlove, S. Miyazaki, Y. Morita, and S. Scott, "Approximability results for stable marriage problems with ties," *Theoretical Computer Science*, 2003.

[86] K. Iwama, S. Miyazaki, and N. Yamauchi, "A 1.875: approximation algorithm for the stable marriage problem," in *SODA*, 2007.

[87] L. S. Shapley and M. Shubik, "The assignment game i: The core," *International Journal of game theory*, 1971.

[88] V. P. Crawford and E. M. Knoer, "Job matching with heterogeneous firms and workers," *Econometrica*, 1981.

[89] A. S. Kelso Jr and V. P. Crawford, "Job matching, coalition formation, and gross substitutes," *Econometrica*, 1982.

[90] J. W. Hatfield and P. R. Milgrom, "Matching with contracts," *The American Economic Review*, 2005.

[91] G. Demange and D. Gale, "The strategy structure of two-sided matching markets," *Econometrica*, 1985.

[92] Y. Singer and M. Mittal, "Pricing tasks in online labor markets." in *HCOMP*, 2011.

[93] ——, "Pricing mechanisms for crowdsourcing markets," in *WWW*, 2013.

[94] G. Goel, A. Nikzad, and A. Singla, "Mechanism design for crowdsourcing markets with heterogeneous tasks," in *HCOMP*, 2014.

[95] C.-J. Ho and J. W. Vaughan, "Online task assignment in crowdsourcing markets." in *AAAI*, 2012.

[96] C.-J. Ho, S. Jabbari, and J. W. Vaughan, "Adaptive task assignment for crowdsourced classification," in *ICML*, 2013.

[97] M. Yin, Y. Chen, and Y.-A. Sun, "The effects of performance-contingent financial incentives in online labor markets," in *AAAI*, 2013.