A Non-asymptotic Analysis for Re-solving Heuristic in Online Matching

Hao Wang, Zhenzhen Yan, Xiaohui Bei
Nanyang Technological University

We investigate an online edge-weighted bipartite matching problem with general capacity constraints. In this problem, the resources are offline and non-replenishable with different capacities. Demands arrive online and each requests a certain amount of resources. The goal is to maximize the reward generated by successful matches. We model the offline optimization problem as a deterministic linear program and present multiple randomized online algorithms based on the solution to the offline linear program. We analyze the performance guarantee of each algorithm in terms of 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 significantly improved when re-solving at carefully selected time steps. Finally, we investigate the value of the demand distribution in further improving the algorithm efficiency. We conduct extensive numerical studies to demonstrate the efficiency of the proposed algorithms. The effect of market conditions on the algorithm performance is also investigated.

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

1. Introduction

Consider the following online bipartite matching problem. There is a set of offline resource vertices, each with a certain capacity. All resources are non-replenishable. Online requests arrive sequentially following an identical and independent probability distribution. Each request demands possibly multiple units of a single resource vertex, and upon its arrival, a decision must be made to either match this request to an appropriate resource vertex with the requested demand size or reject this request. Each match made between the request and the resource vertex generates a reward, whose value might depend on both the request and the selected resource vertex. The objective is to maximize the total reward generated from all matches.

This model has applications in various domains. Examples include multi-item order fulfillment in e-commerce, and trip-vehicle assignment in ride-hitch. We now explain some applications.

Multi-Item Order Fulfillment in E-commerce

In an e-commerce platform, purchasing orders arrive online, each with possibly multiple items. Upon its arrival, the platform needs to select one of the warehouses with enough inventory to fulfill the order. Each successful fulfillment consumes the inventory in the selected warehouse and generates a profit whose value depends on both the order value and the shipping cost.

Trip-vehicle Assignment in Ride Hitch

Ride hitch is a recent innovation of on-demand 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 for the petrol and labor costs. A ride request may involve multiple passengers. In each trip, a driver could take multiple ride requests simultaneously as long as the capacity permits. In this application, we treat drivers' offers as offline resources and riders' requests as online demands. This is because drivers usually have a much longer time tolerance to be matched in general. For example, he/she might plan to pick up some friends in the evening but put up an offer in the system in the morning. In contrast, a rider's request often needs to be matched in seconds, or otherwise, the requester will simply leave the platform and the request will result in being rejected. One example of a ride hitch is the Grab Hitch service launched in 2015 by Grab, the leading on-demand ride-sharing company in Southeast Asia. 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 to maximize the total profit.

The online matching model studied in this paper captures many salient features in the aforementioned applications, including non-replenishable resources with a general capacity, multi-unit demand, edge-weighted rewards, and stochastic i.i.d. arrivals. Our model differs from the vast literature in the online matching problem by combining those features. In particular, the multi-unit demand and the edge-weighted rewards make our model especially appropriate for those applications described earlier. Take the ride hitch problem as an example. In ride hitch, different cars have different vacancies to accommodate passengers, and each ride request may involve multiple passengers, hence occupy multiple vacancies. In addition, different rides may generate different profits depending on the route and possibly some surging policies.

In the meanwhile, these features also bring new challenges in deriving effective online matching algorithms. The features of general capacity and multi-unit demand enlarge the state space, hence make the stochastic dynamic programming approach intractable. Another widely used approach in the literature is to derive simple yet near-optimal algorithms with large competitive ratios. In the case of single-unit demand for each arrival, the best-known algorithm is given by Brubach et al. (2020) with competitive ratio 0.705 under the assumption of edge-weighted rewards and stochastic i.i.d. arrival. However, a single-unit demand algorithm cannot be easily generalized to a multi-unit demand case with a competitive ratio loss of demand size. We provide a detailed discussion on the competitive ratio analysis of such a generalization in Appendix C.1.

Due to the challenge in analyzing the non-asymptotic performance of a multi-unit demand algorithm, the existing literature focuses on a special case assuming the rewards are the same as the demand size which is known as a generalized AdWords problem. The best-known algorithm for this problem achieves a competitive ratio of $\max\{0.321, 1-\sqrt{\frac{1}{2\pi d}}+O(\frac{1}{d})\}$ where the demand to capacity ratio is at most $1/d$ (Stein et al. (2020)). Our paper studies a general multi-unit demand model. In particular, our model admits general rewards, which can be different from the demand size. We show in Appendix C.2 that using the state-of-art approach for the generalized AdWords problem proposed by Stein et al. (2020) does not lead to a better competitive ratio than our proposed approach.

Our Contribution

We solve this general online matching problem by proposing a simple randomized algorithm based on linear program rounding. The algorithm allocates an appropriate resource vertex to each request with a certain probability that is based on the solution of an offline linear program. Following the convention in the literature, we analyze the efficiency of this algorithm by its *competitive ratio*, which is defined as the total reward generated from the algorithm, divided by the maximum reward achievable if full information on the arrival of demand requests is known beforehand. Next, as one of the main results in 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 use 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. To the best of our knowledge, this paper is the first one to investigate the performance of resolving in a general online bipartite matching problem under a non-asymptotic, competitive ratio metric. We also investigate the value of demand distribution of the online request to further improve the algorithm's efficiency.

Finally, we conduct extensive numerical studies to test the efficiency of our proposed algorithms. On average, our online algorithm achieves $70\% - 80\%$ of the maximum reward. In particular, this randomized algorithm with re-solving heuristic significantly outperforms other benchmarks. We observe that by re-solving the linear program at our proposed time, the reward obtained can be increased by almost 20% compared to 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 or the market segment becomes more diversified.

The **main contributions** of the paper is summarized as follows:

1. We solve a general online bipartite matching problem in which the offline resource vertices have general 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 built on the solution to an offline linear optimization 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 apply a re-solving heuristic to the randomization algorithm and show both analytically and numerically that re-solving at the *right time* could significantly improve the performance of the algorithm.
4. We also investigate the value of demand distribution of the online request to further improve the algorithm's efficiency.

The rest of the paper is organized as follows. Section 2 reviews the related literature. Section 3 introduces the online matching model. Section 4 provides randomized algorithms for the online matching model and analyzes the performance of the proposed algorithms. In particular, we thoroughly investigate the re-solving heuristic, including the design of re-solving time and the quality of the heuristic. Section 5 further analyzes the performance when the distribution on demand size is available and discusses the value of demand distribution in improving the algorithms' performance. Section 6 performs numerical studies based on a New York city taxi data set to evaluate the algorithms' efficiency. Section 7 concludes the paper. All proofs can be found in Appendix B.

2. Literature Review

Our model falls within the literature of online resource allocation. Vera and Banerjee (2021) provided a clear overview of different problem settings in online resource allocation. In particular, two

important classes in this domain are the *online packing problem* and the *online matching problem*. In an online packing problem, there is a central pool of possibly multiple types of resources. Each online request will consume a certain amount of resources if accepted, and the decision is simply to accept or reject each online request upon its arrival. On the other hand, in an online matching problem, there are multiple options to serve an online request. A decision on which option to choose needs to be made by the decision maker. Once an option is selected by the decision maker without violating the capacity constraints, the online request will consume the corresponding resources and generate a reward. Our problem belongs to this second class. More specifically, it is an *online bipartite matching problem.*

The online bipartite matching problem has been intensively studied in multiple disciplines, and the literature is too vast to survey here. We provide an overview of the work most directly relevant to ours based on the assumptions on the demand arrival process. The first stream of literature assumes that online requests arrive in an adversary manner. Karp et al. (1990) introduced a RANKING algorithm for a single-capacity unweighted online bipartite matching problem. They proved a tight $1-\frac{1}{e}$ competitive ratio with adversary online vertex arrival order. The analysis was later simplified by Devanur et al. (2013). Aggarwal et al. (2011) generalized the problem by considering weighted offline vertices. Mehta et al. (2007) investigated the multi-demand case known as the AdWords problem and presented a $1-\frac{1}{e}$ competitive algorithm.

Another line of works considers a random order model, in which the online demand vertices arrives in a uniformly random order. Karande et al. (2011) and Mahdian and Yan (2011) independently showed that the RANKING algorithm can achieve a competitive ratio which is better than $1-\frac{1}{e}$ in the random order model. Huang et al. (2018b) further generalized the analysis to the vertex-weighted setting. Devanur and Hayes (2009) presented an algorithm achieving $1-\epsilon$ competitive for the AdWords problem in a random order model. A relevant problem to online bipartite matching is the online generalized assignment problem (GAP). In the online GAP, there are m static and capacitated bins. Items arrive online and consume some capacity of the assigned bins. Kesselheim et al. (2014) and Naori and Raz (2019) provided an algorithm with the best-known competitive ratio of $\frac{1}{6.99}$. Our work shares a similar model setting to the online generalized assignment problem and contributes to the literature by considering the i.i.d. arrival process and investigating the value of demand distribution in the performance analysis.

A third line of works, which also includes this work, assumes the arrival of online vertices follows a known independent and identical distribution (Feldman et al. (2009); Bahmani and Kapralov (2010); Manshadi et al. (2012); Jaillet and Lu (2014); Haeupler et al. (2011)). In this line, a closely

related work is Xu et al. (2019). In their paper, there are multiple types of resources and each request could consume at most one unit of each type. The goal is to maximize the vertex-weighted sum of matches. They proposed 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 reward in our paper is defined on edges instead of vertices. The edge-weighted matching is more general and much more nebulous than the vertex-weighted case (Brubach et al. (2020) and Fahrbach et al. (2020)). Second, we assume each arrival only requests one type of resource but could consume multiple units of resources.

Finally, several works have extended the analysis to non-stationary stochastic demand. Wang et al. (2018) model an online advance-scheduling problem as an online weighted bipartite matching problem. The arrival process follows a non-homogeneous Poisson distribution. Each online request is assigned to an offline vertex and consumes one unit of available resource or is rejected. They proposed online algorithms that achieve a competitive ratio of at least 0.5. More recently, Stein et al. (2020) relaxed the single-demand assumption and studied a general Adwords problem with non-stationary arrivals. Specifically, when a request is matched to a offline vertex, it could consume multiple units of resources. The reward generated from the matching is equal to the demand size. They showed that no online algorithm can achieve a competitive ratio better than 0.5 for this generalized Adwords problem, and proposed an algorithm that achieve a competitive ratio of 0.321. Our paper further generalizes this model setting by allowing the reward from the successful matching to be different from the demand size.

Extensions to online bipartite matching 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 (Chen et al. (2009); Adamczyk (2011); Adamczyk et al. (2015); Baveja et al. (2018); Mehta and Panigrahi (2012); Ashlagi et al. (2017); Dickerson et al. (2018); Lowalekar et al. (2020)), and allowing both sides to arrive randomly to the system, which is called the two-sided matching problem (Huang et al. (2018a); Hu and Zhou (2018); Baccara et al. (2020); Özkan and Ward (2020))

2.1. Re-solving Heuristic

Our proposed algorithm adopts a re-solving heuristic, whose idea is to repeatedly re-solve the offline linear program to update the matching probability during the matching process. This intuition has been used in several works in the literature (cf. Reiman and Wang (2008); Jasin and Kumar (2013); Vera and Banerjee (2021); Bumpensanti and Wang (2020)).

Among these works, Vera and Banerjee (2021) considered a general online resource allocation problem, which shares a similar model to ours. They proposed an algorithm that re-solves the offline LP problem before *each* new arrival, which is significantly different from our *in-frequent resolving* algorithm in this work. In addition, their work and ours pursue different objectives. They use regret as the performance measure, and their algorithm achieves $O(1)$ regret bound under mild assumptions on the request arrival process. In contrast, our work adopts the competitive ratio as the performance measure. Note that an $O(1)$ regret bound does not imply any nontrivial competitive ratio, because the expected reward in the offline problem is also in $O(1)$. We further compare our algorithm with theirs in the numerical study and demonstrate our performance differences in maximizing the expected reward.

Another closely related work is by Bumpensanti and Wang (2020), in which authors showed that an infrequent re-solving method can sometimes be more effective than resolving after each arrival. Our work shares a similar high-level idea. But our work differ from theirs from the following perspectives. First, their work studies a quantity-based network revenue management problem, which is an online packing problem, whereas we study an online matching problem. As we mentioned above, there are two completely different classes of problems. As a result, our results are incomparable to each other. Second, both resource capacities and request arrivals in Bumpensanti and Wang (2020) are assumed to scale up in proportion to the planning horizon. In contrast, we follow the model in Vera and Banerjee (2021) and assume a fixed budget but scale up the number of arrivals in the planning horizon. Finally, their re-solving algorithm depends on a threshold whereas we adopt an LP-based randomization policy. The proposed re-solving times are also different from the two works.

3. Model

Consider a bipartite graph $G = (U, V, E)$, where U denotes the set of offline resources, V denotes the set of online requests and E denotes the set of weighted edges. Each offline resource $u \in U$ has 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 resource u can be matched to the request v. The edge is also associated with a reward w_{uv}. We assume that c_u and d_v are all positive integers and each w_{uv} is a non-negative real number. Without loss of generality, we assume all demand and capacity values are mutually prime. Otherwise, we can divide all the values by the most common divisor, and the new problem is the same as before.

We consider a finite time horizon of T rounds and assume an i.i.d. distribution model of online request arrival. That is, the resource set U is made available offline at the beginning of the algorithm. In each round, an online 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 immediately to either reject v, or to match it to some adjacent resource $u \in U$ that still has enough remaining capacity. If a pair (u, v) is matched, a reward of w_{uv} is generated and the capacity of u is decreased by d_v. In practice, each demand d_v is usually a small positive integer and the total number of possible arrivals T can be very large.

Below is the list that summarizes the notation.

- T: total number of online requests.
- p_v: the probability of type v vertex being sampled in each arrival.
- d_v: the demand of online request of type v.
- D: the maximum demand of all online requests.
- c_u: the capacity of resource u.
- C: the maximum capacity of all resources.
- w_{uv}: the weight (i.e., revenue) associated with edge (u, v)
- R: the maximal capacity-demand ratio that is defined as the maximal ratio of resource u's capacity to request v's demand among all the $(u, v) \in E$, i.e., $R \equiv \max_{(u,v)\in E} \frac{c_u}{d_v}$.

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 resources.

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

Here $N_v(s)$ denotes the number of type v request appeared in sequence s. $X_{uv}(s)$ denotes the number of requests of type v matched to resource u. The first set of constraints restricts the total consumption of each resource u below its capacity and the second set of constraints specifies that the matched request cannot exceed the total arrivals.

We denote the optimal solution to (1) as $X^*_{uv}(s)$ and the optimal objective value as $\text{OFF}(s) = \sum_{(u,v)\in E} w_{uv} X^*_{uv}(s)$. Then the expected reward generated by optimally solving each possible arrival realization can be formulated as $\mathbb{E}[\text{OFF}] = \sum_s \mathbb{P}(s)\text{OFF}(s)$, where $\mathbb{P}(s)$ denotes the probability of sequence s among all possible sequences.

Competitive Ratio Analysis Note that one often cannot achieve $\mathbb{E}[\text{OFF}]$ via an online algorithm due to an unforeseen circumstance in the future. To evaluate the performance of an online algorithm L, we adapt *competitive ratio (CR)* as our performance criterion. Competitive ratio is a widely used performance metric for online algorithms with various applications, including the scheduling problems (Wang and Truong (2018); Stein et al. (2020)), order fulfillment problems (Jasin and Sinha (2015); Zhao et al. (2020)), and revenue management problems (Ma et al. (2020a); Ma et al. (2020b)).

Specifically, we denote the reward achieved by an algorithm L on an input sequence s as $\text{ALG}_L(s)$. Then the expected reward achieved by the algorithm is $\mathbb{E}[\text{ALG}_L] = \sum_s \mathbb{P}(s)\text{ALG}_L(s)$. We define the competitive ratio of an algorithm L as follows

$$\text{CR}_L = \frac{\mathbb{E}[\text{ALG}_L]}{\mathbb{E}[\text{OFF}]}. \tag{2}$$

Note that it is usually difficult to calculate the exact value of $\mathbb{E}[\text{OFF}]$. Instead, we upper bound $\mathbb{E}[\text{OFF}]$ by the following linear program (LP) to obtain a lower bound of the competitive ratio.

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

where the first set of constraints (3a) is to specify the capacity constraint of each offline resource, and the second set (3b) is a normalized demand constraint of each online vertex. We use OPT to denote the optimal value of LP (3).

PROPOSITION 1. $\text{OPT} \ge \mathbb{E}[\text{OFF}]$*, i.e.,* $\mathbb{E}[\text{OFF}]$ *is upper bounded by the optimal value of LP (3).*

In the subsequent sections, we will use LP (3) to help design our randomized online algorithms and analyze their competitive ratios.

4. Randomized Algorithms

In this section we discuss a randomization algorithm based on the optimal solution to the LP (3) to solve the matching problem. The performance of the algorithm is analyzed based on its competitive ratio. We then investigate a commonly used heuristic—re-solving heuristic to further improve the algorithm performance. In general, the re-solving heuristic does not always help improve the performance. Yet, we show that re-solving at our proposed times generates better performance.

4.1. Samp(α) Algorithm

Our first randomized algorithm SAMP(α) is shown in Algorithm 1. In this algorithm, we first solve the optimal solution y^*_{uv} from LP (3). Then for each arrival request v, an offer u is randomly chosen to match v with probability αy^*_{uv}. Here α ($0 \le \alpha \le 1$) is a parameter that controls how aggressively the online algorithm makes the matches, and we call it sampling ratio. If the chosen offer u has enough remaining capacity, the match is made successfully and the capacity is updated. Otherwise, we reject the request v. Let $\mathcal{C}_{ut}$ denote the remaining capacity of offer u after running algorithm t rounds ($0 \le t \le T$).

Algorithm 1 SAMP(α)

Solve the LP (3) and get an optimal solution $\boldsymbol{y}^*$
Matching $M = \phi$, capacity $\mathcal{C}_{u0} = c_u$
for $t = 1$ to T **do**
  An online request v arrives
  Randomly choose u with probability αy^*_{uv}
  if $\mathcal{C}_{u(t-1)} \ge d_v$ **then**
    Match u and v: $\mathcal{C}_{ut} = \mathcal{C}_{u(t-1)} - d_v$, $M = M \cup \{(u, v)\}$
  else
    $\mathcal{C}_{ut} = \mathcal{C}_{u(t-1)}$
  end if
end for
return M

Next, we analyze the performance of SAMP(α) . For notation convenience, we first define a function $g(z)$ as below:

$$g(z) = -\frac{D}{2}z^2 + z. \tag{4}$$

It is easy to see that when $z = \frac{1}{D}$, $g(z)$ achieves its maximal value $\frac{1}{2D}$.

We first provide a lower bound of the expected times that edge (u, v) is matched by SAMP(α) during the first t' steps, denoted by $N_\alpha(u, v, t')$.

LEMMA 1. *The expected times that edge (u, v) is matched by* SAMP(α) *during the first t' arrivals $N_\alpha(u, v, t')$ has the following lower bound:*

$$N_\alpha(u, v, t') \ge \left[-\left(\frac{t'}{T}\right)^2 \frac{\alpha^2 D}{2} + \frac{t'}{T}\alpha \right] \cdot T p_v y^*_{uv} = g\left(\alpha \cdot \frac{t'}{T}\right) \cdot T p_v y^*_{uv}.$$

Using Lemma 1 and Proposition 1, we can further provide a lower bound of SAMP(α) 's competitive ratio in the following proposition.

PROPOSITION 2. *The* SAMP(α) *algorithm has a competitive ratio* $\mathrm{CR} \geq \frac{1}{2D}$.

4.2. Re-solving Heuristic

Note that in Algorithm SAMP(α) , the LP (3) is solved only once at the beginning of the algorithm, and the solution to the LP will be used to guide the online algorithm matching probability throughout the whole time span. Yet, in the middle of the process, due to the randomness of the request arriving sequence, it is likely that the capacities of the orders are consumed disproportionately. In such cases, the original LP can no longer capture the correct resource configuration. Specifically, a resource u may not have enough capacity for some requests v while u and v are still considered a valid pair by the original LP. To fix this issue, an intuitive idea is to re-solve the linear program in the middle of the algorithm with the updated capacity information, and update the LP solution to guide the subsequent allocation. We call this refinement step a re-solving step. This re-solving idea has been widely used in several works. For example, Ma and Simchi-Levi (2020) considered an online assortment of products with multiple prices in a simulation study on a publicly accessible hotel data set and found that the re-solving heuristic improves the performance significantly. On the other hand, the effectiveness of the re-solving heuristic varies case by case. Jasin and Kumar (2013) analyzed the performance of the re-solving approach in a revenue management problem known as booking limit and bid-price controls and found that the asymptotic revenue loss of either booking limit or bid price control cannot be reduced by re-solving the deterministic LP problem. Bumpensanti and Wang (2020) studied the performance of the re-solving heuristic in a network revenue management problem, which is essentially an online packing problem. They found that frequently re-solving the LP generates the same order of revenue loss as one could get without re-solving, and proposed an infrequent re-solving with thresholding method to improve the performance measured by a regret bound. To the best of our knowledge, this paper is the first one to investigate the performance of the re-solving heuristic in an online matching problem under the non-asymptotic, competitive ratio metric.

4.2.1. 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, can always help the algorithm to generate a better solution. Intuitively this may seem true. However, we show via a counterexample that the re-solving heuristic may make things worse sometimes. We refer the interested readers to Appendix A.1 for the detailed example.

4.2.2. Re-solving at the right time helps. In this section, we study when to re-solve the LP. We first investigate the re-solving time that helps improve the proposed algorithm's performance if we are only allowed to re-solve the LP once. Let t' denote the re-solving time, i.e., we re-solve the LP right after the t'th arrival (cf. Algorithm 2). Hence, t' is an integer between 0 and $T-1$. Our goal is to find the value of t' that can produce a better competitive ratio than SAMP(α) . We let $T' = T - t'$ denote the remaining periods. We set the sampling ratio $\alpha = \alpha_0$ before re-solving (including t') and $\alpha = \alpha_1$ after re-solving (not including t'). To avoid ambiguity, in the subsequent presentation, when we say re-solving at time t' it means that re-solving after t'th arrivals and t' denotes the re-solving time. We use "before re-solving" to indicate before re-solving including the t'th arrival and "after re-solving" to indicate after re-solving but not including the t'th arrival.

Recall that $\mathcal{C}_{ut'}$ denotes the remaining capacity of u after t' arrivals, we can then re-write the LP as follows.

$$\begin{aligned}
\textbf{max} \quad & \sum_{(u,v)\in E} T' p_v w_{uv} y_{uv} && && (5)\\
\textbf{s.t.} \quad & \sum_{v:(u,v)\in E} p_v d_v y_{uv} \le \frac{\mathcal{C}_{ut'}}{T'}, && \forall u \in U, && (5a)\\
& \sum_{u:(u,v)\in E} y_{uv} \le 1, && \forall v \in V, && (5b)\\
& y_{uv} = 0, && \forall (u,v) \in E, \mathcal{C}_{ut'} < d_v. && (5c)\\
& y_{uv} \ge 0, && \forall (u,v) \in E, &&
\end{aligned}$$

Constraints (5a) and (5b) are similar to LP (3) with an updated capacity of each resource. We add new constraints (5c) in the re-solved LP because when running the algorithm for several rounds, an offline resource u may have less remaining capacity than the demand of a request v. In this case, the edge (u,v) should be deleted from edge set E. To make the analysis simple, we do not update the edge set but add additional constraints to disable the possible match between them. Denote the optimal solution to the re-solved LP (5) as $\boldsymbol{y}'^*$ and its optimal value as OPT$'$. Note that the optimal solution $\boldsymbol{y}'^*$ and the optimal value of the new LP OPT$'$ depend on the remaining capacity, which is a random variable. Hence, $\boldsymbol{y}'^*$ and 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., $\mathcal{C}_{ut'} = c'_u$ for each u, in which case, $\boldsymbol{y}'^*$ and OPT$'$ become deterministic.

Let $N'_\alpha(u,v,T)$ be the expected number of (u,v) that is matched by the re-solving Algorithm 2. We will analyze the expected number of matches before and after the re-solving, respectively.

Algorithm 2 SAMP(α) with re-solving once (t', α_0, α_1)

Solve the LP (3) and get the optimal solution $\boldsymbol{y}^*$
Matching $M = \phi$, capacity $\mathcal{C}_{u0} = c_u$, $\alpha = \alpha_0$
for $t = 1$ to T **do**
  Online request v arrives
  Randomly choose u with probability αy^*_{uv}
  if $\mathcal{C}_{u(t-1)} \geq d_v$ **then**
    Match u and v: $\mathcal{C}_{ut} = \mathcal{C}_{u(t-1)} - d_v$, $M = M \cup \{(u, v)\}$
  else
    $\mathcal{C}_{ut} = \mathcal{C}_{u(t-1)}$
  end if
  if $t = t'$ **then**
    Solve the LP (5) and get the optimal solution $\boldsymbol{y}'^*$
    $\boldsymbol{y}^* = \boldsymbol{y}'^*$, $\alpha = \alpha_1$
  end if
end for
return M

Before Re-solving. According to Lemma 1, the expected number of matches made before the re-solving time point t' (including t') is at least $g\left(\alpha_0 \cdot \frac{t'}{T}\right) \cdot T p_v y^*_{uv}$ where $\alpha_0 \boldsymbol{y}^*$ denotes the matching probability before t'.

After Re-solving. Then we bound the expected number of matches of edge (u, v) after the re-solving (denoted by $N'_{\alpha_1, t'}(u, v, T')$) in the following Lemma 2.

LEMMA 2. *By choosing* $\alpha_1 = \frac{T'/D}{\lceil T'/D \rceil}$, *the expected number of matches of edge* (u, v) *after re-solving, i.e.,* $N'_{\alpha_1, t'}(u, v, T')$ *is bounded from below by* $\frac{1}{2D} \cdot T' p_v y'^*_{uv}$.

Therefore, the expected number of times that (u, v) is matched by Algorithm 2

$$N'_\alpha(u, v, T) \geq g\left(\alpha_0 \cdot \frac{t'}{T}\right) \cdot T p_v y^*_{uv} + \frac{1}{2D} \cdot T' p_v y'^*_{uv}. \tag{6}$$

The corresponding expected reward generated from the re-solving Algorithm 2 can hence be bounded from below as follows.

$$\sum_{uv} w_{uv} N'_\alpha(u, v, T) \geq g\left(\alpha_0 \cdot \frac{t'}{T}\right) \cdot \mathrm{OPT} + \frac{1}{2D} \cdot \mathrm{OPT}'.$$

Based on this inequality, the competitive ratio can be bounded from below as follows.

$$\mathrm{CR}' \geq g\left(\alpha_0 \cdot \frac{t'}{T}\right) + \frac{1}{2D} \cdot \frac{\mathrm{OPT}'}{\mathrm{OPT}}. \tag{7}$$

Bearing in mind that the remaining capacity $\mathcal{C}_{ut'}$ is a random variable, OPT' also should be a random variable. Then to bound the competitive ratio by inequality (7), we need to derive a lower bound of the expectation of the ratio $\frac{\mathrm{OPT}'}{\mathrm{OPT}}$ which is showed in Proposition 3 below.

PROPOSITION 3. $\mathbb{E}\left[\frac{\mathrm{OPT}'}{\mathrm{OPT}}\right] \geq \left(1 - \frac{t'}{T}\right) e^{-R\frac{\alpha_0 t'}{T}}$, *where* $R = \max_{(u',v')\in E} \frac{c_{u'}}{d_{v'}}$.

Proposition 3 establishes a lower bound of the expected ratio between OPT' and OPT. Together with the lower bound of the competitive ratio established in (7), we can further establish the following proposition to demonstrate that re-solving at $t' = \lceil \frac{T}{D} \rceil$ helps to achieve a better lower bound for the competitive ratio.

PROPOSITION 4. *Re-solving once at time step* $t' = \lceil \frac{T}{D} \rceil$ *with* $\alpha_0 = \frac{T/D}{\lceil T/D \rceil}$ *and* $\alpha_1 = \frac{T'/D}{\lceil T'/D \rceil}$ *in Algorithm 2 achieves a competitive ratio of*

$$\mathrm{CR}' \geq \frac{1}{2D} + \frac{1}{2D}\left(1 - \frac{\lceil T/D \rceil}{T}\right) e^{-\frac{R}{D}}. \tag{8}$$

Note that the first term in the established competitive ratio (8) is exactly the one we have derived for SAMP(α) . The second term is non-negative as long as $D \geq 1$, which indicates that re-solving at our proposed time will generate at least the same competitive ratio as SAMP(α) . We also note that the second term increases as R decreases, which indicates that the extra gain from resolving increases. To understand the intuition, note that when resource u does not have enough capacity for a request v at some point in running the algorithm, SAMP(α) algorithm may still try to match these pairs while re-solving can help us avoid this invalid matching. This is how re-solving can help improve the algorithm performance. A smaller R implies a smaller capacity or a larger demand, which will make such invalid cases appear more often. Hence, it will further improve the performance gain from re-solving.

4.2.3. Re-solving many times at the right times can further help improve the performance. Having discussed the algorithm that re-solves the LP only once, we can further generalize the results and discuss what are the right times when the algorithm is allowed to re-solve the LP many times. Note that when the algorithm first re-solves the LP at $\lceil \frac{T}{D} \rceil$, the updated LP represents a smaller subproblem within time horizon $(\lceil \frac{T}{D} \rceil, T]$. This motivates us to apply the resolve-once heuristic recursively to continue improving the performance of the algorithm. Assume

that we can re-solve the LP K times in total. Before the first re-solve, we apply SAMP(α) with $\alpha = \alpha_0$. After ith ($1 \le i \le K$) re-solve, we allocate resources based on SAMP(α_i) and assume the new re-solve time point is at γ proportion of the remaining time. Denote the ith re-solving time point as t_i and the remaining time after the ith re-solve as T_i for $i = 1, \ldots, K$. Set $t_0 = 0$ and $T_0 = T$. Then our proposed re-solving times are at $\mathbf{t} = (t_1, t_2, \ldots, t_K)$, where

$$t_i = \begin{cases} 0, & i = 0 \\ t_{i-1} + \lceil \gamma(T - t_{i-1}) \rceil, & i > 0. \end{cases} \tag{9}$$

Without loss of generality, we assume $t_K \le T - 1$. We call such a re-solving algorithm log-resolving algorithm and present the algorithm details in Algorithm 3.

Algorithm 3 Log-resolving Algorithm (γ, K, α_0, α_1, α_2, $\cdots$, α_K)

Solve the LP (3) and get the optimal solution $\boldsymbol{y}^*$
Calculate re-solve times $\mathbf{t} = (t_1, t_2, \ldots, t_K)$.
Matching $M = \phi$, capacity $\mathcal{C}_{u0} = c_u$, $\alpha = \alpha_0$
for $t = 1$ to T **do**
  Online request v arrives
  Randomly choose u with probability αy^*_{uv}
  if $\mathcal{C}_{u(t-1)} \ge d_v$ **then**
    Match u and v: $\mathcal{C}_{ut} = \mathcal{C}_{u(t-1)} - d_v$, $M = M \cup \{(u, v)\}$
  else
    $\mathcal{C}_{ut} = \mathcal{C}_{u(t-1)}$
  end if
  if $t = t_i$ for $1 \le i \le K$ **then**
    Solve the LP (5) and get the optimal solution $\boldsymbol{y}'^*$
    $\boldsymbol{y}^* = \boldsymbol{y}'^*$, $\alpha = \alpha_i$
  end if
end for
return M

Let $\boldsymbol{y}^{*(i)}$ and OPT_i denote the optimal solution and optimal value of LP after the ith re-solve. Let $\mathcal{C}_u^{(i)}$ denote the remaining capacity of resource u right before ith re-solve. Note that $\mathcal{C}_u^{(i)}$ is a random variable, we first study the bound of the expected ratio between the optimal values of two consecutive linear programs OPT_i and OPT_{i+1} conditioned on a realization of $\mathcal{C}_u^{(i)}$, denoted by

$c_u^{(i)}$. Denote $\Delta_i = t_{i+1} - t_i = \lceil \gamma T_i \rceil$ for $0 \le i \le K-1$. Similar to Proposition 3, we can bound the expected ratio of two consecutive optimal LP values in the following proposition.

PROPOSITION 5. *Conditioned on $\mathcal{C}_u^{(i)} = c_u^{(i)}$ for every u, when $\frac{R}{T_i} << 1$, we have*

$$\mathbb{E}\left[\frac{\mathrm{OPT}_{i+1}}{\mathrm{OPT}_i}\middle|\boldsymbol{c}^{(i)}\right] \ge \left(1 - \frac{\Delta_i}{T_i}\right) e^{-\alpha_i \frac{\Delta_i}{T_i} R} \tag{10}$$

With this proposition, we are now ready to establish our first main result in the paper.

THEOREM 1. *Assuming $T_K \gg \max_u c_u$, the log-resolving algorithm (Algorithm 3) with $\gamma = \frac{1}{D}$ and $\alpha_i = \frac{T_i/D}{\lceil T_i/D \rceil}$ has a competitive ratio of at least $\frac{1}{2D}\frac{1-\theta^{K+1}}{1-\theta}$, where $\theta = \left(1 - \frac{1}{D}\right) e^{-\frac{R}{D}}$.*

Note that $\frac{1}{2D}$ is exactly the one we have derived for SAMP(α) and $\frac{1-\theta^{K+1}}{1-\theta}$ is greater than one for any positive integer value K. Theorem 1 shows that the proposed log-resolving algorithm can indeed help improve the lower bound of the competitive ratio for the general online bipartite matching problem.

Computational Complexity. We now discuss the time complexity of our algorithms. 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 K linear programs for the log-resolving algorithm. According to Equation (9), we have the following inequalities:

$$t_i \ge \begin{cases} 0, & i = 0 \\ t_{i-1} + \gamma(T - t_{i-1}) = (1-\gamma)t_{i-1} + \gamma T, & i > 0 \end{cases}$$

which gives us

$$t_K \ge (1-\gamma)t_{K-1} + \gamma T \ge (1-\gamma)((1-\gamma)t_{K-2} + \gamma T) + \gamma T \ge \cdots \ge \gamma T \sum_{i=0}^{K-1} (1-\gamma)^i = T(1-(1-\gamma)^K).$$

Because $t_K \le T - 1$, we then have $T(1-(1-\gamma)^K) \le T - 1$. This means $K \le \frac{\log T}{-\log(1-\gamma)}$. Therefore, in the log-resolving algorithm, we need to re-solve $K = O(\log T)$ linear programs.

4.3. Quality of the Re-solving Heuristic

We now discuss the quality and necessity of the re-solving heuristic to the LP-based algorithm. Theorem 1 suggests that re-solving can help the algorithm achieve a better competitive ratio. However, this is not a formal statement since the ratios claimed in Proposition 4 and Theorem 1 are only lower bounds, and it is still possible that the actual competitive ratio remains the same with or without the re-solving heuristic.

In the following we formalize the idea that re-solving heuristic can indeed improve the competitive ratio of an LP-based algorithm. More specifically, we will demonstrate through a simple example that an algorithm with the re-solving heuristic can achieve a competitive ratio that is strictly better than the competitive ratio upper bound that any LP-based algorithm without re-solving can achieve.

INSTANCE 1. *Consider the following problem instance:*

- *There are* 2^n *offline resource vertices* $\{u_1, u_2, \ldots, u_{2^n}\}$ *where* $n \gg 1$. *The vertices have the same capacity* $c_i = 2, \forall i \in \{1, 2, \ldots, 2^n\}$.
- *There are two types of online request* v_1 *and* v_2*:*
  - *Each request of type* v_1 *has demand* $d = 1$ *and arrival probability* $q = \frac{1}{2^n}$. *This request can only be matched to* u_1, *with weight* $w_{11} = \frac{1}{2} - \epsilon$ *where* $0 \le \epsilon < \frac{1}{2}$.
  - *Each request of type* v_2 *has demand* $d = 2$ *and arrival probability* $p = 1 - q$. *This request can be matched to any offline resource vertex in the bipartite graph with weight 1. That is,* $w_{i2} = 1, \forall 1 \le i \le 2^n$.
- *The time horizon includes* $T = 2^n$ *rounds.*

First we get the offline optimal value OFF in Lemma 3.

LEMMA 3. *The offline optimal value* OFF *of Instance 1 is* $2^n p + o(2^n)$.

Using Instance 1, we can provide an upper bound of the competitive ratio of any LP-based algorithm without re-solving in Theorem 2.

THEOREM 2 **(Upper Bound)**. *Any LP-based randomized algorithm* SAMP(α) *can not achieve a competitive ratio of* $1 - \frac{1}{e} + c \approx 0.632 + c$ *for any constant* $c > 0$.

Next, we analyze the re-solving algorithms proposed in this paper with Instance 1. Let RES_1 denote Algorithm 2 with our proposed re-solving time $t' = \lceil \frac{T}{D} \rceil$ and parameters $\alpha_0 = \frac{T/D}{\lceil T/D \rceil}$ and $\alpha_1 = \frac{T'/D}{\lceil T'/D \rceil}$. For Instance 1, we re-solve at time $\frac{T}{2}$ and $\alpha_0 = \alpha_1 = 1$.

PROPOSITION 6. *The re-solving algorithm* RES_1 *can achieve a competitive ratio of at least 0.710 with Instance 1.*

Proposition 6 already shows that resolving once (at the right time) can improve the competitive ratio of the LP-based algorithm with the proposed problem instance. Next we continue to analyze what would happen when we can re-solve multiple times. Let RES_K denote the log-resolving algorithm (Algorithm 3) with $\gamma = \frac{1}{D} = \frac{1}{2}$, $\alpha_i = \frac{T_i/D}{\lceil T_i/D \rceil} = 1$ and K re-solvings. We assume that both $2^{n-K} \gg 1$ and $K \gg 1$. According to Algorithm 3, the ith re-solve happens after $T - \frac{T}{2^i}$ arrivals.

PROPOSITION 7. *The log-resolving algorithm* RES_K *can achieve a competitive ratio of at least 0.787 with Instance 1.*

To summarize, for the problem instance considered in this section, if we re-solve at right time ($t' = \lceil \frac{T}{D} \rceil$), we can improve the competitive ratio performance of the algorithm against any LP-based randomization algorithm $\text{SAMP}(\alpha)$ from 0.632 to 0.710. Furthermore, if we are allowed to re-solve multiple times, using the proposed log-resolving algorithm, the competitive ratio can be further improved (from 0.710 to 0.787). These demonstrate that the re-solving technique does have the power to improve the algorithm performance with theoretical guarantees.

5. With Information of Demand Distribution

The previous sections only assume the information on the online arrival process, but no distributional information on demand size. Therefore, the competitive ratio is analyzed using the worst-case analysis of demand size, i.e., the maximal demand in the historical data. In the case when we have enough historical data on the past arrivals, we can further estimate the distribution of demand size. In this section, we consider a model setting with not only the online arrival information but also the distribution of the demand size, and investigate whether incorporating the demand distribution information can further improve the algorithm performance.

To understand the value of distributional information on demand size, we first aggregate the type of each online arrival as follows. For online vertices with the same adjacent offline vertex set and the same weight for each corresponding incident edge, we aggregate them into a group. In other words, the set defined by $Q(v) = \{v' \mid Adj(v') = Adj(v), w_{uv'} = w_{uv}, \forall u \in Adj(v)\}$ includes all the online vertices in the same group as v. Then we can distinguish online vertices by its affiliated group and demand size. We label the group index of each vertex $v \in V$ as $q(v)$, and its affiliated group is defined by $Q_{q(v)}$, i.e., $Q_{q(v)} = Q(v)$. We denote the whole set of group indices as Q. Denote the demand set in a group q for $q \in Q$ as L_q. The probability of getting a vertex v in group q (i.e., $v \in Q_q$) for $q \in Q$ is $\sum_{v \in V: q(v) = q} p_v$. For each vertex in group q, we define the demand distribution as $p_{l|q} = \frac{p_{ql}}{p_q}$, where $p_{ql} = \sum_{v \in V: q(v) = q, d_v = l} p_v$ and $p_q = \sum_{l \in L_q} p_{ql}$.

From the definition of groups, we can get the following lemma.

LEMMA 4. *There exists an optimal solution* $\boldsymbol{y}$ *to LP (3), such that for any* v, v' *satisfying* $q(v) = q(v')$ *and* $d_v = d_{v'}$*, we have* $y_{uv} = y_{uv'}$.

We view the set of vertices in a group q and with a demand size of l as a super vertex and use v_{ql} to denote this super vertex. By the definition of a group, each super vertex has the same adjacent

set as any vertex in the corresponding group. We use E^s to denote the set of edges between super vertices and offline resources. Again by definition, we have that the edge between each vertex in a group and a given offline resource shares the same weight. We use w_{uq} to denote the weight of the edge $(u,v)\in E$ for $v\in Q_q$. Then we can revise LP (3) accordingly as below.

$$\begin{aligned}
\textbf{max} \quad & \sum_{(u,v_{ql})\in E^s} T w_{uq} \sum_l p_{ql} y_{uv_{ql}} \\
\textbf{s.t.} \quad & \sum_{q\in Q}\sum_{l\in L_q} p_{ql} l y_{uv_{ql}} \le \frac{c_u}{T}, && \forall u\in U, \\
& \sum_{u\in Adj(v_{ql})} y_{uv_{ql}} \le 1, && \forall q\in Q, l\in L_q, \\
& y_{uv_{ql}} \ge 0, && \forall (u,v_{ql})\in E^s.
\end{aligned} \tag{11}$$

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

Lemma 5. *LP (11) is equivalent to LP (3). Specifically, for any optimal solution $\{y^*_{uv_{ql}}\}$ to LP (11), we can construct $y^*_{uv} = y^*_{uv_{ql}}$ for any v satisfying $q(v)=q$ and $d_v = l$, and the result $\{y^*_{uv}\}$ is optimal to LP (3).*

In the following analysis, we will focus on the super vertex in each group and solve the revised LP (11). Each super vertex arrives with a probability of p_{ql}. For each arrival, a resource is matched according to the revised LP (11). Denote the optimal solution to LP (11) as $y^*_{uv_{ql}}$ for each u, v_{ql}.

Let $N_\alpha(u,q,l,t')$ be the expected number of matched edge (u,v_{ql}) from $t=1$ to $t=t'$. Following a similar analysis to (19), we have

$$\begin{aligned}
N_\alpha(u,q,l,t') &\ge \left[-\left(\frac{t'}{T}\right)^2 \frac{\alpha^2 c_u}{2(c_u-l+1)} + \frac{t'}{T}\alpha\right] T p_{ql} y^*_{uv_{ql}} \\
&\ge \left[-\left(\frac{t'}{T}\right)^2 \frac{\alpha^2 l}{2} + \frac{t'}{T}\alpha\right] T p_{ql} y^*_{uv_{ql}},
\end{aligned} \tag{12}$$

where $\frac{c_u}{c_u-l+1} = 1 + \frac{l-1}{c_u-l+1} \le l$ since $l \le c_u$.

Let $A_l(t') = -\frac{\alpha^2 l}{2}(\frac{t'}{T})^2 + \frac{t'}{T}\alpha$ and $A_l(t')$ is decreasing in l for any $0 < t' \le T$. For a fixed group q and demand size l, let $W_{ql} = \sum_u w_{uq} y^*_{uv_{ql}}$ represent the expected reward generated by the vertices in group q and with demand l. Then we can write down the expected reward generated from this algorithm:

$$
\begin{aligned}
\mathbb{E}[\textsc{Samp}(\alpha)\,] &= \sum_{u,q,l}[w_{uq}N_\alpha(u,q,l,T)] \\
&\geq \sum_{u,q,l}[w_{uq}N_\alpha(u,q,l,t')] \\
&\geq \sum_{u,q,l} Tp_{ql}w_{uq}A_l(t')y^*_{uv_{ql}} \\
&\geq \sum_{q\in Q}\sum_{l\in L_q} Tp_{ql}A_l(t')\sum_u w_{uq}y^*_{uv_{ql}} \\
&= \sum_{q\in Q}\sum_{l\in L_q} Tp_{ql}A_l(t')W_{ql}
\end{aligned} \tag{13}
$$

For notation simplicity, we omit t' and use A_l to denote $A_l(t')$. To analyze the lower bound of its competitive ratio, we first establish some properties of vertices in the same group in the following lemma.

LEMMA 6. *If $q(v)=q(v')=q$ and $d_v=l<m=d_{v'}$, then $W_{ql}\geq W_{qm}$.*

Lemma 6 indicates that within the same group, the vertices with a smaller demand generates a higher expected reward. Based on this observation, we can build the following lemma.

LEMMA 7. *For each group $q\in Q$, when $\sum_{l\in L_q} Tp_{ql}W_{ql}$ is fixed, we have*

$$
\sum_{l\in L_q} Tp_{ql}A_lW_{ql} \geq \left(-\frac{1}{2}\mathbb{E}[D_q]\left(\frac{t'}{T}\right)^2\alpha^2+\left(\frac{t'}{T}\right)\alpha\right)W_q,
$$

where $\mathbb{E}[D_q]$ is the expected demand of vertices in group q and W_q represents $\sum_{l\in L_q} Tp_{ql}W_{ql}$.

Denote $M_e=\max_{q\in Q}\mathbb{E}[D_q]$, and we are ready to present a new lower bound of the $\textsc{Samp}(\alpha)$ when the demand distribution is available. We formally describe it in Theorem 3.

THEOREM 3. *With distributional information of demand size, $\textsc{Samp}(\alpha)$ has a competitive ratio of $\mathrm{CR}\geq\frac{1}{2M_e}$.*

5.1. Re-solving with Demand Distribution

With distributional information of demand size, we can again apply the re-solving heuristic to see whether we can further improve the competitive ratio. According to Lemma 5, the deterministic linear program (3) can be reformulated as (11). When re-solving the problem, the deterministic linear program can be revised accordingly as follows:

$$
\begin{aligned}
\textbf{max} \quad & \sum_{(u,v_{ql})\in E^s} T w_{uq} \sum_{l} p_{ql} y_{uv_{ql}} \\
\textbf{s.t.} \quad & \sum_{q\in Q}\sum_{l\in L_q} p_{ql} l y_{uv_{ql}} \le \frac{c_u}{T}, && \forall u \in U \\
& \sum_{u\in Adj(v_{ql})} y_{uv_{ql}} \le 1, && \forall q \in Q, l \in L_q \\
& y_{uv_{ql}} \ge 0, && \forall (u, v_{ql}) \in E^s \\
& y_{uv_{ql}} = 0, && \forall (u, v_{ql}) \in E^s, c_u < l
\end{aligned} \tag{14}
$$

Follow a similar analysis in the previous section, we have

$$
\mathbb{E}[\mathrm{ALG}(0,T)] \ge \left[-\left(\frac{t'}{T}\right)^2 \frac{\alpha^2 M_e}{2} + \frac{t'}{T}\alpha \right] \mathrm{OPT} + \frac{1}{2M_e}\mathbb{E}[\mathrm{OPT}'] \tag{15}
$$

Following a similar analysis to Proposition 4, we can derive a new lower bound of the competitive ratio when re-solving once at step $\lceil \frac{T}{M_e} \rceil$. The same analysis also applies to the case with multiple times of re-solves. We summarize the results in Proposition 8.

PROPOSITION 8. *When distributional information of demand size is available,*

1. *re-solving once at* $\lceil T/M_e \rceil$ *has a competitive ratio of at least*

$$
\frac{1}{2M_e} + \frac{1}{2M_e}\left(1 - \frac{\lceil T/M_e \rceil}{T}\right) e^{-\frac{R}{M_e}};
$$

2. *assuming* $T_K \gg \max_u c_u$*, re-solving* K *times following the log-resolving algorithm (Algorithm 3) with* $\gamma = \frac{1}{M_e}$ *and* $\alpha_i = \frac{T_i/M_e}{\lceil T_i/M_e \rceil}$ *has a competitive ratio of at least*

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

5.2. The Value of Demand Distribution

To have a better understanding of how distributional information on demand size can help improve our algorithm, we provide a concrete example as follows.

- 1 offline vertex: $c = 2$
- 3 online vertices: v_0, v_1 and v_2
- v_0: $p_0 = \frac{1}{2}$, $d_0 = 1$, $w_0 = w > 1$
- v_1: $p_1 = \frac{1}{3}$, $d_1 = 1$, $w_1 = 1$
- v_2: $p_2 = \frac{1}{6}$, $d_2 = 2$, $w_2 = 1$
- $T = 4$

According to the definition of *group*, v_1 and v_2 are in the same group. And the expected demand for this group is $\frac{1/3}{1/2} \times 1 + \frac{1/6}{1/2} \times 2 = \frac{4}{3} > 1$. Hence, $M_e = \frac{4}{3}$. According to the first part in Proposition 8, we need to re-solve after three arrivals. In contrast, without the demand distribution, Proposition 4 suggests to re-solve after two arrivals. We use RES_2 to denote the algorithm that re-solves the linear program after two arrivals and RES_3 to denote the one re-solving after three arrivals. We will compare the expected reward generated by the two algorithms.

By solving the offline LP, we know that the algorithm will always accept v_0 as long as the capacity permits and reject v_1 and v_2 before the re-solving time point. To analyze the performance of the algorithms. We consider all possible arrival sequences as follows.

1. If there are two v_0 in the first two arrivals, there is no capacity left and both algorithms will reject all the requests after the two arrivals. Hence RES_2 and RES_3 generate the same reward.

2. If there is only one v_0 in the first two arrivals, one capacity is consumed. After two arrivals, the re-solved LP generates the same optimal strategy as before. Therefore, if the third arrival is v_0, RES_2 and RES_3 generates the same reward. If the third arrival is either v_1 or v_2, no capacity is consumed in this period. After the third arrival, from the re-solved LP we can imply that RES_3 will match the last arrival with probability 1 if it is either v_0 or v_1, and reject v_2 due to insufficient capacity. In contrast, RES_2 will only accept v_0 with probability 1 but reject v_1 and v_2 in the last arrival. Therefore, RES_3 can generate an expected reward higher than RES_2.

3. If there is no v_0 in the first two arrivals, after re-solving, RES_2 will match v_0 and v_1 with probability 1 and v_2 with probability $\frac{1}{2}$ for the subsequent arrivals. In contrast, RES_3 will only match v_0 with probability 1 but reject v_1 and v_2 in the third arrival. If the third arrival is v_0, it is matched by RES_2 and the capacity decreases by 1, hence it will not be able to match v_2 if the fourth arrival is v_2 using RES_2 due to insufficient capacity. In the same case with the third arrival being v_0, RES_3 will match the request and the re-solved LP will suggest RES_3 to match the last arrival with probability 1 if it is either v_0 or v_1, and reject v_2 due to insufficient capacity. Therefore in the case that the third arrival is v_0, both algorithms generate the same reward (see the first row in Table 1). If the third arrival is not v_0, the RES_3 will reject it and the re-solved linear program will suggest to match all possible arrivals for the last arrival. Following a similar analysis, if the third arrival is v_1, RES_2 will accept v_1 in the third arrival and accept v_0 and v_1 with probability 1 but reject v_2 due to insufficient capacity in the fourth arrival. If the third arrival is v_2, RES_2 will accept v_2 with probability $\frac{1}{2}$ in the third arrival. If v_2 is accepted, the resource runs out of capacity and no match will be made in the fourth round. In contrast, if v_2 is reject in the third round, RES_2 will match v_0 and v_1 with probability 1 and v_2 with probability $\frac{1}{2}$ in the last arrival. We enumerate

3rd arrival	RES_2	RES_3
v_0	$w+(\frac{1}{2}w+\frac{1}{3}\times 1+\frac{1}{6}\times 0)$	$w+(\frac{1}{2}w+\frac{1}{3}\times 1+\frac{1}{6}\times 0)$
v_1	$1+(\frac{1}{2}w+\frac{1}{3}\times 1+\frac{1}{6}\times 0)$	$0+(\frac{1}{2}w+\frac{1}{3}\times 1+\frac{1}{6}\times 1)$
v_2	$\frac{1}{2}\times 1+\frac{1}{2}(\frac{1}{2}w+\frac{1}{3}\times 1+\frac{1}{6}\times \frac{1}{2})$	$0+(\frac{1}{2}w+\frac{1}{3}\times 1+\frac{1}{6}\times 1)$

Table 1 Enumerated Cases When There Is No v_0 In The First Arrivals

the conditional expected reward for all cases in Table 1, from which we can compute the different of the conditional expected reward between RES_3 and RES_2 given no v_0 in the first two arrivals, which is $\frac{1}{24}w-\frac{5}{16}$. When w is large enough, e.g., $w\geq 8$, RES_3 generates a higher expected reward than RES_2.

In summary, the analysis above indicates that when demand distribution is available, there are cases where incorporating the expected demand information can help the algorithm to achieve a better performance.

6. Numerical Studies

In this section, we evaluate the performance of the online algorithms proposed in Section 4 by simulating the algorithms on a ride hitch data set. The data set is generated from an available New York city taxi data set. In the following sections, we will first describe the New York city taxi data and introduce the way to pre-process the data based on the ride hitch application. After that, we will present all the benchmarks considered in the paper and compare their performance thoroughly.

6.1. Data Description and Processing

New York City Taxi Data Set. We obtain the data from Donovan and Work (2014). The data set records taxis' trips. Each record includes the number of passengers, pick-up coordinates and time, drop-off coordinates and time. We pre-process the data based on the ride hitch application introduced in Section 1. We normalize all the pick-up and drop-off coordinates and time between 0 and 1 and construct the matching graph as follows.

Graph Construction. Let U, V represent the offline resource and online request set, respectively, and E denote the edge set. To build a bipartite graph $G=(U,V,E)$, we first generate $1,000$ data records from the taxi data and divide them into two sets—online vertex (resource) and offline vertex (request) sets. Let r represent the ratio of the expected number of online vertices and offline vertices. We regard each data record as an offline vertex with probability $\frac{1}{r+1}$ and as an online vertex with probability $\frac{r}{r+1}$. In the experiments, we set $r=1,2,3$. Each offline vertex $u\in U$ is associated with a tuple $u=(t_u,s_u,f_u,c_u)$, where t_u is the proposed starting time of the trip; s_u

and f_u denote the source and destination of the trip; c_u is the capacity of the driver's vehicle, i.e., how many seats the driver can offer for ride hitch. We use the recorded number of passengers for each resource to represent its capacity. Each online vertex $v \in V$ is also associated with a tuple $v = (t_v, s_v, f_v, d_v)$. The first three parameters t_v, s_v and f_v have similar meanings to those in an offer. The last parameter d_v denotes the number of seats requested in this request. We use the number of passengers in each request to represent its demand. We further truncate the maximal demand to 3 from our ride experience.

After partitioning the data to online and offline vertices, we start to build the matching graph. For any resource u and request v, an edge $e = (u, v) \in E$ exists, i.e., the resource can be matched to the request, if the following three conditions are met:

1. The starting time of the two trips are closed to each other. Specifically, $|t_v - t_u| \leq \delta_{time}$ where δ_{time} is a pre-specified time allowance.
2. The detour for the driver to take the rider is no more than a threshold δ_{detour}. Specifically, let $\text{dist}(a, b)$ be a ℓ_1 distance [1] from any location a to location b. Then the condition can be written as
$$\text{dist}(s_u, s_v) + \text{dist}(s_v, f_v) + \text{dist}(f_v, f_u) - \text{dist}(s_u, f_u) \leq \delta_{detour}.$$
3. The number of seats requested in request v is no more than the vehicle u's capacity, i.e., $d_v \leq c_u$.

Note that different thresholds δ_{time} and δ_{detour} correspond to different degrees in the graph. We have tested the algorithms under graphs with various degrees and the obtained results are similar. Without loss of generality, we set $\delta_{time} = 0.05, \delta_{detour} = 0.1$.

After building edges, we randomly set the weight of each edge. In particular, we uniformly sample the weights from an interval $(1, x)$. We use $U(1, x)$ to denote the distribution of the weights. In the following experiment, we test each problem instance using either $U(1, 2)$ or $U(1, 5)$. We assume the arrival probability $p_v = \frac{1}{|V|}, \forall v \in V$. The procedure above converts each data set to a bipartite graph represented by $G(U, V, E)$.

6.2. Experiment

We first introduce the benchmarks used in the experiment. Three benchmarks are compared. The first is a greedy algorithm—a widely used benchmark in the literature (Xu et al. (2019); Dickerson et al. (2018); Lowalekar et al. (2020)). The second benchmark is based on Algorithm 1 in Naori and

[1] The ℓ_1 distance, also known as the Manhattan distance, is defined as the distance between two points measured along axes at right angles. In a plane with p_1 at (x_1, y_1) and p_2 at (x_2, y_2), it is $|x_1 - x_2| + |y_1 - y_2|$.

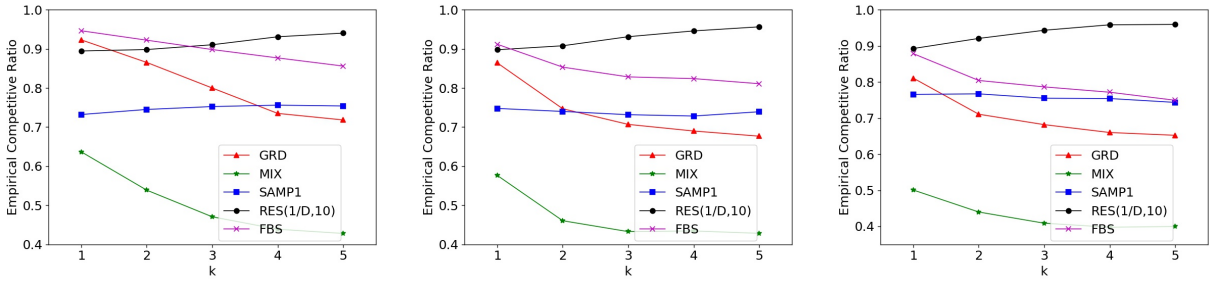
Raz (2019), in which the authors studied an online generalized assignment problem. The problem setting is similar to ours but they assume a random-order arrival model. They design an algorithm by mixing the algorithms for both "heavy" packing options, which is defined as those consume more than half of a bin's capacity, and the "light" options. The mixing method improves the competitive-ratio for the GAP from the best-known $\frac{1}{8.1}$ to $\frac{1}{6.99}$. The detailed algorithm MIX based on Algorithm 1 in Naori and Raz (2019) is shown in Algorithm 4 in Appendix D.1. The third benchmark, denoted by FBS, is based on Algorithm 4 in Vera and Banerjee (2021) in which they studied a general online allocation problem that shares a similar mathematical formula to ours. Their algorithm re-solves the LP after each arrival and can achieve a constant regret bound under mild assumptions on the arrival process.

To examine the value of the re-solving times, we further compare our resolving heuristic with a linear-resolving algorithm. In the linear-resolving algorithm, the matching algorithm follows the SAMP(α) framework but re-solves at time points $t = T\frac{i}{K+1}$ for $i = 1, \ldots, K$, where K denotes re-solve times. We use CONST(K) to denote the linear-resolving algorithm that re-solves K times in total. We set $\alpha = 1$ throughout the section. In summary, we test the following algorithms in this section.

- SAMP1: SAMP(α) algorithm (Algorithm 1) with $\alpha = 1$.
- Greedy (GRD): 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.
- MIX: An algorithm derived from Naori and Raz (2019). Details refer to Algorithm 4 in the Appendix D.1.
- Fluid Bayes Selector (FBS): Algorithm 4 described in Vera and Banerjee (2021). Details refer to Algorithm 5 in the Appendix D.2.
- RES(γ, K): The log-resolving algorithm (Algorithm 3) that re-solves LP K times. We test different values of γ including $\gamma = \frac{1}{D}$, which is equal to $\frac{1}{3}$ in the tested data sets.
- CONST(K): A linear-resolving algorithm that re-solves LP K times. Specifically, the algorithm updates the offline LP at $t = i\lceil\frac{T}{K+1}\rceil$ for $1 \le i \le K$ under the SAMP1 framework.

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

$$\mathrm{ECR}_L = \frac{\sum_{s\in S} P_L(s)}{\sum_{s\in S} \mathrm{OFF}(s)},$$



(a) $r = 1$ (b) $r = 2$ (c) $r = 3$

Figure 1 Effect of Imbalances Between Demand and Supply ($w_{uv} \sim U(1,5)$)

where S denotes the set of sampled request sequences and $P_L(s)$ denotes the reward by algorithm L for the sequence s. Note that different algorithms share the same offline optimal reward, the comparison of empirical competitive ratio is equivalent to a comparison of the reward.

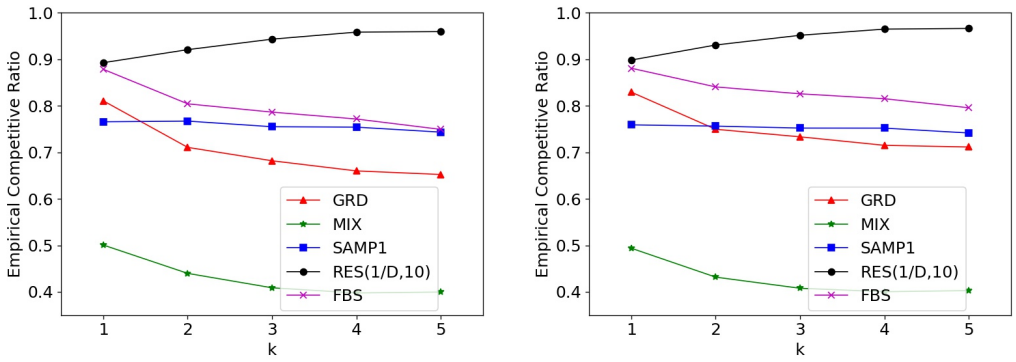
The experiment is organized as follows. We test the aforementioned algorithms under different problem settings. Specifically, we first examine the performance of each algorithm under different market conditions, including the imbalance between demand and supply and the heterogeneity in market segments. After that, the effect of re-solving timing on the performance of the randomized algorithms is investigated.

Effect of Market Conditions We investigate how various algorithms perform under different market conditions. Specifically, we examine the effect of two market metrics: imbalances between demand and supply, and heterogeneity of market segments.

Recall that $r = \frac{\mathbb{E}|V|}{\mathbb{E}|U|}$ denotes the ratio between the expected number of online and offline vertices when randomly partitioning the data, and we use $k = \frac{T}{|V|}$ to denote the ratio between the planning horizon and the number of online vertices. For a fixed graph G, the larger the planing horizon T is, the higher frequency of each online vertex in the request sequences. Hence both r and k measure the imbalances between demand and supply in the market. The larger the parameter k (r), the higher the demand intensity. We test $k = 1, 2, 3, 4, 5$ and $r = 1, 2, 3$. For a given bipartite graph G and a planing horizon T, we uniformly sample each request from the demand pool in constructing the graph and sample the edge weights from $U(1,5)$.

We evaluate the performance of various algorithms under different demand intensity levels and plot the results in Figure 1. From the figures, we can see that in general, our proposed log-resolving method with 10 rounds of re-solves (RES(1/D,10)) outperforms all the benchmarks. In particular, from the comparison between the log-resolving algorithm and SAMP1 algorithm, we can see that

incorporating the re-solving heuristic in the randomization algorithm improves the reward by nearly 20%. The greedy method performs well when the demand intensity level is low but badly when demand increases. In contrast, the performance of our randomized algorithms is more robust across different market conditions. One possible explanation to this phenomenon is that when the demand is high, strategically skipping some inferior requests can lead to a more efficient utilization of the resource capacity. The MIX algorithm performs inferiorly in all cases. One possible reason is that this algorithm was designed for a random-order arrival model instead of i.i.d. arrivals, and a good performance in the random-order model does not necessarily imply a good performance in the i.i.d. arrival model. In fact, according to the algorithm, it always waits and does not make any matches in the first $0.5256T$ arrivals. Finally, the FBS performs the best among the benchmarks. However, it still consistently under-performs our resolving heuristic. Moreover, we observe that it achieves a quite good performance when the demand is less intense but its performance drops when demand increases.



(a) $w_{uv} \sim U(1,5)$ (b) $w_{uv} \sim U(1,2)$

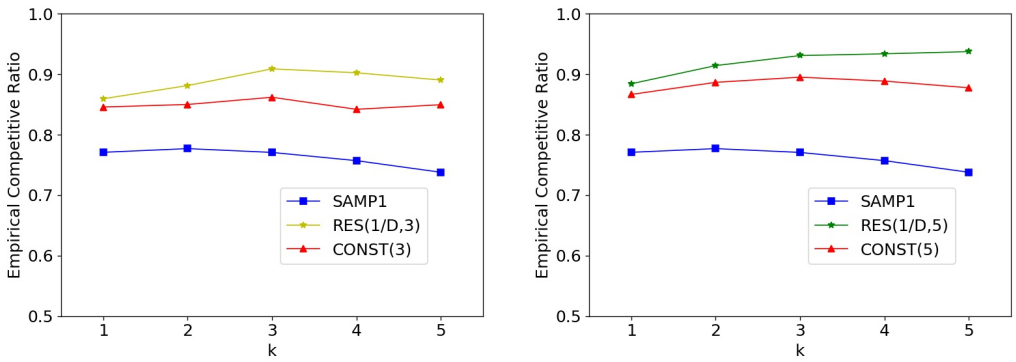
Figure 2 Effect of Heterogeneity in Market Segments ($r = 3$)

We extend the tests to investigate the effect of heterogeneity in the market segment. We measure the market segment by the range of weights in the matching. The wider range the weights are located in, the more diverse the market segment is. We compare the weight range $U(1,5)$ with $U(1,2)$ and plot the performance of each algorithm in Figure 2. The comparisons among algorithms at different values of k are consistent with the observations in Figure 1. From the comparison between figures 2(a) and 2(b), we can see that when the weight range becomes smaller, indicating the market is less diverse, the greedy method's performance increases. In an extreme case where all the weights are equal to one, the problem reduces to maximizing the number of matches. The greedy

method is known to perform well in this case. In general, Figure 2 indicates that our algorithm becomes more significant in solving the weighted matching problem, which is practically relevant.

Effect of Re-solving Timing To understand how important it is to re-solve at the right times, we compare algorithms with different re-solving patterns (linear-resolving versus log-resolving), and various re-solving times and time points under the proposed log-resolving pattern.

Figure 3 compares the log-resolving algorithm $\left(\text{RES}\left(\frac{1}{D}, K\right)\right)$ with the linear-resolving one $(\text{CONST}(K))$. From figures 3(a) and 3(b), we can see that the log-resolving algorithm consistently outperforms the linear-resolving algorithm across different market conditions indicated by k for each fixed number of re-solving times K.



(a) $K = 3$ (b) $K = 5$

Figure 3 **Effect of Re-solving Patterns ($w_{uv} \sim U(1,5)$, $r = 3$)**

Figure 4 shows the effect of different re-solve times K under $\text{RES}\left(\frac{1}{D}, K\right)$. We use MAX to denote the maximal re-solve times. It equals $14, 15, 16, 17, 18$ for $k = 1, 2, 3, 4, 5$, respectively. We find that the more times we re-solve, the better performance we can achieve, which is consistent with our theoretical analysis in Section 4. But the performance improvement is diminishing as K increases. In particular, when K exceeds 10 the improvement is negligible.

Figure 5 tests the effect of different time points under the log-resolving pattern. Specifically, it presents the performance comparison among algorithms with different γ when applying RES(γ,10) algorithm. From the figure, we can see that re-solving at our proposed time which is to set $\gamma = \frac{1}{D} = \frac{1}{3}$ outperforms the other time points. It can achieve up to 12% improvement compared with other tested resolving time points.

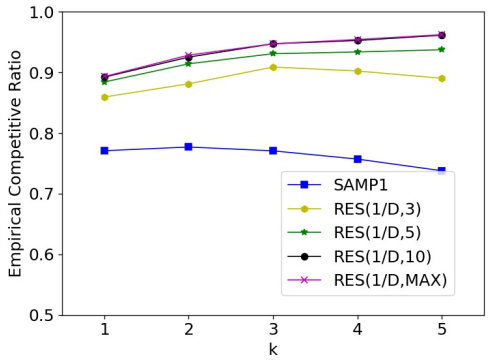
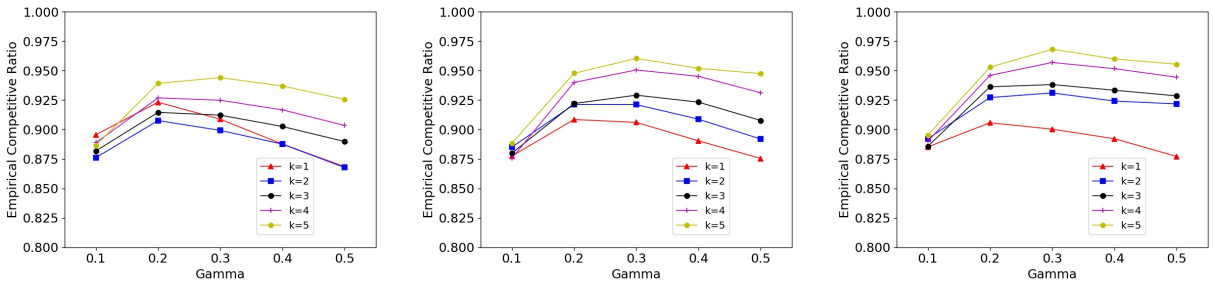


Figure 4 **Effect of Re-solving Times ($w_{uv} \sim U(1,5)$, $r = 3$)**



(a) $r = 1$ (b) $r = 2$ (c) $r = 3$

Figure 5 **Effect of Re-solving Time Points (γ) under Log-resolving Pattern ($w_{uv} \sim U(1,5)$)**

7. Conclusions

We investigate an online weighted bipartite matching problem with the following features: non-replenishable resources with a general capacity, multi-unit demand, edge-weighted rewards, and stochastic i.i.d. arrivals. We propose a randomized algorithm according to the solution to an offline linear program and analyze its competitive ratio. We introduce a re-solving heuristic to the randomized algorithm and demonstrate that re-solving at the right times can significantly improve the performance of the algorithm. We further propose a log-resolving algorithm and demonstrate the quality of this resolving method both theoretically and numerically. Finally, we investigate the value of demand distribution of the online request in further improving the algorithm's efficiency. Extensive numerical experiments based on New York city taxi data are conducted to evaluate the performance of the proposed algorithms. In these numerical tests, our algorithms demonstrate a significant advantage against other benchmarks in the literature under various market conditions, especially in a demand-intensive and weight-diverse market.

The derived algorithms have the potential to be generalized to a more general two-sided online matching setting. Both the LP-based algorithm and the re-solving technique can be adjusted to

handle the case where both sides arrive online. However, the performance analysis will require completely new methods. A detailed study of this setting is a research direction for future study. Note that our current analysis methods rely on the i.i.d. arrival assumptions and the integral assumption of capacities and demands. Hence another interesting future direction is to investigate a more general setting with regard to the arrival process, capacities, and demands.

References

Adamczyk, Marek. 2011. Improved analysis of the greedy algorithm for stochastic matching. *Information Processing Letters* **111**(15) 731–737.

Adamczyk, Marek, Fabrizio Grandoni, Joydeep Mukherjee. 2015. Improved approximation algorithms for stochastic matching. *Algorithms-ESA 2015*. Springer, 1–12.

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

Ashlagi, Itai, Yossi Azar, Moses Charikar, Ashish Chiplunkar, Ofir Geri, Haim Kaplan, Rahul Makhijani, Yuyi Wang, Roger Wattenhofer. 2017. Min-cost bipartite perfect matching with delays. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)* **81** 1–1.

Baccara, Mariagiovanna, SangMok Lee, Leeat Yariv. 2020. Optimal dynamic matching. *Theoretical Economics* **15**(3) 1221–1278.

Bahmani, Bahman, Michael Kapralov. 2010. Improved bounds for online stochastic matching. *European Symposium on Algorithms*. Springer, 170–181.

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

Brubach, Brian, Karthik Abinav Sankararaman, Aravind Srinivasan, Pan Xu. 2020. Online stochastic matching: New algorithms and bounds. *Algorithmica* 1–47.

Bumpensanti, Pornpawee, He Wang. 2020. A re-solving heuristic with uniformly bounded loss for network revenue management. *Management Science* .

Chen, Ning, Nicole Immorlica, Anna R Karlin, Mohammad Mahdian, Atri Rudra. 2009. Approximating matches made in heaven. *International Colloquium on Automata, Languages, and Programming*. Springer, 266–278.

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

Devanur, Nikhil R, Kamal Jain, Robert D Kleinberg. 2013. Randomized primal-dual analysis of ranking for online bipartite matching. *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 101–107.

Dickerson, John P, Karthik A Sankararaman, Aravind Srinivasan, Pan Xu. 2018. Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. *Thirty-Second AAAI Conference on Artificial Intelligence*.

Donovan, Brian, DB Work. 2014. New york city taxi trip data (2010-2013). *Univ. Illinois Urbana-Champaign, Champaign, IL, USA, Tech. Rep* .

Fahrbach, Matthew, Zhiyi Huang, Runzhou Tao, Morteza Zadimoghaddam. 2020. Edge-weighted online bipartite matching. *arXiv preprint arXiv:2005.01929* .

Feldman, Jon, Aranyak Mehta, Vahab Mirrokni, Shan Muthukrishnan. 2009. Online stochastic matching: Beating 1-1/e. *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 117–126.

Haeupler, Bernhard, Vahab S Mirrokni, Morteza Zadimoghaddam. 2011. Online stochastic weighted matching: Improved approximation algorithms. *International workshop on internet and network economics*. Springer, 170–181.

Hu, Ming, Yun Zhou. 2018. Dynamic type matching. *Rotman School of Management Working Paper* (2592622).

Huang, Zhiyi, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, Xue Zhu. 2018a. How to match when all vertices arrive online. *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 17–29.

Huang, Zhiyi, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang. 2018b. Online vertex-weighted bipartite matching: Beating 1-1/e with random arrivals. *arXiv preprint arXiv:1804.07458* .

Jaillet, Patrick, Xin Lu. 2014. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research* **39**(3) 624–646.

Jasin, Stefanus, Sunil Kumar. 2013. Analysis of deterministic lp-based booking limit and bid price controls for revenue management. *Operations Research* **61**(6) 1312–1320.

Jasin, Stefanus, Amitabh Sinha. 2015. An lp-based correlated rounding scheme for multi-item ecommerce order fulfillment. *Operations Research* **63**(6) 1336–1351.

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

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

Kesselheim, Thomas, Andreas Tönnis, Klaus Radke, Berthold Vöcking. 2014. Primal beats dual on online packing lps in the random-order model. *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 303–312.

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

Ma, Will, David Simchi-Levi. 2020. Algorithms for online matching, assortment, and pricing with tight weight-dependent competitive ratios. *Operations Research* **forthcoming**.

Ma, Will, David Simchi-Levi, Chung-Piaw Teo. 2020a. On policies for single-leg revenue management with limited demand information. *Operations Research* **forthcoming**.

Ma, Will, David Simchi-Levi, Jinglong Zhao. 2020b. Dynamic pricing (and assortment) under a static calendar. *Management Science* **forthcoming**.

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

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

Mehta, Aranyak, Debmalya Panigrahi. 2012. Online matching with stochastic rewards. *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE, 728–737.

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

Naori, David, Danny Raz. 2019. Online multidimensional packing problems in the random-order model. *arXiv preprint arXiv:1907.00605* .

Özkan, Erhun, Amy R Ward. 2020. Dynamic matching for real-time ride sharing. *Stochastic Systems* **10**(1) 29–70.

Reiman, Martin I, Qiong Wang. 2008. An asymptotically optimal policy for a quantity-based network revenue management problem. *Mathematics of Operations Research* **33**(2) 257–282.

Stein, Clifford, Van-Anh Truong, Xinshang Wang. 2020. Advance service reservations with heterogeneous customers. *Management Science* **66**(7) 2929–2950.

Vera, Alberto, Siddhartha Banerjee. 2021. The bayesian prophet: A low-regret framework for online decision making. *Management Science* **67**(3) 1368–1391.

Wang, Xinshang, Van-Anh Truong. 2018. Multi-priority online scheduling with cancellations. *Operations Research* **66**(1) 104–122.

Wang, Xinshang, Van-Anh Truong, David Bank. 2018. Online advance admission scheduling for services with customer preferences. *arXiv preprint arXiv:1805.10412* .

Xu, Pan, Yexuan Shi, Hao Cheng, John Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, Yongxin Tong, Leonidas Tsepenekas. 2019. A unified approach to online matching with conflict-aware constraints. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33. 2221–2228.

Zhao, Yanyang, Xinshang Wang, Linwei Xin. 2020. Multi-item online order fulfillment: A competitive analysis. *Working Paper* .

Appendix. Proofs and Omitted Results

A. Examples

A.1. Re-solving does not always help.

In the following we show via a counterexample that the re-solving heuristic may make things worse sometimes.

- 1 offline resource u: capacity $c = 2$
- 2 online requests: v_0 and v_1
- v_0: arrival probability $p_0 = \frac{1}{2}$, demand $d_0 = 1$, weight with u: $w_0 = w > 1$
- v_1: arrival probability $p_1 = \frac{1}{2}$, demand $d_1 = 1$, weight with u: $w_1 = 1$
- Total numbers of online arrivals: $T = 4$

In this example we have $D = 1$, and according to Proposition 2, we set $\alpha = 1$. By solving the LP (3), we know that algorithm SAMP(α) will always accept v_0 as long as the capacity permits and reject v_1. To understand the performance of algorithm SAMP(α) , let s denote the realized sequence. If $s = (v_1, v_1, v_1, v_1)$, then the offline optimal reward is 2 while SAMP(α) gives 0. If there is exactly one v_0 in s, the offline optimal reward is $1 + w$ while SAMP(α) gives w. If there are more than one v_0 in s, both offline optimal reward and the expected reward by SAMP(α) 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 rewards 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:

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

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 accepting v_0 with probability 1 and accept v_1 with probability $\frac{1}{3}$ from the second arrival on. Hence, the expected reward 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 reward of this re-solving method as follows

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

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

B. Proofs

B.1. Proofs of Section 3

PROPOSITION 1 $\mathbb{E}[\text{OFF}] \le \text{OPT}$, *i.e.,* $\mathbb{E}[\text{OFF}]$ *is upper bounded by the optimal value of LP (3).*

Proof Recall that s is the realized arrival sequence and $X^*_{uv}(s)$ is the optimal solution of MILP (1). Let $h_{uv} = \sum_s \mathbb{P}(s) X^*_{uv}(s)$ be the expected offline optimal solution. It is easy to show that $y_{uv} = \frac{h_{uv}}{T p_v}$ is a feasible solution of (3) and moreover, the objective of this solution is exactly the expected reward of offline optimal $\mathbb{E}[\text{OFF}]$. Then we have $\mathbb{E}[\text{OFF}] \le \text{OPT}$. □

B.2. Proofs of Section 4

LEMMA 1 *The expected times that edge (u,v) is matched by* SAMP(α) *during the first t' arrivals $N_\alpha(u,v,t')$ has the following lower bound:*

$$N_\alpha(u,v,t') \geq \left[-\left(\frac{t'}{T}\right)^2 \frac{\alpha^2 D}{2} + \frac{t'}{T}\alpha\right] \cdot T p_v y^*_{uv} = g\left(\alpha \cdot \frac{t'}{T}\right) \cdot T p_v y^*_{uv}.$$

Proof Let $\mathcal{C}_{ut}^{\text{SAMP}(\alpha)}$ and $\mathcal{Z}_{ut}^{\text{SAMP}(\alpha)}$ denote the remaining capacity and consumed capacity of offer u after running SAMP(α) for t rounds ($0 \leq t \leq T$). For notation convenience, we will use $\mathcal{C}_{ut}$ and $\mathcal{Z}_{ut}$ when there is no ambiguity. We always have $\mathcal{C}_{ut} + \mathcal{Z}_{ut} = c_u$. If we ignore the capacity constraint of offer u and still run SAMP(α) , we denote the consumed capacity of u is $\bar{\mathcal{Z}_{ut}}$. We have $\mathcal{Z}_{ut} = \min\{c_u, \bar{\mathcal{Z}_{ut}}\}$. From the algorithm, we have

$$\begin{aligned}\mathbb{E}[\bar{\mathcal{Z}_{ut}}] &= \sum_{i=1}^{t} \sum_{v' \in V} p_{v'} \alpha y^*_{uv'} d_{v'} \\ &\leq \sum_{t'=1}^{t} \frac{\alpha c_u}{T} = \frac{\alpha c_u t}{T},\end{aligned} \tag{16}$$

where the inequality is according to the capacity constraint for offline resources in LP (3). By Markov's inequality, we can derive a lower bound of the probability that $\mathcal{C}_{ut} \geq d_v$, i.e., the remaining capacity of an offer u by time t is sufficient for a request v.

$$\mathbb{P}[\mathcal{C}_{ut} \geq d_v] = \mathbb{P}[\mathcal{Z}_{ut} \leq c_u - d_v] \geq \mathbb{P}[\bar{\mathcal{Z}_{ut}} \leq c_u - d_v] = 1 - \mathbb{P}[\bar{\mathcal{Z}_{ut}} \geq c_u - d_v + 1] \geq 1 - \frac{\alpha c_u t}{T(c_u - d_v + 1)}. \tag{17}$$

The second equality is based on the relation between $\bar{\mathcal{Z}_{ut}}$ and $\mathcal{Z}_{ut}$ under the assumption of positive demand. The third equality comes from the integral assumption of each demand. The last inequality is from Markov inequality. Hence, the probability that v is matched to u at time step $t+1$ can be calculated as

$$\mathbb{P}_\alpha(u,v,t+1) = \alpha p_v y^*_{uv} \mathbb{P}[\mathcal{C}_{ut} \geq d_v] \geq \alpha p_v y^*_{uv} \left[1 - \frac{\alpha c_u t}{T(c_u - d_v + 1)}\right].$$

This inequality helps to bound the expected number of times that edge (u,v) is matched by the algorithm during the first t' steps, denoted by $N_\alpha(u,v,t')$.

$$\begin{aligned}N_\alpha(u,v,t') &\geq \sum_{t=1}^{t'} \mathbb{P}_\alpha(u,v,t) \\ &\geq \sum_{t=1}^{t'} \alpha p_v y^*_{uv} \left[1 - (t-1)\frac{\alpha c_u}{T(c_u - d_v + 1)}\right] \\ &\geq \left[-\left(\frac{t'}{T}\right)^2 \frac{\alpha^2 c_u}{2(c_u - d_v + 1)} + \frac{t'}{T}\alpha\right] T p_v y^*_{uv} \qquad (18) \\ &\geq \left[-\left(\frac{t'}{T}\right)^2 \frac{\alpha^2 D}{2} + \frac{t'}{T}\alpha\right] T p_v y^*_{uv} \qquad (19) \\ &= g\left(\alpha \cdot \frac{t'}{T}\right) \cdot T p_v y^*_{uv}. \qquad (20)\end{aligned}$$

The inequality (18) is according to a re-organization of the summation term and the fact that $\frac{-t'^2+t'}{2} \geq \frac{-t'^2}{2}$. The inequality (19) holds as $1 \leq c_u \leq C$ and $1 \leq d_v \leq D$ and $c_u \geq d_v$, which implies that for every u and v,

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

The equality (20) is according to the definition of function $g(z)$ (equality (4)).

PROPOSITION 2 *The* SAMP(α) *algorithm has a competitive ratio* $\mathrm{CR} \geq \frac{1}{2D}$.

Proof It is easy to show that $g(z)$ achieves the maximal value when $z = \frac{1}{D}$ and the maximal value is $\frac{1}{2D}$. With Lemma 1, we can further give the lower bound of the expected performance of SAMP(α) as follows.

$$\begin{aligned}
\mathbb{E}[\text{SAMP}(\alpha)] &= \sum_{(u,v)\in E} [w_{uv} N_\alpha(u,v,T)] \\
&\geq \sum_{(u,v)\in E} [w_{uv} N_\alpha(u,v,t')] \\
&\geq g\left(\alpha \cdot \frac{t'}{T}\right) \cdot \sum_{(u,v)\in E} T p_v w_{uv} y^*_{uv} \\
&= g\left(\alpha \cdot \frac{t'}{T}\right) \cdot \mathrm{OPT}.
\end{aligned}$$

Then with Proposition 1, we have the lower bound of the competitive ratio of SAMP(α) :

$$\mathrm{CR} = \frac{\mathbb{E}[\text{SAMP}(\alpha)]}{\mathrm{OFF}} \geq \frac{\mathbb{E}[\text{SAMP}(\alpha)]}{\mathrm{OPT}} = g\left(\alpha \cdot \frac{t'}{T}\right).$$

Note that the lower bound $g\left(\alpha \cdot \frac{t'}{T}\right)$ achieves the maximal value $\frac{1}{2D}$ when $\alpha \cdot \frac{t'}{T} = \frac{1}{D}$, and the maximal lower bound is achievable since we can take any integer value of t' between $\lceil \frac{T}{D} \rceil$ and T such that $0 \leq \alpha \leq 1$. Then we have

$$\mathrm{CR} \geq \frac{1}{2D}.$$

LEMMA 2 *By choosing* $\alpha_1 = \frac{T'/D}{\lceil T'/D \rceil}$, *the expected number of matches of edge* (u,v) *after re-solving, i.e.,* $N'_{\alpha_1,t'}(u,v,T')$ *is bounded from below by* $\frac{1}{2D} \cdot T' p_v y'^*_{uv}$.

Proof According to Algorithm 2, the matching probability is $\alpha_1 \boldsymbol{y}'^*$ after the re-solving time point t' but not including t'.

For (u,v) that $c'_u < d_v$, according to the constraint in (5c), we have $y'^*_{uv} = 0$, then $N'_{\alpha_1,t'}(u,v,T') \geq \frac{1}{2D} \cdot T' p_v y'^*_{uv} = 0$ always holds.

For (u,v) that $c'_u \geq d_v$, following a similar analysis of Lemma 1, the expected number of matches between a resource-request pair (u,v) within T' periods from the re-solving time point t' not including t', denoted by $N'_{\alpha_1,t'}(u,v,T')$ can be bounded as follows.

$$\begin{aligned}
N'_{\alpha_1,t'}(u,v,T') \geq N'_{\alpha_1,t'}(u,v,t'') &\geq \left[-(\frac{t''}{T'})^2 \frac{\alpha_1^2 c'_u}{2(c'_u - d_v + 1)} + \frac{t''}{T'}\alpha_1\right] T' p_v y'^*_{uv} \\
&\geq \left[-\left(\frac{t''}{T'}\right)^2 \frac{\alpha_1^2 D}{2} + \frac{t''}{T'}\alpha_1\right] T' p_v y'^*_{uv} \qquad (21) \\
&= g\left(\alpha_1 \cdot \frac{t''}{T'}\right) \cdot T' p_v y'^*_{uv},
\end{aligned}$$

where we introduce the subscript t'' to denote the number of periods counted from the time point t', which should be no more than the remaining time periods T'. The first inequality holds since the expected number of matches made in T' periods is no less than that in t'' periods from t'. The last inequality holds since $c'_u \geq d_v$ and by simple algebra, we have $\frac{c'_u}{c'_u - d_v + 1} = 1 + \frac{d_v - 1}{c'_u - d_v + 1} \leq d_v \leq D$.

Then according to the property of $g(z)$, we can set $\alpha_1 = \frac{T'/D}{\lceil T'/D \rceil} \leq 1$ and $t'' = \lceil \frac{T'}{D} \rceil$, which implies $g\left(\alpha_1 \cdot \frac{t''}{T'}\right) = \frac{1}{2D}$. Then we have $N'_{\alpha_1,t'}(u,v,T') \geq \frac{1}{2D} \cdot T' p_v y'^*_{uv}$. □

PROPOSITION 3 $\mathbb{E}\left[\frac{\text{OPT}'}{\text{OPT}}\right] \geq \left(1-\frac{t'}{T}\right) e^{-R\frac{\alpha_0 t'}{T}}$, *where* $R = \max_{(u',v')\in E} \frac{c_{u'}}{d_{v'}}$.

Proof We first get a lower bound of OPT′. To do this, we construct a feasible solution $\boldsymbol{y}'$ for the resolved LP (5) based on the optimal solution $\boldsymbol{y}^*$ to the original LP (3). Denote U_+ as the set of offline vertices whose capacity does not change, i.e., $c'_u = c_u$ and U_- as the rest offline vertices. Consider such a solution

$$y'_{uv} = \begin{cases} y^*_{uv} & u \in U_+; \\ 0 & u \in U_-. \end{cases} \tag{22}$$

We can easily check that the constructed $\boldsymbol{y}'$ is feasible for the re-solved LP (5). Therefore,

$$\text{OPT}' \geq \sum_{u\in U_+} \sum_{v\in Adj(u)} T' w_{uv} p_v y'_{uv} + \sum_{u\in U_-} \sum_{v\in Adj(u)} T' w_{uv} p_v y'_{uv}.$$

Denote $w(S) = \sum_{u\in S} \sum_{v\in Adj(u)} p_v w_{uv} y^*_{uv}$ and $w'(S) = \sum_{u\in S} \sum_{v\in Adj(u)} p_v w_{uv} y'_{uv}$, where $Adj(u)$ is the set of adjacent vertices of u. We have

$$\frac{\text{OPT}'}{\text{OPT}} \geq \frac{T'(w'(U_+)+w'(U_-))}{T(w(U_+)+w(U_-))} = \left(1-\frac{t'}{T}\right) \frac{w'(U_+)}{w(U_+)+w(U_-)}, \tag{23}$$

where the last equality holds because we have $y'_{uv} = 0$ for $u \in U_-$.

From the analysis above, we have

$$\mathbb{E}\left[\frac{\text{OPT}'}{\text{OPT}}\right] \geq \left(1-\frac{t'}{T}\right) \frac{\mathbb{E}[w'(U_+)]}{w(U_+)+w(U_-)} = \left(1-\frac{t'}{T}\right) \frac{\sum_u \mathbb{P}(\mathcal{C}_{ut'} = c_u) w_u}{w(U_+)+w(U_-)}, \tag{24}$$

where $w_u = \sum_{v\in Adj(u)} p_v w_{uv} y^*_{uv}$ and the equality holds since $\mathbb{E}[w'(U_+)] = \mathbb{E}\left[\sum_u w_u \mathbb{I}\{\mathcal{C}_{ut'} = c_u\}\right] = \sum_u \mathbb{P}(\mathcal{C}_{ut'} = c_u) w_u$. Here $\mathbb{I}\{A\}$ is an indicator function that equals one if A is true and zero otherwise. Note that

$$\begin{aligned}
\mathbb{P}(\mathcal{C}_{ut'} = c_u) &= \mathbb{P}(u \text{ is not matched by } t') \\
&= \mathbb{P}(u \text{ is not matched at time } = 1)^{t'} && (25) \\
&= \left(1 - \sum_{v:(u,v)\in E} \alpha_0 p_v y^*_{uv}\right)^{t'} && (26) \\
&\geq \left(1 - \frac{\alpha_0 R}{T}\right)^{t'} && (27) \\
&\approx e^{-R\frac{\alpha_0 t'}{T}}, && (28)
\end{aligned}$$

where (25) comes from the i.i.d. assumption and (26) is from the matching algorithm before re-solving. According to constraint (3a) of LP (3), inequality (27) holds since

$$\sum_{v,(u,v)\in E} p_v y^*_{uv} = \sum_{v,(u,v)\in E} \frac{c_u}{d_v} \cdot \frac{p_v d_v y^*_{uv}}{c_u} \leq \max_{(u',v')\in E} \left(\frac{c_{u'}}{d_{v'}}\right) \cdot \sum_{v,(u,v)\in E} \frac{p_v d_v y^*_{uv}}{c_u} = R \cdot \sum_{v,(u,v)\in E} \frac{p_v d_v y^*_{uv}}{c_u} \leq \frac{R}{T}.$$

By Taylor expansion of $e^{-\frac{\alpha_0 R}{T}}$ and the condition that $R = \max_{(u',v')\in E} \frac{c_{u'}}{d_{v'}} \ll T$, $e^{-\frac{\alpha_0 R}{T}}$ is a good approximation of $(1-\frac{\alpha_0 R}{T})$. Hence, $\mathbb{P}(\mathcal{C}_{ut'} = c_u) \geq e^{-R\frac{\alpha_0 t'}{T}}$, which further implies that

$$\mathbb{E}\left[\frac{\text{OPT}'}{\text{OPT}}\right] \geq \left(1-\frac{t'}{T}\right) e^{-R\frac{\alpha_0 t'}{T}}$$

according to (24). □

PROPOSITION 4 *Re-solving once at time step* $t' = \lceil \frac{T}{D} \rceil$ *with* $\alpha_0 = \frac{T/D}{\lceil T/D \rceil}$ *and* $\alpha_1 = \frac{T'/D}{\lceil T'/D \rceil}$ *in Algorithm 2 achieves a competitive ratio of*

$$\mathrm{CR}' \geq \frac{1}{2D} + \frac{1}{2D}\left(1 - \frac{\lceil T/D \rceil}{T}\right) e^{-\frac{R}{D}}. \tag{29}$$

Proof According to (7) and Proposition 3, the lower bound of competitive ratio is written as below:

$$\mathrm{CR}' \geq g\left(\frac{\alpha_0 t'}{T}\right) + \frac{1}{2D}\left(1 - \frac{t'}{T}\right) e^{-R\frac{\alpha_0 t'}{T}}, \tag{30}$$

which is derived based on $\alpha_1 = \frac{T'}{D\lceil \frac{T'}{D} \rceil}$ according to Lemma 2. Note that when $\frac{\alpha_0 t'}{T} = \frac{1}{D}$, the first term $g(\frac{\alpha_0 t'}{T}) = \frac{1}{2D}$, which is exactly the lower bound we have derived for SAMP(α) . In other words, the second term indicates the improvement we make from the re-solving heuristic. To find appropriate values of t' and α_0, we first reorganize the second term as $\frac{1}{2D}(1 - \frac{z}{\alpha_0})e^{-Rz}$ where $z = \frac{\alpha_0 t'}{T}$. Under the condition that $\frac{\alpha_0 t'}{T} = \frac{1}{D}$, i.e., $z = \frac{1}{D}$, we want to take the largest possible value of α_0 to render a large lower bound. Bear in mind that $0 \leq \alpha_0 \leq 1$ and t' is an integer number less than T, we can take $\alpha_0 = \frac{T}{D\lceil \frac{T}{D} \rceil} \leq 1$ and $t' = \lceil \frac{T}{D} \rceil$. By doing so, the lower bound is derived as shown in the statement. □

PROPOSITION 5 *Conditioned on* $\mathcal{C}_u^{(i)} = c_u^{(i)}$ *for every* u, *when* $\frac{R}{T_i} << 1$, *we have*

$$\mathbb{E}\left[\frac{\mathrm{OPT_{i+1}}}{\mathrm{OPT_i}}\middle|\boldsymbol{c}^{(i)}\right] \geq \left(1 - \frac{\Delta_i}{T_i}\right) e^{-\alpha_i \frac{\Delta_i}{T_i} R} \tag{31}$$

Proof Recall the notations first. Let $\boldsymbol{y}^{*(i)}$ and OPT_i denote the optimal solution and optimal value of LP after the ith re-solve. Let $\mathcal{C}_u^{(i)}$ denote the remaining capacity of resource u right before ith re-solve. Note that $\mathcal{C}_u^{(i)}$ is a random variable, this proposition studies the bound of the expected ratio between the optimal values of two consecutive linear programs OPT_i and OPT_{i+1} conditioned on a realization of $\mathcal{C}_u^{(i)}$, denoted by $c_u^{(i)}$. As a generalized version of Proposition 3, we notice that

$$\mathbb{E}\left[\frac{\mathrm{OPT}_{i+1}}{\mathrm{OPT}_i}\middle|\boldsymbol{c}^{(i)}\right] \geq \left(1 - \frac{\Delta_i}{T_i}\right) \frac{\sum_u \mathbb{P}(\mathcal{C}_u^{(i+1)} = \mathcal{C}_u^{(i)} = c_u^{(i)})\mathbb{E}[w_u^{(i)}|\mathcal{C}_u^{(i+1)} = \mathcal{C}_u^{(i)} = c_u^{(i)}]}{\sum_u \mathbb{E}[w_u^{(i)}|\mathcal{C}_u^{(i)} = c_u^{(i)}]},$$

where $w_u^{(i)} = \sum_{v \in Adj(u)} p_v w_{uv} y_{uv}^{*(i)}$. Here $\mathcal{C}_u^{(i+1)} = \mathcal{C}_u^{(i)} = c_u^{(i)}$ means that no capacity is consumed between the ith arrival and $i+1$st arrival. The lower bound is obtained by constructing a feasible solution $\boldsymbol{y}'^{(i+1)}$ to the LP after the $i+1$st re-solve in the following way: $y_{uv}'^{(i+1)} = y_{uv}^{*(i)}$ if $\mathcal{C}_u^{(i+1)} = c_u^{(i)}$ and zero otherwise, conditioned on $\mathcal{C}_u^{(i)} = c_u^{(i)}$ for every u. Note that conditioned on $\mathcal{C}_u^{(i)} = c_u^{(i)}$ for every u, $\mathbb{E}[w_u^{(i)}|\mathcal{C}_u^{(i+1)} = c_u^{(i)}] = \mathbb{E}[w_u^{(i)}|\mathcal{C}_u^{(i+1)} = \mathcal{C}_u^{(i)} = c_u^{(i)}]$. Hence, the key part to prove the proposition is to establish a lower bound of $\mathbb{P}(\mathcal{C}_u^{(i+1)} = c_u^{(i)})$ conditioned on $\mathcal{C}_u^{(i)} = c_u^{(i)}$.

$$\begin{aligned}
\mathbb{P}(\mathcal{C}_u^{(i+1)} = c_u^{(i)}) &= \mathbb{P}(u \text{ is not matched during } [t_i, t_{i+1})) \\
&= \mathbb{P}(u \text{ is not matched at time } = t_i + 1)^{\Delta_i} \\
&= \left(1 - \sum_{v:(u,v)\in E} \alpha_i p_v y_{uv}^{*(i)}\right)^{\Delta_i} \\
&\geq \left(1 - \frac{\alpha_i R}{T_i}\right)^{\Delta_i} \approx e^{-\alpha_i \frac{\Delta_i}{T_i} R}
\end{aligned}$$

The analysis follows the same logic as that in Proposition 3. The key part is the following inequality:

$$\sum_{v,(u,v)\in E} p_v y_{uv}^{*(i)} = \sum_{v,(u,v)\in E} \frac{c_u^{(i)}}{d_v} \cdot \frac{p_v d_v y_{uv}^{*(i)}}{c_u^{(i)}} \le \max_{(u',v')\in E} \left(\frac{c_{u'}}{d_{v'}}\right) \cdot \sum_{v,(u,v)\in E} \frac{p_v d_v y_{uv}^{*(i)}}{c_u^{(i)}} = R \cdot \sum_{v,(u,v)\in E} \frac{p_v d_v y_{uv}^{*(i)}}{c_u^{(i)}} \le \frac{R}{T_i}.$$

□

THEOREM 1 *Assuming $T_K \gg \max_u c_u$, the log-resolving algorithm (Algorithm 3) with $\gamma = \frac{1}{D}$ and $\alpha_i = \frac{T_i/D}{\lceil T_i/D \rceil}$ has a competitive ratio of at least $\frac{1}{2D}\frac{1-\theta^{K+1}}{1-\theta}$, where $\theta = \left(1-\frac{1}{D}\right)e^{-\frac{R}{D}}$.*

Proof We first list some notations used in the proof:

- $T_i = T - t_i$ $(0 \le i \le K)$: the remaining time after ith re-solve.
- $\Delta_i = \lceil \gamma T_i \rceil = t_{i+1} - t_i$ $(0 \le i \le K-1)$: the number of periods between the ith and $i+1$th re-solves.
- $\gamma_i = \frac{\Delta_i}{T_i}$ $(0 \le i \le K-1)$: the ratio between Δ_i and T_i.
- OPT_i: the optimal value of the ith re-solved LP.

Next we recall the time for the ith re-solve is t_i, which is defined as

$$t_i = \begin{cases} 0, & i = 0 \\ t_{i-1} + \lceil \gamma(T - t_{i-1}) \rceil, & i > 0. \end{cases} \tag{32}$$

$\mathrm{ALG}(s,T)$ denotes the expected reward generated by arrivals in the interval $(s,T]$ 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. Assume that $\textsc{Samp}(\alpha)$ is applied for $(0,t_1]$ with $\alpha = \alpha_0$ and $\textsc{Samp}(\alpha_K)$ is used for $(t_K,T]$. For $1 \le i \le K-1$, $\textsc{Samp}(\alpha_i)$ is used for $(t_i,t_{i+1}]$ arrivals. Here $0 \le \alpha_i \le 1$ for $0 \le i \le K$. We can derive a lower bound of 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 (19); from t_1 onward the algorithm is based on the designed randomization algorithm. Therefore,

$$\mathbb{E}[\mathrm{ALG}(0,T)] \ge \left(-\frac{\gamma_0^2\alpha_0^2 D}{2} + \gamma_0\alpha_0\right)\mathrm{OPT} + \mathbb{E}[\mathrm{ALG}(t_1,T)]$$

Hence we have

$$\begin{aligned} \frac{\mathbb{E}[\mathrm{ALG}(0,T)]}{\mathrm{OPT}} &\ge -\frac{\gamma_0^2\alpha_0^2 D}{2} + \gamma_0\alpha_0 + \frac{\mathbb{E}[\mathrm{ALG}(t_1,T)]}{\mathrm{OPT}} \\ &= -\frac{\gamma_0^2\alpha_0^2 D}{2} + \gamma_0\alpha_0 + \mathbb{E}\left[\frac{\mathrm{ALG}(t_1,T)}{\mathrm{OPT}_1}\cdot\frac{\mathrm{OPT}_1}{\mathrm{OPT}}\right] \end{aligned}$$

Note 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 $\mathbb{E}\left[\frac{\mathrm{ALG}(t_1,T)}{\mathrm{OPT}_1}\frac{\mathrm{OPT}_1}{\mathrm{OPT}}\right] = \mathbb{E}\left[\frac{\mathrm{ALG}(t_1,T)}{\mathrm{OPT}_1}\middle|\boldsymbol{c}^{(1)}\right]\cdot\mathbb{E}\left[\frac{\mathrm{OPT}_1}{\mathrm{OPT}}\right]$. From Proposition 5, we have $\mathbb{E}\left[\frac{\mathrm{OPT}_1}{\mathrm{OPT}}\right] \ge (1-\gamma_0)e^{-\alpha_0\gamma_0 R}$. Hence we have

$$\frac{\mathbb{E}[\mathrm{ALG}(0,T)]}{\mathrm{OPT}} \ge -\frac{\gamma_0^2\alpha_0^2 D}{2} + \gamma_0\alpha_0 + \mathbb{E}\left[\frac{\mathrm{ALG}(t_1,T)}{\mathrm{OPT}_1}\middle|\boldsymbol{c}^{(1)}\right](1-\gamma_0)e^{-\alpha_0\gamma_0 R}$$

Conditioning on $\boldsymbol{c}^{(1)}$, OPT_1 is deterministic. We can follow a similar analysis to get the following inequality

$$\begin{aligned} &\mathbb{E}\left[\frac{\mathrm{ALG}(t_1,T)}{\mathrm{OPT}_1}\middle|\boldsymbol{c}^{(1)}\right] \\ \ge&\mathbb{E}\left[-\frac{\gamma_1^2\alpha_1^2 D}{2} + \gamma_1\alpha_1\middle|\boldsymbol{c}^{(1)}\right] + \mathbb{E}\left[\frac{\mathrm{ALG}(t_2,T)}{\mathrm{OPT}_2}\frac{\mathrm{OPT}_2}{\mathrm{OPT}_1}\middle|\boldsymbol{c}^{(1)}\right] \\ \ge&-\frac{\gamma_1^2\alpha_1^2 D}{2} + \gamma_1\alpha_1 + \mathbb{E}\left[\frac{\mathrm{ALG}(t_2,T)}{\mathrm{OPT}_2}\middle|\boldsymbol{c}^{(1)},\boldsymbol{c}^{(2)}\right](1-\gamma_1)e^{-\alpha_1\gamma_1 R} \end{aligned} \tag{33}$$

With the same analysis we have for each $i < K$,

$$\mathbb{E}\left[\frac{\mathrm{ALG}(t_i,T)}{\mathrm{OPT}_i}\Big|\boldsymbol{c}^{(1)},\ldots,\boldsymbol{c}^{(i)}\right] \geq -\frac{\gamma_i^2\alpha_i^2 D}{2} + \gamma_i\alpha_i + \\ \mathbb{E}\left[\frac{\mathrm{ALG}(t_{i+1},T)}{\mathrm{OPT}_{i+1}}\Big|\boldsymbol{c}^{(1)},\ldots,\boldsymbol{c}^{(i+1)}\right](1-\gamma)e^{-\alpha_i\gamma R}$$

In the last period ($i = K$), we have

$$\mathbb{E}\left[\frac{\mathrm{ALG}(t_K,T)}{\mathrm{OPT}_K}\Big|\boldsymbol{c}^{(1)},\ldots,\boldsymbol{c}^{(K)}\right] \geq \mathbb{E}\left[\frac{\mathrm{ALG}(t_K,t'')}{\mathrm{OPT}_K}\Big|\boldsymbol{c}^{(1)},\ldots,\boldsymbol{c}^{(K)}\right] \geq -\left(\frac{t''-t_K}{T-t_K}\right)^2\frac{\alpha_K^2 D}{2} + \frac{t''-t_K}{T-t_K}\alpha_K,$$

where t'' denotes an arbitrary time point between t_K and T, i.e., $t_K \leq t'' \leq T$, and the last inequality comes from (19). Set α_K to any value between $\frac{1}{D}$ and 1 such that $\frac{t''-t_K}{T-t_K}\alpha_K = \frac{1}{D}$, we have

$$\mathbb{E}\left[\frac{\mathrm{ALG}(t_K,T)}{\mathrm{OPT}_K}\Big|\boldsymbol{c}^{(1)},\ldots,\boldsymbol{c}^{(K)}\right] \geq \frac{1}{2D}.$$

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

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

and $l_K \geq \frac{1}{2D}$. Set $\alpha_i = \frac{1}{\gamma_i D} = \frac{T_i}{D\lceil\gamma T_i\rceil}, \forall i = 0,...,K-1$ and $\gamma = \frac{1}{D}$. We have $l_i \geq \frac{1}{2D} + l_{i+1}(1-\gamma_i)e^{-\frac{R}{D}}, \forall i = 0,\ldots,K-1$. Note that $\gamma_i = \frac{\lceil\frac{T_i}{D}\rceil}{T_i} = \frac{1}{D} + \frac{\epsilon_i}{T_i}$ where $\epsilon_i = \lceil\frac{T_i}{D}\rceil - \frac{T_i}{D} \in [0,1)$. Then rewrite the inequalities, we get

$$l_i \geq \frac{1}{2D} + l_{i+1}(1-\frac{1}{D}-\frac{\epsilon_i}{T_i})e^{-\frac{R}{D}}, \quad \forall l = 0,...,K-1$$

and $l_K \geq \frac{1}{2D}$. Aggregate all the inequalities, we have

$$l_0 \geq \sum_{i=0}^{K}\frac{1}{2D}\left((1-\frac{1}{D})e^{-\frac{R}{D}}\right)^i + o(\tfrac{\epsilon_i}{T_i})$$

Under the assumption that $T_i \geq T_K \gg \max_u c_u \geq D > \epsilon_i$ for any $i = 0,\ldots,K$, $\frac{\epsilon_i}{T_i}$ is small enough. Hence, we can ignore the term $o(\frac{\epsilon_i}{T_i})$ and obtain the following bound:

$$l_0 \geq \sum_{i=0}^{K}\frac{1}{2D}\left((1-\frac{1}{D})e^{-\frac{R}{D}}\right)^i = \frac{1}{2D}\frac{1-\left((1-\frac{1}{D})e^{-\frac{R}{D}}\right)^{K+1}}{1-(1-\frac{1}{D})e^{-\frac{R}{D}}} = \frac{1}{2D}\frac{1-\theta^{K+1}}{1-\theta},$$

where $\theta = \left(1-\frac{1}{D}\right)e^{-\frac{R}{D}}$. □

LEMMA 3 *The offline optimal value* OFF *of Instance 1 is* $2^n p + o(2^n)$.

Proof Assume that in a realized sequence with 2^n requests, there are k requests of type v_1 and $2^n - k$ requests of type v_2. We know the value of k follows a binomial distribution.

1. When $k = 0$, that is, the arrival sequence only has 2^n requests of type v_2, the optimal matching is to match them to all of the 2^n offline resource. The offline optimal value is 2^n. This event happens with probability p^{2^n}.
2. When $k = 1$, the optimal matching is again to match these requests to all offline resource. The offline optimal value is $\frac{1}{2} - \epsilon + (2^n - 1)$. This even happens with probability $2^n p^{2^n-1} q$.
3. When $k \geq 2$, then at most two requests of type v_1 can be matched (to u_1) and they can together generate a value of at most $1 - 2\epsilon$. The $2^n - k$ requests of type v_2 can all be matched and generate a total value of $2^n - k$. Therefore the offline optimal value is $1 - 2\epsilon + 2^n - k$. The probability of a sequence with k requests of type v_1 is $C(2^n, k)q^k p^{2^n-k}$.

Combining all cases together, we obtain the expected offline optimal:

$$\begin{aligned}\text{OFF} &= p^{2^n}2^n + 2^n p^{2^n-1}q\left(\frac{1}{2}-\epsilon+(2^n-1)\right) + \sum_{k=2}^{2^n} C(2^n,k)q^k p^{2^n-k}(1-2\epsilon+2^n-k)\\ &= 2^n p + 2^n q p^{2^n-1}(\frac{1}{2}-\epsilon) + (1-2^n q p^{2^n-1}-p^{2^n})(1-2\epsilon)\\ &= 2^n p + o(2^n).\end{aligned}$$

The second equality is derived based on $1=(p+q)^{2^n}=\sum_{k=0}^{2^n} C(2^n,k)q^k p^{2^n-k}$. To see the last equality, note that

$$\begin{aligned}&2^n q p^{2^n-1}\frac{1}{2} + (1-2^n q p^{2^n-1}-p^{2^n})\\ =&1-\frac{2^n}{2}qp^{2^n-1}-p^{2^n}\\ =&1-\frac{1}{2}\left(1-\frac{1}{2^n}\right)^{2^n-1}-\left(1-\frac{1}{2^n}\right)^{2^n}\\ =&1-\left(\frac{2^{n-1}}{2^n-1}+1\right)\left(1-\frac{1}{2^n}\right)^{2^n}\\ \approx&1-\frac{3}{2}e^{-1}=o(2^n),\end{aligned}$$

where the second equality is from the definition of q which is equal to $\frac{1}{2^n}$. Hence we have

$$\begin{aligned}&2^n q p^{2^n-1}\left(\frac{1}{2}-\epsilon\right) + (1-2^n q p^{2^n-1}-p^{2^n})(1-2\epsilon)\\ =&2^n q p^{2^n-1}\frac{1}{2} + (1-2^n q p^{2^n-1}-p^{2^n}) - 2\epsilon\left(2^n q p^{2^n-1}\frac{1}{2}+1-2^n q p^{2^n-1}-p^{2^n}\right)\\ =&o(2^n)-2\epsilon o(2^n)=o(2^n)\end{aligned}$$

□

THEOREM 2(**Upper Bound**) *Any LP-based randomized algorithm* SAMP(α) *can not achieve a competitive ratio of* $1-\frac{1}{e}+c\approx 0.632+c$ *for any constant* $c>0$.

Proof We consider the Instance 1. Here we assume that $n\gg 1$. The optimal solution to LP (3) is $y^*_{u2}=\frac{1}{Tp},\forall u\in\{2,\dots,2^n\}$, $y^*_{11}=1$ and $y^*_{12}=1-\frac{T-1}{Tp}=0$. According to the LP-based randomized algorithm, a match is made successfully with a probability of $\alpha\mathbf{y}^*$ for any α between zero and one.

First, we consider the expected reward generated by $u\in\{2,\dots,2^n\}$. Let Z_u be an indicator variable indicating if a resource u is matched during the algorithm process. We have $\mathbb{E}[Z_u]=\mathbb{P}(\text{resource } u \text{ is matched during } T \text{ periods})=1-(1-\alpha p y^*_{u2})^T$. Then we have:

$$\mathbb{E}\left[\sum_{u=2}^{2^n} Z_u\right]=\sum_{u=2}^{2^n}[1-(1-\alpha p y^*_{u2})^T]\approx(2^n-1)(1-e^{-\alpha}).$$

Let $\mathbb{E}[u_1]$ denote the expected reward of $u=1$. Then we obtain the upper bound of $\mathbb{E}[u_1]$ by conditioning on the number of requests of type v_1.

$$\begin{aligned}\mathbb{E}[u_1] &\le C(T,1)qp^{T-1}(\frac{1}{2}-\epsilon)+C(T,2)q^2p^{T-2}(1-2\epsilon)+[1-(1-\alpha p y^*_{12})^T]\\ &= p^{T-2}q\left(\frac{1}{2}-\epsilon\right)(Tp+T-1)=\left(1-\frac{1}{2^n}\right)^{2^n-2}(\frac{1}{2}-\epsilon)\left(p+1-\frac{1}{2^n}\right)\\ &\approx\frac{1}{e}\left(\frac{1}{2}-\epsilon\right)(p+1)=o(2^n).\end{aligned}$$

This allows us to show an upper bound of any LP-based randomization algorithm (SAMP(α)) (combining the analysis above and Lemma 3):

$$\mathrm{CR}(\textsc{Samp}(\alpha)) = \frac{\mathbb{E}\left[\sum_{u=2}^{2^n} Z_u\right] + \mathbb{E}[u_1]}{\mathrm{OFF}} \le \frac{(2^n-1)(1-e^{-\alpha}) + o(2^n)}{2^n p + o(2^n)} \approx \frac{1-e^{-\alpha}}{p} \le \frac{1-e^{-1}}{1-\frac{1}{2^n}}.$$

This value is less than $1-\frac{1}{e}+c$ for any constant c when n is large enough. □

PROPOSITION 6 *The re-solving algorithm* RES_1 *can achieve a competitive ratio of at least 0.710 with Instance 1.*

Proof Recall that RES_1 denote Algorithm 2 with our proposed re-solving time $t' = \lceil \frac{T}{D} \rceil$ and parameters $\alpha_0 = \frac{T/D}{\lceil T/D \rceil}$ and $\alpha_1 = \frac{T'/D}{\lceil T'/D \rceil}$. For Instance 1, the re-solve time is $\frac{T}{2}$ and $\alpha_0 = \alpha_1 = 1$.

To obtain a lower bound of the re-solving heuristic, we focus on the expected reward from u_2 to u_{2^n}. The expected reward before the re-solve step, i.e., during the first $\frac{T}{2}$ arrivals is at least

$$\sum_{u=2}^{2^n} [1-(1-py^*_{u2})^{\frac{T}{2}}] = \sum_{u=2}^{2^n} [1-(1-p\frac{1}{pT})^{\frac{T}{2}}] \approx (2^n-1)(1-e^{-\frac{1}{2}}).$$

Let U_1 be the set of offline vertices with non-zero capacity at the re-solving time point. It is easy to see that $2^{n-1} \le |U_1| \le 2^n$, and one optimal solution to LP (5) is $y^*_{u2} = \frac{1}{|U_1|}, \forall u \ge 2, u \in U_1$ and $y^*_u = 0, \forall u \ge 2, u \notin U_1$[2]. Then the expected reward from the remaining $\frac{T}{2}$ arrivals is at least

$$\sum_{u\ge 2 \text{ and } u\in U_1} [1-(1-py^*_u)^{\frac{T}{2}}] \approx \sum_{u\ge 2 \text{ and } u\in U_1} (1-e^{-p\frac{T}{2|U_1|}}) \ge \frac{|U_1|-1}{|U_1|}\frac{T}{2}(1-e^{-p}).$$

The approximation can be arbitrarily accurate when n is large enough. The inequality holds because $\frac{1}{y}(1-e^{-y})$ is decreasing in y in the interval $(0,1]$. From here we can derive a lower bound of the expected reward of RES_1 as follows.

$$\mathbb{E}[\mathrm{RES}_1] \ge (2^n-1)(1-e^{-\frac{1}{2}}) + \frac{|U_1|-1}{|U_1|}2^{n-1}(1-e^{-p}),$$

which further implies a lower bound of the competitive ratio of RES_1 as

$$\mathrm{CR}(\mathrm{RES}_1) = \frac{\mathbb{E}[\mathrm{RES}_1]}{\mathrm{OFF}} \ge \frac{(2^n-1)(1-e^{-\frac{1}{2}}) + \frac{|U_1|-1}{|U_1|}2^{n-1}(1-e^{-p})}{2^n p + o(2^n)} \approx 1-e^{-\frac{1}{2}} + \frac{1}{2}(1-e^{-1}) \approx 0.710.$$

The approximation can arbitrarily accurate when n is large enough. □

PROPOSITION 7 *The log-resolving algorithm* RES_K *can achieve a competitive ratio of at least 0.787 with Instance 1.*

Proof Recall that RES_K denotes the log-resolving algorithm (Algorithm 3) with $\gamma = \frac{1}{D} = \frac{1}{2}$, $\alpha_i = \frac{T_i/D}{\lceil T_i/D \rceil} = 1$ and K re-solving times. We assume that both $2^{n-K} \gg 1$ and $K \gg 1$. According to Algorithm 3, the ith re-solve happens after $T - \frac{T}{2^i}$ arrivals.

[2] There are many possible optimal solutions to the LP, we choose this solution because it is easy to analyze.

We ignore the reward generated by u_1. Denote $U_i, 1 \le i \le K$ as the set of offline vertices with non-zero capacity at the ith re-solving time point, and we can imply that $2^{n-i} \le |U_i| \le 2^n$. Let $y_{u2}^{(i)}$ denote the optimal solution to LP (5) for the ith re-solving. One optimal solution to this LP is $y_{u2}^{(i)} = \frac{1}{|U_i|}, \forall u \ge 2, u \in U_i$ and $y_{u2}^{(i)} = 0, \forall u \ge 2, u \notin U_i$. According to Algorithm 3, after the ith re-solve, a match is made with a probability of $y_{u2}^{(i)}$. Denote $R_i, 1 \le i \le K-1$ as the reward generated from arrivals between the ith and $(i+1)$th re-solves. Following a similar analysis to the RES_1 case, we have that $\forall i = 1, \dots, K-1$,

$$\mathbb{E}[R_i] \ge \sum_{u\ge 2 \text{ and } u\in U_i} [1-(1-py_{u2}^{(i)})^{\frac{T}{2^{i+1}}}] \approx \sum_{u\ge 2 \text{ and } u\in U_i} (1-e^{-p\frac{T}{2^{i+1}|U_i|}}) \ge \frac{|U_i|-1}{|U_i|}\frac{T}{2^i}(1-e^{-\frac{1}{2}p}).$$

The last inequality follows a similar analysis in Proposition 6. More specifically, $\frac{1}{y}(1-e^{-y})$ is decreasing in y in the interval $(0,1]$, which implies that $\frac{|U_i|}{pT/2^{i+1}}\left(1-e^{-p\frac{T/2^{i+1}}{|U_i|}}\right) \ge \frac{2}{p}(1-e^{-1/2p})$, since $2^{n-i} \le |U_i| \le 2^n$ implying $0 < p\frac{T/2^{i+1}}{|U_i|} \le \frac{1}{2}p$. Next, by applying a similar analysis as in Proposition 6, we can further derive that

$$\mathbb{E}[R_K] \ge \sum_{u\ge 2 \text{ and } u\in U_K} [1-(1-py_u^{(K)})^{\frac{T}{2^K}}] \approx \sum_{u\ge 2 \text{ and } u\in U_K} (1-e^{-p\frac{T}{2^K|U_K|}}) \ge \frac{|U_K|-1}{|U_K|}\frac{T}{2^K}(1-e^{-p}).$$

Similarly, we can further derive that the expected reward of the first $\frac{T}{2}$ arrivals is $(2^n-1)(1-e^{-\frac{1}{2}}) \ge (2^n-1)(1-e^{-\frac{1}{2}p})$. Therefore, we obtain a lower bound of the expected reward of the log-resolving algorithm as follows.

$$\begin{aligned}
\mathbb{E}[\text{RES}_K] \ge& \sum_{i=1}^{K-1}\frac{|U_i|-1}{|U_i|}\frac{T}{2^i}(1-e^{-\frac{1}{2}p}) + \frac{|U_K|-1}{|U_K|}\frac{T}{2^K}(1-e^{-p}) + (2^n-1)(1-e^{-\frac{1}{2}}) \\
\ge& \sum_{i=1}^{K-1}\big(\frac{1}{2^i}-\frac{1}{2^n}\big)T(1-e^{-\frac{1}{2}p}) + \big(\frac{1}{2^K}-\frac{1}{2^n}\big)T(1-e^{-p}) + (2^n-1)(1-e^{-\frac{1}{2}}) \\
=& \sum_{i=1}^{K-1}\frac{1}{2^i}T(1-e^{-\frac{1}{2}p}) + \big(\frac{1}{2^K}\big)T(1-e^{-p}) - \frac{1}{2^n}T(1-e^{-p}) - \frac{K-1}{2^n}T(1-e^{-\frac{1}{2}p}) + (2^n-1)(1-e^{-\frac{1}{2}}) \\
\ge& \sum_{i=0}^{K-1}\frac{1}{2^i}T(1-e^{-\frac{1}{2}p}) + \big(\frac{1}{2^K}\big)T(1-e^{-p}) - \frac{1}{2^n}T(1-e^{-p}) - \frac{K-1}{2^n}T(1-e^{-\frac{1}{2}p}) - (1-e^{-\frac{1}{2}}) \\
\ge& (2-\frac{1}{2^{K-1}}-\frac{K-1}{2^n})(1-e^{-\frac{1}{2}p})2^n + \big(\frac{1}{2^K}-\frac{1}{2^n}\big)(1-e^{-p})2^n - (1-e^{-\frac{1}{2}}).
\end{aligned}$$

Hence, the competitive ratio of RES_K is

$$\begin{aligned}
\text{CR}(\text{RES}_K) = \frac{\mathbb{E}[\text{RES}_K]}{\text{OFF}} \ge& \frac{(2-\frac{1}{2^{K-1}}-\frac{K-1}{2^n})(1-e^{-\frac{1}{2}p})2^n + \big(\frac{1}{2^K}-\frac{1}{2^n}\big)(1-e^{-p})2^n - (1-e^{-\frac{1}{2}})}{2^n p + o(2^n)} \\
\approx& \frac{(2-\frac{1}{2^{K-1}}-\frac{K-1}{2^n})(1-e^{-\frac{1}{2}p}) + \big(\frac{1}{2^K}-\frac{1}{2^n}\big)(1-e^{-p})}{p} \\
\ge& \frac{(2-\frac{1}{2^{K-1}}-\frac{K-1}{2^n})(1-e^{-\frac{1}{2}(1-\frac{1}{2^n})})}{1-\frac{1}{2^n}},
\end{aligned}$$

where the approximation can be arbitrarily accurate when n is large enough, and the second inequality is because $K \le n$. When K is large enough, we can obtain a lower bound of $\text{CR}(\text{RES}_K)$ as

$$\text{CR}(\text{RES}_K) \approx 2(1-e^{-\frac{1}{2}}) \approx 0.787.$$

□

B.3. Proofs of Section 5

LEMMA 4 *There exists an optimal solution $\boldsymbol{y}$ to LP (3), such that for any v, v' satisfying $q(v) = q(v')$ and $d_v = d_{v'}$, we have $y_{uv} = y_{uv'}$.*

Proof Consider an optimal solution to (3) denoted by $\boldsymbol{y}^*$. Define the set of online vertices in group q and demand size l as $S_{ql} = \{v : q(v) = q, d_v = l\}$. We show that another solution $\boldsymbol{y}'$ defined by $y'_{uv} = \sum_{v' \in S_{ql}} \frac{p_{v'}}{p_q} y^*_{uv'}$ for any $v \in S_{ql}$ is also optimal to (3), where $p_q = \sum_{v \in S_{ql}} p_v$ as defined earlier. We first check that it is feasible. For each $u \in U$,

$$\begin{aligned}\sum_{v:(u,v)\in E} p_v d_v y'_{uv} &= \sum_{q:(u,\hat{v})\in E \text{ for some } \hat{v}\in Q_q} \sum_{l\in L_q} \sum_{v\in S(ql)} l p_v y'_{uv} \\ &= \sum_{q:(u,\hat{v})\in E \text{ for some } \hat{v}\in Q_q} \sum_{l\in L_q} l p_q y'_{uv} \\ &= \sum_{q:(u,\hat{v})\in E \text{ for some } \hat{v}\in Q_q} \sum_{l\in L_q} l p_q \sum_{v'\in S_{ql}} \frac{p_{v'}}{p_q} y^*_{uv'} \le \frac{c_u}{T},\end{aligned}$$

where the second equality holds since y'_{uv} equals to the same value ($\sum_{v' \in S_{ql}} \frac{p_{v'}}{p_q} y^*_{uv'}$) for any $v \in S_{ql}$. For any $v \in V$, if $q(v) = q, d_v = l$,

$$\sum_{u,(u,v)\in E} y'_{uv} = \sum_{u,(u,v)\in E} \sum_{v'\in S_{ql}} \frac{p_{v'}}{p_q} y^*_{uv'} \le \sum_{u,(u,v_0)\in E} y^*_{uv_0} \le 1,$$

where $y^*_{uv_0} = \max_{v' \in S(q,l)} y^*_{uv'}$.
The objective value at $\boldsymbol{y}'$ is

$$\begin{aligned}\sum_{(uv)\in E} T p_v w_{uv} y'_{uv} &= \sum_u \sum_{q:(u,\hat{v})\in E \text{ for some } \hat{v}\in Q_q} \sum_{l\in L_q} \sum_{v\in S(ql)} T p_v w_{uq} y'_{uv} \\ &= \sum_u \sum_{q:(u,\hat{v})\in E \text{ for some } \hat{v}\in Q_q} \sum_{l\in L_q} T w_{uq} p_q \sum_{v'\in S_{ql}} \frac{p_{v'}}{p_q} y^*_{uv'} \\ &= \sum_u \sum_{q:(u,\hat{v})\in E \text{ for some } \hat{v}\in Q_q} \sum_{l\in L_q} \sum_{v'\in S_{ql}} T w_{uq} p_{v'} y^*_{uv'} \\ &= \sum_{(uv')\in E} T p_v w_{uv} y^*_{uv'}.\end{aligned}$$

Here with a slight abuse of notation, we use w_{uq} to denote w_{uv} for $v \in Q_q$. By definition, we have $w_{uv} = w_{uq}$ for all $v \in Q_q$. Therefore, $\boldsymbol{y}'$ is also optimal to (3). □

LEMMA 5 *LP (11) is equivalent to LP (3). Specifically, for any optimal solution $\{y^*_{uv_{ql}}\}$ to LP (11), we can construct $y^*_{uv} = y^*_{uv_{ql}}$ for any v satisfying $q(v) = q$ and $d_v = l$, and $\{y^*_{uv}\}$ will be optimal to LP (3).*

Proof To prove the two LPs are equivalent to each other. We first prove LP (11) is a relaxation of LP (3). To see that, consider any feasible solution $y_{uv_{ql}}$ to LP (11) and construct $y_{uv} = y_{uv_{ql}}$ for any v satisfying $q(v) = q, d_v = l$. Next we show that the constructed solution $\boldsymbol{y}$ is feasible to LP (3).

$$\begin{aligned}\sum_{v\in\mathcal{V}} p_v d_v y_{uv} &= \sum_{q\in Q} \sum_{l\in L_q} \sum_{v:q(v)=q,d_v=l} p_v l y_{uv_{ql}} \\ &= \sum_{q\in Q} \sum_{l\in L_q} p_{ql} l y_{uv_{ql}} \le \frac{c_u}{T}\end{aligned}$$

And for any $v \in S(q,l)$, $\sum_{u \in Adj(v)} y_{uv} = \sum_{u \in Adj(v_{ql})} y_{uv_{ql}} \le 1$ according to the definition of group. Hence $y_{uv} = y_{uv_{ql}}$ is one feasible solution to LP (3). In other words, LP (11) provide an lower bound to OPT.

We next show LP (11) provides optimal solutions. From Lemma 4, we know there exists an optimal solution to LP (3) such that $y_{uv} = y_{uv'}$ if $q(v) = q(v')$ and $d_v = d_{v'}$. Denote such a solution as $\boldsymbol{y}^*$. Consider such a solution $y_{uv_{ql}} = y^*_{uv}$ for $q = q(v)$ and $l = d_v$. We can check that it is feasible to LP (11) since

$$\sum_{q\in Q}\sum_{l\in L_q} p_{ql} l y_{uv_{ql}} = \sum_{q\in Q}\sum_{l\in L_q}\sum_{v\in\mathcal{V}, q(v)=q, d_v=l} p_v l y^*_{uv} = \sum_{v\in\mathcal{V},(u,v)\in E} p_v d_v y^*_{uv} \le \frac{c_u}{T} \quad \text{for any } u \in \mathcal{V}$$

and

$$\sum_{u\in Adj(q,l)} y_{uv_{ql}} = \sum_{u\in Adj(v)} y^*_{uv} \le 1 \quad \text{for } q(v) = q, d_v = l, q \in Q, l \in L_q.$$

Moreover, the objective function $\sum_{(u,v_{ql})\in E^s} T w_{uq} \sum_{l\in L_q} p_{ql} y_{uql}$ becomes $\sum_{(u,v_{ql})\in E^s} T w_{uq} \sum_{l\in L_q}\sum_{v\in\mathcal{V}, q(v)=q, d_v=l} p_v y^*_{uv} = \sum_{(u,v)\in E} T w_{uv} p_v y^*_{uv}$, which achieves the optimal value of (3). Hence the relaxation is exact. From the analysis above, we can imply that for any optimal solution $\{y^*_{uv_{ql}}\}$ to LP (11), we can construct $y^*_{uv} = y^*_{uv_{ql}}$ for any v satisfying $q(v) = q$ and $d_v = l$, and $\{y^*_{uv}\}$ is optimal to LP (3). □

LEMMA 6 *If $q(v) = q(v') = q$ and $d_v = l < m = d_{v'}$, then $W_{ql} \ge W_{qm}$.*

Proof Prove by contradiction. Assume $W_{ql} < W_{qm}$. Construct a new solution $y'_{uv_{ql}} = y'_{uv_{qm}} = \frac{l p_{ql} y^*_{uv_{ql}} + m p_{qm} y^*_{uv_{qm}}}{l p_{ql} + m p_{qm}}$ and other variables keep the same as the optimal solution $\boldsymbol{y}^*$. We first check the feasibility of this new solution. It is clear to see the total capacity consumption of u by v and v' does not change as $p_{ql} l y'_{uv_{ql}} + p_{qm} m y'_{uv_{qm}} = p_{ql} l \frac{l p_{ql} y^*_{uv_{ql}} + m p_{qm} y^*_{uv_{qm}}}{l p_{ql} + m p_{qm}} + p_{qm} m \frac{l p_{ql} y^*_{uv_{ql}} + m p_{qm} y^*_{uv_{qm}}}{l p_{ql} + m p_{qm}} = l p_{ql} y^*_{uv_{ql}} + m p_{qm} y^*_{uv_{qm}}$. We next prove that $\sum_u y'_{uv_{ql}} \le 1$ and $\sum_u y'_{uv_{qm}} \le 1$.

$$\sum_u y'_{uv_{ql}} = \sum_u \frac{l p_{ql} y^*_{uv_{ql}} + m p_{qm} y^*_{uv_{qm}}}{l p_{ql} + m p_{qm}} \le 1,$$

where the last equality holds because $\sum_u y^*_{uv_{ql}} \le 1$ and $\sum_u y^*_{uv_{qm}} \le 1$. Hence the new solution is feasible.

Now we look at the change of the objective.

$$\begin{aligned} &\sum_u T w_{uq}((p_{ql} + p_{qm}) y'_{uv_{ql}} - p_{ql} y^*_{uql} - p_{qm} y^*_{uv_{qm}}) \\ =& T \frac{(m-l) p_{qm} p_{ql}}{l p_{ql} + m p_{qm}} \sum_u w_{uq}(y^*_{uv_{qm}} - y^*_{uv_{ql}}) \end{aligned} \tag{34}$$

From the assumption $l < m$ and $W_{ql} = \sum_u w_{uq} y^*_{uv_{ql}} < \sum_u w_{uq} y^*_{uv_{qm}} = W_{qm}$, we can see the new solution generates a strictly larger reward which contradicts the fact that $\boldsymbol{y}^*$ is the optimal solution. □

LEMMA 7 *For each group $q \in Q$, when $\sum_{l\in L_q} T p_{ql} W_{ql}$ is fixed, we have*

$$\sum_{l\in L_q} T p_{ql} A_l W_{ql} \ge \left(-\frac{1}{2}\mathbb{E}[D_q] \left(\frac{t'}{T}\right)^2 \alpha^2 + \left(\frac{t'}{T}\right)\alpha \right) W_q,$$

where $\mathbb{E}[D_q]$ is the expected demand of vertices in Group q and W_q represents $\sum_{l\in L_q} T p_{ql} W_{ql}$.

Proof Consider the following linear program:

$$\begin{aligned} \mathbf{min}_{\boldsymbol{\omega}} & \sum_{l\in L_q} Tp_{ql}A_l\omega_l \\ \mathbf{s.t.} & W_q = \sum_{l\in L_q} Tp_{ql}\omega_l \\ & \omega_l \geq \omega_{l'}, \qquad \forall 1 \leq l \leq l' \\ & \omega_l \geq 0, \qquad \forall l \in L_q \end{aligned} \tag{35}$$

Let $K_q = \sum_{l\in L_q} Tp_{ql} = Tp_q$. We show that $\omega_l^* = \frac{W_q}{K_q} := \omega_0, \forall l \in L_q$ is optimal to (35). Suppose in the optimal solution $\boldsymbol{\omega}$, not all the elements are the same. Then there must exist a certain l and l' such that $\omega_l > \omega_0$ and $\omega_{l'} < \omega_0$. Denote $j = \max\{l : \omega_l > \omega_0\}$ and $k = \min\{l' : \omega_{l'} < \omega_0\}$. From the second constraint we have $j < k$, and for $j < l < k$, $\omega_l = \omega_0$. Define $\delta = min\{Tp_{qj}(\omega_j - \omega_0), Tp_{qk}(\omega_0 - \omega_k)\} > 0$. Construct a new solution defined by $\omega_j' = \omega_j - \frac{\delta}{Tp_{qj}}$ and $\omega_k' = \omega_k + \frac{\delta}{Tp_{qk}}$. $\omega_l' = \omega_l$ for all the other l.

We first show the new solution satisfies the second constraint in (35). From the definition of δ, we have

$$Tp_{qj}(\omega_j - \omega_0) \geq \delta \text{ and } Tp_{qk}(\omega_0 - \omega_k) \geq \delta,$$

hence

$$\omega_j' - \omega_k' = \omega_j - \omega_k - \frac{\delta}{Tp_{qj}} - \frac{\delta}{Tp_{qk}} \geq \omega_0 + \frac{\delta}{Tp_{qj}} + \frac{\delta}{Tp_{qk}} - \omega_0 - \frac{\delta}{Tp_{qj}} - \frac{\delta}{Tp_{qk}} = 0.$$

For $j < l < k$, we know $\omega_j' - \omega_l' = \omega_j - \frac{\delta}{Tp_{qj}} - \omega_0 \geq 0$ and $\omega_l' - \omega_k' = \omega_0 - \frac{\delta}{Tp_{qk}} - \omega_k \geq 0$. For $l < j$, we know $\omega_l' = \omega_l \geq \omega_j > \omega_j'$ and for $l > k$, $\omega_l' = \omega_l \leq \omega_k < \omega_k'$.

We next check that the first constraint $\sum_{l\in L_q} Tp_{ql}\omega_l' = \sum_{l\in L_q} Tp_{ql}\omega_l + \delta - \delta = W_q$. Therefore, $\boldsymbol{\omega}'$ is feasible to (35). The change of the objective function is

$$\sum_{l\in L_q} Tp_{ql}A_l\omega_l' - \sum_{l\in L_q} Tp_{ql}A_l\omega_l = \delta(A_k - A_j) < 0.$$

The last inequality holds because A_l is strictly decreasing in l by definition. In other words, $\boldsymbol{\omega}'$ generates lower objective while satisfying all the constraints, which contradicts the optimality assumption of $\boldsymbol{\omega}$. Therefore, in the optimal solution, we must have all the elements are the same. The first constraint implies that $\omega_l = \frac{W_q}{\sum_{l\in L_q} Tp_{ql}}$ for all l. The corresponding optimal value is

$$\sum_{l\in L_q} \frac{Tp_{ql}A_lW_q}{T\sum_{l\in L_q} p_{ql}} = \sum_{l\in L_q} p_{l|q}A_lW_q,$$

where $p_{l|q}$ denotes the conditional probability mass of demand size in group q. Note that W_{ql} satisfies the second constraint and the non-negativity constraint in (35), hence when $\sum_{l\in L_q} Tp_{ql}W_{ql}$ is fixed to W_q, it is a feasible solution to (35). Therefore,

$$\begin{aligned} \sum_{l\in L_q} Tp_{ql}A_lW_{ql} &\geq \sum_{l\in L_q} p_{l|q}A_lW_q \\ &= W_q \sum_{l\in L_q} p_{l|q}\left(-\left(\frac{t'}{T}\right)^2\frac{\alpha^2 l}{2} + \left(\frac{t'}{T}\right)\alpha\right) \\ &= \left(-\mathbb{E}[D_q]\frac{1}{2}\left(\frac{t'}{T}\right)^2\alpha^2 + \left(\frac{t'}{T}\right)\alpha\right)W_q \end{aligned} \tag{36}$$

□

THEOREM 3 *With distributional information of demand size,* SAMP(α) *generates a competitive ratio of* $\mathrm{CR} \geq \frac{1}{2M_e}$.

Proof From Lemma 7, we have that for every possible type q,

$$\sum_{l \in L_q} T p_{ql} A_l W_{ql} \geq \left(-\mathbb{E}[D_q] \frac{1}{2} \left(\frac{t'}{T} \right)^2 \alpha^2 + \left(\frac{t'}{T} \right) \alpha \right) W_q \geq \left(-M_e \frac{1}{2} \left(\frac{t'}{T} \right)^2 \alpha^2 + \left(\frac{t'}{T} \right) \alpha \right) W_q.$$

Therefore,

$$\sum_{q \in Q} \sum_{l \in L_q} T p_{ql} A_l W_{ql} \geq \left(-M_e \frac{1}{2} \left(\frac{t'}{T} \right)^2 \alpha^2 + \left(\frac{t'}{T} \right) \alpha \right) \sum_{q \in Q} W_q = \left(-M_e \frac{1}{2} \left(\frac{t'}{T} \right)^2 \alpha^2 + \left(\frac{t'}{T} \right) \alpha \right) W.$$

According to (13) and the fact that W is the optimal value of LP (11), we get a competitive ratio

$$\mathrm{CR} \geq -M_e \frac{1}{2} \left(\frac{t'}{T} \right)^2 \alpha^2 + \left(\frac{t'}{T} \right) \alpha.$$

By setting $\alpha \frac{t'}{T} = \frac{1}{M_e}$, the lower bound is maximized to $\frac{1}{2M_e}$. In particular, when t' takes any integer value between $\lceil \frac{T}{M_e} \rceil$ and T, we can ensure that $0 \leq \alpha \leq 1$. In other words, this maximal value is achievable. □

C. Generalizing the State-of-Art Algorithms for Special Cases to Our General Model

In this section, we will illustrate the challenge in generalizing the algorithms for two special online matching problems—the single-unit demand and the generalized AdWords models—to the general multi-unit demand considered in this paper.

C.1. From Single-Unit Demand to Multi-Unit Demand

To illustrate our contribution in generalizing the online matching problems from single-unit demand to multi-unit demand, we first show how to convert an algorithm for a single-unit demand setting to that for a multi-unit demand setting. Then by applying the best algorithms for the single-unit demand setting we get its corresponding competitive ratio in the multi-unit demand setting and compare it with the ratio from our algorithms.

MILP formulation. Recall from Section 3 that the optimal offline solution to the multi-unit demand setting can be solved from a MILP. In particular, given a realized arrival sequence s of requests, we can solve the following MILP to optimally match them to the resources (same as MILP (1)).

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

Here $N_v(s)$ denotes the number of type v vertex appeared in sequence s. $X_{uv}(s)$ denotes the decision variables that represent how many requests of type v are matched to the resource u. The first set of constraints restricts the total consumption of each resource u below its capacity and the second set of constraints specifies that the matched request cannot exceed the total arrivals. Let $\mathrm{OFF}^{\mathrm{MUL}}$ denote the optimal value of this program.

Recall that $D = \max_v d_v$. Next we consider the MILP below:

$$\begin{aligned} \textbf{max} \quad & \sum_{(u,v)\in E} w_{uv} X_{uv}(s) \\ \textbf{s.t.} \quad & \sum_{v:(u,v)\in E} D X_{uv}(s) \le c_u, && \forall u \in U \\ & \sum_{u:(u,v)\in E} X_{uv}(s) \le N_v(s), && \forall v \in V \\ & X_{uv}(s) \in \mathbb{N}, && \forall (u,v) \in E \end{aligned} \tag{37}$$

Because $X_{uv}(s)$ is an integer, this MILP can be written as:

$$\begin{aligned} \textbf{max} \quad & \sum_{(u,v)\in E} w_{uv} X_{uv}(s) \\ \textbf{s.t.} \quad & \sum_{v:(u,v)\in E} X_{uv}(s) \le \left\lfloor \frac{c_u}{D} \right\rfloor, && \forall u \in U \\ & \sum_{u:(u,v)\in E} X_{uv}(s) \le N_v(s), && \forall v \in V \\ & X_{uv}(s) \in \mathbb{N}, && \forall (u,v) \in E \end{aligned} \tag{38}$$

The new MILP now represents a single-unit consumption problem. We let OFF^{SIN} denote its optimal value.

LEMMA 8. *Any algorithm with competitive ratio α for single-unit demand setting can be converted to another algorithm with competitive ratio $\alpha\beta$ for multi-unit demand setting, where $\beta = \frac{OFF^{SIN}}{OFF^{MUL}}$.*

Proof An algorithm with competitive ratio α has an output such that $\frac{\mathbb{E}[\text{ALG}]}{\mathbb{E}[\text{OFF}^{\text{SIN}}]} \ge \alpha$. By the definition of β, we then have $\frac{\mathbb{E}[\text{ALG}]}{\mathbb{E}[\text{OFF}^{\text{MUL}}]} \ge \alpha\beta$. This means if we assume that every request v has demand D, then the multi-unit demand becomes the single-unit demand setting. Running this algorithm can therefore give us a competitive ratio of $\alpha\beta$ as desired. □

Next we can further bound the value of β via the following lemma.

LEMMA 9. $\beta = \frac{OFF^{SIN}}{OFF^{MUL}} \ge \min_u \frac{\lfloor \frac{c_u}{D} \rfloor}{c_u}$.

Proof Let $\{X^m_{uv}\}$ be an optimal solution to MILP (1). Recall that X^m_{uv} means the number of requests of type v that are matched to u. For a given resource vertex u, we build a match set for u that contains all requests matched to u. Because X^m_{uv} can be larger than 1, we treat requests of the same type v as different requests in the match set. Then we choose top $\lfloor \frac{c_u}{D} \rfloor$ requests according to the request weights. Following this new allocation, we can build another solution X'_{uv}, and we have

$$\sum_{v:(u,v)\in E} w_{uv} X'_{uv} \ge \frac{\lfloor \frac{c_u}{D} \rfloor}{c_u} \sum_{v:(u,v)\in E} w_{uv} X^m_{uv}.$$

It is also easy to check that $\{X'_{uv}\}$ is a feasible solution to MILP (38). This gives us the following inequalities:

$$\text{OFF}^{\text{SIN}} \ge \sum_u \sum_{v:(u,v)\in E} w_{uv} X'_{uv} \ge \left(\min_u \frac{\lfloor \frac{c_u}{D} \rfloor}{c_u} \right) \left(\sum_u \sum_{v:(u,v)\in E} w_{uv} X^m_{uv} \right) = \left(\min_u \frac{\lfloor \frac{c_u}{D} \rfloor}{c_u} \right) \text{OFF}^{\text{MUL}}.$$

It now becomes obvious that $\beta = \frac{\text{OFF}^{\text{SIN}}}{\text{OFF}^{\text{MUL}}} \ge \min_u \frac{\lfloor \frac{c_u}{D} \rfloor}{c_u}$. □

Note that this ratio of $\min_u \frac{\lfloor \frac{c_u}{D} \rfloor}{c_u}$ is always no larger than $\frac{1}{D}$.

REMARK 1. We can further improve the lower bound of β from $\min_u \frac{\lfloor \frac{c_u}{D} \rfloor}{c_u}$ to $\min_u \frac{\lfloor \frac{c_u}{D_u} \rfloor}{c_u}$ where $D_u = \max_{v:(u,v)\in E} d_{uv}$. This can be obtained by replacing D in MILP 38 by D_u for the constraint

$$\sum_{v:(u,v)\in E} X_{uv}(s) \le \left\lfloor \frac{c_u}{D} \right\rfloor,$$

and use D_u instead of D in Lemma 9.

Brubach et al. (2020) provide an algorithm that achieves 0.705 competitive ratio for single-unit demand setting. If we apply Lemma 8 and Lemma 9 to this algorithm, it gives us a new algorithm with a competitive ratio of $0.705 \cdot \min_u \frac{\lfloor \frac{c_u}{D} \rfloor}{c_u}$. In the meanwhile, our algorithm with the re-solving heuristic has a competitive ratio of $\frac{1}{2D(1-\theta)}$ where $\theta = (1-\frac{1}{D})e^{-\frac{B}{D}}$ (when we assume the re-solving times K is large). To compare the two competitive ratios, consider $D \ge 2$, $\min_v d_v = 1$ and $c_u = 2D-1$ for all u, then we have $\min_u \frac{\lfloor \frac{c_u}{D} \rfloor}{c_u} = \frac{1}{2D-1}$ and the ratio of their algorithm becomes $\frac{0.705}{2D-1}$. On the other hand, our work gives a ratio of $\frac{1}{2D(1-\theta)} > \frac{1}{2D}\frac{1}{1-\frac{1}{2}e^{-2}} \approx \frac{1.0726}{2D}$, which is always strictly better than $\frac{0.705}{2D-1}$. For example, when $D=2$, we improve the competitive ratio from 0.235 to 0.281, and when $D=3$, we improve the competitive ratio from 0.141 to 0.191.

C.2. From the Generalized AdWords Model to the General Multi-unit Demand Model

Note that a special type of multi-unit demand model has been widely studied in the literature, which is known as a generalized AdWords problem. In a generalized AdWords problem, the rewards are assumed to be the same as the demand size. The best known algorithm for the problem is derived by Stein et al. (2020). We try to apply their algorithm to our model but unfortunately, it does not lead to a better competitive ratio for our problem. Specifically, we first write down a similar LP (Section 5 in Stein et al. (2020)) and replace u_{ij} by w_{ij} in the objective and demand u_{ij} by d_i in the constraints. Then we apply the Large-Or-Small algorithm from Section 6. In the proof of Theorem 1 of Stein et al. (2020), it shows that the expected amount of resource j allocated to customers is at least $\frac{1}{2}(1-\frac{1}{e})U_j$. Follow the similar notation in Stein et al. (2020), we can prove that the expected amount of resource j allocated to customers/requests is at least $\frac{1}{2}(1-\frac{1}{e})U_j$, where $U_j = \sum_i x^*_{ij} d_i$ in our setting. On the other hand, the expected reward generated by resources j if it is fully utilized is at least $\frac{w}{D}U_j$ where $w = \min_{i,j} w_{ij}$ and $D = \max_i d_i$. Furthermore, we have $\frac{w}{D}U_j \ge \frac{w}{D}\sum_i x^*_{ij} \ge \frac{w}{D}\frac{\sum_i x^*_{ij} w_{ij}}{W}$ where $W = \max_{i,j} w_{ij}$. Therefore, the expected reward generated by the Large-Or-Small algorithm is at least $\frac{1}{2}\left(1-\frac{1}{e}\right)\frac{w}{D}\frac{\sum_j \sum_i x^*_{ij} w_{ij}}{W}$. Note that the optimal objective of LP is $\sum_j \sum_i x^*_{ij} w_{ij}$ which is an upper bound of the offline optimal. This shows that the Large-Or-Small algorithm from Stein et al. (2020) can achieve a competitive ratio of $\frac{1}{2D}\frac{w}{W}(1-\frac{1}{e})$ for our problem. Note that this value is always smaller than $\frac{1}{2D}$ which is the first bound obtained in our paper.

D. Baseline Algorithms

D.1. MIX Algorithm

We design the MIX algorithm based on Algorithm 1 in (Naori and Raz 2019). For an edge (u,v), we call it heavy edge if $d_v > \frac{1}{2}c_u$, and light edge otherwise. Considering a problem instance of our model I, we divide I into two sub-instances I_h and I_l: both consist of original resources and requests while I_h includes all heavy edges and I_l includes all light edges. For an instance I and a set S of requests, let $I|_S$ denote the sub-instance of I that only has the requests in S. We denote $G(I)$ as the weighted bipartite graph for the instance I. The MIX algorithm tested in the paper is shown as follows, where the coefficients q_1 and q_2 take the same value as those in (Naori and Raz 2019).

D.2. FBS Algorithm

We design the FBS Algorithm (Algorithm 5) according to the Algorithm 4 provided in Vera and Banerjee (2021).

Algorithm 4 MIX Algorithm

$\boldsymbol{y}^* = \mathbf{0}$
Matching $M = \phi$
Capacity $C_{u0} = c_u$
$\alpha = 1$
$q_1 = 0.5256$ and $q_2 = 0.69$.
Online vertices set $S = \phi$
for $t = 1$ to T **do**
  Online vertex v arrives
  $S = S + v$
  if $t \le q_1 T$ **then**
    Continue
  end if
  if $q_1 T + 1 \le t \le q_2 T$ **then**
    Calculate maximum-weight matching M_0 of $G(I_h|_S)$
    if there is u that $(u, v) \in M_0$ and $C_{ut-1} \ge d_v$ **then**
      Match u and v: $C_{ut} = C_{u(t-1)} - d_v$
      $M = M \cup \{(u, v)\}$
    end if
  end if
  if $t \ge q_2 T + 1$ **then**
    if $\boldsymbol{y}^* = 0$ **then**
      Solve the LP (5) of I_l and get the optimal solution $\boldsymbol{y}^*$
    end if
    Randomly choose u with probability αy^*_{uv}
    if $C_{u(t-1)} \ge d_v$ **then**
      Match u and v: $C_{ut} = C_{u(t-1)} - d_v$
      $M = M \cup \{(u, v)\}$
    end if
  end if
end for
return M

Algorithm 5 FBS Algorithm

Solve the LP (3) and get the optimal solution $\boldsymbol{y}^*$
Matching $M = \phi$
Capacity $\mathcal{C}_{u0} = c_u$
for $t = 1$ to T **do**
  Online request v arrives
  Choose u with largest y^*_{uv}
  if $\mathcal{C}_{u(t-1)} \geq d_v$ **then**
    Match u and v: $\mathcal{C}_{ut} = \mathcal{C}_{u(t-1)} - d_v$
    $M = M \cup \{(u, v)\}$
  else
    $\mathcal{C}_{ut} = \mathcal{C}_{u(t-1)}$
  end if
  Solve the LP (5) and get the optimal solution $\boldsymbol{y}'^*$
  $\boldsymbol{y}^* = \boldsymbol{y}'^*$
end for
return M