Allocation Problems in Ride-Sharing Platforms: Online Matching with Offline Reusable Resources

JOHN P. DICKERSON, University of Maryland, College Park, USA
KARTHIK A. SANKARARAMAN, Facebook, USA
ARAVIND SRINIVASAN, University of Maryland, College Park, USA
PAN XU, New Jersey Institute of Technology, Newark, USA

Bipartite-matching markets pair agents on one side of a market with agents, items, or contracts on the opposing side. Prior work addresses online bipartite-matching markets, where agents arrive over time and are dynamically matched to a known set of disposable resources. In this paper, we propose a new model, *Online Matching with (offline) Reusable Resources under Known Adversarial Distributions (*OM-RR-KAD*)*, in which resources on the offline side are *reusable* instead of disposable; that is, once matched, resources become available again at some point in the future. We show that our model is tractable by presenting an LP-based non-adaptive algorithm that achieves an online competitive ratio of $\frac{1}{2} - \epsilon$ for any given constant $\epsilon > 0$. We also show that no adaptive algorithm can achieve a ratio of $\frac{1}{2} + o(1)$ based on the same benchmark LP. Through a data-driven analysis on a massive openly-available dataset, we show our model is robust enough to capture the application of taxi dispatching services and ride-sharing systems. We also present heuristics that perform well in practice.

Additional Key Words and Phrases: Online-Matching, Ride-Sharing, Randomized Algorithms, Auction Design

ACM Reference Format:
John P. Dickerson, Karthik A. Sankararaman, Aravind Srinivasan, and Pan Xu. 2018. Allocation Problems in Ride-Sharing Platforms: Online Matching with Offline Reusable Resources. *ACM Transactions on Economics and Computation* 0, 0, Article 0 (January 2018), 17 pages. https://doi.org/0000001.0000001

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.

Work done when KAS was a PhD student at University of Maryland, College Park. There, KAS supported in part by NSF Awards CNS 1010789 and CCF 1422569. PX was partially supported by NSF CRII Award IIS-1948157 and partially by NSF Awards CNS 1010789 and CCF 1422569. AS supported in part by NSF Awards CNS-1010789, CCF-1422569 and CCF-1749864, and by research awards from Adobe, Inc., Amazon, Inc., and Google Inc. JD was supported in part by NSF CAREER Award IIS-1846237, NIST MSE Award #20126334, DARPA GARD #HR00112020007, DARPA SI3-CMD #S4761, DoD WHS Award #HQ003420F0035, and a Google Faculty Research Award. A preliminary version of this work appeared in the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18) under the same title.
Authors' addresses: John P. Dickerson, University of Maryland, College Park, USA, john@cs.umd.edu; Karthik A. Sankararaman, Facebook, USA, karthikabinavs@gmail.com; Aravind Srinivasan, University of Maryland, College Park, USA, asriniv1@umd.edu; Pan Xu, New Jersey Institute of Technology, Newark, USA, pxu@njit.edu.

XXXX-XXXX/2018/1-ART0 $15.00
https://doi.org/0000001.0000001

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" u in G 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 Karp et al. [29], there has been a large body of research on related variants (overviewed by Mehta [40]). One particular flavor of problems is online-matching with known identical independent distributions (OM-KIID) [14, 21, 26, 28, 36]. In this context, 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 (*i.e.,* they are not i.i.d.). Consider the following motivating examples.

Taxi Dispatching Services and Ride-Sharing Systems. Traditional taxi services and ride-sharing systems such as Uber and Didi Chuxing match drivers to would-be riders [30, 31, 50, 55]. 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. Though our model is primarily motivated by taxi-dispatching services and ride-sharing platforms, we acknowledge that it does not address the full generality of these applications. In particular, our work focuses on the temporal components of rideshare; spatial components such as match-specific driver movement are not modeled [17, 18, 33]. Since we make the simplifying assumption that the stochastic process that determines the parameters in the environment is independent of the algorithm's decisions, this may not fully capture the scenario where requests are heterogenous with vastly-different ride times. One potential direction to extend this model is to consider correlated reusable rates, which makes the theory much more challenging.

Organ Allocation. Chronic kidney disease affects tens of millions of people worldwide at great societal and monetary cost [43, 49]. 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 [12, 20, 37], 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. Similarly, in some blood or plasma donation settings, donors might reappear after some number of weeks.

Similar scenarios can be seen in other areas such as wireless network connection management (SPs are different wireless access points) [59] and online cloud computing service scheduling [42, 60]. 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 $t \in [T]$ [1], 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 [4, 5] and is known as 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 $t \in [T]$.

Competitive Ratio. Let $\mathrm{E}[\mathrm{ALG}(\mathcal{I}, \mathcal{D})]$ denote the expected value obtained by an algorithm ALG on an input $\mathcal{I}$ and arrival distribution $\mathcal{D}$. Let $\mathbf{E}[\mathrm{OPT}(\mathcal{I})]$ denote the expected *offline optimal value*, which refers to the optimal solution when we are allowed to make choices after observing the entire sequence of online arrival vertices. Then, the competitive ratio is defined as $\min_{\mathcal{I}, \mathcal{D}} \frac{\mathrm{E}[\mathrm{ALG}(\mathcal{I}, \mathcal{D})]}{\mathrm{E}[\mathrm{OPT}(\mathcal{I})]}$. It is a common technique to use an LP optimal value to upper bound $\mathbf{E}[\mathrm{OPT}(\mathcal{I})]$ (called the benchmark LP) and hence get a valid lower bound on the resulting competitive ratio.

Adaptive vs. non-adaptive algorithms. An online algorithm ALG is called *non-adaptive* if the strategy for all time-steps t, denoted by $\mathrm{ALG}(t)$, is pre-computed *before* the realizations of the online process. In contrast, adaptive algorithms can choose the strategy for time-step t *after* seeing the random realizations of all the random processes (*e.g.,* random seeds used in the algorithm, arrivals of online requests and occupation times of drivers) in steps $1, 2, \ldots, t-1$.

1.1 Our Contributions

First, we propose the new model of OM-RR-KAD to capture a wide range of real-world applications related to online scheduling, organ allocation, rideshare dispatch, among others. We claim that this model is tractable enough to obtain good algorithms with theoretically provable guarantees and general enough to capture many real-life instances. Our model assumptions take a significant step forward from the usual assumptions in the online-matching literature where the offline side is assumed to be *single-use* or *disposable*. This leads to a larger range of potential applications which can be modeled by online-matching. The first part of the paper focusses on the abstract model and though the experiments are motivated using ride-share, we do not get into the application-specific assumptions/issues one may run into. For specific discussion on issues that arise when applied to ride-share, we defer to Section 5.

Second, we show how this model can be *cleanly* analyzed under a theoretical framework. We first construct a linear program (LP henceforth) LP (1) which we show is a valid upper bound on the expected offline optimal value (note that the latter is hard to characterize). Next, we propose

[1]Throughout this paper, we use $[N]$ to denote the set $\{1, 2, \ldots, N\}$, for any positive integer N.

an efficient *non-adaptive* algorithm that achieves a competitive ratio of $\frac{1}{2} - \epsilon$ for any given constant $\epsilon > 0$. This algorithm solves the LP and obtains an optimal fractional solution. It uses this optimal solution as a guide in the online phase. Using Monte-Carlo simulations (called simulations henceforth), and combining with this optimal solution, our algorithm makes the online decisions. In particular, Theorem 1 describes our first theoretical results formally.

THEOREM 1. LP (1) *is a valid benchmark for* OM-RR-KAD. *There exists a non-adaptive online algorithm that achieves an online competitive ratio of* $\frac{1}{2} - \epsilon$ *for any given* $\epsilon > 0$ *against the benchmark* LP (1).

Third, we show that the online-competitive analysis of the non-adaptive algorithm is *tight* with respect to the choice of our benchmark LP. Specifically, we show that when restricted to LP (1), no adaptive algorithm can achieve a competitive ratio better than $\frac{1}{2}$ (Theorem 2), and no non-adaptive algorithm can beat $\frac{1}{2}$ even when all C_e are deterministic and equal (Theorem 3).

THEOREM 2. *No adaptive algorithm can achieve a competitive ratio better than* $\frac{1}{2} + o(1)$ *against the benchmark* LP (1). *Here,* $o(1)$ *is a vanishing term when* T *is sufficiently large.*

THEOREM 3. *No non-adaptive algorithm can achieve a competitive ratio better than* $\frac{1}{2} + o(1)$ *against the benchmark* LP (1) *even when all* C_e *are deterministic and equal. Here,* $o(1)$ *is a vanishing term when both of* C_e *and* T/C_e *are sufficiently large.*

Finally, through a data-driven analysis on a massive openly-available dataset we show that our model is robust enough to capture the setting of taxi hailing/sharing at least. Additionally, we provide certain *simpler* heuristics which also give good performance. Hence, we can combine these theoretically grounded algorithms with such heuristics to obtain further improved ratios in practice. Section 5 provides a detailed qualitative and quantitative discussion.

1.2 Other Related Work

In addition to the arrival assumptions of KIID and KAD, there are several other important, well-studied variants of online-matching problems. Under *adversarial ordering*, an adversary can arrange the arrival order of all items in an arbitrary way (e.g., online-matching [29, 53] and AdWords [16, 41]). Under a *random arrival order*, all items arrive in a random permutation order (e.g., online-matching [35] and AdWords [24]). Finally, under *unknown distributions*, in each round, an item is sampled from a fixed but unknown distribution. (e.g., [19]). For each of the categories above, we list only a few examples considered under that setting. For a more complete list, please refer to the book by Mehta [40].

Despite the fact that our model is inspired by online bipartite-matching, it also overlaps with stochastic online scheduling problems (SOS) [38, 39, 52]. 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. In a recent concurrent work [22], they consider the dynamic assortment of reusable resources. In their context, the offline side is the set of reusable resources, while the online side is the set of consumers. They consider the arrival setting of KAD (which they call the Bayesian model with non-identical distributions). The critical difference is that they assume that in each round, on the arrival of an online customer, the algorithm should assign her a set of offline resources, where each offline resource has a given budget. The benchmark LP has an exponential number of variables, which is not solvable in polynomial time. They overcome this by assuming an offline oracle, that returns an optimal solution to the benchmark LP for any given arrival sequence of online customers. They design a similar simulation-based online policy that achieve a competitive ratio of $1/2$.

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 [11, 13, 46]. 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.,* [2, 10, 17, 33, 44, 47, 58]). 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, [20, 31, 54, 55] have studied algorithms for matching in various dynamic bipartite markets such as kidney exchange, spatial crowdsourcing, labor markets, and so on. A similar line of work on general graphs is also prominent in the literature (*e.g.,* [3, 6–8]).

2 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 $\sum_{v \in V} p_{v,t} \leq 1$[2] (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 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 (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 (2) is exactly the expected number of assignments made at t for v. It should be no more than the probability that v arrives at t, which is the RHS of (2). Constraint (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 probability that u is not available at t while the second term (summation over $e \in E_u$) is the probability that u is assigned to some driver at t, which is no larger than probability u is available at t. Thus, the sum of the first term and second term on LHS is no larger than 1.[3] This argument implies that the LP forms a valid upper-bound on the offline optimal solution and hence we have the first part of Lemma 4.

LEMMA 4. *The optimal value to* LP (1) *is a valid upper bound for the offline optimal. Moreover, suppose for some $\delta \geq 0$, we have an estimate $f(e, y)$ of* $\Pr[C_e > y]$ *for all edges e and $y \geq 0$, where*

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

[3]We would like to point out that our LP constraint (3) on u is inspired by Ma [34]. The proof is similar to that by Alaei et al. [4] and Alaei et al. [5].

$$\text{maximize} \sum_{t\in[T]}\sum_{e\in E} w_e x_{e,t} \tag{1}$$

$$\text{subject to} \sum_{e\in E_v} x_{e,t} \le p_{v,t} \qquad \forall v\in V, t\in[T] \tag{2}$$

$$\sum_{t'<t}\sum_{e\in E_u} x_{e,t'} \Pr[C_e > t-t'] + \sum_{e\in E_u} x_{e,t} \le 1 \qquad \forall u\in U, t\in[T] \tag{3}$$

$$0 \le x_{e,t} \le 1 \qquad \forall e\in E, t\in[T] \tag{4}$$

$f(e,y)/\Pr[C_e > y]$ always lies in $[1/(1+\delta), 1+\delta]$. Then, by using $f(e,t-t')$ in the LP instead of $\Pr[C_e > t-t']$ and scaling down the resultant vector $\mathbf{x}$ by $(1+\delta)$, we only get a further loss of $(1+\delta)$ in the competitive ratio.

PROOF. Fix any offline optimal algorithm OPT. For each assignment $e=(u,v)$ and t, let $X_{e,t}$ denote an indicator random variable for the event that e is matched at t in OPT, which includes the event that v arrives at time t. Let $\mathbb{E}[X_{e,t}] = x_{e,t}$. Therefore, by linearity of expectation, the expected performance of OPT is $\mathbf{E}[\text{OPT}] = \sum_t \sum_e w_e x_{e,t}$. Now we justify that the solution $\{x_{e,t}\}$ is feasible to all constraints in LP (1).

In constraint (2), the LHS denotes the probability that v is matched at t, which should be no larger than the probability that vertex v arrives at time t. In constraint (3), the first part $\sum_{t'<t}\sum_{e\in E_u} x_{e,t'}\Pr[C_e > t-t']$ denotes the probability that u is occupied due to assignments made prior to t while the second part $\sum_{e\in E_u} x_{e,t}$ is the probability that an assignment incident to u is made at time t. Thus, the sum of these two parts should be no larger than 1. Constraint (4) is satisfied since $\{x_{e,t}\}$ are all probability values. Therefore, we have shown that the values $\{x_{e,t}\}$ is feasible to all constraints in LP (1), which implies that the optimal value of LP (1) is a valid upper bound for the performance of any offline optimal.

The second part follows directly from the fact that $\mathbf{x}$ is scaled down by a factor $(1+\delta)$ and hence the objective is scaled down by a factor $(1+\delta)$. □

3 SIMULATION-BASED ALGORITHM

In this section, we present a simulation-based algorithm. We will first give a gentle introduction to simulation-based algorithms, that has been developed and used in prior works ([1] and [15]).

Simulation-based algorithms. We use the term *simulation* throughout this paper to refer to Monte Carlo simulation and the term *simulation-based attenuation* to refer to the simulation and attenuation techniques as shown in [1] and [15][4]. At a high level, suppose we have a randomized algorithm such that for some event E (*i.e., u* is available at t) we have $\Pr[E] \ge c$, then we modify the algorithm as follows: (i) We first use simulation to estimate a value $\hat{E}$ that lies in the range $[\Pr[E], (1+\epsilon)\Pr[E]]$ with probability at least $1-\delta$. (ii) By "ignoring" E (i.e., *attenuation*, in a problem-specific manner) with probability $\sim 1 - c/\hat{E}$, we can ensure that the final effective value of $\Pr[E]$ is arbitrarily close to c, i.e., in the range $[c/(1+\epsilon), c]$ with probability at least $1-\delta$. This simple idea of attenuating the probability of an event to come down approximately to a certain value c is what we term simulation-based attenuation. The number of samples needed to obtain the estimate $\hat{E}$ is $\Theta(\frac{1}{c\epsilon^2}\cdot\log(\frac{1}{\delta}))$ via a standard Chernoff-bound argument. In our applications, we will take $\epsilon = 1/\text{poly}(N)$ where N is the problem-size, and the error ϵ will only impact lower-order terms in our approximations.

[4]This is called "dumping factor" in [1]. See Appendix B in [1] for a formal treatment.

We adapt the above simulation-based attenuation technique to our setting. Let $\mathbf{x}^*$ denote an optimal solution to LP (1). Suppose we aim to develop an online algorithm achieving a ratio of $\gamma \in [0,1]$. In particular, for every edge $e \in E$, we want to ensure that the Pr[e is matched] $\geq \gamma$. Consider an assignment $e = (u, v)$ when some v arrived at time t. Let $\mathrm{SF}_{e,t}$ be the event that e is safe at t, *i.e.,* u is available at t. Using Monte-Carlo simulations we obtain an approximate estimate $\beta_{e,t}$ of the quantity $\Pr[\mathrm{SF}_{e,t}]$. Note that to obtain $\beta_{e,t}$, we need the estimates $\beta_{e,t'}$ for every $t' < t$. With the estimated quantity $\beta_{e,t}$, we match a safe edge e at time t with probability $\frac{x^*_{e,t}}{p_{v,t}} \frac{\gamma}{\beta_{e,t}}$. This is a valid probability (*i.e.,* not exceeding 1), if and only if $\gamma \leq \beta_{e,t}$. We denote an algorithm ADAP(γ) to be *valid* if and only if for every $e \in E$ and every $t \in [T]$ we have $\gamma \leq \beta_{e,t}$.

When the condition $\gamma \leq \beta_{e,t}$ is satisfied for all $e \in E$ and $t \in [T]$, we have that the probability an edge $e = (u, v)$ is matched conditioned on the event that v arrives and u is available is at least $\gamma x^*_{e,t}$. Thus, using linearity of expectation this immediately implies that the expected reward obtained by the algorithm is at least $\gamma \sum_{t \in [T]} \sum_{e \in E} w_e x^*_{e,t}$. Therefore, the competitive ratio is at least γ.

At the outset, this looks similar to the Inverse Propensity Scoring (IPS) used in the multi-armed bandit literature [9]. 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 [34] and stochastic matching [1]. Throughout the analysis, we assume that we know the exact value of $\beta_{e,t} := \Pr[\mathrm{SF}_{e,t}]$ 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 1: A simulation-based adaptive algorithm ADAP(γ)

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

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

Remarks. Our simulation-based adaptive algorithm described above is *non-adaptive* according to our definition. Assume ADAP is valid with respect to a given $\gamma \in (0, 1)$. We can first solve the benchmark LP (1) and get an optimal solution $\{x^*_{e,t}\}_{e \in E, t \in [T]}$. Then by simulating the random arrivals of online agents according to known distributions $\{p_{v,t}\}_{v \in V, t \in [T]}$ and ADAP(γ) itself sequentially from $t = 1, 2, \ldots, T$, we can get an arbitrarily accurate estimate of $\beta_{e,t}$ for each e and t. All these procedures can be done in an offline manner (*i.e.,* before the online process).

LEMMA 5. *ADAP(γ) is a valid algorithm (*i.e., *$\gamma \leq \beta_{e,t}$ for every $e \in E$ and $t \in [T]$) when $\gamma = \frac{1}{2}$.*

PROOF. Essentially we need to show that $\beta_{e,t} \geq \gamma = \frac{1}{2}$ for all e and t. We prove it by induction on t as follows.

When $t = 1$, $\beta_{e,t} = 1$ for all $e = (u, *)$. Therefore, we are done. Assume for all $t' < t$, $\beta_{e,t'} \geq 1/2$ and ADAP(γ) is valid for all rounds t'. In other words, we assume each e is assigned with probability *exactly* equal to $x^*_{e,t'} \cdot \frac{1}{2}$ 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,

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

The last inequality above is due to Constraint (3) in the benchmark LP (1). Therefore, we have $\beta_{e,t} \ge 1/2$ and we are done. □

The main Theorem 1 follows directly from Lemmas 4 and 5.

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 (3) on u in the benchmark LP (1) to the following. We have that $\forall u \in U, t \in [T]$,

$$\sum_{t'<t} \sum_{e \in E_u} x_{e,t'} \Pr[C_{e,t'} > t - t'] + \sum_{e \in E_u} x_{e,t} \le 1 \tag{5}$$

The rest of our algorithm remains the same as before. We can verify that (1) LP (1) with constraints (3) replaced by (5) is a valid benchmark; (2) ADAP achieves a competitive ratio of $\frac{1}{2} - \epsilon$ for any given $\epsilon > 0$ for the new model based on the new valid benchmark LP. The modifications to the analysis transfer through in a straightforward way and for brevity we omit the details here.

4 TIGHTNESS OF ONLINE ANALYSIS AGAINST THE BENCHMARK LP

In this section we prove the main result as stated in Theorem 2. Consider the following example.

Example 1. *Consider a star graph $G = (U, V, E)$, where U consists of one single node u and $V = \{v_1, v_2\}$. Set $T = N + 1$. For $j = 1, 2$ and $t \in [T]$, let $p_{j,t}$ denote the arrival probability of the vertex of type v_j at time t. For $t = 1$, $p_{11} = 1$ and $p_{21} = 0$. For $2 \le t \le T$, $p_{1,t} = 0$ and $p_{2,t} = 1/N$. In other words, with probability $1 - 1/N$, no vertex will arrive (or we can assume a dummy node will arrive) during round $t \ge 2$. Let $w_1 \doteq w_{(u,v_1)} = 1 - 1/N$ and $w_2 \doteq w_{(u,v_2)} = 1$. For $C_1 \doteq C_{(u,v_1)}$, it takes value of T and 1 with respective probabilities $1 - 1/N$ and $1/N$. For $C_2 \doteq C_{(u,v_2)}$, it takes the value 1 with probability 1.*

Lemma 6. *The benchmark* LP (1) *has an optimal value of* $2 - 1/N$ *on Example 1.*

Proof. For ease of exposition, let x_t and y_t denote the probabilities that $e = (u, v_1)$ and $e = (u, v_2)$ get chosen at time t in any offline optimal, respectively. Thus, the updated benchmark LP is as follows:

$$\text{maximize} \sum_{t \in [T]} (1 - 1/N) x_t + \sum_{t \in [T]} y_t \tag{6}$$

$$\text{s.t. } x_1 \le 1, y_1 \le 0 \tag{7}$$

$$x_t \le 0, y_t \le 1/N \qquad \forall t \ge 2 \tag{8}$$

$$x_1(1 - 1/N) + y_t \le 1 \qquad \forall t \ge 2 \tag{9}$$

$$0 \le x_t, y_t \le 1 \qquad \forall t \in [T] \tag{10}$$

We can verfiy that the optimal solution to the above LP is as follows: $x_1 = 1$ and $y_t = 1/N$ for all $t \ge 2$, all the rest are zeros. Therefore, the optimal value is $(1 - 1/N) + (1/N) \cdot (T - 1) = 2 - 1/N$. □

Lemma 7. *The optimal (adaptive) online algorithm has an expected performance of* $1 + 1/N$ *on Example 1.*

PROOF. Let ALG be an optimal online algorithm and suppose that it matches the edge $e = (u, v_1)$ with probability $\alpha \in [0, 1]$ at $t = 1$. Let $\mathbf{E}[\text{ALG}]$ be the expected total weight of all matches achieved by ALG. Thus,

$$\mathbf{E}[\text{ALG}] = \alpha\Big(1 + (1/N) \cdot (1/N) \cdot N\Big) + (1 - \alpha) \cdot (1/N) \cdot N = 1 + \alpha/N.$$

Thus, the optimal online algorithm will choose $\alpha = 1$ and the resultant expected performance is $1 + 1/N$. □

Proof of Theorem 2.

PROOF. Combining the two lemmas above, we see that the optimal algorithm can achieve a competitive ratio of $(1 + 1/N)/(2 - 1/N)$ on Example 1 w.r.t. to benchmark LP (1). This completes the proof. □

4.1 The special case of deterministic C_e being a constant

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 deterministically equal to 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 $1 \le \ell \le n - K + 1$. The benchmark LP (1) then reduces to the following.

$$\max \sum_{t \in [T]} \sum_{e \in E} x_{e,t} \tag{11}$$

$$\text{s.t.} \sum_{e \in E_v} x_{e,t} \le \frac{1}{n^2} \qquad \forall v \in V, t \in [T] \tag{12}$$

$$\sum_{t \in W_\ell} \sum_{e \in E_u} x_{e,t} \le 1 \qquad \forall u \in U, 1 \le \ell \le n - K + 1 \tag{13}$$

$$0 \le x_{e,t} \le 1 \qquad \forall e \in E, t \in [T] \tag{14}$$

We can verify that an optimal solution to the above LP is as follows: $x^*_{e,t} = 1/(n^2 K)$ 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 $\sum_{u \in E_v} \alpha_{u,v} \le 1$ 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,

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

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 8, with $\gamma_{u,t} = 1, t = 1$.

LEMMA 8. *$\forall 1 < t \le n$, we have*

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

PROOF. The inequality for $t = 1$ is due to the fact that u is safe at $t = 1$. For each time $t > 1$, Let $\mathrm{SF}_{u,t}$ 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, $\{\mathrm{SF}_{u,t}, A_{u,t'}, t-K+1 \le t' < t\}$ are mutually exclusive and collectively exhaustive events. Therefore,

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

□

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.

$$\begin{aligned} \max \quad & \frac{\sum_u \sum_t \beta_u \gamma_{u,t}}{n} && (15) \\ \text{s.t.} \quad & \sum_{u \in U} \beta_u \le 1 && (16) \\ & \gamma_{u,t} + \beta_u \sum_{t-K+1 \le t' < t} \gamma_{u,t'} = 1 \qquad \forall 1 < t \le n, u \in U && (17) \\ & \beta_u \ge 0, \gamma_{u,1} = 1 \qquad \forall u \in U && (18) \end{aligned}$$

We prove the following Lemma which implies Theorem 3.

LEMMA 9. *The optimal value of the program* (15) *is at most* $\frac{1}{2-1/K} + K/n$.

PROOF. Focus on a given vertex $u \in U$. Notice that $\gamma_{u,t} + \beta_u \sum_{t-K+1 \le t' < t} \gamma_{u,t'} = 1$ for all $1 \le t \le n$. Summing both sides over $t \in [n]$, we have the following.

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

Therefore we have,

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

Define $H_u \doteq \sum_t \beta_u \gamma_{u,t}$. From the above analysis, we have that $H_u \le \frac{n\beta_u}{1+\beta_u(K-1)} + 1$. Thus the objective value in the program (15) can be upper-bounded as follows.

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

We claim that the optimal value to the program (15) can be upper bounded by the following maximization program.

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

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. □

Hardness against the ex-post optimal solution. Here we show a hardness result against the ex-post optimal solution. Manshadi et al. [36] prove that for the online-matching problem under known IID distributions (but disposable offline vertices), no algorithm can achieve a ratio better than 0.823. Since our setting generalizes this, their hardness result directly applies to our problem as well. It is worth noting that the gap between the LP relaxation (1) and the ex-post optimal solution is currently unknown. Thus, either the hardness result can be improved or the algorithm can use a tighter relaxation of the offline optimal solution.

5 EXPERIMENTS

To validate the approaches presented in this paper, we use the New York City Yellow Cabs dataset,[5] 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 5.2). We test our model on two specific distributions, namely a *normal* distribution and a *power-law* distribution (see Figure 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).

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).

[5] http://www.andresmh.com/nyctaxitrips/

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.

5.2 Justifying The Two Important Model Assumptions

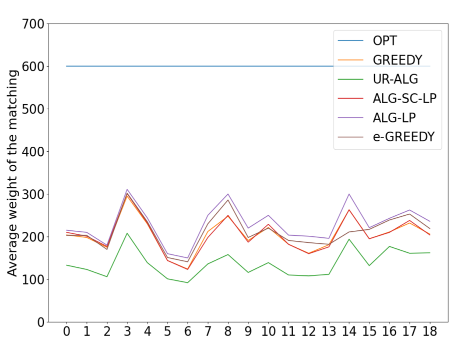

Fig. 1. OTD is normal distribution under KIID

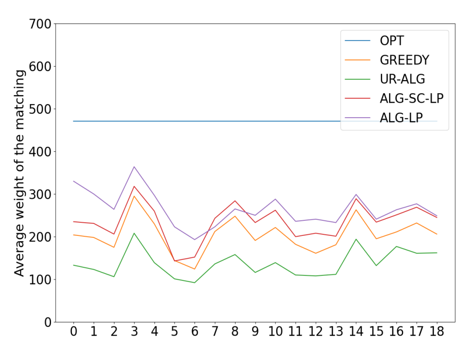

Fig. 2. OTD is normal distribution under KAD

Known Adversarial Distributions. Figure 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

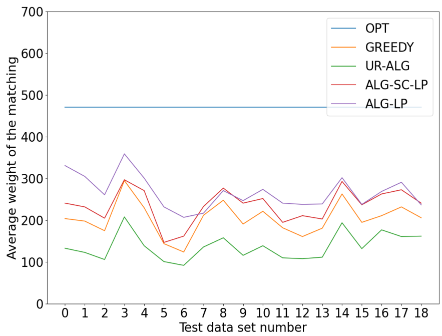

Fig. 3. OTD is power law distribution under KAD

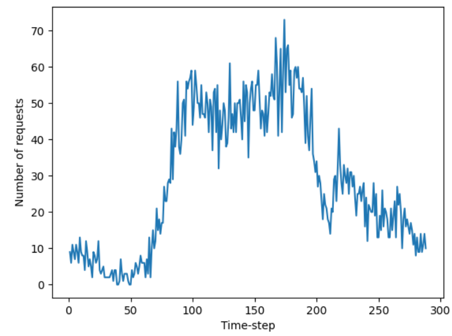

Fig. 4. The number of requests of a given type at various time-steps. x-axis: time-step, y-aixs: number of requests

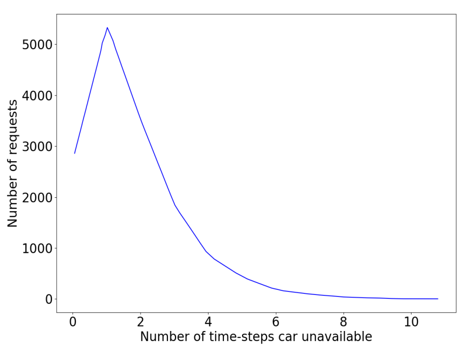

Fig. 5. Occupation time distribution of all cars. x-axis: number of time-steps, y-axis: number of requests

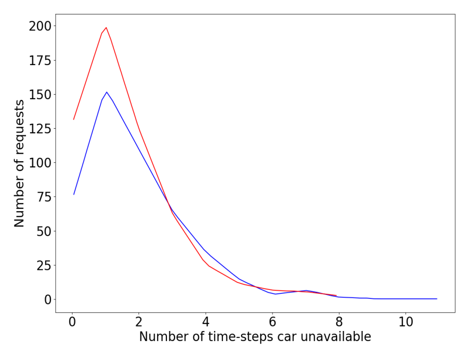

Fig. 6. Occupation time distribution of two different cars. x-axis: number of time-steps, y-axis: number of requests

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 results compared to the assumption of Known Identical Independent Distributions (KIID). This is confirmed by our experimental results shown in Figures 1 and 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 5 and 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 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.

5.3 Results

Inspired by the experimental setup by [55, 56], 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 1, 2, and 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. 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 2 and 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.

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 [27], etc.). Our work opens several important research directions. The first direction is to generalize the online model to the *batch* setting (*e.g.,* this subsequent work). 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 [23, 31, 45, 51, 57]. Following the initial conference publication of this paper, subsequent work has appeared that addresses some of the aforementioned directions. The multi-capacitated version of the problem in the *batch* setting was studied in [32] where the authors devise an algorithm that has a competitive ratio of 0.317. Driver-rider matching in rideshare was modeled as a Markovian system [18]; they show that under a practical condition this system convergences to the stationary distribution very fast. The unweighted multi-capacity version of the problem considered in this paper under the adversarial arrival model was studied in [25]. They proved an optimal competitive ratio $1 - 1/e$ improving over the ratio of $1/2$ proved in [22, 48].

Acknowledgements. The authors would like to thank the anonymous reviewers of AAAI-2018 and the TEAC reviewers for valuable comments on the presentation of this paper.

REFERENCES

[1] Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. Improved approximation algorithms for stochastic matching. In *ESA-15*. 2015.
[2] Daniel Adelman. Price-directed control of a closed logistics queueing network. *Operations Research*, 2007.
[3] Mohammad Akbarpour, Shengwu Li, and Shayan Oveis Gharan. Dynamic matching market design. *EC*, 2014.
[4] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. In *EC-12*, 2012.
[5] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. The online stochastic generalized assignment problem. In *APPROX-RANDOM-13*. 2013.
[6] Ross Anderson, Itai Ashlagi, David Gamarnik, and Yash Kanoria. Efficient dynamic barter exchange. *Operations Research*, 65(6):1446–1459, 2017.
[7] Itai Ashlagi, Patrick Jaillet, and Vahideh H Manshadi. Kidney exchange in dynamic sparse heterogenous pools. *EC*, 2013.
[8] Itai Ashlagi, Maximilien Burq, Patrick Jaillet, and Vahideh Manshadi. On matching and thickness in heterogeneous dynamic markets. 2017.
[9] 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.
[10] Siddhartha Banerjee, Ramesh Johari, and Carlos Riquelme. Dynamic pricing in ridesharing platforms. *ACM SIGecom Exchanges*, 2016.
[11] Siddhartha Banerjee, Daniel Freund, and Thodoris Lykouris. Pricing and optimization in shared vehicle systems: An approximation framework. In *EC*, 2017.
[12] Dimitris Bertsimas, Vivek F Farias, and Nikolaos Trichakis. Fairness, efficiency, and flexibility in organ allocation for kidney transplantation. *Operations Research*, 61(1), 2013.
[13] Anton Braverman, Jim G Dai, Xin Liu, and Lei Ying. Empty-car routing in ridesharing systems. *arXiv preprint arXiv:1609.07219*, 2016.

[14] Brian Brubach, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. New algorithms, better bounds, and a novel model for online stochastic matching. In *ESA-16*, 2016.

[15] Brian Brubach, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. Attenuate locally, win globally: An attenuation-based framework for online stochastic matching with timeouts. In *Sixteenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*. 2017.

[16] Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA-07*, 2007.

[17] Juan Camilo Castillo, Dan Knoepfle, and Glen Weyl. Surge pricing solves the wild goose chase. In *EC*, 2017.

[18] Michael Curry, John P Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, Yuhao Wan, and Pan Xu. Mix and match: Markov chains and mixing times for matching in rideshare. In *International Conference on Web and Internet Economics (WINE)*, pages 129–141. Springer, 2019.

[19] Nikhil R Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *EC-11*, 2011.

[20] John P. Dickerson and Tuomas Sandholm. FutureMatch: Combining human value judgments and machine learning to match in dynamic environments. In *AAAI-15*, 2015.

[21] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *FOCS-09*, 2009.

[22] Yiding Feng, Rad Niazadeh, and Amin Saberi. Linear programming based online policies for real-time assortment of reusable resources. *Available at SSRN 3421227*, 2019.

[23] 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.

[24] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA-08*, 2008.

[25] Vineet Goyal, Garud Iyengar, and Rajan Udwani. Online allocation of reusable resources: Achieving optimal competitive ratio. *arXiv preprint arXiv:2002.02430*, 2020.

[26] Bernhard Haeupler, Vahab S Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In *WINE-11*, 2011.

[27] Chien-Ju Ho and Jennifer Wortman Vaughan. Online task assignment in crowdsourcing markets. In *AAAI-12*, 2012.

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

[29] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC-90*, 1990.

[30] 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.

[31] Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. Online spatio-temporal matching in stochastic and dynamic domains. In *AAAI-16*, 2016.

[32] Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. Competitive ratios for online multi-capacity ridesharing. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 771–779, 2020.

[33] Hongyao Ma, Fei Fang, and David C Parkes. Spatio-temporal pricing for ridesharing platforms. In *ACM Conference on Economics and Computation (EC)*, pages 583–583, 2019.

[34] Will Ma. Improvements and generalizations of stochastic knapsack and multi-armed bandit approximation algorithms. In *SODA-14*, 2014.

[35] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *STOC-11*, 2011.

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

[37] Nicholas Mattei, Abdallah Saffidine, and Toby Walsh. Mechanisms for online organ matching. In *IJCAI-17*, 2017.

[38] Nicole Megow, Marc Uetz, and Tjark Vredeveld. Stochastic online scheduling on parallel machines. In *WAOA-04*, 2004.

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

[40] Aranyak Mehta. Online matching and ad allocation. *Theoretical Computer Science*, 8(4), 2012.

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

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

[43] Brendon L Neuen, Georgina E Taylor, Alessandro R Demaio, and Vlado Perkovic. Global kidney disease. *The Lancet*, 382(9900), 2013.
[44] Afshin Nikzad. Thickness and competition in ride-sharing markets. Technical report.
[45] Eoin O'Mahony and David B Shmoys. Data analysis and optimization for (citi) bike sharing. In *AAAI-15*, 2015.
[46] Erhun Ozkan and Amy Ward. Dynamic matching for real-time ridesharing. *Working Paper.*, 2017.
[47] Jiang Rong, Tao Qin, and Bo An. Dynamic pricing for reusable resources in competitive market with stochastic demand. In *AAAI*, 2018.
[48] Paat Rusmevichientong, Mika Sumida, and Huseyin Topaloglu. Dynamic assortment optimization for reusable products with random usage durations. *Management Science*, 2020.
[49] 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.
[50] 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.
[51] 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.
[52] Martin Skutella, Maxim Sviridenko, and Marc Uetz. Unrelated machine scheduling with stochastic processing times. *Mathematics of operations research*, 41(3), 2016.
[53] Xiaoming Sun, Jia Zhang, and Jialin Zhang. Near optimal algorithms for online weighted bipartite matching in adversary model. *Journal of Combinatorial Optimization*, 2016.
[54] Yong Sun, Jun Wang, and Wenan Tan. Online algorithms of task allocation in spatial crowdsourcing. In *ICDE*, 2017.
[55] 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*, 9(12), 2016.
[56] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. Online mobile micro-task allocation in spatial crowdsourcing. In *ICDE-16*, 2016.
[57] Tanvi Verma, Pradeep Varakantham, Sarit Kraus, and Hoong Chuin Lau. Augmenting decisions of taxi drivers through reinforcement learning for improving revenues. 2017.
[58] Ariel Waserhole and Vincent Jost. Pricing in vehicle sharing systems: Optimization in queuing networks with product forms. *EURO Journal on Transportation and Logistics*, 2016.
[59] Man Lung Yiu, Kyriakos Mouratidis, Nikos Mamoulis, et al. Capacity constrained assignment in spatial databases. In *SIGMOD-08*, 2008.
[60] 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.