ABSTRACT

Title of dissertation: Matching Algorithm Design in E-Commerce: Harnessing the Power of Machine Learning via Stochastic Optimization

Pan Xu, Doctor of Philosophy, 2019

Dissertation directed by: John Dickerson and Aravind Srinivasan
Department of Computer Science

Internet-based matching markets have gained great attention during the last decade, such as Internet advertising (matching keywords and advertisers), ridesharing platforms (pairing riders and drivers), crowdsourcing markets (assigning tasks to workers), online dating (pairing romantically attracted partners), etc. A fundamental challenge is the presence of *uncertainty*, which manifests in the following two ways. The first is on the arrival of agents in the system, *e.g., drivers* and *riders* in ridesharing services, *keywords* in the Internet advertising, and *online workers* in crowdsourcing markets. The second is on the outcome of interaction. For example, two users may *like* or *dislike* each other after a dating arranged by a match-making firm, a user may *click* or *not click* the link of an advertisement shown by an Ad company, to name a few.

We are now living in the era of big data, fortunately. Thus, by applying powerful machine learning techniques to huge volumes of historical data, we can often get very accurate estimates of the uncertainty in the system as described

above. Given this, the question then is as follows: *How can we exploit estimates for our benefits as a matching-policy designer*?

This dissertation aims to address this question. We have built an AI toolbox, which takes as input the estimates over uncertainty in the system, appropriate objectives (*e.g.,* maximization of the total profit, maximization of fairness, etc.), and outputs a matching policy which works well both theoretically and experimentally on those pre-specified targets. The key ingredients are two matching models: stochastic matching and online matching. We have made several foundational algorithmic progress for these two models. Additionally, we have successfully utilized these two models to harness estimates from powerful machine learning algorithms, and designed improved matching policies for various real matching markets including ridesharing, crowdsourcing, and online recommendation applications.

Matching Algorithm Design in E-Commerce: Harnessing the Power of Machine Learning via Stochastic Optimization

by

Pan Xu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Assistant Professor John Dickerson, Chair/Advisor
Professor Andrew Childs
Assistant Professor Furong Huang
Professor David Mount
Professor Aravind Srinivasan, Advisor
Professor Prakash Narayan

Acknowledgments

I am fortunate to receive lots of help and support from many people during my second Ph.D. program, whose names I would like to mention here.

First and foremost, I wholeheartedly thank my two advisors, John Dickerson, and Aravind Srinivasan, for their encouragement and guidance that have helped me overcome the most stressful time in my study. I want to express my special thanks to Aravind, who has offered me lots of tips regarding how to behave as a graceful scholar.

Besides my advisors, I would like to thank my close collaborators, Amit Chavan, Brian Brubach, and Karthik Sankararaman. We have spent countless time together exploring and discussing various research topics. In particular, I want to offer my deep appreciation to Karthik, who has helped me a lot to improve my writing and presentation skills.

My sincere gratefulness also goes to Prof. Andrew Childs, Prof. Furong Huang, Prof. David Mount, and Prof. Prakash Narayan, for being in my dissertation committee. I also want to thank Prof. Andrew Childs, Prof. Soheil Feizi, and Prof. Furong Huang, for their time attending my job practice talk and their useful constructive comments and feedbacks.

In the latter stage of my Ph.D. program, I have visited Tsinghua and Beihang universities, and we have very productive collaborations together. Thanks to my two hosts for their generous financial support: Prof. Jian Li and Prof. Yongxin Tong. Also, I feel exceedingly grateful to those fantastic graduate students there,

including Hao Cheng, Hao Fu, Yexuan Shi, and Yuxiang Zeng: their diligence and excellence impressed me a lot!

Finally, I want to thank my housemate, Lutgarda Barnachea: we have spent enjoyable six and a half years together. Also, I would like to extend my paramount appreciation to my family: my parents and sister for supporting me spiritually throughout this tough Ph.D. program.

To my wife Yulu:

I could not have gone through this trek without her love and support!

Acknowledgments

I am fortunate to receive lots of help and support from many people during my second Ph.D. program, whose names I would like to mention here.

First and foremost, I wholeheartedly thank my two advisors, John Dickerson, and Aravind Srinivasan, for their encouragement and guidance that have helped me overcome the most stressful time in my study. I want to express my special thanks to Aravind, who has offered me lots of tips regarding how to behave as a graceful scholar.

Besides my advisors, I would like to thank my close collaborators, Amit Chavan, Brian Brubach, and Karthik Sankararaman. We have spent countless time together exploring and discussing various research topics. In particular, I want to offer my deep appreciation to Karthik, who has helped me a lot to improve my writing and presentation skills.

My sincere gratefulness also goes to Prof. Andrew Childs, Prof. Furong Huang, Prof. David Mount, and Prof. Prakash Narayan, for being in my dissertation committee. I also want to thank Prof. Andrew Childs, Prof. Soheil Feizi, and Prof. Furong Huang, for their time attending my job practice talk and their useful constructive comments and feedbacks.

In the latter stage of my Ph.D. program, I have visited Tsinghua and Beihang universities, and we have very productive collaborations together. Thanks to my two hosts for their generous financial support: Prof. Jian Li and Prof. Yongxin Tong. Also, I feel exceedingly grateful to those fantastic graduate students there,

including Hao Cheng, Hao Fu, Yexuan Shi, and Yuxiang Zeng: their diligence and excellence impressed me a lot!

Finally, I want to thank my housemate, Lutgarda Barnachea: we have spent enjoyable six and a half years together. Also, I would like to extend my paramount appreciation to my family: my parents and sister for supporting me spiritually throughout this tough Ph.D. program.

Table of Contents

List of Tables

List of Figures

List of Notations and Abbreviations

$\ln(\cdot)$	the natural logarithm function
$\exp(\cdot)$	the natural exponential function
$\text{Pois}(\lambda)$	the Poisson distribution with mean λ
$[T]$	the set of $\{1, 2, \ldots, N\}$ for any natural number T
WS	the worst scenario (structure) arranged by an adversary
WLOG	without loss of generality
IID	identical and independent distributions
KIID	known identical and independent distributions
KAD	known adversarial distributions
OPT	the performance of an optimal algorithm
SM	Stochastic Matching (offline)
SHM	Stochastic Hypergraph Matching (offline)
OM	Online Matching

Chapter 1: Introduction

We are now living in an era of Internet Economy: we prefer to visit giant online marketplaces such as Amazon or eBay for shopping instead of brick-and-mortar stores; we have become more reliable on online dating websites as opposed to friends or relatives to find "The One"; We have shifted from Taxicabs to Uber and Lyft when traveling outside; we are increasingly likely to use hospitality services from Airbnb instead of hotels when making our trip plans. Unlike traditional business models, Internet companies do not directly offer services to customers. Instead, they provide convenient online platforms and facilitate "deals" (or matches) among users from typically two parties, *e.g., buyers and sellers* in Amazon, *guests and hosts* in Airbnb, *passengers and drivers* in Uber, *advertisers and impressions (users)* in Google, and *task manager and (online) workers* in crowdsourcing markets (Amazon Mechanical Turk). Notably, for most Internet companies, their main income comes exactly from these successful "deals" completed through the online platforms and thus, they try to manage as many "successful" matches as possible such that they can profit most from them.

There are several fundamental challenges in the way, however. One among them is *uncertainty*, which is inherent in modern data science. The first kind of

uncertainty lies in the arrival of agents into the system. Due to the nature of the Internet Economy, agents from at least one party arrive in a stochastic manner (see, *e.g.,* keywords in the advertising business, drivers and users in ridesharing platforms). The second is due to the inherent risk of failure associated with each "deal" (or match). Consider Amazon for example: Amazon offers a discounted price to a potential buyer while the buyer rejects the offer. Another example is displaying advertisement on Google: Google displays an ad to some online user, while the user shows no interest in the ad and chooses to ignore it. Note that in the modern pay-per-click model, Google will receive a contracted payment from each advertiser only when a user clicks on the ad (impression). Imaginably, the inherent uncertainty due to the agents' arrivals and risk of failure impose significant challenges when Internet companies try to optimize the matching policy.

Fortunately, we are now blessed with big data and powerful machine learning techniques. There is tons of past and ongoing research which deals with how to apply various deep learning techniques to estimate the uncertainty as described above. For example, [1–3] utilized neutral networks to predict the riders' arrival patterns and the drivers' arrival time in ridesharing platforms.

Given this, one of the main questions is as follows: *How to best leverage these estimates from machine learning algorithms to optimize the matching policy*? This dissertation aims to answer this question. My research has designed such an AI toolbox that it takes as input the estimates over the uncertain parameters in the system and appropriate objective functions, and outputs a matching policy which provably works well while being practically useful. Sample objectives include *maximization of*

the total profit obtained in the market, *maximization of users' satisfaction* (*e.g.,* minimization of riders' waiting time in ridesharing platforms), and *maximization of the fairness among all users*.

The rest of this chapter is organized as follows: Section 1.1 will discuss in detail the sources of uncertainty in E-Commerce, and Section 1.2 will present a brief overview of two fundamental matching models: offline and online stochastic matchings.

1.1 Challenges: Two Kinds of Uncertainties

In this section, we present a detailed discussion regarding the challenging issue of uncertainty in E-Commerce.

Uncertainty on outcomes. In many applications, we use a graph to capture the complex relations between different agents involved. In online dating, we formulate a graph where each vertex represents an online user, and each edge connects a pair of users whose attributes, as shown in their profiles, match each other well. In online advertising, we have such a natural bipartite graph: a set of advertisers and a set of users, each edge links an advertiser u and a user user v who shows interests toward the ads from u. Suppose we run an online dating company or an ads platform (*e.g.,* Google) and our goal is to arrange as many matches as possible. Unlike the traditional matching model, each edge e appearing in the final matching should go through two steps: an attempt of adding e (called *probing*) and another stochastic process for its existence. In online dating, "probing" an edge $e = (u, v)$ means

that we make a recommendation of u to v: it may end up a failure or a success (a final match), either of which occurs with a certain chance. In the online advertising business, "probing" an edge $e = (u, v)$ means that we display an ad from u to a user v: v may not click on the ads or click (a final match). In both these models, we can gain a profit w_e only when we get a final match of e. Thus, an inherent challenging problem arises: given a weighted "stochastic" graph where edges each exist with a certain (known) probability, how to probe them sequentially such that the final random matching obtained has a maximum expected weight. This question is later formalized as the (offline) Stochastic Matching problem (SM) after adding several practical constraints.

Uncertainty on arrivals. Consider the online advertising scenario as mentioned before. In practice, for an ads company, only the set of advertisers is known in advance while each user comes in a *stochastic* manner. We often try to identify the arrival of a certain type of users by their online activities in different venues such as search engines, emails, and social networking websites. Another example is crowdsourcing markets (*e.g.,* Amazon Mechanical Turk and Crowdflower), which are powerful platforms for task managers to crowdsource online workers. The problem facing a task manager can be naturally modeled by a bipartite graph as well: one side is the set of tasks while the other side is the set of online workers. Again as in online advertising, only tasks are known beforehand while the workers join the platform in a *stochastic* manner. In both applications, we have a special bipartite graph where one side of vertices is known beforehand (called *offline* side) while the other side is

revealed sequentially in a random way (called *online* side). This stochastic arrival feature often comes together with another particular requirement for our decision process: an *immediate and irrevocable* decision is required upon the arrival of an online vertex. In online advertising, once we detect the arrival of an online user, we have to display the relevant ad immediately, since users typically stay on a website for a very short time. The same applies to crowdsourcing markets: once a worker arrives online to bid a set of tasks, we should figure out the final assignment for her quickly. In short, we cannot afford to optimize our decisions after observing the full arrival sequence from the online side. A natural question is: how to optimize our real-time matching decisions to well address the stochasticity in the arrivals from the online side? This problem is later formalized as the Online Matching (OM) problem.

1.2 Overview of Two Matching Models

In this section, we present a brief summarization of the two fundamental matching models: offline and online stochastic matchings.

Stochastic Matching (offline). Stochastic matching (SM) models capture lots of real matching markets which features inherent uncertainty in the matching outcomes. Typically it has no real-time decision-making requirement. Samples include online dating, kidney exchange, public housing allocation, etc. SM takes as input a general graph used to model the network of agents in the market. Many cases, the target network studied by SM is complicated enough that we can't simply use

a bipartite graph to capture the complex structure. Consider online dating in the *bisexual* setting for example: We can imagine that some users might be romantically attracted to users of both genders; thus, we have to use a general graph to model the network.

Online Matching. Online matching models capture a large variety of matching markets which share the following two features. The first feature is that at least part of agents arrive in a dynamic way. For example, keywords (impressions) in Google, riders in Uber and Lyft, etc. The second is the real-time decision-making requirement. Upon the arrival of an online agent, we should decide immediately and irrevocably which current existing relevant agent(s) in the system we should match it to, if possible. Sample markets include Internet advertising business (Google), ridesharing platforms (Uber and Lyft), crowdsourcing markets, etc. Typically, OM takes as input a bipartite graph, where the two sets of vertices are used to model the respective groups of static and dynamic agents in the system, which are called *offline* and *online* specifically. Note that in SM only the set of offline vertices is known in advance: the other online set is revealed sequentially in a certain random way.

The main differences between SM and OM are summarized in Table 1.1. Let Challenge I, Challenge II and Challenge III denote respectively the uncertainty on outcomes, the uncertainty on arrivals, and the real-time decision-making requirement.

The rest of this dissertation is organized as follows: Chapters 2 and 3 will

	Input	Challenge I	Challenge II	Challenge III
SM	General graphs	✓	✗	✗
OM	Bipartite graphs	✓	✓	✓

Table 1.1: The main differences between SM and OM

present the two fundamental matching models respectively in detail and relevant foundational algorithmic progress; Chapter 4 will offer a specific application of online matching in ridesharing platforms; Chapter 5 will briefly discuss applications of online matching in several other real matching markets including online recommendations, taxi-dispatching services, and online task assignment platforms. We conclude the dissertation and discuss some future directions in Chapter 6.

Chapter 2: Stochastic Matching (Offline)

2.1 Preliminaries

Approximation Ratio. Owing to the computational intractability (known, conjectured, or otherwise) of problems in combinatorial optimization, a powerful approach that has developed over more than four decades is that of *approximation algorithms*, where we aim at efficiently computing solutions that are within a guaranteed factor of optimal; see, e.g., the textbooks [4, 5]. For maximization problems with a non-negative objective function, a ρ-approximation algorithm, for $\rho \geq 1$, is a

polynomial-time algorithm that always delivers a solution of value at least $1/\rho$ times optimal; for randomized algorithms, the expected solution-value output should be at least $1/\rho$ times optimal, where this expectation is over the internal randomization of the algorithm. In the context of stochastic optimization (maximization), we need to be a little more careful, since the objective function value is random due to the randomness in the stochastic input; letting OPT denote the maximum-possible expected objective-function value over all possible terminating algorithms with no constraint on the running time, a ρ-approximation algorithm is one that outputs a solution of expected value at least OPT/ρ, where the expectation is over the uncertainty of the input, and over any internal randomization of the algorithm. This will be the notion of approximation employed in the next sections, where we discuss our approximation algorithms for stochastic matching in a model that posits the uncertain data as being independent with known distributions.

Random permutation and FKG inequality. We will often consider a uniformly random permutation π on a set of items $I = \{e_1, e_2, \ldots, e_\ell\}$. We can assume that π is chosen as follows: for each item e, we pick independently and uniformly at random a real number $\pi(e) = a_e \in [0, 1]$, and then sort these in increasing order to obtain π. Note that we abuse notation by letting π denote both the permutation and the reals chosen; however, this choice will be clear from the context.

In the context of such a randomly-chosen permutation π of our set I, the FKG inequality [6] will be quite useful to us, as follows. A Boolean function $f : \{0,1\}^t \to \{0,1\}$ is termed *increasing* if for each input $x = (x_1, x_2, \ldots, x_t) \in \{0,1\}^t$, turning

any x_i from 0 to 1 cannot change the value of $f(x)$ from 1 to 0; i.e., the value of f either remains unchanged by this bit-flip, or increases from 0 to 1. Similarly, $g : \{0,1\}^t \to \{0,1\}$ is *decreasing* if for each $x = (x_1, x_2, \ldots, x_t) \in \{0,1\}^t$, turning any x_i from 1 to 0 cannot change the value of $g(x)$ from 1 to 0. The FKG inequality states that if we have *independent* random bits $R_1, R_2, \ldots, R_t$, then for all k and for all increasing or all decreasing $f_1, f_2, \ldots, f_k$ that map $\{0,1\}^t$ to $\{0,1\}$,

$$\Pr\left[\bigwedge_{i=1}^{k} \Big(f_i(R_1, R_2, \ldots, R_t) = 1\Big)\right] \geq \prod_{i=1}^{k} \mathbb{E}\big[f_i(R_1, R_2, \ldots, R_t)\big];$$

In our analyses, we will often condition on an event A of the form "$\pi(e) = x$" (where π is our random permutation as above and $x \in [0,1]$), and will need to lower-bound certain probabilities of the form $\Pr\left[\bigwedge_{i=1}^{k} B_i \mid A\right]$; the FKG inequality is quite useful if these events B_i have a certain structure [7, 8]. For all $f \in I$ such that $f \neq e$, define a random bit R_f that is 1 if $\pi(f) \leq x$, and 0 otherwise; note that even conditional on the event A, these R_f are all independent. Now, if the B_i are Boolean functions of the tuple of bits R_f such that the B_i are all increasing or all decreasing, then the FKG inequality applied to the space where we condition on A, yields

$$\Pr\left[\bigwedge_{i=1}^{k} B_i \mid A\right] \geq \prod_{i=1}^{k} \Pr[B_i \mid A]. \tag{2.1}$$

Chernoff-Hoeffding bound. In this chapter, we will also make use of the following form of the Chernoff-Hoeffding bound [9]:

Definition 2.1.1 (Chernoff-Hoeffding Bound)**.** *Let $X_1, \ldots, X_n$ be n independent*

random variables with $0 \leq X_i \leq 1$. *Let* $X = X_1 + \ldots + X_n$ *and* $\mu = \mathbb{E}[X]$. *Then for any* $\epsilon > 0$,

$$\Pr[X \geq (1+\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2}{2+\epsilon}\mu\right), \text{ and}$$
$$\Pr[X \leq (1-\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2}{2}\mu\right)$$

2.2 Overview and Related Work

The Stochastic Matching (unweighted) was first introduced in Chen *et al.* [10]. Suppose we are given an undirected and unweighted graph $G = (V, E)$; each vertex v has an integral patience parameter, say t_v, and each edge has a presence probability p_e; at any step of the algorithm, only an edge $e = (u, v) \in E$ such that $t_u > 0$ and $t_v > 0$ can be probed. Upon probing such an edge e, one of the following happens: (1) with probability p_e, e exists; we are forced to add e into the final matching while both u and v are removed from G; or (2) with probability $(1 - p_e)$, e does not exist; e is removed while t_u and t_v are reduced by 1. All these edge-existence events are independent. Our goal is to find an adaptive strategy of probing edges such that the expected cardinality of the final matching is maximized. As for the weighted version, each edge e has a weight $w_e \geq 0$ and the target is updated as finding a matching with a maximum expected total weight.

As for the above unweighted stochastic matching on a general graph, Chen *et al.* [10] gave a simple greedy algorithm which achieves an approximation ratio of 4. Later, it was shown in [11] that greedy achieves an improved ratio of 2 actually.

Bansal *et al.* [12] was the first to consider the weighted version of Stochastic Matching. They proposed the following benchmark LP to upper bound the OPT, which represents the performance of an optimal algorithm[1]. For each vertex v, let $\partial(v)$ be the set of edges incident to v. Let y_e be the probability that edge $e = (u, v)$ gets probed in an OPT, and $x_e = y_e p_e$ the probability that e gets matched.

$$\text{maximize} \sum_{e \in E} w_e x_e \tag{2.2}$$

$$\text{subject to} \sum_{e \in \partial(v)} x_e \leq 1 \qquad \forall v \in V \tag{2.3}$$

$$\sum_{e \in \partial(v)} y_e \leq t_v \qquad \forall v \in V \tag{2.4}$$

$$x_e = y_e p_e \geq 0, \ y_e \leq 1 \qquad \forall e \in E \tag{2.5}$$

Note that the above LP is inspired by the work of Dean *et al.* [13], which was the first work to use an LP to upper bound the performance of an optimal adaptive algorithm for the stochastic knapsack problem. Bansal *et al.* [12] considered weighted stochastic matching on a bipartite graph and gave a 3-approximation algorithm as follows.

From [14], we see that $\{Y_e\}$ at the end of Step 2 of Algorithm 1 satisfies these three properties: (1) $\mathbb{E}[Y_e] = y_e$ for each e; (2) $\{Y_e\}$ is negatively correlated; (3) For each v, $\sum_{e \in \partial(v)} Y_e \leq \lceil \sum_{e \in \partial(v)} y_e \rceil$. Note that the last property guarantees that we can ignore patience constraint on each vertex during the random probing on

[1] We use OPT to refer to an optimal algorithm as well when the context is clear.

[2] Our statement here is slightly different from the original version.

Algorithm 1: Stochastic matching on a bipartite graph [12][2]

1 Solve the LP (2.2) and let $\{y_e\}$ be an optimal solution.
2 Apply dependent rounding [14] to $\{y_e\}$ and let $\{Y_e\}$ be the random integral vector returned.
3 Follow a random order π over all rounded edges e with $Y_e = 1$ and check if each e is safe, *i.e.,* if probing e will not violate any patience and matching constraint.
4 If an edge e is safe, then probe it; otherwise, skip it.

Step 3 and 4. [12] shows that Algorithm 1 achieves an approximation ratio of 3 by using the benchmark LP (2.2). Based on this result, they continued to present a 4-approximation algorithm for a general graph.

Adamczyk *et al.* [15] considered the same weighted stochastic matching problem and improved the ratio of 3 and 4 to 2.845 and 3.709 respectively on a bipartite and general graph. They used the same benchmark LP and processed the optimal solution in the same way as shown in Step 1 and 2 of Algorithm 1. The key idea there is instead of choosing a random order, they chose a weighted permutation over all rounded edges such that edges with small p_e values have a higher chance to come earlier in the order than those with large p_e values. In this way, they ended up with an algorithm which favors edges with small p_e values. After that, they designed another algorithm which favors edges with large p_e values and then combined the two to get the improvement.

In the setting of stochastic matching, the requirement of adding each existing edge into the final matching is typically called as *query-commitment*. There are several other work which considered stochastic matching in the context of kidney exchange without that constraint (*e.g.*, [16–18]). In particular, [16] considered

stochastic matching without query-commitment assuming each vertex has a patience of 2 while [17] considered the same model assuming each vertex has a patience bounded by a constant, *i.e.,* we are allowed to query a constant number of edges for each vertex.

Another interesting related work is due to [19], which considered the weighted stochastic matching with query-commitment but without patience constraints. Putting into our context, each vertex has a patience of infinity, *i.e.,* it has no restrictions on the number of probes. They gave an algorithm achieving at least a fraction of 0.573 of the OPT.

2.3 Main Techniques and Our Results

We focus on the case of weighted stochastic matching with query-commitment on a general graph. Both of [12] and [15] attacked this case by reducing a general graph to a bipartite one (randomly splitting all vertices into two parties) and then invoke the algorithm for the bipartite case as a black box. In this paper, we try to treat a general graph in a direct way while the main issue is: We can not apply dependent rounding as before to round a fractional solution from the benchmark LP to an integral one such that patience constraint can be ignored afterwards. We overcome this issue through a careful analysis of our algorithm and manage to identify an elegant structure for the worst scenario. Our algorithm is pretty simple, which states as follows:

Theorem 2.3.1 ([20]). *The approximation ratio achieved by Algorithm 2 is* 3.22

Algorithm 2: Stochastic matching on a general graph [20]

1 Choose a random permutation π on E.
2 For each edge $e \in E$, generate a random bit $Y_e = 1$ independently with probability y_e. Let E' be the set of edges with $Y_e = 1$.
3 Follow the random order π to inspect edges in E'
4 If an edge e is safe, then probe it; otherwise, skip it.

for the weighted stochastic matching problem on a general graph. What is more, the ratio achieved by Algorithm 2 is improved to 2.67 *under the same setting when patience constraints are allowed to be violated by* 1.

After a detailed analysis of Algorithm 2, we can rigorously prove that the edge $e = (u, v)$ achieving the worst performance has the following setting, which is referred to as the Worst Scenario (WS): (1) $y_e \sim 0$ and $t_u = t_v = 2$; (2) both of $E(u)$ and $E(v)$ have such a structure: one big edge f with $y_f = p_f = 1$ and a bunch of N tiny edges each has $y_f = 1/N$ and $p_f = 0$ while $N \to \infty$. Note that in the WS, the big edge tightens the matching constraint such that $\sum_{e \in \partial(u)} y_e p_e = 1$ while the bunch of all rest tiny edges help tighten the patience constraint in the way that $\sum_{e \in \partial(u)} y_e = 2$. We can verify that in the WS, edge e will be probed with probability exactly equal to $0.301 y_e$ in Algorithm 2, which leads to the final ratio.

2.4 Proofs of Main Results in Theorem 2.3.1

In this section, we present proofs in details for the first part of results in Theorem 2.3.1, which states in the following proposition. For the rest of proofs, please check Section 5 of [20].

Proposition 2.4.1. *Algorithm 2 achieves an approximation ratio at least* 3.22 *for*

the weighted stochastic matching problem on a general graph.

To analyze the performance of our algorithm, we conduct an edge-by-edge analysis. Recall that $y_e p_e$ is the probability that e is matched in LP (2.2), and the optimal value of the LP is exactly $\sum_{e \in E} w_e p_e y_e$. Let ALG be the expected weight output by Algorithm 2. We have that

$$\begin{aligned}
\mathbb{E}[\text{ALG}] &= \sum_{e \in E} w_e p_e \cdot \Pr[e \in E'] \cdot \Pr[e \text{ gets probed} | e \in E'] \\
&= \sum_{e \in E} w_e p_e y_e \cdot \Pr[e \text{ gets probed} | e \in E'] \\
&\geq \sum_{e \in E} w_e p_e y_e \lambda
\end{aligned}$$

The last inequality is obtained by assuming $\Pr[e \text{ gets probed} | e \in E'] \geq \lambda$. This gives us a λ-approximation algorithm.

The subsequent discussion focuses on how to lower-bound the value of λ. Consider a specific edge $e = e(u, v)$, and let $E(u)$ be the set of edges incident to u *excluding* e itself, *i.e.,* $E(u) = \partial(u) \setminus \{e\}$. Let $\pi(e) = x, 0 < x < 1$. Conditioned on $\pi(e) = x$, with $0 < x < 1$, and $Y_e = 1$, let $\mathcal{P}_u$ be the probability that e is *not blocked* by any of the edges in $E(u)$ in Algorithm 2. We say that e is *blocked* by some edge f in $E(u)$ if f gets matched or the patience constraint of u gets tight resulting from probing f (*i.e.,* $t_u = 0$). We assume without loss of generality that $|E(u)| \geq t_u$, otherwise the patience constraint for node u is redundant.

A little thought gives us the following lower bound on $\mathcal{P}_u$:

$$\mathcal{P}_u \geq P_u = \sum_{S \subseteq E(u), |S| \leq t_u - 1} x^{|S|} \prod_{f \in S} y_f (1 - p_f) \prod_{f \notin S} (1 - x y_f) \tag{2.6}$$

To see why this is true, let Y'_f (for any $f \in E(u)$) be the indicator random variable that is 1 if and only if f gets matched when probed, i.e., $\Pr[Y'_f = 1] = p_f$. For each $S \subseteq E(u)$ such that $|S| \leq t_u - 1$, we associate an event A_S that happens when both of the following conditions are met: (1) Each edge $f \in S$ falls before e in π with $Y_f = 1$ and $Y'_f = 0$; and (2) each edge $f \notin S$ either falls after e in π or $Y_f = 0$. We can see that this event guarantees that e will not be blocked by any edge of S. Thus, $\mathcal{P}_u$ should be at least the probability that one or more of A_S happen, which is exactly P_u.

Next, we focus on *adverserial configurations* of $E(u)$, i.e, how are the edges in $E(u)$ arranged so as to minimize the value of P_u subject to the constraints: (1) $\sum_{f \in E(u)} y_f p_f \leq 1$, (2) $\sum_{f \in E(u)} y_f \leq t_u$ and (3) $0 \leq y_f, p_f \leq 1$ for each $f \in E(u)$. Here we view x as a (given) parameter. We denote such adversarial configurations of $E(u)$ as the worst-case structure (WS) of $E(u)$. Notice that we give the (hypothetical) adversary extra power of manipulating the values of p_f and number of edges in $E(u)$, both of which are actually part of the input.

Lemma 2.4.1. *In WS, there will be at most one edge with $p_f = 1$ and at most one edge with $0 < p_f < 1$. All other edges must have $p_f = 0$.*

Proof. We prove by contradiction. Assume there are two edges, say $p_1 = p_2 = 1$ in WS. Then, $y_1 + y_2 \leq 1$ since $\sum_i y_i p_i \leq 1$. We perturb the current configuration

as follows: merge the two edges into a single edge e_3 where $y_3 = y_1 + y_2$ and $p_3 = 1$. After this perturbation, both values, $\sum_{f \in E(u)} y_f p_f$ and $\sum_{f \in E(u)} y_f$, remain unchanged. Thus, both the matching and patience constraints are maintained at u, and our perturbation gives a feasible configuration.

The change brought by this perturbation to the value P_u is as follows: for each non-zero term in P_u associated with some $S \subseteq E(u)$ where $e_1 \notin S, e_2 \notin S$, the term $(1 - xy_1)(1 - xy_2)$ will be replaced with $(1 - x(y_1 + y_2))$, which results in a strictly lower value of P_u. This is a contradiction.

Now assume there are two edges a, b with $0 < p_a, p_b < 1$ in WS. Consider the following perturbation: for some small $\varepsilon \neq 0$, set $p'_a = p_a + \varepsilon/y_a$ and $p'_b = p_b - \varepsilon/y_b$. After this perturbation, both of $\sum_{f \in E(u)} y_f p_f$ and $\sum_{f \in E(u)} y_f$ remain unchanged and the perturbed configuration is still feasible.

Let $f(\varepsilon)$ be the value of P_u after this update. In the expression of P_u, the terms contributing to ε^2 must be those associated with S where $a, b \in S$. Notice that

$$(1 - p'_a)(1 - p'_b) = (1 - p_a - \varepsilon/y_a)(1 - p_b + \varepsilon/y_b)$$

has a negative coefficient of ε^2, implying that the second derivative f'' is negative. Therefore we can always find a non-zero value of ε to make P_u strictly smaller. Again a contradiction. □

Let $E_1(u)$ and $E_0(u)$ be the set of edges in the WS which have $p_f = 1$ and $p_f = 0$ respectively. Let a be the potential edge taking a floating value, $0 < p_a < 1$. Lemma 2.4.1 tells us $E_1(u)$ contains at most one such edge in the WS. Let $A =$

$\sum_{f \in E_1(u)} y_f$.

Based on Lemma 2.4.1, we can update the expression of P_u as

$$P_u = (1 - xA)(1 - xy_a) \Pr[Z_u \leq t_u - 1] + (1 - xA)xy_a(1 - p_a) \Pr[Z_u \leq t_u - 2] \quad (2.7)$$

where $Z_u = \sum_{f \in E_0(u)} Z_f$ and the $(Z_f : f \in E_0(u))$ are independent Bernoulli random variables with $\Pr[Z_f = 1] = xy_f, \forall f \in E_0(u)$. (We are abusing notation in the equation $Z_u = \sum_{f \in E_0(u)} Z_f$ by reusing the symbol Z for the left hand side and right hand side; this will not cause any confusion as the identity of Z will always be clear from the context.)

Lemma 2.4.2. *In WS, $p_a = 0$.*

We defer the proof of the above lemma to Appendix A of [20]. From Lemma 2.4.2, we can claim that there is no edge f which has $p_f \in (0, 1)$. Thus, we can further simplify the expression of P_u in equation (2.7) as

$$P_u = (1 - xA) \Pr[Z_u \leq t_u - 1]. \quad (2.8)$$

Lemma 2.4.3 reveals additional structure of the WS.

Lemma 2.4.3. *In WS, we have $A = 1$ and $Z_u \sim \text{Pois}(x(t_u - 1))$.*

Proof. We show $A = 1$ by contradiction. Assume $A < 1$ in WS. Notice that $E_0(u)$ is non-empty since $\mathbb{E}[Z_u] = \sum_{f \in E_0(u)} \mathbb{E}[Z_f] = x(t_u - A) > 0$. Next, consider an

arbitrary edge $f \in E_0(u)$ with $y_f \in (0,1]$. Let $Z'_u = Z_u - Z_f$. Then,

$$\begin{aligned} P_u &= (1 - xA)\Pr[Z_u \leq t_u - 1] \\ &= (1 - xA)\left(\Pr[Z'_u \leq t_u - 2] + (1 - y_f x)\Pr[Z'_u = t_u - 1]\right) \\ &= (1 - xA)\Pr[Z'_u \leq t_u - 2] + (1 - (y_f + A)x + y_f A x^2)\Pr[Z'_u = t_u - 1]. \end{aligned}$$

We have two cases:

(i) $A < y_f$. In this case, P_u can be decreased by interchanging the values A and y_f.

(ii) $A \geq y_f$. In this case, P_u can be decreased by perturbing as $A' = A + \varepsilon$ and $y'_f = y_f - \varepsilon$ for some small $\epsilon > 0$.

Notice that in case **(i)**, after interchanging the values A and y_f, the value $\sum_{f \in E(u)} y_f p_f$ will change from A to y_f and thus is at most 1, since $y_f \leq 1$ for each $f \in E$. As for case **(ii)**, the value $\sum_{f \in E(u)} y_f p_f$ will change from A to $A + \epsilon$. Since $A < 1$, we can always find a $\epsilon > 0$ such that $A + \epsilon \leq 1$ such that the constraint $\sum_{f \in E(u)} y_f p_f \leq 1$ is maintained. Thus, the value $(A + y_f)$ remains unchanged after perturbation in both cases and the constraint $\sum_{f \in E(u)} y_f \leq t_u$ is maintained. In either case, we end up at a feasible configuration in which P_u is strictly lower than that in WS. This yields a contradiction.

We defer the proof of the second part of this lemma, $Z_u \sim \text{Pois}(x(t_u - 1))$, to Appendix A of [20]. □

At this point, we have all the ingredients to prove Proposition 2.4.1.

Proof. We have $\Pr[e \text{ gets probed } | Y_e = 1] = \int_0^1 \mathcal{P}_u \mathcal{P}_v dx \geq \int_0^1 P_u P_v dx$, *i.e.,* at least

$$H(t_u, t_v) \doteq \int_0^1 (1-x)^2 \Pr[Z_u \leq t_u - 1] \Pr[Z_v \leq t_v - 1] dx,$$

where $Z_u \sim \text{Pois}(x(t_u - 1))$ and $Z_v \sim \text{Pois}(x(t_v - 1))$. We verified that the above expression has a minimum value of $0.31016 = 1/3.224$ at $t_u = t_v = 2$. All our numerical computations were done on Mathematica 10 with precision at least up to the fourth digit after the decimal point. We split the whole verifications into the following three cases: (1) $1 \leq t_u, t_v \leq 20$; (2) $t_u, t_v \geq 20$ and (3) $1 \leq t_u \leq 20$ while $t_v \geq 20$. Notice that $H(t_u, t_v)$ is symmetric in the two variables and thus our verifications are complete.

- For $1 \leq t_u, t_v \leq 20$, we can numerically verify that $H(t_u, t_v)$ achieves its minimum value of $0.31016 = 1/3.224$ at $t_u = t_v = 2$.

- For $t_u, t_v \geq 20$, the Chernoff bound from Definition 2.1.1 implies that $H(t_u, t_v)$ should be at least

$$\int_0^1 (1-x)^2 \left[1 - \exp\left(\frac{-\epsilon^2 x(t_u - 1)}{2+\epsilon}\right)\right] \left[1 - \exp\left(\frac{-\epsilon^2 x(t_v - 1)}{2+\epsilon}\right)\right] dx,$$

  where $\epsilon = \epsilon(x) = \frac{1}{x} - 1$; by plugging in $t_u = t_v = 20$, we can verify numerically that this integral is at least 0.316324.

- Similarly, for $1 \leq t_u \leq 20$ while $t_v \geq 20$, we can verify numerically (by checking all integers $1 \leq t_u \leq 20$) that with $\epsilon = \frac{1}{x} - 1$,

$$H(t_u, t_v) \;\geq\; \int_0^1 (1-x)^2 \Pr(Z_u \leq t_u - 1) \left[1 - \exp\left(\frac{-\epsilon^2 x(20-1)}{2+\epsilon}\right)\right] dx,$$

which is at least 0.312253.

This establishes the key claim that $\Pr[e \text{ gets probed } | Y_e = 1] \geq 0.3101$ for each $e \in E$. □

2.5 Extension to Stochastic Hypergraph Matching

We now consider Stochastic Hypergraph Matching (SHM) on a k-uniform hypergraph, *i.e.,* a hypergraph where all edges have size exactly k. However, unlike before, we do not consider patience constraints (the work of [7] proceeds similarly). The following LP can be obtained by naturally extending the LP in (2.2), where $\partial(v)$ denotes the set of hyperedges incident to v:

$$\max \sum_{e \in E} w_e y_e p_e \;\; s.t. \;\; \sum_{e \in \partial(v)} y_e p_e \leq 1, \forall v \in V; \;\; 0 \leq y_e \leq 1, \forall e \in E \tag{2.9}$$

Theorem 2.5.1 and Theorem 2.5.2 improve upon the $(k+1)$–approximation of [7] for weighted matching in k-uniform hypergraphs. Both of these algorithms classify the hyperedges as "small" or "large" based on the LP values, and treat each group separately. The difference is as follows. The algorithm of Theorem 2.5.1 attenuates the small edges to boost the performance of large edges; the algorithm of Theorem 2.5.2 uses a "weighted permutation" of the hyperedges such that each

large edge has a higher chance to fall behind a small edge. Although Theorem 2.5.2 is asymptotically better, we present both theorems since their ideas can be useful elsewhere.

Note that the LP-based methods of [7] and ours cannot in general do better than $k - 1 + 1/k$ [21]; hence, we are close to optimal for LP-based approaches.

Theorem 2.5.1. *There is a $(k + \frac{1}{2} + o(1))$–approximation algorithm for SHM on a k-uniform hypergraph, where the "$o(1)$" term is a function of k that goes to zero as k becomes large.*

Theorem 2.5.2. *For any given $\epsilon > 0$, there is a $(k + \epsilon + o(1))$–approximation algorithm for SHM on a k-uniform hypergraph, where the "$o(1)$" term is a function of k that goes to zero as k becomes large.*

We next resent the algorithms and proofs for these two theorems.

2.6 Proof of Theorem 2.5.1

In this section, we present an algorithm achieving an approximation ratio at least $(k + 1/2 + o(1))$. For notational convenience, let $\{y_e\}$ be an optimal solution to LP (2.9). At a high level, our algorithm proceeds according to the outline below. Let $c \geq 1/2$ be a parameter, which will be optimized at $1/2$ later.

1. Divide the edges into two sets, the "small" edge set $E_S = \{e | y_e p_e \leq c\}$, and the "large" edge set $E_L = E \setminus E_S$.

2. Choose a random permutation π of E_S.

3. Sample each edge $e \in E_S$ with probability y_e, independent of other edges. Let E'_S be the set of sampled edges.

4. Follow the order π to inspect if each (small) edge $e \in E'_S$ is safe or not. If e is safe, probe it with probability h_e; otherwise, skip it. Here $0 < h_e \leq 1$ is a parameter to be determined later.

5. After inspecting all small edges, remove all the unsafe large edges from E_L, and probe others with probability 1 (in arbitrary order).

Roughly speaking, an edge e being "safe" means that none of the edges in the neighborhood of e are matched. Later, we will give a definition that is both stronger and *exactly* computable. Based on the new definition, we compute an *attenuation factor* h_e for each $e \in E_S$, such that at the end of the algorithm, e is probed with probability *exactly equal* to y_e/λ. Here, $\lambda \geq 1$ is our target approximation ratio. All that remains is to analyze the performance of each large edge $e \in E_L$ and show that e is probed with probability at least y_e/λ. This, then, will give us a λ–approximation algorithm.

We redefine the notion of a small edge e being safe. Suppose π is the random order on E_S and $\pi(e) = x, 0 < x < 1$. Let $N_S[e]$ be the set of small edges in the neighborhood of e. For each $f \in N_S[e]$, let X_f, Y_f, Z_f be three random variables such that: $X_f = 1$ if f falls before e in π, $Y_f = 1$ if $f \in E'_S$ and $Z_f = 1$ if f exists in the hypergraph when probed. Note that the collection of random variables $\{X_f, Y_f, Z_f | f \in N_S[e]\}$ are mutually independent. For each $f \in N_S[e]$, let A_f be the event that $(X_f + Y_f + Z_f \leq 2)$ and $\mathsf{S}_e = \wedge_{f \in N_S[e]} A_f$. *We define e to be safe iff* S_e

happens. Lemma 2.6.1 computes the probability that a small edge e is safe in our algorithm.

Lemma 2.6.1.

$$\Pr[\mathcal{S}_e] = \int_0^1 \Pr[\mathcal{S}_e | \pi(e) = x] dx = \int_0^1 \prod_{f \in N_S[e]} (1 - x y_f p_f) dx. \tag{2.10}$$

Proof. By definition, $\Pr[X_f = 1 | \pi(e) = x] = x$. Note that $\Pr[Y_f = 1] = y_f$, $\Pr[Z_f = 1] = p_f$, and that these two random variables are independent of $\pi(e)$. Thus, given $\pi(e) = x$, A_f will occur with probability $(1 - x y_f p_f)$. Since the A_f are independent for $f \in N_S[e]$, the proof is completed. □

Here are two interesting points for the event $\mathcal{S}_e$: (1) When $\mathcal{S}_e$ happens, e must be safe according to our initial definition, *i.e.,* none of the edges in its neighborhood get matched; the contrary is not true. Thus the new definition is more strict. (2) On checking e in the algorithm, we might not know if $\mathcal{S}_e$ occurs or not due to some missing Z_f for $f \in N_S[e]$. For instance, suppose some $f \in N_S[e]$ gets blocked by some small edge $f' \in N_S[f]$ while $X_f = Y_f = 1$. In this case, we do not know the value of Z_f since f will not be probed. In order to continue our algorithm, we simulate Z_f by generating a random bit $Z_f = 1$ with probability p_f and $Z_f = 0$ otherwise. Notice that if $Z_f = 1$, we will view e as not safe and will not probe it, even though it might be safe according to our initial definition.

The full picture can be seen in Algorithm 3.

Algorithm 3: Stochastic hypergraph matching on a k-uniform hypergraph

Initially all edges are safe.
Split the edges into two sets, the "small" edge set $E_S = \{e | y_e p_e \leq c\}$ and the "large" edge set $E_L = E \setminus E_S$ where $c \geq 1/2$.
Choose a random permutation π on E_S.
For each $e \in E_S$, generate a random bit $Y_e = 1$ with probability y_e. Let E'_S be the set of (small) edges with $Y_e = 1$.
Follow the random order π to check if $\mathcal{S}_e$ happens or not for each $e \in E'_S$.
    if $\mathcal{S}_e$ *happens* **then**
        Probe e with probability h_e.
        if *e is matched (exists)* **then**
            Set $Z_e = 1$ and mark all its neighboring large edges as unsafe.
        else
            Set $Z_e = 0$.
    else
        Generate a random bit $Z_e = 1$ with probability p_e.
    Probe each safe large edge with probability 1 in an arbitrary order.

2.6.1 Analysis of Algorithm 3

We first analyze the performance of a small edge. For each edge $e \in E_S$,

$$\Pr[e \text{ gets probed}] = y_e h_e \Pr[e \text{ is safe} | Y_e = 1] = y_e h_e \Pr[\mathcal{S}_e].$$

To ensure that each small edge $e \in E_S$ is probed with probability *equal* to y_e/λ, we can set $h_e = 1/(\lambda \Pr[\mathcal{S}_e])$ if we can ensure that $\Pr[\mathcal{S}_e] \geq 1/\lambda$. The following lemma states that this goal is achievable. Recall that $c \geq 1/2$ is the threshold such that an edge e is small iff $y_e p_e \leq c$.

Lemma 2.6.2.

$$\Pr[\mathcal{S}_e] \geq \frac{1 - (1-c)^{k/c+1}}{k+c}$$

Proof. Consider a small edge e, say $e = (v_1, v_2, \cdots, v_k)$ and $\pi_e = x$. Let $E(v_i)$ be the set of edges incident to v_i excluding e itself. Notice that $N_S[e] = \cup_{i=1}^{k} E(v_i)$. Therefore by Lemma 2.6.1, we have

$$\Pr[\mathcal{S}_e | \pi(e) = x] = \prod_{f \in N_S[e]} (1 - x y_f p_f) \geq \prod_{i=1}^{k} \prod_{f \in E(v_i)} (1 - x y_f p_f)$$

From the proof of Lemma 2.7.2, we see that $\prod_{f \in E(v_i)} (1 - x y_f p_f) \geq (1 - xc)^{1/c}$ for each $1 \leq i \leq k$. Thus by an application of the FKG inequality as in (2.1), we get that $\Pr[\mathcal{S}_e | \pi(e) = x] \geq (1 - xc)^{k/c}$.

Integrating over $[0, 1]$, we get

$$\Pr[\mathcal{S}_e] = \int_0^1 \Pr[\mathcal{S}_e | \pi(e) = x] \geq \frac{1 - (1 - c)^{k/c+1}}{k + c} dx.$$

□

At this point, we have all the ingredients to prove Theorem 2.5.1.

Proof. For small edges, Lemma 2.6.2 gives us a sufficient condition to guarantee that each small edge is probed with probability *exactly equal* to y_e/λ, *i.e.,*

$$\Pr[\mathcal{S}_e] \geq \frac{1 - (1 - c)^{k/c+1}}{k + c} \geq \frac{1}{\lambda}. \tag{2.11}$$

We now analyze the performance of large edges in SM_3. For each $e \in E_L$, let $\mathcal{S}_e$ be the event that e is safe when considered in SM_3, *i.e.,* none of small edges in the neighbor of e gets matched. Since each small edge f gets matched with probability

equal to $\frac{y_f p_f}{\lambda}$, we have that for each large item $e \in E_L$, $\Pr[\mathcal{S}_e] \geq 1 - \frac{(1-c)k}{\lambda}$ by applying the union bound.

In order to ensure that each large edge gets probed with probability at least $\frac{y_e}{\lambda}$, it suffices to set

$$\Pr[\mathcal{S}_e] \geq 1 - \frac{(1-c)k}{\lambda} \geq \frac{1}{\lambda} \tag{2.12}$$

Observe that for a small edge e, the lower bound of $\Pr[\mathcal{S}_e]$ from (2.11) is a decreasing function of c, while for a large edge e, the lower bound in (2.12) is an increasing function of c. Thus to find the optimal value for λ, we choose c that maximizes the minimum of the two,

$$1 - \frac{(1-c)k}{\lambda} = \frac{1-(1-c)^{k/c+1}}{k+c} = \frac{1}{\lambda}$$

The solution above is $c = \frac{1}{k+1} + o(\frac{1}{k+1})$. However, this is not feasible because by assumption, $c \geq 1/2$. Thus the optimal c^* equals $1/2$, in which case $\frac{1}{\lambda} = \frac{1}{k+1/2} - O(1/(k4^k))$, and each small edge is safe to probe with probability $\frac{1}{\lambda}$ while each large edge is safe with probability $\frac{1}{2} + o(1/k)$. □

2.7 Proof of Theorem 2.5.2

In this section, we present a simple randomized algorithm that achieves an approximation ratio of $(k + \epsilon + o(1))$ for stochastic matching on a k-uniform hypergraph, where $\epsilon > 0$ is a given constant.

Let (x, y) be an optimal solution to the LP (2.9). Assume w.l.o.g. $1/\epsilon = L$

where N is an integer. Let a be a constant such that $1 - 1/L < a < 1$. We say an edge e is "large" if $y_e p_e > 1/L$; otherwise we call e "small". For each small edge e, we draw a random real number x_e uniformly from $[0, 1]$. For each large edge e, we draw a random real number x_e from $[0, \delta]$ with density a and from $(\delta, 1]$ with density $(1 - a\delta)/(1 - \delta)$, where $\delta = \min(1, L(1 - a^{1/(L-1)})$. Then we derive a random permutation π by sorting $\{x_e, e \in E\}$ in increasing order. Assuming L is sufficiently large, the value δ is at most $1/L + o(1/L)$. Notice that L, a and δ are all fixed constants. Based on π, we sketch our randomized algorithm as follows. Here we say an edge is *safe* iff none of its neighbors gets matched.

Algorithm 4: Stochastic hypergraph matching on a k-uniform hypergraph

1 Initially all edges are safe.
2 Follow the random order π to check each edge $e \in E$ if it is safe or not.
3 If e is safe, then probe it with probability y_e; otherwise, skip it.

The lemmas below are useful for the proof of Theorem 2.5.2.

Lemma 2.7.1. *For any $c > 1/L$ and $0 < x < \delta$, we have*

$$1 - axc > (1 - x/L)^{cL}$$

Proof. Define $F(x) = 1 - axc - (1 - x/L)^{cL}$. We can verify that: (1) $F(0) = 0$, and (2) $F'(x) > 0$ for any $0 \le x < \delta$. This gives the desired result.

□

For each edge e, define $c_e = y_e p_e$. Consider an edge $e = (v_1, v_2, \cdots, v_k)$. Suppose $y_e p_e = c_e < 1 - 1/L$ and $x_e = x, 0 < x < \delta$. For each $1 \le i \le k$, let $E(v_i)$

denote the set of edges incident to v_i excluding e itself. Denote by $\mathcal{S}_i$ the event that none of the edges in $E(v_i)$ come before e and get matched.

Lemma 2.7.2.

$$\Pr[\mathcal{S}_i] \geq (1 - x/L)^{(1-c_e)L}.$$

Proof. From LP (2.9), we see $\sum_{f \in E(v_i)} y_f p_f \leq 1 - c_e$. Let A and B be the set of small edges and large edges in $E(v_i)$ respectively. Observe that

$$\Pr[\mathcal{S}_i] \geq \prod_{f \in A} (1 - x c_f) \prod_{f \in B} (1 - a x c_f). \tag{2.13}$$

Now we investigate how an adversary can minimize the right hand side (RHS) of (2.13) subject to the constraint $\sum_{f \in E(v_i)} y_f p_f \leq 1 - c_e$. By Lemma 2.7.1, the adversary will not put any large edge f in B: otherwise it could further decrease the RHS by splitting f into $c_f L$ copies of small edges f' with each $c_{f'} = 1/L$ while maintaining the constraint. Thus the adversary aims to minimize $\prod_{f \in A}(1 - x c_f)$ subject to $\sum_{f \in E(v_i)} c_f \leq 1 - c_e$ with $0 \leq c_f \leq 1/L$ for each f. By applying a local perturbation as in Lemma 2.4.1, the RHS will be minimized when there are $(1-c_e)L$ small edges in A, with each such small edge f having $c_f = 1/L$. □

We now prove Theorem 2.5.2.

Proof. We consider two cases.

1. Consider a small edge e, say $e = (v_1, v_2, \cdots, v_k)$ and $x_e = x$. From Lemma 2.7.2, we see $\Pr[\mathcal{S}_i] \geq (1 - x/L)^L$ for each $1 \leq i \leq k$. Thus by applying the

FKG inequality (2.1), we get $\Pr\left[\bigwedge_i \mathcal{S}_i\right] \geq (1-x/L)^{kL}$, which is followed by

$$\Pr[\ e \text{ is checked as safe }] \geq \int_0^{\delta} (1-x/L)^{kL} dx = \frac{1}{k+1/L} - O(k_0^k/k),$$

where $k_0 = (1-\delta/L)^L < 1$ is bounded away from 1.

2. Consider a large edge e, say $e = (v_1, v_2, \cdots, v_k)$ and $x_e = x$. From Lemma 2.7.2, we see $\Pr[\mathcal{S}_i] \geq (1-x/L)^{L-1}$ for each $1 \leq i \leq k$. Thus by applying FKG, we see when $x \leq \delta$, $\Pr\left[\bigwedge_i \mathcal{S}_i\right] \geq (1-x/L)^{k(L-1)}$, which is followed by

$$\begin{aligned} \Pr[\ e \text{ is checked as safe }] &\geq \int_0^{\delta} a(1-x/L)^{k(L-1)} dx \\ &\geq \frac{aL}{L-1} \frac{1}{k+1/(L-1)} - O(k_1^k/k) > \frac{1}{k} \end{aligned}$$

where $k_1 = (1-\delta/L)^{L-1} < 1$ is bounded away from 1; we use the fact that $a > 1 - 1/L$ to get the last inequality above.

□

2.8 Open Problems

Here we list a few open problems related to stochastic matching:

- Can we improve the ratio stated in Theorem 2.3.1 for the general graph? Note that in the WS as described above, the big edge f has a pretty much a higher chance of getting probed than $0.301y_f$. That suggests that we can potentially break the WS by choosing a weighted permutation favoring small p_e values,

the same idea as shown in [15].

- As for the weighted stochastic matching without patience constraints as introduced in [19], can we improve the best ratio of 1/0.573? We can reanalyze Algorithm 2 and potentially land at a different WS.

Chapter 3: Online Matching

3.1 Introduction

Applications to Internet advertising have driven the study of online matching problems in recent years [22]. In these problems, we consider a bipartite graph $G = (U, V, E)$ in which the set of vertices U is available *offline* while the set of vertices in V arrive *online*. Whenever some vertex v arrives, it must be matched immediately (and irrevocably) to (at most) one vertex in U. Each offline vertex u can be matched to at most one v. In the context of Internet advertising, U is the set of advertisers and V is the set of impressions. The edges E define the impressions that interest a particular advertiser. When an impression v arrives, we must choose an available advertiser (if any) to match with it. We consider the case where $v \in V$ can be matched at most once upon arriving. Since advertising forms the key source of revenue for many large Internet companies, finding good matching algorithms and obtaining even small performance gains can have high impact.

In the arrival mode of *Known Independent Identical Distributions* (KIID), we are given a bipartite graph $G = (U, V, E)$ and a finite online time horizon T (in most cases, we assume $T = |V| = n$ and say the online phase takes place over n rounds). In each round, a vertex v is sampled with replacement from a known distribution over V. The sampling distributions are independent and identical over all of the T online rounds. This captures the fact that we often have historical data about the impressions and can predict the frequency with which each type of impression will arrive. Edge-weighted matching [23] is a general model in the context of advertising: every advertiser gains a given revenue for being matched to a particular type of impression. Here, a *type* of impression refers to a class of users (*e.g.,* a demographic group) who are interested in the same subset of advertisements. Each arrival of a type $v \in V$ is considered a distinct vertex (user) that can be matched to up to one $u \in U$. For example, if the same v arrives three times, we consider this three separate vertices (or *copies* of v) that can potentially be matched to three different vertices in U. A special case of this model is vertex-weighted matching [24], where weights are associated only with the advertisers (the offline set U). In other words, a given advertiser has the same revenue generated for matching any of the user types interested in it.

In some modern business models, revenue is not generated upon matching advertisements, but only when a user *clicks* on the advertisement: this is the *pay-per-click* model. From historical data, one can assign the probability of a particular advertisement being clicked by a type of user. Works including [25, 26] capture this notion of *stochastic rewards* by assigning a probability to each edge.

3.2 Preliminaries

In the *Unweighted Online KIID Stochastic Bipartite Matching* problem, we are given a bipartite graph $G = (U, V, E)$. The set U is available offline while the vertices v arrive online and are drawn with replacement from an *independent identical* distribution on V. For each $v \in V$, we are given an *arrival rate* r_v, which is the expected number of times v will arrive. We refer to the case when all $r_v \in \mathbb{Z}^+$ as (the setting of) integral arrival rates; otherwise, we call non-integral or fractional arrival rates. For reasons described in [27], we can further assume WLOG that each v has $r_v = 1$ under the assumption of integral arrival rates. In this case, we have that $|V| = n$ where n is the total number of online rounds.

In the **vertex-weighted** variant, every vertex $u \in U$ has a weight w_u and we seek a maximum weight matching. In the **edge-weighted** variant, every edge $e \in E$ has a weight w_e and we again seek a maximum weight matching. In the **stochastic rewards** variant, each edge has both a weight w_e and a probability p_e of being present once we probe edge e[1] and we seek to maximize the expected weight of the matching.

Asymptotic assumptions and notations. We will always assume n is large and analyze algorithms as n goes to infinity: *e.g.,* if $x \leq 1 - (1 - 2/n)^n$, we will just write this as "$x \leq 1 - 1/e^2$" instead of the more-accurate "$x \leq 1 - 1/e^2 +$

[1]The edge realization process is independent for different edges. At each step, the algorithm "probes" an edge. With probability p_e the edge e exists and with the remaining probability it does not. Once realization of an edge is determined, it does not affect the random realizations for the rest of the edges. We consider the query-commit model where an edge that is probed and found to exist must be matched.

$o(1)$". These suppressed $o(1)$ terms will subtract at most $o(1)$ from our competitive ratios. Algorithms can be **adaptive** or **non-adaptive**. When v arrives, an adaptive algorithm can modify its online actions based on the realization of the online vertices (and edges in the stochastic rewards model) thus far, but a non-adaptive algorithm has to specify all of its actions before the start of the online phase. Throughout, we use "WS" to refer to the worst case instance for various algorithms.

Competitive Ratio. Competitive ratio is a commonly-used metric to evaluate the performance of online algorithms. Consider an online maximization problem for example. Let $\text{ALG}(\mathcal{I}) = \mathbb{E}_{I\sim\mathcal{I}}[\text{ALG}(I)]$ denote the expected performance of ALG on an input $\mathcal{I}$, where the expectation is taken over the random arrival sequence I. Let $\text{OPT}(\mathcal{I}) = \mathbb{E}[\text{OPT}(I)]$ denote the expected *offline optimal*, where $\text{OPT}(I)$ refers to the optimal value after we observe the full arrival sequence I. Then, competitive ratio is defined as $\min_{\mathcal{I}} \frac{\text{ALG}(\mathcal{I})}{\text{OPT}(\mathcal{I})}$. It is a common technique to use an LP to upper bound the $\text{OPT}(\mathcal{I})$ (called the benchmark LP) and hence get a valid lower bound on the target competitive ratio.

3.3 Overview and Related work

Online Matching. The Online Matching (OM) was first introduced by Karp *et al.* [28]. Suppose we have an unweighted bipartite graph $G = (U, V, E)$ where U and V represent the offline and online parties respectively. We have a time horizon, say T rounds, and during each round $t \in [T] \doteq \{1, 2, \ldots, T\}$ a vertex $v \in V$ arrives. Upon the arrival of v, we observe its neighbors $\delta(v) \subseteq U$ and need to make an

immediate and irrevocable decision: either reject v or match v to one its neighbor $u \in \delta(v)$ (in this case u will be not available afterwards). Our task is to design a matching algorithm such that the expected size of the final matching is maximized. Note that we have no any idea in advance regarding V and the arrival pattern in each round, which we refer to as the adversarial order or adversarial when context is clear.

For the unweighted OM under adversarial, Karp *et al.* [28] gave an optimal algorithm with an online competitive ratio of $(1 - 1/e)$. Two other variants of OM under adversarial are also studied which can be viewed as generalizations of the unweighted case: vertex-weighted (each offline vertex has a specific weight) and edge-weighted while the goal is updated to maximization of the total expected weight in the final matching. The vertex-weighted was introduced by Aggarwal *et al.* [24], where they gave an optimal $\left(1 - \frac{1}{e}\right)$ ratio. Feldman *et al.* [23] introduced the edge-weighted version, where they consider an additional relaxation of "free-disposal"; otherwise the ratio can be arbitrarily bad.

Two other well-studied variants of OM, Adwords and Display Ads, have received lots of attention as well. The models of Adwords and Display Ads generalize the matching constraint in OM in two different ways. In Adwords, each edge has a bid $w_e \geq 0$ and each u has a budget B_u; each time after matching $e = (u, v)$, we gain a profit of w_e while the budget of u is reduced by w_e; our goal is to maximize the expected total profit obtained. In contrast, Display Ads has a flavor of B-matching: each u has a capacity of $B_u \in \mathbb{Z}_+$ and each edge has a weight $w_e \geq 0$; each time when matching $e = (u, v)$, we obtain a profit of w_e while the capacity of u is reduced

by 1; the goal is to maximize the total profit. These two models can be viewed as a generalization of OM. As for the arrival setting, there are several other alternatives in addition to the adversarial arrival order. The following is a brief summarization.

1. Adversarial Order: the adversary can arrange the arrival order of all items in an arbitrary way (*e.g.,* Online Stochastic Matching [28, 29] and Adwords [30, 31]).

2. Random Arrival Order: all items arrive in a random permutation order (*e.g.,* Online Stochastic Matching [32, 33] and Adwords [34, 35]).

3. Unknown Distributions: in each round, each vertex is sampled from a fixed but unknown distribution. If the sampling distributions are required to be the identical and independent during each round, we refer to it as *Unknown Identical and Independent Distributions* (UIID) (*e.g.,* [36, 37]); otherwise, we call it *unknown adversarial distributions* (*e.g.,* [36])[2].

4. Known Distributions: in each round, an item is sampled from a known distribution. In particularly, we have KIID (*e.g.,* [38–42]) and *known adversarial distributions* (*e.g.,* [43, 44]), depending on if the sampling distributions are allowed to be different over time.

For each of the above four categories, we list only a few examples. For a more complete list, please refer to the book [22].

In this paper, we focus on OM under KIID. Here are a few related work. Feldman *et al.* [38] considered the unweighted OM case and they were the first to

[2]In [36, 37], it is referred to as adversarial stochastic input.

beat $1 - 1/e$ with a competitive ratio of 0.67. Later, Manshadi *et al.* [40] improved that ratio to 0.705 and they showed no algorithm could achieve a ratio better than $1 - e^{-2} \approx 0.86$ with integral arrival rate (the expected total arrivals of each vertex is integral). Finally, Jaillet *et al.* [41] presented a strengthened LP to achieve a ratio of 0.725 and $1 - 2e^{-2} \approx 0.729$ for the vertex-weighted and unweighted case respectively. As for the edge-weighted case, Haeupler *et al.* [39] were the first to beat $1 - 1/e$ by achieving a competitive ratio of 0.667. They use a *discounted* LP with tighter constraints than the basic matching LP (a similar LP can be seen in (3.1)) and they employ the *power of two choices* by constructing two matchings offline to guide their online algorithm.

Online Matching with Stochastic Rewards. Mehta *et al.* [25] introduced an interesting variant to OM: each edge e is associated with a Bernoulli random reward, which is equal to some $w_e \geq 0$ with probability $p_e \geq 0$ and 0 otherwise. This model can be viewed as a combination of the Stochastic Matching and Online Matching, which was called as as *Online Matching with Stochastic Rewards* (OM-SR) in [25]. Mehta *et al.* [25] focused on the simple case when $w_e = 1$ and $p_e = p$ for all edges e under the setting of adversarial arrival order and they gave a deterministic algorithm which achieves a ratio of $1/0.567$ for vanishing probabilities ($p \to 0$). Mehta *et al.* [26] considered the same setting but each e is allowed to have a distinct probability p_e. They gave a $1/0.534$-approximation algorithm when all $p_e \to 0$.

3.4 Main Techniques and Our Results

In this paper, we focus on the edge-weighted OM under KIID with integral arrival rates. For each e, let f_e be the probability that e is added in the offline optimal. For each vertex $w \in U \cup V$, let $\partial(w)$ be the set of edges adjacent to and let $f_w = \sum_{e \in \partial(w)} f_e$. Consider the following benchmark LP, which is used to upper bound the offline OPT for the unweighted version:

$$\text{maximize} \quad \sum_{e \in E} f_e \tag{3.1}$$

$$\text{subject to} \quad \sum_{e \in \partial(u)} f_e \leq 1 \qquad \forall u \in U \tag{3.2}$$

$$\sum_{e \in \partial(v)} f_e \leq 1 \qquad \forall v \in V \tag{3.3}$$

$$0 \leq f_e \leq 1 - 1/e \qquad \forall e \in E \tag{3.4}$$

$$f_e + f_{e'} \leq 1 - 1/e^2 \qquad \forall e, e' \in \partial(u), \forall u \in U \tag{3.5}$$

Here are a few variants. The objective function is maximization of $\sum_{u \in U} \sum_{e \in \partial(u)} f_e w_u$ in the vertex-weighted version and that of $\sum_{e \in E} f_e w_e$ in the edge-weighted version, where w_u and w_e refer to the weight on vertex u and edge e respectively.

Constraint (3.2) is the matching constraint for vertices in U. Constraint (3.3) is valid because each vertex in V has an arrival rate of 1. Constraint (3.4) is used in [40] and [39]. It captures the fact that the expected number of matches for any edge is at most $1 - 1/e$. This is valid for large n because the probability that a

given vertex doesn't arrive after n rounds is $1/e$. Constraint (3.5) is similar to the previous one, but for pairs of edges. For any two neighbors of a given $u \in U$, the probability that neither of them arrive is $1/e^2$. Therefore, the sum of variables for any two distinct edges in $\partial(u)$ cannot exceed $1 - 1/e^2$. Notice that constraints (3.4) and (3.5) reduces the gap between the optimal LP solution and the performance of the optimal online algorithm.

Feldman *et al.* [38] first proposed the idea of "two suggested matchings" and they used it to attack unweighted OM under KIID and beat the golden ratio of $1 - 1/e$. Later Haeupler *et al.* [39] generalized this idea to attack the weighted version and get current best ratio of 0.667. We generalize this idea further by generating the two matchings in a random way and improve the ratio to 0.688 for the the edge-weighted version. Here is the main picture. First we solve the LP (3.1) and let $\mathbf{f} = \{f_e\}$ be an optimal solution. Second we apply the dependent rounding in [14] to $2 * \mathbf{f}$ and suppose we get an integral vector $\mathbf{F}$. Let $G_{\mathbf{F}}$ be the sparse graph induced by $\mathbf{F}$ where each edge e has $\mathbf{F}_e$ copies. Note that since each $f_e \leq 1 - 1/e$, we have $\mathbf{F}_e \in \{0, 1, 2\}$ and from the property of dependent rounding, we see each vertex on $G_{\mathbf{F}}$ has a degree at most 2. We then apply Hall's Theorem to $G_{\mathbf{F}}$ and decompose it into two matchings. The formal description of all algorithm is shown as below.

Theorem 3.4.1. *Algorithm 5 achieves a ratio of* 0.688 *for the edge-weighted online stochastic matching with integral arrival rates.*

Inspired from the work [25, 26], we introduce the model of OM-SR under KIID

Algorithm 5: Edge-weighted Online Matching under KIID

1 Construct and solve the benchmark LP (3.1) for the input instance.
2 Let $\mathbf{f}$ be an optimal fraction solution vector. Apply dependent rounding to $2 * \mathbf{f}$ to get an integral vector $\mathbf{F}$.
3 Create the graph $G_{\mathbf{F}}$ with $\mathbf{F}_e$ copies of each edge $e \in E$ and decompose it into two matchings.
4 Randomly permute the matchings to get a *random ordered* pair of matchings, say $[M_1, M_2]$.
5 When a vertex v arrives for the first time, try to assign v to some u_1 if $(u_1, v) \in M_1$; when v arrives for the second time, try to assign v to some u_2 if $(u_2, v) \in M_2$.
6 When a vertex v arrives for the third time or more, do nothing in that step.

with arbitrary arrival rates. For each edge e and vertex v, let f_e be the probability that e gets matched in an offline optimal and r_v be the expected total arrivals. Consider the following benchmark LP:

$$\max \quad \sum_{e \in E} w_e f_e p_e : \tag{3.6}$$

$$\text{s.t.} \quad \sum_{e \in \partial(u)} f_e p_e \leq 1, \forall u \in U \tag{3.7}$$

$$\sum_{e \in \partial(v)} f_e \leq r_v, \forall v \in V \tag{3.8}$$

We present a very simple non-adaptive algorithm, which achieves a ratio of $1 - 1/e$. Note that Manshadi *et al.* [40] show that no non-adaptive algorithm could possibly achieve a ratio better than $(1 - 1/e)$ for the non-integral arrival rates, even for the case of all $p_e = 1$. Thus, our algorithm is an optimal non-adaptive algorithm for this model.

Theorem 3.4.2. *Algorithm 6 achieves a ratio of* $1 - 1/e$ *for the edge-weighted* OM-SR *under* KIID *with arbitrary arrival rates. This is an optimal non-adaptive algorithm.*

Algorithm 6: OM-SR under KIID with arbitrary arrival rates

1 Construct and solve LP (3.6). WLOG assume $\{f_e | e \in E\}$ is an optimal solution.
2 When a vertex v arrives, assign v to each of its neighbor u with a probability $\frac{f_{(u,v)}}{r_v}$.

3.5 Proofs of Theorems 3.4.1 and 3.4.2

We show that Algorithm 5 achieves a competitive ratio of 0.688. Let $[M_1, M_2]$ be our randomly ordered pair of matchings. Note that there might exist some edge e which appears in both matchings if and only if $f_e > 1/2$. Therefore, we consider three types of edges. We say an edge e is of type ψ_1, denoted by $e \in \psi_1$, if and only if e appears *only* in M_1. Similarly $e \in \psi_2$, if and only if e appears *only* in M_2. Finally, $e \in \psi_b$, if and only if e appears in *both* M_1 and M_2.

Let P_1, P_2, and P_b be the probabilities of getting matched for $e \in \psi_1$, $e \in \psi_2$, and $e \in \psi_b$, respectively. According to the result in Haeupler *et al.* [27], Lemma 3.5.1 bounds these probabilities.

Lemma 3.5.1 (Proof details in section 3 of [27])**.** *Given M_1 and M_2, in the worst case (1) $P_1 = 0.5808$; (2) $P_2 = 0.14849$ and (3) $P_b = 0.632$.*

We can use Lemma 3.5.1 to prove that Algorithm 5 achieves a ratio of 0.688 by examining the probability that a given edge becomes type ψ_1, ψ_2, or ψ_b.

Proof of Theorem 3.4.1.

Proof. Consider the following two cases.

- **Case 1:** $0 \leq f_e \leq 1/2$: By the marginal distribution property of dependent rounding, there can be at most one copy of e in $G_{\mathbf{F}}$ and the probability of including e in $G_{\mathbf{F}}$ is $2f_e$. Since an edge in $G_{\mathbf{F}}$ can appear in either M_1 or M_2 with equal probability $1/2$, we have $\Pr[e \in \psi_1] = \Pr[e \in \psi_2] = f_e$. Thus, the ratio is $(f_e P_1 + f_e P_2)/f_e = P_1 + P_2 = 0.729$.

- **Case 2:** $1/2 \leq f_e \leq 1 - 1/e$: Similarly, by marginal distribution, $\Pr[e \in \psi_b] = \Pr[F_e = \lceil 2f_e \rceil] = 2f_e - \lfloor 2f_e \rfloor = 2f_e - 1$. It follows that $\Pr[e \in \psi_1] = \Pr[e \in \psi_2] = (1/2)(1 - (2f_e - 1)) = 1 - f_e$. Thus, the ratio is (noting that the first term is from case 1 while the second term is from case 2) $((1 - f_e)(P_1 + P_2) + (2f_e - 1)P_b)/f_e \geq 0.688$, where the WS is for an edge e with $f_e = 1 - 1/e$.

□

Proof of Theorem 3.4.2.

Proof. Let $B(u,t)$ be the event that u is safe at beginning of round t and $A(u,t)$ be the event that vertex u is matched during the round t conditioned on $B(u,t)$. From the algorithm, we know $\Pr[A(u,t)] \leq \sum_{v \sim u} \frac{r_v}{n} \frac{f_{u,v}}{r_v} p_e \leq \frac{1}{n}$, which is followed by $\Pr[B(u,t)] = \Pr\left[\bigwedge_{i=1}^{t-1} (\neg A(u,i))\right] \geq \left(1 - \frac{1}{n}\right)^{t-1}$.

Consider an edge $e = (u,v)$ in the graph. Notice that the probability that e gets matched in Algorithm 6 should be as follows.

$$\begin{aligned} \Pr[e \text{ is matched}] &= \sum_{t=1}^{n} \Pr[\text{v arrives at } t \text{ and } B(u,t)\ ] \cdot \frac{f_e p_e}{r_v} \\ &\geq \sum_{t=1}^{n} \left(1 - \frac{1}{n}\right)^{t-1} \frac{r_v}{n} \frac{f_e p_e}{r_v} \geq \left(1 - \frac{1}{e}\right) f_e p_e \end{aligned}$$

Note that Manshadi *et al.* [40] show that no non-adaptive algorithm could possibly achieve a ratio better than $(1 - 1/e)$ for the non-integral arrival rates, even for the case of all $p_e = 1$. Thus, our algorithm is optimal among all possible non-adaptive algorithms. □

3.6 Open Problems and Future Directions

Consider the simple case of OM-SR under KIID but with integral arrival rates. We observe that the LP (3.6) is updated to

$$\max \quad \sum_{e \in E} w_e f_e p_e : \tag{3.9}$$

$$\text{s.t.} \quad \sum_{e \in \partial(u)} f_e p_e \leq 1, \forall u \in U \tag{3.10}$$

$$\sum_{e \in \partial(v)} f_e \leq 1, \forall v \in V \tag{3.11}$$

Given an instance I, let $\text{LP}(I)$ be the optimal value over I and $\text{OPT}(I)$ be the performance of an optimal (adaptive) algorithm on I. For a given LP, we define the Stochasticity Gap (StochGap) as the maximum ratio of $\text{LP}(I)/\text{OPT}(I)$ over all possible feasible instances. For the above LP (3.9), we can show its StochGap is equal to $1/(1 - 1/e)$.

Lemma 3.6.1. *The StochGap of LP (3.9) is equal to* $\frac{1}{1-1/e}$.

Proof. Consider such an instance I^*: $G = (U, V, E)$, where G is a unweighted complete star graph; $|U| = 1, |V| = T = n$, $p_e = 1/n$, $r_v = 1$ for $v \in V$ and all $w_e = 1$. We can verify that $\text{LP}(I^*) = 1$ and $\text{OPT}(I^*) = 1 - 1/e$. This implies that

the StochGap of LP (3.9) is at least $1/(1-1/e)$. From Theorem 3.4.1, we see the StochGap is at most $1/(1-1/e)$. Summarizing all analysis we reach our claim. □

An interesting question is: can we beat $1-1/e$ for the edge-weighted OM-SR under KIID with integral arrival rates? The first task facing us might be to add some extra constraints to tighten the StochGap for the LP (3.9).

Chapter 4: An Application of Online Matching in Ridesharing

4.1 Introduction

In bipartite matching problems, agents on one side of a market are paired with agents, contracts, or transactions on the other. Classical matching problems—assigning students to schools, papers to reviewers, or medical residents to hospitals—take place in a static setting, where all agents exist at the time of matching, are simultaneously matched, and then the market concludes. In contrast, many matching problems are dynamic, where one side of the market arrives in an *online* fashion and is matched sequentially to the other side.

Online bipartite matching problems are primarily motivated by Internet advertising. In the basic version of the problem, we are given a bipartite graph $G = (U, V, E)$ where U represents the offline vertices (advertisers) and V represents the online vertices (keywords or impressions). There is an edge $e = (u, v)$ if

advertiser u bids for a keyword v. When a keyword v arrives, a central clearinghouse must make an instant and irrevocable decision to either reject v or assign v to one of its "neighbors" (*i.e.,* set of incident edges) u and obtain a profit w_e for the match $e = (u, v)$. When an advertiser u is matched, it is no longer available for matches with other keywords (in the most basic case) or its budget is reduced. The goal is to design an efficient online algorithm such that the expected total weight (profit) of the matching obtained is maximized. Following the seminal work of [28], there has been a large body of research on related variants (overviewed by [22]). One particular flavor of problems is Online Matching with Known Identical Independent Distributions (OM-KIID) [38–42]. In this flavor, agents arrive over T rounds, and their arrival distributions are assumed to be *identical and independent* over all T rounds; additionally, this distribution is known to the algorithm beforehand.

Apart from the Internet advertising application, online bipartite matching models have been used to capture a wide range of online resource allocation and scheduling problems. Typically we have an offline and an online party representing, respectively, the service providers (SP) and online users; once an online user arrives, we need to match it to an offline SP immediately. In many cases, the service is *reusable* in the sense that once an SP is matched to a user, it will be gone for some time, but will then rejoin the system afterwards. Besides that, in many real settings the arrival distributions of online users do change from time to time. Consider the following motivational examples.

Taxi Dispatching Services and Ridesharing Systems. Traditional taxi ser-

vices and rideshare systems like Uber and Didi Chuxing match drivers to would-be riders [45–48]. Here, the offline SPs are different vehicle drivers. Once an online request (potential rider) arrives, the system matches it to a nearby driver instantly such that the rider's waiting time is minimized. In most cases, the driver will rejoin the system and can be matched again once she finishes the service. Additionally, the arrival rates of requests changes dramatically across the day. Consider the online arrivals during peak hours and off-peak hours for example: the arrival rates in the former case can be much larger than the latter.

Organ Allocation. Chronic kidney disease affects tens of millions of people worldwide at great societal and monetary cost [49, 50]. Organ donation—either via a deceased or living donor—is a lifesaving alternative to organ failure. In the case of kidneys, a donor organ can last up to 15 years in a patient before failing again. Various nationwide organ donation systems exist and operate under different ethical and logistical constraints [51–53], but all share a common online structure: the offline party is the set of patients (who reappear every 5 to 15 years based on donor organ longevity), and the online party is the set of donors or donor organs, who arrive over time.

Similar scenarios can be seen in other areas such as wireless network connection management (SPs are different wireless access points) [54] and online cloud computing service scheduling [55, 56]. Inspired by the above applications, we generalize the model of OM-KIID in the following two ways.

Reusable Resources. Once we assign v to u, u will rejoin the system after C_e

rounds with $e = (u, v)$, where $C_e \in \{0, 1, \ldots, T\}$ is an integral random variable with known distribution. In this paper, we call C_e the *occupation time* of u w.r.t. e. In fact, we show that our setting can directly be extended to the case when C_e is time sensitive: when matching v to u at time t, u will rejoin the system after $C_{e,t}$ rounds. This extension makes our model adaptive to nuances in real-world settings. For example, consider the taxi dispatching or ride-sharing service: the occupation time of a driver u from a matching with an online user v does depend on both the user type of v (such as destination) and the time when the matching occurs (peak hours can differ significantly from off-peak hours).

Known Adversarial Distributions (KAD**)**. Suppose we have T rounds and that for each round $t \in [T]$, a vertex v is sampled from V according to an arbitrary known distribution $\mathcal{D}$ where the marginal for v is $\{p_{v,t}\}$ such that $\sum_{v \in V} p_{v,t} \leq 1$ for all t. Also, the arrivals at different times are independent (and according to these given distributions). The setting of KAD was introduced by [43, 44] and is called Prophet Inequality matching.

We call our new model Online Matching with (offline) Reusable Resources under Known Adversarial Distributions (OM-RR-KAD, henceforth). Note that the OM-KIID model can be viewed as a special case when C_e is a constant (with respect to T) and $\{p_{v,t} | v \in V\}$ are the same for all $t \in [T]$.

4.2 Related Work

Despite the fact that our model is inspired by online bipartite matching, it also overlaps with stochastic online scheduling problems (SOS) [57–59]. We first restate our model in the language of SOS: we have $|U|$ *nonidentical* parallel machines and $|V|$ jobs; at every time-step a single job v is sampled from V with probability $p_{v,t}$; the jobs have to be assigned immediately after its arrival (or rejected right away); additionally each job v can be processed *non-preemptively* on a specific subset of machines; once we assign v to u, we get a profit of w_e and u will be occupied for C_e rounds with $e = (u, v)$, where C_e is a random variable with known distribution. Observe that the key difference between our model and SOS is in the objective: the former is to maximize the expected profit from the completed jobs, while the latter is to minimize the total or the maximum completion time of all jobs.

Research in ridesharing platforms and similar allocation problems is an active area of research within multiple fields, including computer science, operations research and transportation engineering. State-independent policies were studied previously using theory from control and queuing systems [60–62]. The role of pricing in the dynamics of drivers in ridesharing platforms is also an active area of research in computational economics and AI/ML (*e.g.,* [63–68]). Our problem is a form of *online matching in dynamic environments*, which is an active area of research within the AI/ML community. In particular, [45, 46, 52, 69] have studied algorithms for matching in various dynamic bipartite markets such as kidney exchange, spatial crowdsourcing, labor markets, and so on. Similar line of work on general graphs is

also prominent in the literature (*e.g.,* [70–73]).

4.3 Main Model and Techniques

4.3.1 Main Model

In this section, we present a formal statement of our main model. Suppose we have a bipartite graph $G = (U, V, E)$ where U and V represent the offline and online parties respectively. We have a finite time horizon T (known beforehand) and for each time $t \in [T]$, a vertex v will be sampled (we use the term v *arrives*) from a known probability distribution $\{p_{v,t}\}$ such that $\sum_{v \in V} p_{v,t} \leq 1$[1] (noting that such a choice is made independently for each round t). The expected number of times v arrives across the T rounds, $\sum_{t \in [T]} p_{v,t}$, is called the *arrival rate* for vertex v. Once a vertex v arrives, we need to make an *irrevocable decision immediately*: either to reject v or assign v to one of its neighbors in U. For each u, once it is assigned to some v, it becomes unavailable for C_e rounds with $e = (u, v)$, and subsequently rejoins the system. Here C_e is an integral random variable taking values from $\{0, 1, \ldots, T\}$ and the distribution is known in advance. Each assignment e is associated with a weight w_e and our goal is to design an online assignment policy such that the total expected weights of all assignments made is maximized. Following prior work, we assume $|V| \gg |U|$ and $T \gg 1$. Throughout this paper, we use edge $e = (u, v)$ and assignment of v to u interchangeably.

For an assignment e, let $x_{e,t}$ be the probability that e is chosen at t in any

[1] Thus, with probability $1 - \sum_{v \in V} p_{v,t}$, none of the vertices from V will arrive at t.

offline optimal algorithm. For each u (likewise for v), let E_u (E_v) be the set of neighboring edges incident to u (v). We use the LP (4.1) as a benchmark to upper bound the offline optimal. We now interpret the constraints. For each round t, once an online vertex v arrives, we can assign it to at most one of its neighbors. Thus, we have: if v arrives at t, the total number of assignments for v at t is at most 1; if v does not arrive, the total is 0. The LHS of (4.2) is exactly the expected number of assignments made at t for v. It should be no more than the prob. that v arrives at t, which is the RHS of (4.2). Constraint (4.3) is the *most* novel part of our problem formulation. Consider a given u and t. In the LHS, the first term (summation over $t' < t$ and $e \in E_u$) refers to the prob. that u is not available at t while the second term (summation over $e \in E_u$) is the prob. that u is assigned to some worker at t, which is no larger than prob. u is available at t. Thus, the sum of the first term and second term on LHS is no larger than 1.[2] This argument implies that the LP forms a valid upper-bound on the offline optimal solution, which is formally stated in the below lemma.

Lemma 4.3.1. *The optimal value to LP (4.1) is a valid upper bound for the offline optimal.*

Extension from C_e to $C_{e,t}$. Consider the case when the occupation time of u from e is sensitive to t. In other words, each u will be unavailable for $C_{e,t}$ rounds from the assignment $e = (u, v)$ at t. We can accommodate the extension by simply updating the constraints (4.3) on u in the benchmark LP (4.1) to the following. We have that

[2]We would like to point out that our LP constraint (4.3) on u is inspired by [74]. The proof is similar to that by [43] and [44].

$$\text{maximize} \sum_{t\in[T]} \sum_{e\in E} w_e x_{e,t} \tag{4.1}$$

$$\text{subject to} \sum_{e\in E_v} x_{e,t} \le p_{v,t} \qquad \forall v \in V, t \in [T] \tag{4.2}$$

$$\sum_{t'<t} \sum_{e\in E_u} x_{e,t'} \Pr[C_e > t - t'] + \sum_{e\in E_u} x_{e,t} \le 1 \qquad \forall u \in U, t \in [T] \tag{4.3}$$

$$0 \le x_{e,t} \le 1 \qquad \forall e \in E, t \in [T] \tag{4.4}$$

$\forall u \in U, t \in [T]$,

$$\sum_{t'<t} \sum_{e\in E_u} x_{e,t'} \Pr[C_{e,t'} > t - t'] + \sum_{e\in E_u} x_{e,t} \le 1 \tag{4.5}$$

The rest of our algorithm remains the same as before. We can verify that LP (4.1) with constraints (4.3) replaced by (4.5) is a valid benchmark.

4.3.2 A Simulation-Based Algorithm

Now we present a simulation-based algorithm. The main idea is as follows. Let $\mathbf{x}^*$ denote an optimal solution to LP (4.1). Suppose we aim to develop an online algorithm achieving a ratio of $\gamma \in [0, 1]$. Consider an assignment $e = (u, v)$ when some v arrived at time t. Let $\text{SF}_{e,t}$ be the event that e is safe at t, *i.e.,* u is available at t. By simulating the current strategy up to t, we can get an estimation of $\Pr[\text{SF}_{e,t}]$, say $\beta_{e,t}$, within an arbitrary small error. Therefore in the case where e is safe at t, we can sample it with probability $\frac{x^*_{e,t}}{p_{v,t}} \frac{\gamma}{\beta_{e,t}}$, which leads to the fact that e is sampled with probability $\gamma x^*_{e,t}$ unconditionally. Hence, we call any algorithm that satisfies $\gamma \le \beta_{e,t}$ as *valid*. At the outset, this looks similar to the Inverse Propensity

Scoring (IPS) used in the multi-armed bandit literature [75]. However, there is a key difference between IPS estimates and our estimates. In the bandit literature, one usually scales the value by the probability of playing an action, since this is the *cost* of observing only bandit feedback. However, here we scale by a quantity that depends on the probability of a certain event happening during the *run* of the algorithm, because of playing other actions. The linear program gives a distribution over the edges assuming that all the neighbors are available. Hence this scaling can be interpreted as the *cost* the algorithm needs to incur when some neighbors are already matched.

The simulation-based attenuation technique has been used previously for other problems, such as stochastic knapsack [74] and stochastic matching [76]. Throughout the analysis, we assume that we know the exact value of $\beta_{e,t} := \Pr[\text{SF}_{e,t}]$ for all t and e. (It is easy to see that the sampling error can be folded into a multiplicative factor of $(1-\epsilon)$ in the competitive ratio by standard Chernoff bounds and hence, ignoring it leads to a cleaner presentation.). The formal statement of our algorithm, denoted by ADAP(γ), is as follows. For each v and t, let $E_{v,t}$ be the set of *safe* assignments for v at t.

Algorithm 7: Simulation-based adaptive algorithm (ADAP(γ))

1 For each time t, let v denote the request arriving at time t.

2 If $E_{v,t} = \emptyset$, then reject v; otherwise choose $e \in E_{v,t}$ with prob. $\frac{x^*_{e,t}}{p_{v,t}} \frac{\gamma}{\beta_{e,t}}$ where $e = (u, v)$.

Here are two main theoretical results regarding the algorithm ADAP(γ) and our model.

Theorem 4.3.1. LP (4.1) *is a valid benchmark for* OM-RR-KAD. *There exists an online algorithm, based on* LP (4.1)*, achieving an online competitive ratio of* $\frac{1}{2} - \epsilon$ *for any given* $\epsilon > 0$.

Theorem 4.3.2. *No non-adaptive algorithm, based on benchmark* LP (4.1)*, can achieve a competitive ratio better than* $\frac{1}{2} + o(1)$ [3] *even when all* C_e *are constants.*

4.4 Proofs of Theorems 4.3.1 and 4.3.2

4.4.1 Proof of Theorem 4.3.1

Lemma 4.4.1. ADAP(γ) *is valid with* $\gamma = \frac{1}{2}$.

Proof. We show by induction on t as follows. When $t = 1$, $\beta_{e,t} = 1$ for all $e = (u, *)$, we are done. This is because of the following.

$$\sum_{e \in E_{v,t}} \frac{x^*_{e,t}}{p_{v,t}} \frac{\gamma}{\beta_{e,t}} \leq \sum_{e \in E_v} \frac{x^*_{e,t}}{p_{v,t}} \gamma \leq \frac{1}{2}$$

Assume for all $t' < t$, $\beta_{e,t'} \geq 1/2$ and ADAP(γ) is valid for all rounds t'. In other words, each e is assigned with probability *exactly* equal to $x^*_{e,t'} * \frac{1}{2}$ for all $t' < t$. Now consider a given $e = (u, v)$. Observe that e is unsafe at t iff u is assigned with some v' at $t' < t$ such that the assignment $e' = (u, v')$ makes u unavailable at t. Therefore

$$1 - \beta_{e,t} = 1 - \Pr[\mathrm{SF}_{e,t}] = \sum_{t' < t} \sum_{e \in E_u} \frac{x^*_{e,t'}}{2} \Pr[C_e > t - t']$$

[3] $o(1)$ is a vanishing term when both of C_e and T/C_e are sufficiently large.

Thus from the constraints (4.3) in our benchmark LP, we have the following.

$$\beta_{e,t} = 1 - \sum_{t'<t} \sum_{e \in E_u} \frac{x^*_{e,t'}}{2} \Pr[C_e > t - t'] \geq \frac{1}{2} + \frac{1}{2} \sum_{e \in E_u} x^*_{e,t} \geq \frac{1}{2}$$

Hence we have, $\sum_{e \in E_{v,t}} \frac{x^*_{e,t}}{p_{v,t}} \frac{\gamma}{\beta_{e,t}} \leq \sum_{e \in E_v} \frac{x^*_{e,t}}{p_{v,t}} \leq 1$ and thus we are done. □

The main Theorem 4.3.1 follows directly from Lemmas 4.3.1 and 4.4.1.

4.4.2 Proof of Theorem 4.3.2

Consider a complete bipartite graph $G = (U, V, E)$ where $|U| = K$, $|V| = n^2$. Suppose we have $T = n$ rounds and $p_{v,t} = \frac{1}{n^2}$ for each v and t. In other words, in each round t, each v is sampled uniformly from V. For each e, let C_e be a constant of K, which implies that each u will be unavailable for a constant K rounds after each assignment. Assume all assignments have a uniform weight (*i.e.*, $w_e = 1$ for all e). Split the whole online process of n rounds into $n - K + 1$ consecutive windows $\mathcal{W} = \{W_\ell\}$ such that $W_\ell = \{\ell, \ell + 1, \ldots, \ell + K - 1\}$ for each $1 \leq \ell \leq n - K + 1$. The benchmark LP (4.1) then reduces to the following.

$$\max \sum_{t \in [T]} \sum_{e \in E} x_{e,t} \tag{4.6}$$

$$\text{s.t.} \sum_{e \in E_v} x_{e,t} \leq \frac{1}{n^2} \qquad \forall v \in V, t \in [T] \tag{4.7}$$

$$\sum_{t \in W_\ell} \sum_{e \in E_u} x_{e,t} \leq 1 \qquad \forall u \in U, 1 \leq \ell \leq n - K + 1 \tag{4.8}$$

$$0 \leq x_{e,t} \leq 1 \qquad \forall e \in E, t \in [T] \tag{4.9}$$

We can verify that an optimal solution to the above LP is as follows: $x^*_{e,t} = 1/(n^2K)$ for all e and t with the optimal objective value of n. We investigate the performance of any optimal non-adaptive algorithm. Notice that the expected arrivals of any v in the full sequence of online arrivals is $1/n$. Thus for any non-adaptive algorithm NADAP, it needs to specify the allocation distribution $\mathcal{D}_v$ for each v during the first arrival. Consider a given NADAP parameterized by $\{\alpha_{u,v} \in [0,1]\}$ for each v and $u \in E_v$ such that $\sum_{u \in E_v} \alpha_{u,v} \leq 1$ for each v. In other words, NADAP will assign v to u with probability $\alpha_{u,v}$ when v comes for the first time and u is available.

Let $\beta_u = \sum_{v \in E_u} \alpha_{u,v} * \frac{1}{n^2}$, which is the probability that u is matched in each round if it is safe at the beginning of that round, when running NADAP. Hence,

$$\sum_{u \in U} \beta_u = \sum_{u \in U} \sum_{v \in E_u} \alpha_{u,v} \cdot \frac{1}{n^2} = \sum_{v \in V} \sum_{u \in E_v} \alpha_{u,v} \cdot \frac{1}{n^2} \leq 1$$

Consider a given u with β_u and let $\gamma_{u,t}$ be the probability that u is available at t. Then the expected number of matches of u after the n rounds is $\sum_t \beta_u \gamma_{u,t}$. We have the recursive inequalities on $\gamma_{u,t}$ as in Lemma 4.4.2, with $\gamma_{u,t} = 1, t = 1$.

Lemma 4.4.2. *$\forall 1 < t \leq n$, we have*

$$\gamma_{u,t} + \beta_u \sum_{t-K+1 \leq t' < t} \gamma_{u,t'} = 1$$

Proof. The inequality for $t = 1$ is due to the fact that u is safe at $t = 1$. For each time $t > 1$, Let $\text{SF}_{u,t}$ be the event that u is safe at t and $A_{u,t}$ be the event that u is matched

at t. Observe that for each window of K time slots, $\{\text{SF}_{u,t}, A_{u,t'}, t-K+1 \leq t' < t\}$ are are mutually exclusive and collectively exhaustive events. Therefore,

$$\begin{aligned} 1 &= \Pr[\text{SF}_{u,t}] + \sum_{t-K+1 \leq t' < t} \Pr[A_{u,t'}] \\ &= \gamma_{u,t} + \beta_u \sum_{t-K+1 \leq t' < t} \gamma_{u,t} \end{aligned}$$

□

Note that the OPT of our benchmark LP is n while the performance of NADAP is $\sum_u \sum_t \beta_u \gamma_{u,t}$. The resulting competitive ratio achieved by an optimal NADAP is captured by the following maximization problem.

$$\begin{aligned} \max \quad & \frac{\sum_u \sum_t \beta_u \gamma_{u,t}}{n} && && (4.10) \\ \text{s.t.} \quad & \sum_{u \in U} \beta_u \leq 1 && && (4.11) \\ & \gamma_{u,t} + \beta_u \sum_{t-K+1 \leq t' < t} \gamma_{u,t'} = 1 && \forall 1 < t \leq n, u \in U && (4.12) \\ & \beta_u \geq 0, \gamma_{u,1} = 1 && \forall u \in U && (4.13) \end{aligned}$$

We prove the following Lemma which implies Theorem 4.3.2.

Lemma 4.4.3. *The optimal value to the program (4.10) is at most* $\frac{1}{2-1/K} + o(1)$ *when* $K = o(n)$.

Proof. Focus on a given vertex $u \in U$. Notice that $\gamma_{u,t} + \beta_u \sum_{t-K+1 \leq t' < t} \gamma_{u,t'} = 1$

for all $1 \leq t \leq n$. Summing both sides over $t \in [n]$, we have the following.

$$\begin{aligned}\Big(1+\beta_u(K-1)\Big)\sum_{t\in[n]}\gamma_{u,t} &= n+\beta_u(K-1)\gamma_{u,n}+\beta_u(K-2)\gamma_{u,n-1}+\cdots+\beta_u\gamma_{u,n-K+2}\\ &\leq n+K-1\end{aligned}$$

Therefore we have,

$$\sum_{t\in[n]}\gamma_{u,t} \leq \frac{n}{1+\beta_u(K-1)}+\frac{K-1}{1+\beta_u(K-1)} \leq \frac{n}{1+\beta_u(K-1)}+\frac{1}{\beta_u}$$

Define $H_u \doteq \sum_t \beta_u\gamma_{u,t}$. From the above analysis, we have that $H_u \leq \frac{n\beta_u}{1+\beta_u(K-1)}+1$. Thus the objective value in the program (4.10) can be upper-bounded as follows.

$$\frac{\sum_u\sum_t \beta_u\gamma_{u,t}}{n} = \sum_{u\in U}\frac{H_u}{n} \leq \sum_{u\in U}\frac{\beta_u}{1+\beta_u(K-1)}+\frac{K}{n}$$

We claim that the optimal value to the program (4.10) can be upper bounded by the following maximization program.

$$\left\{\max \sum_{u\in[U]}\frac{\beta_u}{1+\beta_u(K-1)}+\frac{K}{n}: \quad \sum_{u\in U}\beta_u = 1, \beta_u \geq 0, \forall u \in U\right\}$$

According to our assumption $K = o(n)$, the second term can be ignored. Let $g(x) = x/(1+x(K-1))$. For any $K \geq 2$, it is a concave function, which implies that maximization of g subject to $\sum_u \beta_u = 1$ will be achieved when all $\beta_u = 1/K$. The resultant value is $\frac{1}{2-1/K}+o(1)$. Thus we are done. □

4.5 Experiments

To validate the approaches presented in this paper, we use the New York City Yellow Cabs dataset,[4] which contains the trip records for trips in Manhattan, Brooklyn, and Queens for the year 2013. The dataset is split into 12 months. For each month we have numerous records each corresponding to a single trip. Each record has the following structure. We have an anonymized license number which is the primary key corresponding to a car. For privacy purposes a long string is used as opposed to the actual license number. We then have the time at which the trip was initiated, the time at which the trip ended, and the total time of the trip in seconds. This is followed by the starting coordinates (*i.e.,* latitude and longitude) of the trip and the destination coordinates of the trip.

Assumptions. We make two assumptions specific to our experimental setup. Firstly, we assume that every car starts and ends at the same location, for *all* trips that it makes. Initially, we assign every car a location (potentially the same) which corresponds to its *docking* position. On receiving a request, the car leaves from this docking position to the point of pick-up, executes the trip and returns to this docking position. Secondly, we assume that occupation time distributions (OTD) associated with all matches are identically (and independently) distributed, *i.e.,* $\{C_e\}$ follow the same distribution. Note that this is a much stronger assumption than what we made in the model, and is completely inspired by the dataset (see Section 4.5.2). We test our model on two specific distributions, namely a *normal*

[4] `http://www.andresmh.com/nyctaxitrips/`

distribution and the *power-law* distribution (see Figure 4.5). The docking position of each car and parameters associated with each distribution are all learned from the training dataset (described below in the **Training** discussion).

4.5.1 Experimental Setup

For our experimental setup, we randomly select 30 cabs (each cab is denoted by u). We discretize the Manhattan map into cells such that each cell is approximately 4 miles (increments of 0.15 degrees in latitude and longitude). For each pair of locations, say (a, b), we create a request *type* v, which represents all trips with starting and ending locations falling into a and b respectively. In our model, we have $|U| = 30$ and $|V| \approx 550$ (variations depending on day to day requests with low variance). We focus on the month of January 2013. We split the records into 31 parts, each corresponding to a day of January. We choose a random set of 12 parts for *training* purposes and use the remaining for *testing* purposes.

The edge weight w_e on $e = (u, v)$ (*i.e.,* edge from a car u to type v) is set as a function of two distances in our setup. The first is the trip distance (*i.e.,* the distance from the starting location to the ending location of v, denoted L_1) while the second is the docking distance (*i.e.,* the distance from the docking position of u to the starting/ending location of v, denoted L_2). We set $w_e = \max(L_1 - \alpha L_2, 0)$, where α is a parameter capturing the subtle balance between the positive contribution from the trip distance and negative contribution from the docking distance to the final profit. We set $\alpha = 0.5$ for the experiments. We consider each single day as the

time horizon and set the total number of rounds $T = \frac{24*60}{5} = 288$ by discretizing the 24-hour period into a time-step of 5 minutes. Throughout this section, we use time-step and round interchangeably.

Training. We use the training dataset of 12 days to learn various parameters. As for the arrival rates $\{p_{v,t}\}$, we count the total number of appearances of each request type v at time-step t in the 12 parts (denote it by $c_{v,t}$) and set $p_{v,t} = c_{v,t}/12$ under KAD (Note that $c_{v,t}$ is at most 12 and hence this value is always less than 1). When assuming KIID, we set $p_v = p_{v,t} = (c_v/12)/T$ where we have $c_v = \sum_{t\in[T]} c_{v,t}$ (*i.e.,* the arrival distributions are assumed the same across all the time-steps for each v). The estimation of parameters for the two different occupation time distributions are processed as follows. We first compute the average number of seconds between two *requests* in the dataset (note this was 5 minutes in the experimental setup). We then assume that each *time-step* of our online process corresponds to a time-difference of this average in seconds. We then compute the sample mean and sample variance of the trip lengths (as number of seconds taken by the trip divided by five minutes) in the 12 parts. Hence we use the normal distribution obtained by this sample mean and standard deviation as the distribution with which a car is unavailable. We assign the docking position of each car to the location (in the discretized space) in which the majority of the requests were initiated (*i.e.,* starting location of a request) and matched to this car.

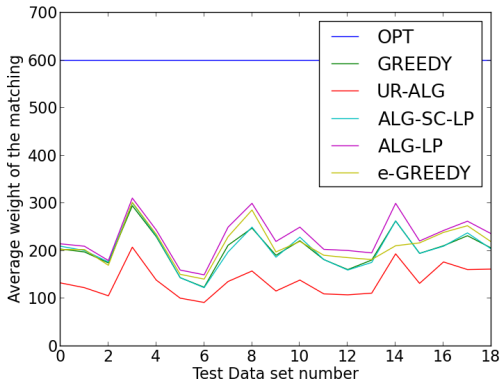

Figure 4.1: OTD is normal distribution under KIID

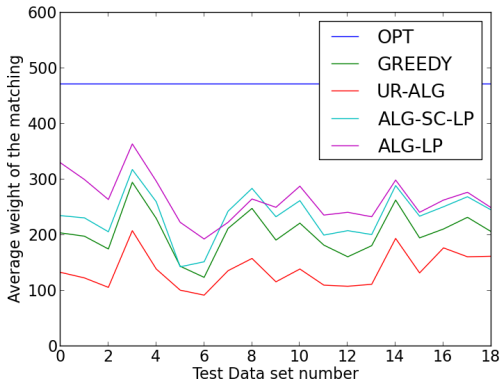

Figure 4.2: OTD is normal distribution under KAD

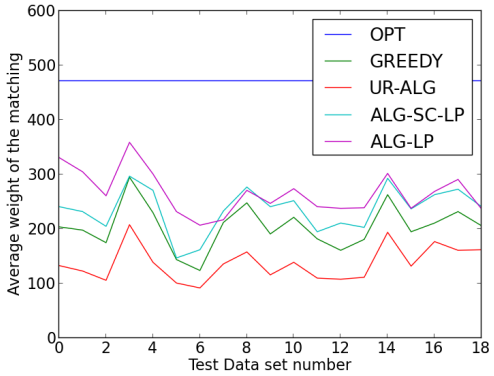

Figure 4.3: OTD is power law distribution under KAD

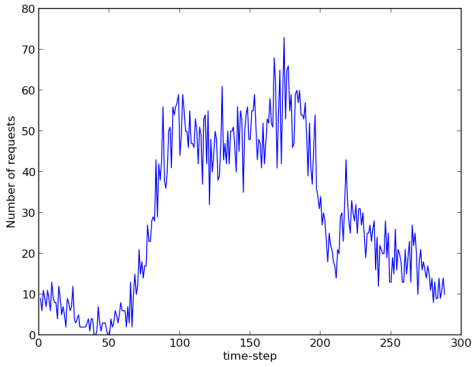

Figure 4.4: The number of requests of a given type at various time-steps. x-axis: time-step, y-aixs: number of requests

4.5.2 Justifying The Two Important Model Assumptions

Known Adversarial Distributions. Figure 4.4 plots the number of arrivals of a particular type at various times during the day. Notice the significant increase in the number of requests in the middle of the day as opposed to the mornings and nights. This justified our arrival assumption of KAD which assumes different arrival distributions at different time-steps. Hence the LP (and the corresponding algorithm) can exploit this vast difference in the arrival rates and potentially obtain improved

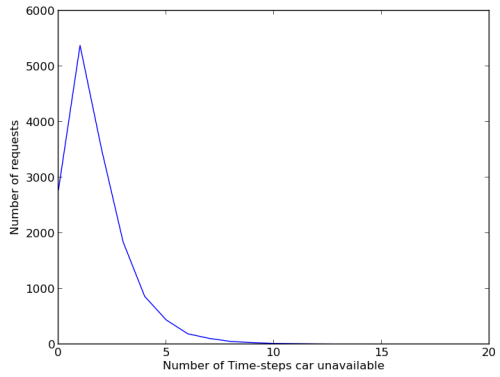

Figure 4.5: Occupation time distribution of all cars. x-axis: number of time-steps, y-axis: number of requests

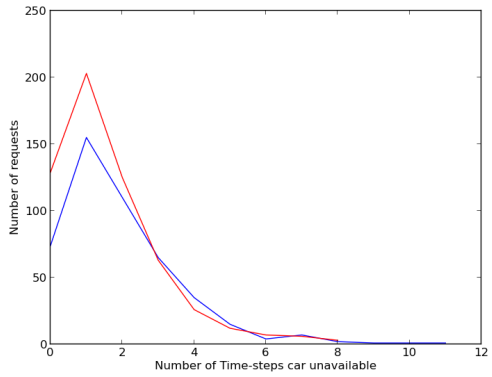

Figure 4.6: Occupation time distribution of two different cars. x-axis: number of time-steps, y-axis: number of requests

results compared to the assumption of Known Identical Independent Distributions (KIID). This is confirmed by our experimental results shown in Figures 4.1 and 4.2.

Identical-Occupation-Time Distribution. We assume each car will be available again via an independent and identical random process regardless of the matches it received. The validity of our assumptions can be seen in Figures 4.5 and 4.6, where the x-axis represents the different occupation time and the y-axis represents the corresponding number of requests in the dataset responsible for each occupation time. It is clear that for most requests the occupation time is around 2-3 time-steps and dropping drastically beyond that with a long tail. Figure 4.6 displays occupation times for two representative (we chose two out of the many cars we plotted, at random) cars in the dataset; we see that the distributions roughly coincide with each other, suggesting that such distributions can be learned from historical data and used as a guide for future matches.

4.5.3 Results

Inspired by the experimental setup by [45, 77], we run five different algorithms on our dataset. The first algorithm is the ALG-LP. In this algorithm, when a request v arrives, we choose a neighbor u with probability $x^*_{e,t}/p_{v,t}$ with $e = (u, v)$ if u is available. Here $x^*_{e,t}$ is an optimal solution to our benchmark LP and $p_{v,t}$ is the arrival rate of type v at time-step t. The second algorithm is called ALG-SC-LP. Recall that $E_{v,t}$ is the set of "safe" or available assignments with respect to v when the type v arrives at t. Let $x_{v,t} = \sum_{e \in E_{v,t}} x^*_{e,t}$. In ALG-SC-LP, we sample a safe assignment for v with probability $x^*_{e,t}/x_{v,t}$. The next two algorithms are heuristics oblivious to the underlying LP. Our third algorithm is called GREEDY which is as follows. When a request v comes, match it to the safe neighbor u with the highest edge weight. Our fourth algorithm is called UR-ALG which chooses one of the safe neighbors uniformly at random. Finally, we use a combination of LP-oblivious algorithm and LP-based algorithm called ϵ-GREEDY. In this algorithm when a type v comes, with probability ϵ we use the greedy choice and with probability $1 - \epsilon$ we use the optimal LP choice. In our algorithm, we optimized the value of ϵ and set it to $\epsilon = 0.1$. We summarize our results in the following plots. Figures 4.1, 4.2, and 4.3 show the performance of the five algorithms and OPT (optimal value of the benchmark LP) under the different assumptions of the OTD (normal or power law) and online arrives (KIID or KAD). In all three figures the x-axis represents test data-set number and the y-axis represents average weight of matching.

Discussion. From the figures, it is clear that both the LP-based solutions, namely

ALG-LP and ALG-SC-LP, do better than choosing a free neighbor uniformly at random. Additionally, with distributional assumptions the LP-based solutions outperform greedy algorithm as well. We would like to draw attention to a few interesting details in these results. Firstly, compared to the LP optimal solution, our LP-based algorithms have a competitive ratio in the range of 0.5 to 0.7. We believe this is because of our experimental setup. In particular, we have that the rates are high (> 0.1) only in a few time-steps while in all other time-steps the rates are very close to 0. This means that it resembles the structure of the *theoretical* worst case example we showed in Section 4.4.2. In future experiments, running our algorithms during *peak* periods (where the request rates are significantly larger than 0) may show that competitive ratios in those cases approach 1. Secondly, it is surprising that our algorithm is fairly robust to the *actual* distributional assumption we made. In particular, from Figures 4.2 and 4.3 it is clear that the difference between the assumption of normal distribution versus power-law distribution for the unavailability of cars is *negligible*. This is important since it might not be easy to learn the *exact* distribution in many cases (*e.g.,* cases where the sample complexity is high) and this shows that a close approximation will still be as good.

Simulation based algorithm. We omit the results of the simulation based algorithm, since the performance was similar to the algorithm without the scaling (*i.e.,* ALG-LP). Here we briefly describe the implementation details on performing the simulations efficiently in practice. The estimates are computed even before the start of the algorithm. We first simulate the entire sequence of T requests, δ times.

Using these δ samples we first compute the estimates for the first time-step. We now re-use the same δ samples and the computed estimates in the first time-step to obtain the estimates for the second time-step. Hence in a sequential manner, we compute estimates at time t using the samples from time-steps $1, 2, \ldots, t-1$. The overall run-time of this implementation is $O(\delta T + \delta T\kappa)$, where κ denotes the running time of ADAP in every time-step. Hence during the online phase, the running time of ADAP is same as that of ALG-LP.

4.6 Conclusion and Future Directions

In this work, we provide a model that captures the application of assignment in ride-sharing platforms. One key aspect in our model is to consider the *reusable* aspect of the offline resources. This helps in modeling many other important applications where agents enter and leave the system *multiple* times (*e.g.,* organ allocation, crowdsourcing markets [78], etc.). Our work opens several important research directions. The first direction is to generalize the online model to the *batch* setting. In other words, in each round we assume multiple arrivals from V. This assumption is useful in crowdsourcing markets (for example) where multiple tasks—but not all—become available at some time. The second direction is to consider a Markov model on the driver starting position. In this work, we assumed that each driver returns to her docking position. However, in many ride-sharing systems, drivers start a new trip from the position of the last-drop off. This leads to a Markovian system on the offline types, as opposed to the assumed static types in the present work. Finally,

pairing our current work with more applied stochastic optimization and reinforcement learning approaches would be of practical interest to policymakers running taxi and bikeshare services [46, 79–82].

Chapter 5: More Applications of Online Matching

In this chapter, we briefly discuss applications of online matching models in several other matching markets, such as online recommendation systems, taxi-dispatching platforms, and online task assignment platforms.

Online Recommendations. Most prior research on online matching focuses on maximizing the total weight of the final matching [22], which captures the quality/relevance of all the matches. In many matching markets, we also care about the *diversity* of the final matching along with relevance. Consider the example of matching academic papers to potential reviewers: just maximizing the relevance (the quality of each match) could potentially assign a paper to multiple scholars in a single lab due to shared expertise, which is undesirable. Instead, we want to assign each paper to *relevant* experts with diverse backgrounds to obtain comprehensive feedback. Maximizing diversity[1] is of particular importance in various recommendation systems, ranging from recommendations of new books and movies on eBay [84] to returning search-engine queries [85]. A common strategy to address diversity is to

[1]Both individual and aggregate diversity [83].

first formulate a specific objective (typically maximization over a submodular function) capturing the balance of diversity and relevance and then design an efficient algorithm—typically a greedy one—to solve it (*e.g.,* [86] and references within).

We proposed a new model, Online Submodular bipartite matching, which effectively captures notions such as relevance and diversity in matching markets. Many applications such as advertising, hiring diverse candidates, recommending movies or songs naturally fit within this framework. We designed two algorithms, one based on contention-resolution schemes and the other based on using the solution of the mathematical program directly; we gave theoretical guarantees on their performance. The algorithm using the mathematical program directly is essentially tight even for the special case of linear objectives. We also showed that our algorithms do well in practice via intensive synthetic and real experiments. Additionally, we proposed heuristics, some of which perform well on specialized submodular functions, and showed that our general algorithm is competitive with such algorithms as well. More details can be checked in the paper [87].

Taxi-Dispatching Platforms. It is important for taxi dispatching platforms to account for human factors such as preferences of workers and tasks. Prior work has integrated the preferences of *either* workers *or* tasks into the optimization objectives. For example, some researchers propose to minimize the sum of distances between the origin of each task and the matched worker over all matches [88–90]. This way the overall waiting time of all passengers is reduced. Others maximize the total utility obtained through all successful matches, where the utility is defined to

depict the workers' preference on payment [91, 92]. Despite these pioneer studies, the preference of only *one side* (workers *or* tasks) is considered. We argue that the matching policies should reflect the preferences of *both sides* (workers *and* tasks), which we will illustrate via the following example.

Image during the rush hours of Monday, Alice requested a taxi on Uber to take her from home to office for a short-distance trip. At the same time, Bob appeared on Uber as driver and he happened to be close to Alice's home. To minimize the waiting time of Alice, Uber should match Alice and Bob. However, this might hurt Bob's interest. This is because during rush hours passengers far outnumber drivers. Thus Bob preferred to wait for requests of long-distance rides to earn more profit. A question arises whether Uber should reject Alice or assign her to Bob. This is a common example in on-demand taxi dispatching platforms, where the preferences of workers and tasks may differ or even conflict with each other. That is, a passenger may prefer to be picked up immediately by a driver nearby while the driver may prefer to wait for long-distance rides. A natural question is: How can we design matching policies to reconcile the preferences of both the workers and tasks such that they are satisfied to a best degree?

We proposed an online stable matching model under KIID to address the preference-aware task assignment problem in taxi-dispatching applications. The model features two objectives: maximization of the total profit and minimization of the overall dissatisfaction about preferences among workers and tasks. We constructed an LP, which proves to be a valid upper bound on the expected maximum profit on the offline optimal stable matching. We further proposed an LP-based on-

line algorithm, which achieves an online ratio of at least $1-1/e$ on the first objective and maintains the ratio of the expected number of total blocking edges to that of the total edges at most 0.6. More details can be see in the paper [93].

Online Task Assignment Platforms. In most matching models for online task assignment problems, we assume that tasks are static (known in advance). This fails to capture various applications where the tasks are not all available at once and come in an online manner similar to the workers. This is a common scenario in spatial crowdsourcing platforms. [94] considered a practical worker-task assignment under a converse setting to that in a typical crowdsourcing human resource market, where the spatial tasks come dynamically while the workers are static. The worker has to travel to the specific location of the task to finish it. [45] studied a generalized setting where both workers and tasks come online which was motivated from a spatial crowdsourcing platform on university campus, where anyone on campus can both post micro-tasks, (*e.g.,* buying drinks or collecting a package), and perform tasks as a worker. They assumed that the arrivals is sampled from the distribution over all permutations of both workers and tasks together and is unknown to the algorithm. They tested their algorithms on two real-world crowdsourcing datasets, namely gMission [95] and EverySender.

Inspired by the above work ([45,94]), we proposed the online task assignment with two-sided arrival where both workers and tasks come online but under the arrival setting of KIID. Let us first briefly review a typical setting of task assignment under KIID — a known bipartite graph $G = (U, V, E)$ is given as input (this graph is

also called the compatibility graph throughout this paper), where U and V represent the respective set of worker-types and task-types; we have a finite time horizon of T in which vertices in U are revealed step-by-step in each time-step (while all vertices in V are already given). In every time-step a worker of a particular type is sampled from a known distribution over U and the samples are independent across all the T rounds. We generalized the KIID setting from one-sided arrival to two-sided arrival in the following natural way — in each round (for a total of T rounds) a worker of type u is sampled from a known distribution over U, while simultaneously a task of type v is sampled from another known distribution over V independently.

We presented an optimal non-adaptive algorithm which achieves an online competitive ratio of 0.295. For the special case where the reward is a function of just the worker type, we present an improved algorithm (which is adaptive) and achieves a competitive ratio of at least 0.343. On the hardness side, along with showing that the ratio obtained by our non-adaptive algorithm is the best possible among all non-adaptive algorithms, we further show that no (adaptive) algorithm can achieve a ratio better than 0.581 (unconditionally), even for the special case with homogenous tasks (*i.e.,* all rewards are same). At the heart of our analysis lies a new technical tool (which is a refined notion of the birth-death process), called the two-stage birth-death process. We also performed numerical experiments on two real-world datasets obtained from crowdsourcing platforms to complement our

theoretical results. More details can be seen in the paper [96].

Chapter 6: Conclusion and Future Work

This dissertation presented two fundamental matching models, namely stochastic matching and online matching, and an application of online matching in ridesharing platforms. It offers examples regarding we utilize online matching techniques to leverage those estimations from machine learning to optimize the matching policy in various matching markets. In the following, we brief list a few future directions.

The first direction is to remove the independence from the arrival assumption. Current literature mainly considers the following three arrival assumptions: adversarial (no information is known for the full arrival sequence), random arrival order (the full arrival sequence forms a random permutation order) and known distributions (each time a vertex is sampled from a known distribution and the sampling is independent over time). My current research primarily focuses on the last one and in the future, I plan to consider the following variant of known distributions: what if the sampling distributions are dependent across different time-steps? The correlation on the arrivals of different types of online agents (*e.g.,* users, workers and tasks) can often be observed in real datasets. That will be a great challenge to online algorithm design.

The second direction is about variance reduction in online algorithm design.

Due to the nature of online algorithms, it can be implemented only once and thus, the optimization over the expected performance (often captured by maximization or minimization an expectation) is far from enough. We hope to design an online algorithm, which achieves not only high expected performance (*i.e.,* effectiveness) but also low variance (*i.e.,* robustness). How to balance these two objectives? Is there any tradeoff between the two, similar to the bias-variance tradeoff common in machine learning?

Bibliography

[1] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 2588–2595, 2018.

[2] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. Multi-task representation learning for travel time estimation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1695–1704, 2018.

[3] Zheng Wang, Kun Fu, and Jieping Ye. Learning to estimate the travel time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 858–866, 2018.

[4] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[5] David P Williamson and David B Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[6] C. M. Fortuin, P. W. Kasteleyn, and J. Ginibre. Correlation inequalities on some partially ordered sets. *Comm. Math. Phys.*, 22(2):89–103, 1971.

[7] Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012.

[8] H. Shachnai and A. Srinivasan. Finding large independent sets in graphs and hypergraphs. *SIAM Journal on Discrete Mathematics*, 18:488–500, 2004.

[9] Noga Alon and Joel H Spencer. Wiley interscience series in discrete mathematics and optimization. *The probabilistic method*, pages 353–354, 2008.

[10] Ning Chen, Nicole Immorlica, Anna R Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. *Automata, Languages and Programming*, pages 266–278, 2009.

[11] Marek Adamczyk. Greedy algorithm for stochastic matching is a 2-approximation. *arXiv preprint arXiv:1007.3036*, 2010.

[12] Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012.

[13] Brian C Dean, Michel X Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

[14] Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM (JACM)*, 53(3):324–360, 2006.

[15] Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. Improved approximation algorithms for stochastic matching. In *Algorithms-ESA 2015*, pages 1–12. Springer, 2015.

[16] Avrim Blum, Anupam Gupta, Ariel Procaccia, and Ankit Sharma. Harnessing the power of two crossmatches. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 123–140. ACM, 2013.

[17] Avrim Blum, John P Dickerson, Nika Haghtalab, Ariel D Procaccia, Tuomas Sandholm, and Ankit Sharma. Ignorance is almost bliss: Near-optimal stochastic matching with few queries. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 325–342. ACM, 2015.

[18] Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem with (very) few queries. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 43–60. ACM, 2016.

[19] Kevin Costello, Prasad Tetali, and Pushkar Tripathi. Stochastic matching with commitment. *Automata, Languages, and Programming*, pages 822–833, 2012.

[20] Alok Baveja, Amit Chavan, Andrei Nikiforov, Aravind Srinivasan, and Pan Xu. Improved bounds in stochastic matching and optimization. *Algorithmica*, 80(11):3225–3252, 2018.

[21] Zoltán Füredi, Jeff Kahn, and Paul D. Seymour. On the fractional matching polytope of a hypergraph. *Combinatorica*, 13(2):167–180, 1993.

[22] Aranyak Mehta. Online matching and ad allocation. *Foundations and Trends in Theoretical Computer Science*, 8(4):265–368, 2012.

[23] Jon Feldman, Nitish Korula, Vahab Mirrokni, S Muthukrishnan, and Martin Pál. Online ad assignment with free disposal. In *International Workshop on Internet and Network Economics*, pages 374–385. Springer, 2009.

[24] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1253–1264. SIAM, 2011.

[25] Aranyak Mehta and Debmalya Panigrahi. Online matching with stochastic rewards. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 728–737. IEEE, 2012.

[26] Aranyak Mehta, Bo Waggoner, and Morteza Zadimoghaddam. Online stochastic matching with unequal probabilities. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1388–1404. SIAM, 2014.

[27] Bernhard Haeupler, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In *Internet and Network Economics*, volume 7090 of *Lecture Notes in Computer Science*, pages 170–181. Springer Berlin Heidelberg, 2011.

[28] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358. ACM, 1990.

[29] Xiaoming Sun, Jia Zhang, and Jialin Zhang. Near optimal algorithms for online weighted bipartite matching in adversary model. *Journal of Combinatorial Optimization*, pages 1–17, 2016.

[30] Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *European Symposium on Algorithms*, pages 253–264. Springer, 2007.

[31] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22, 2007.

[32] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 597–606. ACM, 2011.

[33] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 587–596. ACM, 2011.

[34] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 982–991. Society for Industrial and Applied Mathematics, 2008.

[35] Nikhil R Devanur and Thomas P Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 71–78. ACM, 2009.

[36] Nikhil R Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 29–38. ACM, 2011.

[37] Nikhil R Devanur, Balasubramanian Sivan, and Yossi Azar. Asymptotically optimal algorithm for stochastic adwords. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 388–404. ACM, 2012.

[38] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 117–126. IEEE, 2009.

[39] Bernhard Haeupler, Vahab S Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In *International Workshop on Internet and Network Economics*, pages 170–181. Springer, 2011.

[40] Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573, 2012.

[41] Patrick Jaillet and Xin Lu. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 39(3):624–646, 2013.

[42] Brian Brubach, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. New algorithms, better bounds, and a novel model for online stochastic matching. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 57. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, ESA, 2016.

[43] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 18–35. ACM, 2012.

[44] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. The online stochastic generalized assignment problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 11–25. Springer, 2013.

[45] Yongxin Tong, Jieying She, Bolin Ding, Lei Chen, Tianyu Wo, and Ke Xu. Online minimum matching in real-time spatial data: experiments and analysis. *Proceedings of the VLDB Endowment*, 2016.

[46] Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. Online spatio-temporal matching in stochastic and dynamic domains. *Artificial Intelligence*, 261:71–112, 2018.

[47] Der-Horng Lee, Hao Wang, Ruey Cheu, and Siew Teo. Taxi dispatch system based on current demands and real-time traffic conditions. *Transportation Research Record: Journal of the Transportation Research Board*, (1882), 2004.

[48] Kiam Tian Seow, Nam Hai Dang, and Der-Horng Lee. A collaborative multiagent taxi-dispatch system. *IEEE Transactions on Automation Science and Engineering*, 7(3), 2010.

[49] Brendon L Neuen, Georgina E Taylor, Alessandro R Demaio, and Vlado Perkovic. Global kidney disease. *The Lancet*, 382(9900), 2013.

[50] Rajiv Saran, Yi Li, Bruce Robinson, John Ayanian, Rajesh Balkrishnan, Jennifer Bragg-Gresham, JT Chen, Elizabeth Cope, Debbie Gipson, Kevin He, et al. US renal data system 2014 annual data report: Epidemiology of kidney disease in the United States. *American Journal of Kidney Diseases*, 65(6 Suppl 1), 2015.

[51] Dimitris Bertsimas, Vivek F Farias, and Nikolaos Trichakis. Fairness, efficiency, and flexibility in organ allocation for kidney transplantation. *Operations Research*, 61(1), 2013.

[52] John P Dickerson and Tuomas Sandholm. Futurematch: Combining human value judgments and machine learning to match in dynamic environments. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[53] Nicholas Mattei, Abdallah Saffidine, and Toby Walsh. Mechanisms for online organ matching. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 345–351, 2017.

[54] Man Lung Yiu, Kyriakos Mouratidis, Nikos Mamoulis, et al. Capacity constrained assignment in spatial databases. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 15–28. ACM, 2008.

[55] Michael Miller. *Cloud computing: Web-based applications that change the way you work and collaborate online*. Que publishing, 2008.

[56] Andrew J Younge, Gregor Von Laszewski, Lizhe Wang, Sonia Lopez-Alarcon, and Warren Carithers. Efficient resource management for cloud computing environments. In *International Green Computing Conference*, 2010.

[57] Nicole Megow, Marc Uetz, and Tjark Vredeveld. Stochastic online scheduling on parallel machines. In *International Workshop on Approximation and Online Algorithms*, pages 167–180. Springer, 2004.

[58] Nicole Megow, Marc Uetz, and Tjark Vredeveld. Models and algorithms for stochastic online scheduling. *Mathematics of Operations Research*, 31(3), 2006.

[59] Martin Skutella, Maxim Sviridenko, and Marc Uetz. Unrelated machine scheduling with stochastic processing times. *Mathematics of operations research*, 41(3), 2016.

[60] Erhun Ozkan and Amy Ward. Dynamic matching for real-time ridesharing. *Working Paper.*, 2017.

[61] Anton Braverman, Jim G Dai, Xin Liu, and Lei Ying. Empty-car routing in ridesharing systems. *arXiv preprint arXiv:1609.07219*, 2016.

[62] Siddhartha Banerjee, Daniel Freund, and Thodoris Lykouris. Pricing and optimization in shared vehicle systems: An approximation framework. *arXiv preprint arXiv:1608.06819*, 2016.

[63] Daniel Adelman. Price-directed control of a closed logistics queueing network. *Operations Research*, 2007.

[64] Ariel Waserhole and Vincent Jost. Pricing in vehicle sharing systems: Optimization in queuing networks with product forms. *EURO Journal on Transportation and Logistics*, 2016.

[65] Jiang Rong, Tao Qin, and Bo An. Dynamic pricing for reusable resources in competitive market with stochastic demand. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[66] Siddhartha Banerjee, Ramesh Johari, and Carlos Riquelme. Dynamic pricing in ridesharing platforms. *ACM SIGecom Exchanges*, 2016.

[67] Juan Camilo Castillo, Dan Knoepfle, and Glen Weyl. Surge pricing solves the wild goose chase. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 241–242. ACM, 2017.

[68] Afshin Nikzad. Thickness and competition in ride-sharing markets. *Available at SSRN 3065672*, 2017.

[69] Yong Sun, Jun Wang, and Wenan Tan. Online algorithms of task allocation in spatial crowdsourcing. In *Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing*, pages 205–208. ACM, 2017.

[70] Mohammad Akbarpour, Shengwu Li, and Shayan Oveis Gharan. Dynamic matching market design. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, EC '14, pages 355–355, New York, NY, USA, 2014. ACM.

[71] Ross Anderson, Itai Ashlagi, David Gamarnik, and Yash Kanoria. Efficient dynamic barter exchange. *Operations Research*, 65(6):1446–1459, 2017.

[72] Itai Ashlagi, Maximilien Burq, Patrick Jaillet, and Vahideh Manshadi. On matching and thickness in heterogeneous dynamic markets. *Available at SSRN 3067596*, 2018.

[73] Itai Ashlagi, Patrick Jaillet, and Vahideh H. Manshadi. Kidney exchange in dynamic sparse heterogenous pools. In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce*, EC '13, pages 25–26, New York, NY, USA, 2013. ACM.

[74] Will Ma. Improvements and generalizations of stochastic knapsack and multi-armed bandit approximation algorithms. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1154–1163. Society for Industrial and Applied Mathematics, 2014.

[75] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[76] Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. Improved approximation algorithms for stochastic matching. In *Algorithms-ESA 2015*, pages 1–12. Springer, 2015.

[77] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. Online mobile micro-task allocation in spatial crowdsourcing. In *2016 IEEE 32Nd international conference on data engineering (ICDE)*, pages 49–60. IEEE, 2016.

[78] Chien-Ju Ho and Jennifer Wortman Vaughan. Online task assignment in crowdsourcing markets. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, volume 12, pages 45–51, 2012.

[79] Divya Singhvi, Somya Singhvi, Peter I Frazier, Shane G Henderson, Eoin O'Mahony, David B Shmoys, and Dawn B Woodard. Predicting bike usage for new york city's bike sharing system. In *AAAI-15 Workshop on Computational Sustainability*, 2015.

[80] Eoin O'Mahony and David B Shmoys. Data analysis and optimization for (citi) bike sharing. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[81] Tanvi Verma, Pradeep Varakantham, Sarit Kraus, and Hoong Chuin Lau. Augmenting decisions of taxi drivers through reinforcement learning for improving

revenues. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*, 2017.

[82] Supriyo Ghosh, Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. Dynamic repositioning to reduce lost demand in bike sharing systems. *Journal of Artificial Intelligence Research (JAIR)*, 58:387–430, 2017.

[83] Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):896–911, 2012.

[84] C. Chen, L. Zheng, V. Srinivasan, A. Thomo, K. Wu, and A. Sukow. Conflict-aware weighted bipartite b-matching and its application to e-commerce. *IEEE TKDE*, 2016.

[85] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the second ACM international conference on web search and data mining*, pages 5–14. ACM, 2009.

[86] Faez Ahmed, John P Dickerson, and Mark Fuge. Diverse weighted bipartite b-matching. *arXiv preprint arXiv:1702.07134*, 2017.

[87] John P Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. Balancing relevance and diversity in online bipartite matching via submodularity. *arXiv preprint arXiv:1811.05100*, 2018.

[88] Xiaohui Bei and Shengyu Zhang. Algorithms for trip-vehicle assignment in ride-sharing. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 3–9, 2018.

[89] Yongxin Tong, Jieying She, Bolin Ding, Lei Chen, Tianyu Wo, and Ke Xu. Online minimum matching in real-time spatial data: experiments and analysis. *PVLDB*, 9(12):1053–1064, 2016.

[90] Nikhil Bansal, Niv Buchbinder, Anupam Gupta, and Joseph Naor. A randomized $o(log^2 k)$-competitive algorithm for metric bipartite matching. *Algorithmica*, 68(2):390–403, 2014.

[91] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. Online mobile micro-task allocation in spatial crowdsourcing. In *32nd IEEE International Conference on Data Engineering*, pages 49–60, 2016.

[92] John P. Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1007–1014, 2018.

[93] Boming Zhao, Pan Xu, Yexuan Shi, Yongxin Tong, Zimu Zhou, and Yuxiang Zeng. Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2019.

[94] U. U. Hassan and E. Curry. A multi-armed bandit approach to online spatial task assignment. In *11th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC-14)*, pages 212–219, Dec 2014.

[95] Zhao Chen, Rui Fu, Ziyuan Zhao, Zheng Liu, Leihao Xia, Lei Chen, Peng Cheng, Caleb Chen Cao, Yongxin Tong, and Chen Jason Zhang. gmission: A general spatial crowdsourcing platform. *Proceedings of the VLDB Endowment*, 7(13), 2014.

[96] John P. Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. Assigning tasks to workers based on historical data: Online task assignment with two-sided arrivals. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pages 318–326, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.