Online Weighted Bipartite Matching with Capacity Constraints

Hao Wang, Zhenzhen Yan, Xiaohui Bei
Nanyang Technological University
Singapore

ABSTRACT

We investigate an online edge-weighted bipartite matching problem with capacity constraints. In this problem, the supply vertices are offline with different capacities. Demand vertices arrive online and each consumes a certain amount of resources. The goal is to maximize the total weight of the matching. This framework can capture several real-world applications such as the trip-vehicle assignment problem in ridesharing. We model the offline optimization problem as a deterministic linear program and provide several randomized online algorithms based on the solution to the offline linear program. We analyze the performance guarantee of each algorithm measured by its competitive ratio. Importantly, we introduce a re-solving heuristic that periodically re-computes the offline linear program and uses the updated offline solution to guide the online algorithm decisions. We find that the algorithm's competitive ratio can be improved when re-solving at carefully selected time steps. Finally, we investigate the value of the demand distribution in further improving the algorithm efficiency.

KEYWORDS

online bipartite matching, randomized algorithm, re-solving heuristic, competitive ratio

1 INTRODUCTION

In a typical online bipartite matching problem, requests arrive sequentially following some probability distribution. Upon the arrival of each request, a decision has to be made to either match the request to an appropriate resource, or reject it. If the request gets matched, it consumes a certain amount of resources. Resources can be replenishable or non-replenishable. Each match made between the request and the resource generates a profit. The goal is to maximize the total profit generated from all matches.

The online bipartite matching has been widely applied to various resource allocation problems. Examples include airline seat allocation, clinic appointment slot allocation, and car allocation in the online ride hitch problem. Particularly, ride hitch is a recent innovation of ride sharing. It refers to a mode of transportation in which private car drivers offer to share their journeys to multiple passengers based on coordination through a centralized dispatch. For example, drivers may share part of their ride on the way to work with other passengers who have similar itineraries, and the drivers will receive remuneration to compensate the petrol and labor costs. An example is the grab hitch service launched by Grab in 2015, the leading super app in Southeast Asia. A ride request may involve multiple passengers. In each trip, a driver could take multiple ride requests as long as the capacity permits. Each ride trip of a driver can be regarded as a non-replenishable resource. Once the capacity is used up, it becomes unavailable in demand fulfillment. This new generation of ridesharing significantly increases vehicle occupancy rates and the efficiency of urban transportation systems, consequently reducing congestion and pollution.

In the grab hitch platform, drivers are not allowed to pick up passengers by themselves via self-arrangements, but can only take ride requests assigned by the platform. Therefore, one key problem faced by the platform is to automatically match ride requests to available drivers in real-time so as to maximize the total profit.

A rich literature has been devoted to the study of online matching algorithms with different models and under different assumptions. Karp et al. [19] first studied the online bipartite matching problem to maximize the number of matches under the assumption that the arrival process is determined by the adversary. In their paper, each resource has a single unit of capacity and each request is assumed to consume only one unit of resource. Feldman et al. [14] revisited the problem by assuming the arrival process to be stochastic, that is, the arrival of online vertices follows a known independent and identical distribution (i.i.d). Brubach et al. [8] extended the earlier work to the model to maximize the vertex-weighted sum of matches and the edge-weighted sum of matches. They again assumed unit demand size for each online request and unit capacity for each offline resource. However, in practice, the resources often have general capacities, and each request may consume multiple units of capacity in one match. The ride hitch problem described earlier is an example. Different cars have different number of vacancies to accommodate passengers. Each ride request could involve multiple passengers, hence occupy multiple vacancies. The model studied in this paper considers such a general problem settings. We assume there are multiple types of resources and each type of resource has a general capacity. Each online request consumes a single type of resource but by multiple units. We also assume an i.i.d arrival process. Our goal is to maximize the edge-weighted sum of matches. To the best of our knowledge, this is the first paper considering the online matching problem in such a general setting.

Our Contribution

To solve this general online matching problem, we start by proposing a simple randomized algorithm based on linear program rounding. The algorithm allocates an appropriate resource to each request with certain probability that is based on the solution of an offline linear program. We analyze the performance of this simple algorithm. Following the convention in the literature, we measure the efficiency of an online algorithm by the *competitive ratio*, which is defined as the total profit generated from the algorithm, divided by the maximum profit achievable if full information on the arrival of demand requests is known beforehand. Next, as the main result of this paper, we introduce a re-solving heuristic to the randomized algorithm. The idea of re-solving is to periodically re-compute the offline linear program and uses the updated offline solution to guide the online algorithm. We show that re-solving *at a right time* could help significantly improve the performance of the algorithm. We also investigate the value of demand distribution of the online request to further improve the algorithm's efficiency.

Finally, we conduct extensive experimental studies to test the efficiency of our proposed algorithms. On average, our online algorithm achieves 70% – 80% of the optimal profit on both synthetic and real-world datasets. In particular, the proposed randomized algorithm with the re-solving heuristic significantly outperforms all the other algorithms. We observe that by re-solving the linear program at our proposed time, the profit obtained is increased by almost 20% compared with the standard randomized online algorithm. We also show that the advantage of our proposed algorithms becomes more salient when the demand in the market increases.

We summarize our **main contributions** as follows:

(1) We solve a general online bipartite matching problem where the offline resources are equipped with multiple capacities and each online request consumes multiple units of capacity once matched. The arrival process is assumed to be i.i.d and the goal is to maximize the edge-weighted sum of matches.
(2) We propose a randomization algorithm based on the solution to an offline linear program and establish its competitive ratio as a function of the maximal demand among requests. In a special case that the maximal demand is 2, the competitive ratio is $\frac{1}{4}$, which is comparable to the existing results on similar problem settings.
(3) We further introduce a re-solving heuristic to the randomization algorithm and show that re-solving at the *right time* could significantly improve the performance of the algorithm.
(4) We investigate the value of demand distribution of the online request to further improve the algorithm's efficiency.

2 RELATED WORK

The problem of online bipartite matching has been intensively studied, and the literature is too vast to survey here. We provide an overview of the work most directly relevant to ours. The first algorithm for the single-capacity unweighted online bipartite matching problem was given by Karp et al. [19]. They introduced the RANKING algorithm and proved a tight $1 - \frac{1}{e}$ competitive ratio with adversary online vertex arrival order. The analysis was later simplified by Devanur et al. [11]. Aggarwal et al. [3] generalized the problem by considering weigthed offline vertices. Mehta et al. [25] investigated the multi-demand case known as the AdWords problem and presented a $1 - \frac{1}{e}$ competitive algorithm.

Another line of works considers the random arriving model, in which the online vertices arrives in a uniformly random order. Karande et al. [18] and Mahdian et al. [22] independently showed that the RANKING algorithm can achieve a competitive ratio better than $1 - \frac{1}{e}$ in the random arrival model. Huang et al. [16] further generalized the analysis to the vertex-weighted setting. Devanur and Hayes [10] presented a $1 - \epsilon$ competitive algorithm for the AdWord problem in the random arrival model.

Finally, a third line of works, which also includes this work, assumes the arrival of online vertices follows a known independent and identical distribution [6, 14, 15, 17, 23]. In this model, a closely related work is Xu et al. [27].In their paper, there are multiple types of resources and each request could consume at most one unit of each type. The objective is to maximize the vertex-weighted sum of matches. They designed an algorithm which achieves $\frac{1}{4\Delta}$ competitive ratio, where Δ denotes the maximal number of resources requested by arrivals. Although their paper shares a similar setting to ours, our paper distinguishes from theirs in the following aspects: First, the profit in our paper is defined on edges instead of vertices. The edge-weighted matching is known to be much more nebulous than the vertex-weighted case [13]. Second, we assume each arrival only requests one type of resource but could consume multiple units of resources.

Another relevant problem to online bipartite matching is the online generalized assignment problem. In the online generalized assignment problem, there are m (static) bins each with a capacity limit. Items arrive online and consume some capacity of the assigned bins. Alaei et al. [4] provided an algorithm for this problem with $1 - \frac{1}{k}$ competitive ratio, assuming that no item consumes more than $\frac{1}{k}$ fraction of any bin's capacity. Kesselheim et al. [20] and Naori et al. [26] considered the online generalized assignment problem in the random arrival model and provided the best-known competitive ratio of $\frac{1}{6.99}$.

Other extensions include generalizing the graph to a general network structure and allowing a matching delay, i.e. the request is allowed to wait for some time before being matched (c.f. Chen et al. [9], Adamczyk et al. [1], Adamczyk et al. [2], Baveja et al. [7], Mehta et al. [24], Ashlagi et al. [5], Dickerson et al. [12] and Lowalekar et al. [21]).

3 MODEL

We define our online bipartite matching problem in a ride hitch context. Consider a bipartite graph $G = (U, V, E)$, where U denotes the set of drivers' offers and V denotes the set of possible riders' requests. Each offer $u \in V$ is associated with a capacity c_u, and each request v has a demand d_v. An edge (u, v) exists for $u \in U$ and $v \in V$ if the offer u can can be matched to the request v. The edge is also associated with a weight (i.e. revenue) w_{uv}.

In our problem definition, we will treat drivers' offers as offline resources and riders' requests as online demands. This is because generally, the driver has a much longer time tolerance to be matched. He/She might plan to pick up some friends in the evening but put up an offer in system in the morning. In contrast, a rider's request usually needs to be matched in seconds. If no suitable drivers can be found within one minute, the request will be rejected.

We assume an i.i.d. distribution model of request arrival over an online time horizon of T rounds. That is, the set U is made available offline beforehand. In each round, a rider request v is sampled with replacement from a known distribution $\{p_v\}$ over V. The distribution is independent and identical in every round. Upon the arrival of request v, a decision has to be made to either reject v, or to match it to some neighbor offer $u \in U$ that still has enough remaining capacity. If a pair (u, v) is matched, a revenue of w_{uv} is generated and the capacity of u is decreased by d_v.

Below is the list that summarizes the notations.

- T: total numbers of online requests.
- p_v: the probability of type v vertex in each arrival.

- d_v: the demand of online request v.
- D: the maximal demand of all online requests.
- c_u: the capacity of offer u.
- C: the maximal capacity of all offers.
- w_{uv}: the weight (i.e., revenue) associated with edge (u, v)

MILP formulation. Given a realized arrival sequence s of requests, we can solve a mixed integer linear program (MILP) to optimally match them to the offers.

$$\begin{aligned} \textbf{max} \quad & \sum_{(u,v)\in E} w_{uv} X_{uv}(s) \\ \textbf{s.t.} \quad & \sum_{v:(u,v)\in E} d_v X_{uv}(s) \le c_u, \forall u \in U \\ & \sum_{u:(u,v)\in E} X_{uv}(s) \le N_v(s), \forall v \in V \\ & X_{uv}(s) \in \{0,1\}, \forall (u,v) \in E \end{aligned} \tag{1}$$

Here $N_v(s)$ denotes the number of type v vertex appeared in sequence s. The first set of constraint restricts the total consumption of each resource u below its capacity and the second set of constraint specifies the matched request cannot exceed the total arrivals.

We denote the optimal solution to (1) as $H(s)$ and the optimal objective value as $OFF(s)$. Then the expected revenue generated by optimally solving each possible arrival realization can be formulated as $\mathbb{E}[OFF] = \sum_s \mathbb{P}(s) OFF(s)$, where $\mathbb{P}(s)$ denotes the probability of sequence s among all possible sequences.

Competitive Ratio Analysis. Note that one usually cannot achieve $\mathbb{E}[OFF]$ via an online algorithm due to an unforeseen circumstance in the future. In this paper, we aim to design online algorithms to achieve as a larger expected revenue as possible. To evaluate the performance of an online algorithm L, we adopt a commonly used performance criterion— *competitive ratio (CR)*. For a given sequence s, we denote the outcome achieved by an algorithm L as $ALG_L(s)$. Then the expected outcome by the algorithm is $\mathbb{E}[ALG_L] = \sum_s \mathbb{P}(s) ALG_L(s)$. The competitive ratio of an algorithm L is defined as

$$CR_L = \frac{\mathbb{E}[ALG_L]}{\mathbb{E}[OFF]} \tag{2}$$

Noticed that $OFF(s)$ of (1) is a concave function in $N_v(s)$ for each sequence s. By taking the expectation over all possible arriving sequences, replacing $\mathbb{E}[N_v(s)]$ by $p_v T$, and defining $y_{uv} = \frac{x_{uv}}{p_v T}$, we get upper bound $\mathbb{E}[OFF]$ by the following linear program.

$$\begin{aligned} \textbf{max} \quad & \sum_{(u,v)\in E} T p_v w_{uv} y_{uv} \\ \textbf{s.t.} \quad & \sum_{v:(u,v)\in E} p_v d_v y_{uv} \le \frac{c_u}{T}, \forall u \in U \\ & \sum_{u:(u,v)\in E} y_{uv} \le 1, \forall v \in V \\ & y_{uv} \ge 0, \forall (u,v) \in E \end{aligned} \tag{3}$$

We use OPT to denote the optimal value of (3).

Proposition 3.1. *$OPT \ge \mathbb{E}[OFF]$, i.e., $\mathbb{E}[OFF]$ is upper bounded by the optimal value of (3).*

Algorithm 1: Samp(α) Algorithm

```
Result: Online matchings $M$
Solve the LP (3) and get the optimal solution $y^*$;
Time $t = 1$;
Matchings $M = \phi$;
while $t \le T$ do
    Online vertex $v$ arrives;
    Randomly choose $u$ with probability $\alpha y^*_{uv}$;
    if $c_u \ge d_v$ then
        Match $u$ and $v$: $c_u = c_u - d_v$, $M = M + (u, v)$;
    else
        Reject $v$;
    end
    $t = t + 1$;
end
```

We omit the proof due to space constraints.

In the subsequent sections, we will use linear program (3) to aid our design of the randomized online algorithms and analyze their competitive ratios.

4 A RANDOMIZED ALGORITHM — SAMP (α)

Our first online algorithm takes a common linear-program-rounding approach. First solve the optimal solution y^*_{uv} from (3). Then for each arrival request v, an offer u is randomly chosen to match v with probability αy^*_{uv}. Here α is a parameter that controls how aggressively the online algorithm makes the matches.

Let C_{ut} denote the amount of capacity of offer u consumed by the requests before time t. We have

$$\begin{aligned} \mathbb{E}[C_{ut}] &= \sum_{t'=1}^{t-1} \sum_{v' \in V} p_{v'} \alpha y^*_{uv'} d_{v'} = \sum_{t'=1}^{t-1} \sum_{v' \in V} \alpha y^*_{uv'} d_{v'} p_{v'} \\ &\le \sum_{t'=1}^{t-1} \frac{\alpha c_u}{T} = \frac{\alpha c_u (t-1)}{T} \end{aligned} \tag{4}$$

where the last inequality is from offline vertex's capacity constraint. By Markov's inequality, we can write the probability that v is matched to u at time step t as

$$Pr_\alpha(u, v, t) \ge \alpha p_v y^*_{uv} \left[1 - (t-1) \frac{\alpha c_u}{T(c_u - d_v + 1)} \right]$$

This allows us to bound the expected number of times that edge (u, v) is matched by the algorithm during the first t' steps.

$$\begin{aligned} N_\alpha(u, v, t') &\ge \sum_{t=1}^{t'} Pr_\alpha(u, v, t) \\ &= \sum_{t=1}^{t'} \alpha p_v y^*_{uv} \left[1 - (t-1) \frac{\alpha c_u}{T(c_u - d_v + 1)} \right] \\ &\ge \left[-\left(\frac{t'}{T}\right)^2 \frac{\alpha^2 c_u}{2(c_u - d_v + 1)} + \frac{t'}{T}\alpha \right] T p_v y^*_{uv} \\ &\ge \left[-\left(\frac{t'}{T}\right)^2 \frac{\alpha^2 D}{2} + \frac{t'}{T}\alpha \right] T p_v y^*_{uv} \end{aligned} \tag{5}$$

The last inequality holds as $1 \le c_u \le C$ and $1 \le d_v \le D$ and $c_u \ge d_v$, which implies that for every u and v,

$$\frac{c_u}{c_u - d_v + 1} = 1 + \frac{d_v - 1}{c_u - d_v + 1} \le 1 + \frac{D-1}{1} = D.$$

Therefore, we can further bound the expected performance of the Samp(α) algorithm as follows.

$$\begin{aligned} ALG_\alpha &= \sum_{(u,v)\in E} [w_{uv} N_\alpha(u,v,T)] \\ &\ge \left(-\frac{\alpha^2 D}{2} + \alpha\right) \sum_{(u,v)\in E} T p_v w_{uv} y^*_{uv} \ge \left(-\frac{\alpha^2 D}{2} + \alpha\right) OPT \end{aligned} \tag{6}$$

Proposition 4.1. *The* Samp(α) *algorithm has competitive ratio* $CR \ge \frac{1}{2D}$.

The proof of the proposition is a straightforward result from the analysis above and is omitted here.

4.1 Re-solving Heuristic

Note that in algorithm Samp(α), the linear program (3) is solved once at the beginning of the algorithm. Then the solution will be used to guide the online algorithm matching probability throughout the whole time span. However, in the middle of the process, due to the randomness of the request arriving sequence, it is probably that the capacities of the orders are consumed disproportionately. In such cases, the original LP can no longer capture the correct resource configuration. As such, we need to *re-solve* the linear program with the updated capacity information, and update the LP solution to guide the subsequent allocation. We call this refinement step the *re-solving heuristic.*

In this section, we will analyze the re-solving heuristic. Our goal is to decide whether and when this heuristic will improvement the algorithm performance.

4.1.1 Re-solving does not always help. First we try to answer the question of whether the re-solving heuristic, regardless of when it is applied, always helps the algorithm to generate a better solution. Intuitively this may seem true. However, in the following we show via a counterexample that the re-solving heuristic may make things worse sometimes.

- 1 offline vertex: $c = 2$
- 2 online vertices: v_0 and v_1
- v_0: $p_0 = \frac{1}{2}$, $d_0 = 1$, $w_0 = w > 1$
- v_1: $p_1 = \frac{1}{2}$, $d_1 = 1$, $w_1 = 1$
- $T = 4$

In this case, $D = 1$ hence $\alpha = \frac{1}{D} = 1$. By solving the offline LP (3), we know that algorithm Samp will always accept v_0 as long as the capacity permits and reject v_1. To understand the performance of algorithm Samp, let s denote the realized sequence. If $s = (v_1, v_1, v_1, v_1)$, then the offline optimal profit is 2 while Samp gives 0. If there is exactly one v_0 in s, the offline optimal profit is $1 + w$ while Samp gives w. If there are more than one v_0 in s, both offline optimal profit and the expected profit by Samp are $2w$. Let q_0 denote the probability that s has no v_0 and q_1 denote the probability that s has one v_0. In summary, the expected profits generated by the offline linear program and Samp are different only if the realized sequence has less than two v_0. According to the i.i.d assumption, we have $q_0 = \frac{1}{16}$ and $q_1 = \frac{1}{4}$. Therefore we have:

$$\mathbb{E}[OFF] = \mathbb{E}[ALG] + 2q_0 + q_1 = \mathbb{E}[ALG] + \frac{3}{8}$$

Next we study the effect of re-solving heuristic. Suppose we re-solve the linear program after the first arrival. Consider such a realized sequence $s_0 = (v_1, v_1, v_0, v_0)$. The re-solved LP will suggest to accept v_0 with probability 1 and accept v_1 with probability $\frac{1}{3}$ from the second arrival on. Hence the expected profit generated under this sequence is $\frac{2}{3} \times 2w + \frac{1}{3}(1+w)$ by the re-solving method, which is equivalent to say the expected loss compared to the offline optimal solution is $\frac{w-1}{3}$. Noticed that the probability of having a sequence $s = s_0$ is $\frac{1}{16}$. Then we can upper bound the expected outcome of this re-solving method as follows

$$\begin{aligned} \mathbb{E}[ALG'] &\le \mathbb{P}(s_0)\left[OFF(s_0) - \frac{w-1}{3}\right] + \sum_{s \ne s_0} \mathbb{P}(s) OFF(s) \\ &\le \mathbb{E}[OFF] - \frac{w-1}{48} \end{aligned}$$

By setting $w > 19$, we have $\mathbb{E}[ALG'] < \mathbb{E}[ALG]$. In other words, to re-solve LP could generate a worse performance.

4.1.2 Re-solving at the right time helps. In this section we focus on the problem of when to re-solve. Consider the following question: if we are only allowed to re-solve the LP once during the whole time span, when should we re-solve the LP to maximize the expected performance of the algorithm? Suppose we re-solve at time $t' = (1-\beta)T = \gamma T$. Let T' denote the remaining time period. Then $T' = \beta T$. Let X_u denote the remaining capacity of u at time t'. Then the new LP becomes

$$\begin{aligned} \mathbf{max} \quad & \sum_{(u,v)\in E} T' p_v w_{uv} y_{uv}, \\ \mathbf{s.t.} \quad & \sum_{v:(u,v)\in E} p_v d_v y_{uv} \le \frac{X_u}{T'}, \forall u \in U, \\ & \sum_{u:(u,v)\in E} y_{uv} \le 1, \forall v \in V, \\ & y_{uv} \ge 0, \forall (u,v) \in E \\ & y_{uv} = 0, \forall (u,v) \in E, X_u < d_v. \end{aligned} \tag{7}$$

We add the last constraint because when running the algorithm for several rounds, some offline vertex might have less remaining capacity than the demand. In this case, the edge between them still exists but the matching is not possible. Denote the optimal solution of the re-solved linear program as $\boldsymbol{y}'^*$ and the optimal value as OPT'. The optimal solution and optimal value of the new linear program depend on the remaining capacity, which is a random variable. Hence the optimal solution $\boldsymbol{y}'^*$ and optimal value OPT' are also random variables. To simplify the analysis, we condition on a particular realization of the remaining capacity in the subsequent presentation, i.e. $X_u = c'_u$, in which case, $\boldsymbol{y}'^*$ and optimal value OPT' are deterministic.

Let $N'_\alpha(u,v,T)$ be the expected number of (u,v) that is matched by re-solved algorithm, which is to match the resource based on $\alpha \boldsymbol{y}^*$ before t' and on $\alpha_1 \boldsymbol{y}'^*$ after t'. Following a similar analysis above, we have from the re-solve point t', the expected match between a resource-request pair (u, v) can be analyzed as follows:

$$\begin{aligned} N'_\alpha(u,v,T') &\geq \left[-(\frac{T'}{T'})^2 \frac{\alpha_1^2 c'_u}{2(c'_u - d_v + 1)} + \frac{T'}{T'}\alpha_1\right] T p_v y'^{*}_{uv} \\ &\geq \left[-\frac{\alpha_1^2 D}{2} + \alpha_1\right] T p_v y'^{*}_{uv} \end{aligned} \tag{8}$$

By choosing $\alpha_1 = \frac{1}{D}$, we have $N'_\alpha(u,v,T') \geq \frac{1}{2D} T p_v y'^{*}_{uv}$. The second inequality holds as $y'^{*}_{uv} = 0$ if $c'_u < d_v$, which implies for any $y'^{*}_{uv} > 0$, $c'_u \geq d_v$, hence $\frac{c'_u}{c'_u - d_v + 1} \leq 1 + \frac{d_v - 1}{c'_u - d_v + 1} \leq d_v$. Therefore,

$$N'_\alpha(u,v,T) \geq \left[-\frac{\alpha^2 D}{2}\gamma^2 + \alpha\gamma\right] T p_v y^{*}_{uv} + \frac{1}{2D} T p_v y'^{*}_{uv} \tag{9}$$

The expected outcome of re-solve algorithm is

$$\sum_{uv} w_{uv} N'_\alpha(u,v,T) \geq \left[-\frac{\alpha^2 D}{2}\gamma^2 + \alpha\gamma\right] OPT + \frac{1}{2D} OPT'$$

Then we can bound the competitive ratio from below, i.e.,

$$CR' \geq \left[-\frac{\alpha^2 D}{2}\gamma^2 + \alpha\gamma\right] + \frac{1}{2D}\frac{OPT'}{OPT} = -\frac{\gamma^2\alpha^2 D}{2} + \gamma\alpha + \frac{1}{2D}\frac{OPT'}{OPT}. \tag{10}$$

Now we want to show the lower bound (LB) of $\frac{OPT'}{OPT}$ and check if we need to re-solve according to this LB. We construct a solution $\boldsymbol{y'}$ for LP' based on the optimal solution of LP. We denote U_+ as the set of offline vertices whose capacity without change, i.e., $c'_u = c_u$ and U_- as the rest of the offline vertices. Consider such a solution

$$y'_{uv} = \begin{cases} y^{*}_{uv} & u \in U_+ \\ 0 & u \in U_- \end{cases} \tag{11}$$

It is easy to check that $\boldsymbol{y'}$ is feasible for the re-solved linear program (7). Therefore,

$$\frac{OPT'}{OPT} \geq \frac{T'(w'(U_+) + w'(U_-))}{T(w(U_+) + w(U_-))} = \beta \frac{w'(U_+)}{w(U_+) + w(U_-)} \tag{12}$$

where $w_u = \sum_{v \in Adj(u)} p_v w_{uv} y^{*}_{uv}$, $w(S) = \sum_{u \in S} w_u$ and $w'(S) = \sum_{u \in S} \sum_{v \in Adj(u)} p_v w_{uv} y'_{uv}$. Bearing in mind that the remaining capacity X_u is a random variable, OPT' is also random. From the analysis above, we have

Proposition 4.2. *Define* $R = \frac{max_u c_u}{min_v d_v}$, $\mathbb{E}\left[\frac{OPT'}{OPT}\right] \geq (1-\gamma)e^{-\alpha\gamma R}$

We omit the proof of the proposition due to space constraints. Based on the result in Proposition 4.2. Let $\alpha = 1$ and $\gamma = \frac{1}{D}$. We can further establish the following proposition to demonstrate that re-solving at γT helps to achieve a better lower bound for the competitive ratio.

Proposition 4.3. *Re-solving at* $\frac{T}{D}$ *helps to generate a better competitive ratio which is* $CR' \geq \frac{1}{2D} + \frac{1}{2D}(1 - \frac{1}{D})e^{-\frac{R}{D}}$

The first term in the established competitive ratio is exactly the one we have built for algorithm Samp. The second term is non-negative as long as $D \geq 1$, which indicates that to re-solve at our proposed time will generate at least the same competitive ratio as Samp(α).

4.1.3 *Re-solving Many Times At the Right Time Further Helps to Improve the Bound.*

Consider such a randomization algorithm. Randomly allocate the resource based on Samp(α) until γT. After ith re-solve, the allocation is based on Samp(α_i) and the new re-solve time point is at γ proportion of the remaining time period. We call such an algorithm as a log-resolving algorithm with parameter γ to specify the re-solving time point. Denote the remaining time period at ith re-solve as $T^{(i)}$. $T^{(0)} = T$. Let $X_u^{(i)}$ denote the remaining capacity right before ith re-solve. Notice that $X_u^{(i)}$ is a random variable, we first study the bound of the expected ratio between the optimal values of two consecutive linear programs conditioned on a realization of $X_u^{(i)}$.

Proposition 4.4. *Conditioned on* $X_u^{(i)} = c_u^{(i)}, \forall u$, *under the condition that* $\frac{R}{T^{(i)}} << 1$, *we have*

$$\mathbb{E}\left[\frac{OPT_{i+1}}{OPT_i}\middle| \boldsymbol{c}^{(i)}\right] \geq (1-\gamma)e^{-\alpha\gamma R} \tag{13}$$

We omit the proof here as it is similar to Proposition 4.2. With this proposition, we are now ready to establish our first main result in the paper.

Theorem 4.5. *The log-resolving algorithm with* $\gamma = \frac{1}{D}$ *gives a competitive ratio at least* $\frac{1}{2D}\frac{1-C^{K+1}}{1-C}$, *where* $C = \left(1 - \frac{1}{D}\right)e^{-\frac{R}{D}}$. *This value increases with the number of re-solving time* K *and converges to* $\frac{1}{2D(1-C)}$ *when* K *approaches infinity.*

Proof: To start the proof, we first define some notations: denote the time for ith re-solve as t_i, the ratio between the maximal remaining capacity and the minimum demand as R_i and the optimal value of the ith re-solved LP as OPT_i. Samp(α) is applied at the beginning and after ith re-solve, Samp(α_i) is applied. $ALG(s,T)$ denotes the expected outcome from time s to T using the designed randomization algorithm, which is to match the resource according to the solution from the corresponding re-solved deterministic linear program in each time interval. We can derive a lower bound if the expected performance of our re-solving heuristic in the following way: from the beginning of the time period to the first re-solve point t_1, the lower bound is derived from (5); from t_1 onward the algorithm is based on the designed randomization algorithm. Therefore,

$$\mathbb{E}[ALG(0,T)] \geq \left(-\frac{\gamma^2\alpha^2 D}{2} + \gamma\alpha\right) OPT + \mathbb{E}[ALG(t_1,T)]$$

Hence we have

$$\begin{aligned} \frac{\mathbb{E}[ALG(0,T)]}{OPT} &\geq -\frac{\gamma^2\alpha^2 D}{2} + \gamma\alpha + \frac{\mathbb{E}[ALG(t_1,T)]}{OPT} \\ &= -\frac{\gamma^2\alpha^2 D}{2} + \gamma\alpha + \mathbb{E}\left[\frac{ALG(t_1,T)}{OPT_1}\frac{OPT_1}{OPT}\right] \end{aligned}$$

Notice that the remaining capacity at time t_1 is a random variable, hence the re-solved LP is also a random variable determined by the remaining capacity. Hence we can reformulate the last term as $\mathbb{E}\left[\frac{ALG(t_1,T)}{OPT_1}\frac{OPT_1}{OPT}\right] = \mathbb{E}\left[\frac{ALG(t_1,T)}{OPT_1}\middle|\boldsymbol{c}^{(1)}\right] \cdot \mathbb{E}\left[\frac{OPT_1}{OPT}\right]$. From proposition 4.4, we have $\mathbb{E}\left[\frac{OPT_1}{OPT}\right] \geq (1-\gamma)e^{-\alpha\gamma R}$. Hence we have

$$\frac{\mathbb{E}[ALG(0,T)]}{OPT} \geq -\frac{\gamma^2\alpha^2 D}{2} + \gamma\alpha + \mathbb{E}\left[\frac{ALG(t_1,T)}{OPT_1}\middle|\boldsymbol{c}^{(1)}\right](1-\gamma)e^{-\alpha\gamma R}$$

Conditioning on $\mathbf{c}^{(1)}$, OPT_1 is deterministic, hence $\mathbb{E}\left[\frac{ALG(t_1,T)}{OPT_1}\middle|\mathbf{c}^{(1)}\right]$ represents the expected performance of the randomization algorithm. Starting from t_1, we can follow a similar analysis to get the following inequality

$$\begin{aligned}
&\mathbb{E}\left[\frac{ALG(t_1,T)}{OPT_1}\middle|\mathbf{c}^{(1)}\right]\\
\geq\;&\mathbb{E}\left[-\frac{\gamma^2\alpha_1^2 D}{2}+\gamma\alpha_1\middle|\mathbf{c}^{(1)}\right]+\mathbb{E}\left[\frac{ALG(t_2,T)}{OPT_2}\frac{OPT_2}{OPT_1}\middle|\mathbf{c}^{(1)}\right]\\
\geq\;&\mathbb{E}\left[-\frac{\gamma^2\alpha_1^2 D}{2}+\gamma\alpha_1\middle|\mathbf{c}^{(1)}\right]+\mathbb{E}\left[\frac{ALG(t_2,T)}{OPT_2}\middle|\mathbf{c}^{(1)},\mathbf{c}^{(2)}\right](1-\gamma)e^{-\alpha_1\gamma R}\\
\geq\;&-\frac{\gamma^2\alpha_1^2 D}{2}+\gamma\alpha_1+\mathbb{E}\left[\frac{ALG(t_2,T)}{OPT_2}\middle|\mathbf{c}^{(1)},\mathbf{c}^{(2)}\right](1-\gamma)e^{-\alpha_1\gamma R}
\end{aligned}\tag{14}$$

With the same analysis we have for each i,

$$\begin{aligned}
&\mathbb{E}\left[\frac{ALG(t_i,T)}{OPT_i}\middle|\mathbf{c}^{(1)},\ldots,\mathbf{c}^{(i)}\right]\geq-\frac{\gamma^2\alpha_i^2 D}{2}+\gamma\alpha_i+\\
&\mathbb{E}\left[\frac{ALG(t_{i+1},T)}{OPT_{i+1}}\middle|\mathbf{c}^{(1)},\ldots,\mathbf{c}^{(i+1)}\right](1-\gamma)e^{-\alpha_i\gamma R}
\end{aligned}$$

In the last period, we have

$$\mathbb{E}\left[\frac{ALG(t_K,T)}{OPT_K}\middle|\mathbf{c}^{(1)},\ldots,\mathbf{c}^{(K)}\right]\geq-\frac{\alpha_K^2 D}{2}+\alpha_K$$

Denote $E[\frac{ALG(t_i,T)}{OPT_i}\mid\mathbf{c}^{(1)},\ldots,\mathbf{c}^{(i)}]$ as l_i. Hence we have

$$l_i\geq-\frac{\gamma^2\alpha_i^2 D}{2}+\gamma\alpha_i+l_{i+1}(1-\gamma)e^{-\alpha_i\gamma R},\forall l=0,\ldots,K-1$$

and $l_K\geq-\frac{\alpha_K^2 D}{2}+\alpha_K$. Set $\alpha_K=\frac{1}{D}$, $\alpha_i=1,\forall i=0,\ldots,K-1$ and $\gamma=\frac{1}{D}$. We have $l_K\geq\frac{1}{2D}$ and $l_i\geq\frac{1}{2D}+l_{i+1}(1-\gamma)e^{-\frac{R}{D}},\forall i=0,\ldots,K-1$. Aggregate all the inequalities, we have

$$\begin{aligned}
l_0\;&\geq\sum_{i=0}^{K}\frac{1}{2D}\left((1-\gamma)e^{-\frac{R}{D}}\right)^i=\frac{1}{2D}\frac{1-\left((1-\gamma)e^{-\frac{R}{D}}\right)^{K+1}}{1-(1-\gamma)e^{-\frac{R}{D}}}\\
&\to\frac{1}{2D}\frac{1}{1-(1-\gamma)e^{-\frac{R}{D}}}=\frac{1}{2D}\frac{1}{1-(1-\frac{1}{D})e^{-\frac{R}{D}}}
\end{aligned}$$

□

4.1.4 Time Complexity. For each arrival, the matching decision is done by flipping a coin based on the optimal solution to a linear program and checking the capacity availability. Hence the computation efficiency is mainly determined by the computation time of a linear program. For SAMP(1), we need to solve $O(1)$ linear programs, and it requires $O(\log|V|)$ linear programs for the re-solving heuristic.

4.2 With Information of Demand Distribution

Previous section has established the competitive ratio as a function of the maximal demand. But we have not made use of the demand distribution. To see the value of the information, we first aggregate the type of each online arrival as follows: For online vertices with the same adjacent offline vertices and the same weight for each corresponding incident edges, we aggregate them into a group. In other words, the set defined by $Q(v)=\{v'\mid Adj(v')=Adj(v),w_{uv'}=w_{uv},\forall u\in Adj(v)\}$ contains all the online vertices in the same group as v. Then we can distinguish online arrival vertices by its affiliated group and demand size. We denote the whole set of groups as Q and the demand set in group q as L_q. For any vertex $v\in\mathcal{V}$, its group is defined by $q(v)\in Q$. The probability of getting a vertex in group $q\in Q$ is $\sum\limits_{v\in\mathcal{V}:q(v)=q}p_v$. For vertex in group q, we define the demand distribution as $p_{l|q}=\frac{p_{ql}}{p_q}$, where $p_{ql}=\sum\limits_{v\in\mathcal{V}:q(v)=q,d_v=l}p_v$ and $p_q=\sum\limits_{l\in L_q}p_{ql}$.

From the definition of groups, it is easy to get

LEMMA 4.6. *There exists an optimal solution to (3), such that $y_{uv}=y_{uv'}$ is $q(v)=q(v')$ and $d_v=d_{v'}$.*

Then we can revise the linear program (3) accordingly as below

$$\begin{aligned}
\mathbf{max}\quad&\sum_{(u,(q,l))\in E}Tw_{uq}\sum_l p_{ql}y_{uql}\\
s.t.\quad&\sum_{q\in Q}\sum_{l\in L_q}p_{ql}ly_{uql}\leq\frac{c_u}{T},\forall u\in U\\
&\sum_{u\in Adj(q,l)}y_{uql}\leq 1,\forall(q,l)\in V\\
&y_{uql}\geq 0,\forall(u,(q,l))\in E
\end{aligned}\tag{15}$$

We prove in the following lemma that the revised LP is equivalent to (3).

LEMMA 4.7. *(15) is equivalent to (3). Specifically, For all v that $q(v)=q,d_v=l$, we have an optimal solution y^* to (15) such that $y^*_{uql}=y^*_{uv}$ for every adjacent vertex u, where y^*_{uv} is an optimal solution to (3).*

In other words, we can regard the vertices in the same group and capacity as a super vertex with arrival probability p_{ql} and solve the revised linear program. For each arrival, a resource is matched according to the revised linear program. The randomization algorithm based on the revised linear program will generate the same results as the original linear program. In the following analysis, we will focus on the revised linear program. Denote the optimal solution to (15) as y^*_{uql} for each u,q,l.

Let $N_\alpha(u,q,l,t')$ be the expected number of matched edge (u,q,l) from $t=1$ to $t=t'$. Following a similar analysis to (5), we have

$$\begin{aligned}
N_\alpha(u,q,l,t')&\geq\left[-(\frac{t'}{T})^2\frac{\alpha^2c_u}{2(c_u-l+1)}+\frac{t'}{T}\alpha\right]Tp_{ql}y^*_{uql}\\
&\geq\left[-(\frac{t'}{T})^2\frac{\alpha^2 l}{2}+\frac{t'}{T}\alpha\right]Tp_{ql}y^*_{uql}
\end{aligned}\tag{16}$$

as $\frac{c_u}{c_u-l+1}\leq 1+\frac{l-1}{c_u-l+1}\leq l$ since $l\leq c_u$.

Let $A_l(t')=-(\frac{t'}{T})^2\frac{\alpha^2 l}{2}+\frac{t'}{T}\alpha$ and $A_l(t')$ is decreasing in l for any $0<t'\leq T$.

Then write down the expected outcome of this algorithm:

$$\begin{aligned}
ALG_\alpha&=\sum_{u,q,l}[w_{uq}N_\alpha(u,q,l,T)]\\
&\geq\sum_{u,q,l}Tp_{ql}w_{uq}A_l(T)y^*_{uql}\\
&\geq\sum_{q\in Q}\sum_{l\in L_q}Tp_{ql}A_l(T)\sum_u w_{uq}y^*_{uql}\\
&=\sum_{q\in Q}\sum_{l\in L_q}Tp_{ql}A_l(T)W_{ql}
\end{aligned}\tag{17}$$

For notation simplicity, we omit T and use A_l to denote $A_l(T)$. To further analyze the lower bound, we first establish some properties of vertices in the same group. We define $W_{ql}=\sum_u w_{uq}y^*_{uql}$ for a fixed group q to represent the expected profit generated by the vertices in the group.

Lemma 4.8. *If $q(v) = q(v') = q$ and $d_v = l < m = d_{v'}$, then $W_{ql} \geq W_{qm}$.*

Lemma 4.8 indicates that within the same group, the vertices with a smaller demand generates a higher expected profit. Based on such an observation, we can build the following lemma.

Lemma 4.9. *For each type $q \in Q$, given a fixed $\sum_{l \in L_q} Tp_{ql}W_{ql}$, we have $\sum_{l \in L_q} Tp_{ql}A_l W_{ql} \geq (-\frac{1}{2}\mathbb{E}[D_q]\alpha^2 + \alpha)W_q$, where $\mathbb{E}[D_q]$ is the expected demand of type q arrivals and W_q represents $\sum_{l \in L_q} Tp_{ql}W_{ql}$.*

Denote $M_e = \max_{q \in Q} \mathbb{E}[D_q]$. Then we are ready to present a new lower bound of the Samp(α) when demand distribution is available.

Theorem 4.10. *With the distribution of demand,* Samp(α) *generates a competitive ratio $\frac{1}{2M_e}$, where $M_e = \max_{q \in Q} \mathbb{E}[D_q]$.*

Proof: From Lemma 4.9, we have for all possible type q,

$$\sum_{l \in L_q} Tp_{ql}A_l W_{ql} \geq (-\mathbb{E}[D_q]\frac{1}{2}\alpha^2 + \alpha)W_q \geq (-M_e\frac{1}{2}\alpha^2 + \alpha)W_q$$

Therefore, $\sum_{q \in Q}\sum_{l \in L_q} Tp_{ql}A_l(T)W_{ql} \geq (-M_e\frac{1}{2}\alpha^2+\alpha)\sum_{q \in Q} W_q = (-M_e\frac{1}{2}\alpha^2 + \alpha)W$. In other words, the competitive ratio is lower bounded by $-M_e\frac{1}{2}\alpha^2 + \alpha$. By setting $\alpha = \frac{1}{M_e}$, we can get a constant a competitive ratio $\frac{1}{2M_e}$. □

4.2.1 Re-solve with Demand Distribution. With information of demand distribution, we can again apply the re-solving heuristic to see whether we can further improve the competitive ratio. According to Lemma 4.7, the deterministic linear program (3) can be reformulated as (15). When revolving the problem, the deterministic linear program can be revised accordingly as follows:

$$\begin{aligned} \textbf{max} \quad & \sum_{(u,(q,l)) \in E} Tw_{uq}\sum_l p_{ql}y_{uql} \\ \textbf{s.t.} \quad & \sum_{q \in Q}\sum_{l \in L_q} p_{ql}ly_{uql} \leq \frac{c_u}{T}, \forall u \in U \\ & \sum_{u \in Adj(q,l)} y_{uql} \leq 1, \forall (q,l) \in V \\ & y_{uql} \geq 0, \forall (u,(q,l)) \in E \\ & y_{uql} = 0, \forall (u,(q,l)) \in E, c_u < l \end{aligned} \tag{18}$$

Follow a similar analysis in the previous section, we have

$$\mathbb{E}[ALG(0,T)] \geq \left[-\left(\frac{t'}{T}\right)^2 \frac{\alpha^2 M_e}{2} + \frac{t'}{T}\alpha\right] OPT + \frac{1}{2M_e}\mathbb{E}[OPT'] \tag{19}$$

Following a similar analysis to Proposition 4.3, we can easily derive the new competitive ratio $\frac{1}{2M_e} + \frac{1}{2M_e}(1 - \frac{1}{M_e})e^{-\frac{R}{M_e}}$ if re-solving once at $\frac{T}{M_e}$. The same analysis applies to the case with multiple times of re-solve.

Theorem 4.11. *When demand distribution is available, the log-resolving algorithm with γ gives a competitive ratio after Kth re-solve is:*

$$\frac{1}{2M_e}\frac{1 - \left((1 - \frac{1}{M_e})e^{-\frac{R}{M_e}}\right)^{K+1}}{1 - (1 - \frac{1}{M_e})e^{-\frac{R}{M_e}}}$$

If the number of re-solving time K increases the ratio can be better. When K is large, this ratio converges to

$$\frac{1}{2M_e}\frac{1}{1 - (1 - \frac{1}{M_e})e^{-\frac{R}{M_e}}}$$

5 EXPERIMENT

We test the online algorithms proposed in Section 4 over several synthetic data sets and a New York city taxi data set. Each data set specifies a bipartite graph, which is represented by $G(U, V, E)$, where U, V represents the offline and online request set respectively and E denotes the arc set. Let $r = \frac{|V|}{|U|}$ denote the ratio between the number of online and offline vertices. We compare our algorithms to the greedy algorithm, which is a widely used benchmark in the literature (c.f. Xu et al. [27], Dickerson et al. [12] and Lowalekar et al.[21]). Specifically, we analyze the following algorithms in this section.

- Greedy: Assign an arriving request v to the resource u with the largest weight on the edge (u, v) among all the available resources; if no available resource found, reject v.
- Samp(1): Refer to the Samp(α) algorithm and choose $\alpha = 1$.
- RES-γ: Under the Samp(1) framework, update the offline linear program at $t = T(1 - (1 - \gamma)^i)$ for $1 \leq i \leq K$, where K is the maximal re-solve times. Set $K = 10$. We test over different γ including $\gamma = \frac{1}{D} = \frac{1}{3}$ as $D = 3$ in the data.

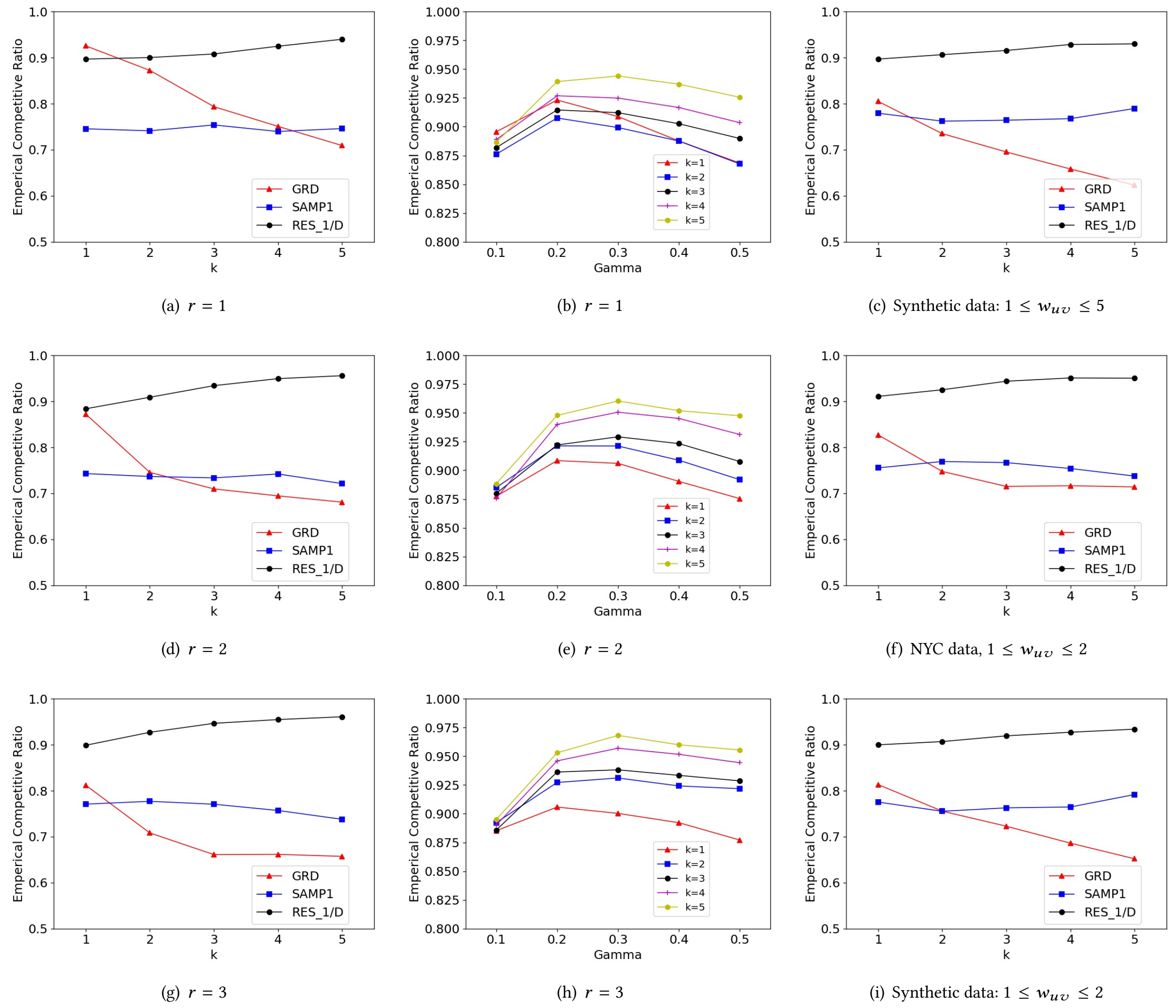
The comparison is based on the *empirical competitive ratio (ECR)*. The *empirical competitive ratio (ECR)* is defined as the ratio between the total profit generated from an algorithm and the total offline optimal profit for a sample of request sequences.

$$ECR_L = \frac{\sum_{s \in S} P_L(s)}{\sum_{s \in S} OFF(s)},$$

where S denotes the set of sampled request sequences and $P_L(s)$ denotes the profit by algorithm L for the sequence s. Noticed that different algorithms share the same offline optimal profit, the comparison of empirical competitive ratio is equivalent to a comparison of profit. For a given bipartite graph G and a planing horizon T, we generate M request sequences by uniformly sampling each request from the demand pool. We run tests for $T = k|V|$ where $k = 1, 2, 3, 4, 5$.

The test is organized in the following manner: We first test different algorithms in New-York city taxi data, with the request weights ranging from 1 to 5. We examine the performance of each algorithm in different markets and investigate the effect of different re-solving time points. Lastly, we extend the test to some synthetic data generated and test different request weight ranges.

The first column of Figure 1 presents the performance of different algorithms in different markets. The parameter k and r indicate the unbalance between the supply and demand in the market. The larger they are, the more overwhelming the demand is. From the figure we can see that in general, the re-solving method outperforms both greedy method and Samp(1) algorithm. On average, incorporating re-solving heuristic in the randomization algorithm improves the profit by 20%. Greedy method performs well when demand is not high. But its performance drops significantly when



(a) $r = 1$

(b) $r = 1$

(c) Synthetic data: $1 \leq w_{uv} \leq 5$

(d) $r = 2$

(e) $r = 2$

(f) NYC data, $1 \leq w_{uv} \leq 2$

(g) $r = 3$

(h) $r = 3$

(i) Synthetic data: $1 \leq w_{uv} \leq 2$

Figure 1: Result Summary

demand increases. In contrast, the performance of the proposed randomization algorithms is more robust across different markets. One possible reason is that when demand is high, strategically skipping some inferior demand can better utilize the resource capacity.

To further understand how important it is to re-solve at a right time. We compare algorithms with different re-solving time points. The second column in Figure 1 compares the performance of algorithms with different γ when applying RES-γ algorithm. From the figure we can see that resolving at our proposed time which is to set $\gamma = \frac{1}{D} = \frac{1}{3}$ outperforms the other time points. It could achieve up to 12% improvement compared to some other resolving time. Consistent with the observation in the column in Figure 1, the randomization algorithms gets better performance when γ becomes larger, for all the tested γs.

We extend the test to the synthetic data generated and plot the results in Figure (a) in the last column of Figure 1. The observations are consistent with those in the first column . We further test the data with edge weights in a smaller range. It is observed that when the weight range becomes smaller, online algorithms' performance increases. This observation is more significant when it applies to the greedy method. It implies that the randomization algorithm is more robust across different data sets.

6 CONCLUSIONS

We study the online weighted bipartite matching problem with capacity constraints in this paper. We propose a randomized algorithm based on the solution to an offline linear program and analyze its competitive ratio. We further introduce a re-solving heuristic to the randomized algorithm and demonstrate that re-solving at the *right times* could significantly improve the performance of the algorithm. Finally, we investigate the value of demand distribution of the online request to further improve the algorithm's efficiency. Several experiments are conducted to test the performance of the proposed algorithms based on an application in ride hitch.

REFERENCES

[1] Marek Adamczyk. 2011. Improved analysis of the greedy algorithm for stochastic matching. *Inform. Process. Lett.* 111, 15 (2011), 731–737.
[2] Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. 2015. Improved approximation algorithms for stochastic matching. In *Algorithms-ESA 2015*. Springer, 1–12.
[3] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. 2011. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 1253–1264.
[4] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. 2013. The online stochastic generalized assignment problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 11–25.
[5] Itai Ashlagi, Yossi Azar, Moses Charikar, Ashish Chiplunkar, Ofir Geri, Haim Kaplan, Rahul Makhijani, Yuyi Wang, and Roger Wattenhofer. 2017. Min-cost bipartite perfect matching with delays. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)* 81 (2017), 1–1.
[6] Bahman Bahmani and Michael Kapralov. 2010. Improved bounds for online stochastic matching. In *European Symposium on Algorithms*. Springer, 170–181.
[7] Alok Baveja, Amit Chavan, Andrei Nikiforov, Aravind Srinivasan, and Pan Xu. 2018. Improved bounds in stochastic matching and optimization. *Algorithmica* 80, 11 (2018), 3225–3252.
[8] Brian Brubach, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. 2016. Online Stochastic Matching: New Algorithms and Bounds. *arXiv preprint arXiv:1606.06395* (2016).
[9] Ning Chen, Nicole Immorlica, Anna R Karlin, Mohammad Mahdian, and Atri Rudra. 2009. Approximating matches made in heaven. In *International Colloquium on Automata, Languages, and Programming*. Springer, 266–278.
[10] Nikhil R Devanur and Thomas P Hayes. 2009. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM conference on Electronic commerce*. ACM, 71–78.
[11] Nikhil R Devanur, Kamal Jain, and Robert D Kleinberg. 2013. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 101–107.
[12] John P Dickerson, Karthik A Sankararaman, Aravind Srinivasan, and Pan Xu. 2018. Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[13] Matthew Fahrbach, Zhiyi Huang, Runzhou Tao, and Morteza Zadimoghaddam. 2020. Edge-Weighted Online Bipartite Matching. *arXiv preprint arXiv:2005.01929* (2020).
[14] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and Shan Muthukrishnan. 2009. Online stochastic matching: Beating 1-1/e. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 117–126.
[15] Bernhard Haeupler, Vahab S Mirrokni, and Morteza Zadimoghaddam. 2011. Online stochastic weighted matching: Improved approximation algorithms. In *International workshop on internet and network economics*. Springer, 170–181.
[16] Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. 2018. Online Vertex-Weighted Bipartite Matching: Beating 1-1/e with Random Arrivals. *arXiv preprint arXiv:1804.07458* (2018).
[17] Patrick Jaillet and Xin Lu. 2014. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research* 39, 3 (2014), 624–646.
[18] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. 2011. Online bipartite matching with unknown distributions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 587–596.
[19] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. 1990. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. ACM, 352–358.
[20] Thomas Kesselheim, Andreas Tönnis, Klaus Radke, and Berthold Vöcking. 2014. Primal beats dual on online packing LPs in the random-order model. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 303–312.
[21] Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. 2020. Competitive Ratios for Online Multi-capacity Ridesharing. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 771–779.
[22] Mohammad Mahdian and Qiqi Yan. 2011. 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*. 597–606.
[23] Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. 2012. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research* 37, 4 (2012), 559–573.
[24] Aranyak Mehta and Debmalya Panigrahi. 2012. Online matching with stochastic rewards. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE, 728–737.
[25] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. 2007. Adwords and generalized online matching. *Journal of the ACM (JACM)* 54, 5 (2007), 22.
[26] David Naori and Danny Raz. 2019. Online multidimensional packing problems in the random-order model. *arXiv preprint arXiv:1907.00605* (2019).
[27] Pan Xu, Yexuan Shi, Hao Cheng, John Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, Yongxin Tong, and Leonidas Tsepenekas. 2019. A Unified Approach to Online Matching with Conflict-Aware Constraints. (2019).