Online Task Assignment Problems with Reusable Resources

Hanna Sumita,[1] Shinji Ito,[2] Kei Takemura,[2] Daisuke Hatano,[3] Takuro Fukunaga,[4]
Naonori Kakimura,[5] Ken-ichi Kawarabayashi,[6]

[1] Tokyo Institute of Technology, [2] NEC Corporation, [3] RIKEN AIP, [4] Chuo University,
[5] Keio University, [6] National Institute of Informatics
sumita@c.titech.ac.jp, {kei_takemura, i-shinji}@nec.com, daisuke.hatano@riken.jp, fukunaga.07s@chuo-u.ac.jp,
kakimura@math.keio.ac.jp, k_keniti@nii.ac.jp

Abstract

We study online task assignment problem with reusable resources, motivated by practical applications such as ridesharing, crowdsourcing and job hiring. In the problem, we are given a set of offline vertices (agents), and, at each time, an online vertex (task) arrives randomly according to a known time-dependent distribution. Upon arrival, we assign the task to agents immediately and irrevocably. The goal of the problem is to maximize the expected total profit produced by completed tasks. The key features of our problem are (1) an agent is reusable, i.e., an agent comes back to the market after completing the assigned task, (2) an agent may reject the assigned task to stay the market, and (3) a task may accommodate multiple agents. The setting generalizes that of existing work in which an online task is assigned to one agent under (1).

In this paper, we propose an online algorithm that is $1/2$-competitive for the above setting, which is tight. Moreover, when each agent can reject assigned tasks at most Δ times, the algorithm is shown to have the competitive ratio $\Delta/(3\Delta - 1) \geq 1/3$. We also evaluate our proposed algorithm with numerical experiments.

1 Introduction

Online task assignment problem has attracted extensive attention recently in combinatorial optimization and artificial intelligence. The problem models practical situations that assign agents to tasks arriving online. For example, in a rideshare platform (Dickerson et al. 2021; Dong et al. 2021; Lowalekar, Varakantham, and Jaillet 2020; Nanda et al. 2020) such as Uber and Lyft, we match drivers to riders where requests from riders arrive one by one. Other applications include crowdsourcing (Assadi, Hsu, and Jabbari 2015; Ho and Vaughan 2012; Xu et al. 2017) and job hiring (Anagnostopoulos et al. 2018; Dickerson et al. 2019).

In this paper, we study online task assignment problem under *known adversarial distributions*. In the problem, we are given a time horizon T and a bipartite graph $G = (U, V; E)$, where U corresponds to a set of agents and V represents *types* of tasks. A type of a task is usually defined by attributes of the task, e.g., pick-up/drop-off locations of a rider in a rideshare platform and language skills required to complete a task in a crowdsourcing platform. An edge $(u, v) \in E$ means that u has ability to process an online task v. In the problem, the set U of agents is known in advance (offline vertices), while we are given a vertex with a type $v \in V$ (an online vertex) at each time, randomly according to a time-dependent distribution D_t over V. We identify an online vertex with its type v. On arrival of v, we immediately and irrevocably either assign some agents to v, or discard the chance of processing v. Then we obtain a profit produced by an assigned agent if she completes a task. The goal is to design an online algorithm that maximizes the total profit. We here assume that D_t is known in advance, which is called a *Known Adversarial Distribution (KAD) model* (Dickerson et al. 2021); see also (Alaei, Hajiaghayi, and Liaghat 2012). If D_t is the same for all t, it is called a *Known I.I.D. (KIID) model*.

Motivated by practical applications, it has been studied to generalize the online task assignment problem with additional specific conditions. An example is *reusability* of agents (Dickerson et al. 2021; Dong et al. 2021). In the standard online task assignment problem, when an agent is assigned a task, she will leave immediately from the market. However, if agents are reusable, an agent comes back to the market after completing the task, and she can be matched to a new task again. Such situation especially arises in a rideshare platform with drivers and riders. Another example is *rejections* by agents (Nanda et al. 2020). It is natural that an agent is allowed to reject an assigned task if it is not satisfactory. Moreover, each agent may have an upper bound on the number of rejections, that is, when she rejects an assigned task Δ times, then she must leave from the market. It should be noted that the previous work mentioned above studied the *online bipartite matching* where only one agent can be assigned to an online task, and it is not straightforward to extend[1] the results to the online task assignment problem.

In this paper, we introduce the online task assignment problem in the KAD model with all the constraints mentioned above, that is, for the problem with the following constraints:

[1] We can consider a simple reduction that replaces an online vertex v with capacity b_v as b_v vertices with capacity 1 arriving sequentially. Such reduction, however, would not work in the KIID/KAD model because an online vertex is chosen independently at each time.

A. **(reusability)** an agent comes back to the market after she completes a task, where the occupation time is drawn from a known distribution,

B. **(rejections)** an agent u rejects an assignment with some probability, and has to leave after rejecting Δ_u times, and

C. **(task assignment)** an online vertex v can accommodate at most b_v offline vertices.

See Section 2 for a formal problem definition.

1.1 Our Contribution

Our main result is to propose an online algorithm for the above problem. We prove that our algorithm has the following theoretical guarantees:

- $1/2$-competitive for the unlimited rejection case, i.e., when Δ_u is $+\infty$ for all u (Theorem 1),
- $\Delta/(3\Delta - 1)$-competitive for the general case (Theorem 2), where $\Delta = \max_{u \in U} \Delta_u$.

We here evaluate the performance of online algorithms with the common measure called *competitive ratio*. We say that an algorithm is α-competitive if the expected profit obtained by the algorithm is at least α times the offline optimal value (See Section 2.2).

Our results and existing results are summarized in Table 1. Note that our algorithm is useful for the problem without reusability. In fact, we prove that our algorithm has the competitive ratio $(1 - 1/e^2)/2 > 0.432$ in the KIID model without reusability, which improves on the competitive ratio $1/e$ (Nanda et al. 2020) for the same setting (Theorem 3).

In addition, we show that no (adaptive) algorithm can achieve the competitive ratio better than $1/2$. This is implied by a well-known example for the prophet inequality problem (Krengel and Sucheston 1977, 1978). We note that this hardness result is applied even in the setting of (Dickerson et al. 2021). Thus our result complements their result, which considers only non-adaptive algorithms.

Due to the space limitation, we omit some results and the details of the proofs; see the full version. [2] Below let us describe technical highlights of the proof of our algorithms.

Technique To design an online algorithm, we first construct an offline linear program (LP) that gives an upper bound on the offline optimal value. This is a similar approach to the online bipartite matching in the KIID/KAD model (Alaei, Hajiaghayi, and Liaghat 2012; Dickerson et al. 2021; Nanda et al. 2020). Our LP has variables $x_{uv,t}$ for each $(u, v) \in E$ and t, and has linear inequalities that incorporate the constraints (A)–(C) above. See Section 2 for the formal definition of our LP. Intuitively, each variable $x_{uv,t}$ corresponds to a probability that v arrives at time t and u is assigned to v. We shall use an LP optimal solution $x^*_{uv,t}$ to determine how to assign agents to v at each time. There are, however, several difficulties to obtain our results as described below.

First, in each time, we need to find a set S of at most b_v agents according to a probability distribution defined by LP optimal solution $x^*_{uv,t}$. For the online bipartite matching, this is easy; we just choose an edge e with probability proportional to $x^*_{e,t}$ at time t. However, for the online task assignment problem, we need to construct a distribution of feasible sets so that the probability of choosing e at time t is $x^*_{e,t}$. To do it, we use Carathéodory's theorem in convex analysis. The theorem allows us to decompose $x^*_{e,t}$ to a polynomial number of feasible sets, which can be regarded as a probability distribution on feasible sets.

Another difficulty is that the LP optimal value may give a loose upper bound on the offline optimal value. In this case, finding a feasible set based on LP is not sufficient; we may select an agent which should not be chosen. In fact, there exists a problem instance such that an algorithm similarly to Nanda et al. (2020) that just assigns an agent according to an LP optimal solution would fail to obtain the competitive ratio more than $1/3$ (See the full version for a specific example). To overcome this difficulty, we adapt the idea of Alaei, Hajiaghayi, and Liaghat (2012), who analyzed a different online matching problem in the KAD model. Specifically, after finding a feasible set S based on $x^*_{e,t}$, we decide whether to assign a task v to u for each agent $u \in S$. We compute the expected profit that u earns at and after the current time t by assigning u to v, and, if it is larger than the one when not assigning, then we decide to assign u to v.

To prove that the proposed algorithm admits desired competitive ratio, we evaluate the expected profit that each vertex u earns. Let $R^d_{u,t}$ be the expected profit that u earns at and after time t, when u has a remaining budget d. Then $R^d_{u,t}$ can be represented recursively with respect to t and d. Since the recursive equation is linear, we can bound $R^d_{u,t}$ by introducing another linear program and its dual. We note that when Δ_u is infinite, the recursive equation becomes simpler depending on only t, that gives a better competitive ratio.

Let us describe the differences from (Dickerson et al. 2021; Nanda et al. 2020). The algorithm of Dickerson et al. (2021) assumes that the algorithm can access a probability that $u \in U$ is *available* at time t, i.e., u is not occupied by a task at time t. This assumption may be problematic because it is not easy to obtain such information precisely. In contrast, we adopt expected profit after t to decide the assignment, which can be computed deterministically and efficiently by dynamic programming. Our algorithm is also different from Nanda et al. (2020), as their algorithm just uses $x^*_{e,t}$ as a probability mentioned above, which works only for the KIID model. Our algorithm is shown to admit a better competitive ratio in their setting, which improves on the competitive ratio $1/e$ by Nanda et al. (2020) (Theorem 3).

Experiments We evaluate the performance of our algorithm through experiments. We use a synthetic dataset and the real-world dataset of taxi trip records, similarly to previous work (Dickerson et al. 2021; Nanda et al. 2020). Our algorithm performs the best in most cases, and runs practically fast enough. The results imply the superiority of our algorithm.

[2] https://alg.c.titech.ac.jp/sumita/preprint/AAAI2022.pdf

	Reusability	# Rejections	# Assigned agents	Online vertices	Comp. ratio	Hardness
This work	✓	$+\infty$	≥ 1	KAD	$1/2$	$1/2$
This work	✓	$\leq \Delta$	≥ 1	KAD	$\Delta/(3\Delta-1)$	
(Dickerson et al. 2021)	✓	NA	1	KAD	$1/2$	$(1/2)$
This work		$\leq \Delta$	≥ 1	KIID	> 0.432	
(Nanda et al. 2020)		$\leq \Delta$	1	KIID	$1/e$	$1-1/e$

Table 1: Summary of our results and previous work. The algorithm (Dickerson et al. 2021) requires to know in advance the probability that an agent is available at each time.

1.2 Related Work

The online task assignment problem is a generalization of the online bipartite matching (where each vertex can match to at most one neighbor). Nowadays there exist a large body of literature on online matching. We here mention some of them closely related to our problem. See Mehta (2013) for the detailed survey.

The online bipartite matching was introduced by Karp, Vazirani, and Vazirani (1990). They considered the adversarial input model, that is, the model that online vertices arrive in an adversarial order, and proposed a randomized $(1-1/e)$-approximation algorithm for the unweighted case. It is known that the ratio is tight (Birnbaum and Mathieu 2008; Mehta et al. 2007). When online vertices arrive according to an i.i.d. distribution (i.e., the KIID model), Manshadi, Gharan, and Saberi (2012) proposed a 0.702-competitive algorithm and showed that the ratio cannot be better than 0.823. For the edge-weighted case, a 0.667-competitive algorithm is known (Haeupler, Mirrokni, and Zadimoghaddam 2011). It is also known that we can do better if arrival rates (the expected number of arrivals in T rounds) are integral (Brubach et al. 2016).

Online stochastic matching, introduced by Mehta and Panigrahi (2012), is the problem that an offline vertex accepts an assignment with a probability. This can be viewed as that infinite number of rejections are allowed in the process. For the problem in the adversarial input model, Mehta, Waggoner, and Zadimoghaddam (2015) presented a 0.534-competitive algorithm for the unweighted case when edge probabilities go to 0. For the KIID model, Brubach et al. (2016) gave a $(1-1/e)$-competitive algorithm, which works also in the edge-weighted case.

Online bipartite matching in the KAD model was introduced by Alaei, Hajiaghayi, and Liaghat (2012) under a name of *prophet-inequality* matching, as the problem includes the prophet inequality problem (Krengel and Sucheston 1977, 1978) as a special case. The KAD model includes the KIID model as a special case. Alaei, Hajiaghayi, and Liaghat (2012) extended to the problem that an offline vertex has some capacity. Dickerson et al. (2021) gave a $1/2$-competitive algorithm for the problem with the reusability conditions. Our problem can be viewed as a general framework that unifies the problem of Dickerson et al. and online stochastic matching with limited number of rejections.

When offline vertices have capacities, the online bipartite matching is often called the AdWords problem (Mehta et al. 2007). For the KIID model, there is a $(1-\varepsilon)$-competitive algorithm (Devanur and Hayes 2009) under the small bid assumption. The AdWords problem in the KAD model is studied in a line of researches for ridesharing (Lowalekar, Varakantham, and Jaillet 2020).

Nanda et al. (2020) discussed the online bipartite matching problem that maximizes fairness, instead of the total profit. Here, the fairness means the smallest acceptance ratio in tasks. Note that our results can easily be extended to their setting to balance a trade-off between profit and fairness.

2 Model

In this section, we describe a formal definition of our problem. For a positive integer k, we denote $[k] = \{1, \dots, k\}$. We are given a bipartite graph $G = (U, V; E)$ with edge weight $w_e \geq 0$, where U is the set of offline vertices and V is the set of types of online vertices. We are also given a time horizon T. Each offline vertex $u \in U$ has a positive integer Δ_u, which is a budget of allowed rejections in the process. In addition, each edge $(u, v) \in E$ has a random variable $C_e \in [T]$ that represents the occupation time for u to complete the task v. In other words, when u accepts v, u is absent from the market, and will be available at time $t + C_e$, where C_e is drawn from a given distribution. We say that an offline vertex u is *available* at time t if u is in the market (i.e., not being occupied by some task) and u has not rejected online vertices Δ_u times.

For each time $t \in [T]$, an online vertex v arrives according to a probability distribution $\{p_{v,t}\}_v$.[3] Upon arrival of a vertex v, we immediately and irrevocably either assign at most b_v neighbors to v that are available, or discard v. When $u \in U$ is assigned, u either accepts v with probability q_e or rejects v with probability $1 - q_e$, where $e = (u, v)$. When u accepts v, we obtain a profit w_e and u becomes absent from the market during the occupation time C_e.

We note that we may assume $\Delta_u > 0$ for all $u \in U$. When no rejection is allowed for u, i.e., $\Delta_u = 0$, only edges e with $q_e = 1$ can be matched to u. Therefore, we can remove all the edges e such that e is incident to u and $q_e < 1$, and set Δ_u to 1.

Our goal is to design an online (randomized) algorithm that maximizes the expected total profit. The performance of an online algorithm is evaluated by the competitive ratio. In the subsequent sections, we define the offline optimal value and the competitive ratio.

We note that our model includes the online bipartite matching problem studied in (Dickerson et al. 2021; Nanda et al. 2020) as special cases. We assume that $b_v = 1$ for

[3] Nothing arrives at time t with probability $1 - \sum_{v \in V} p_{v,t}$.

each $v \in V$. When $\Delta_u = +\infty$ for any $u \in U$ and the acceptance probability q_e is one for each $e \in E$, we can ignore the constraint on rejections, and hence our model is identical to the one in (Dickerson et al. 2021). On the other hand, when probability distribution $\{p_{v,t}\}_v$ is the same for all $t \in [T]$ and $\Pr[C_e = T] = 1$ for any $e \in E$, our model is the exactly one in the KIID model without reusable resources, which was studied in (Nanda et al. 2020). We also note that our model and the problem of Alaei, Hajiaghayi, and Liaghat (2012) have no inclusion relationships.

2.1 Offline Optimal Algorithms

Given a problem instance, a sequence of online vertices is determined according to probability distributions $\{p_{v,t}\}_{v,t}$. We denote by $\mathcal{I}$ the probability distribution over all input sequences of online vertices. In the offline setting, we suppose that we know the sequence I of online vertices in advance. An offline algorithm that maximizes the expected profit is called an *offline optimal algorithm for* I. We note that the algorithm does not know whether each offline vertex accepts or rejects an online vertex at each time. We denote the expected profit of the offline optimal algorithm for I by $\mathrm{OPT}(I)$. Define the *offline optimal value* by $\mathbb{E}_{I\sim\mathcal{I}}[\mathrm{OPT}(I)]$. Thus we consider the best algorithm for each I in the offline optimal value.

We denote the set of edges incident to u by $E_u = \{(u,v) \mid (u,v) \in E\}$ for each $u \in U$, and similarly E_v for $v \in V$. We introduce an offline LP whose optimal value is an upper bound of the offline optimal value:

$$
\begin{aligned}
&(\texttt{Off})\\
&\max \sum_{t\in[T]} \sum_{e\in E} w_e q_e x_{e,t}\\
&\text{s.t.} \sum_{t'<t} \sum_{e\in E_u} x_{e,t'} q_e \Pr[C_e \geq t - t' + 1]\\
&\qquad + \sum_{e\in E_u} x_{e,t} q_e \leq 1 \quad (u \in U, t \in [T]) \qquad (1)\\
&\qquad \sum_{t\in[T]} \sum_{e\in E_u} x_{e,t} \left(1 - q_e \Pr[C_e \leq T - t]\right) \leq \Delta_u\\
&\qquad\qquad (u \in U) \qquad (2)\\
&\qquad \sum_{e\in E_v} x_{e,t} \leq p_{v,t} b_v \quad (v \in V, t \in [T]) \qquad (3)\\
&\qquad 0 \leq x_{e,t} \leq p_{v,t} \quad (v \in V, e \in E_v, t \in [T]) \qquad (4)
\end{aligned}
$$

For each edge $e = (u,v)$ and time t, a variable $x_{e,t}$ corresponds to a probability that e is chosen at time t. Intuitively, constraints (1)–(3) corresponds to the constraints (A)–(C) mentioned in the introduction, respectively.

Lemma 1. *The optimal value of LP* (Off) *is at least the offline optimal value* $\mathbb{E}_{I\sim\mathcal{I}}[\mathrm{OPT}(I)]$.

Proof. For each input sequence I, an offline optimal algorithm for I determines a probability that, when an online vertex v arrives at time t, v is assigned to offline vertices u for each $e = (u,v) \in E$ and $t \in [T]$. By taking expectations over $\mathcal{I}$, we set $x_{e,t}$ to be the probability that an online vertex v arrives at t and an offline vertex u is assigned to v for each $e = (u,v) \in E$ and $t \in [T]$. We show that x defined above is a feasible solution to the LP. By definition, $x_{e,t}$ is nonnegative and at most the probability $p_{v,t}$, and hence (4) is satisfied.

Let us see that the first constraint (1) is valid. Fix an input sequence I, $u \in U$, and $t \in [T]$. Let $v^{t'}$ be an online vertex arriving at time $t' < t$ (if exists). For any realization of acceptances/rejections and C_e's, we have any one of the following: u has no assignment, or u is occupied by $v^{t'}$ for some $t' < t$ with occupation time at least $t - t' + 1$. Since acceptances/rejections and C_e's are sampled independently, the expected number of online vertices occupying u is $\Pr[u \text{ accepts } v^t \mid I] + \sum_{t'<t} \Pr[u \text{ accepts } v^{t'} \text{ at } t' \mid I] \Pr[C_{(u,v^{t'})} \geq t - t' + 1]$. Since $\Pr[u \text{ accepts } v^t \mid I] = x_{(u,v)} q_{(u,v)}$ and the expected number is at most 1, (1) holds.

To see the second constraint (2), let us fix an input sequence I and $u \in U$. For any realization of acceptances/rejections and C_e's, the number of rejections plus assignments with u never coming back (i.e., C_e being at least $T - t + 1$) is at most Δ_u. Then the expected number of rejections is $\sum_{t\in[T]} \Pr[u \text{ rejects some } v \in V \text{ at } t \mid I] = \sum_{e\in E_u} x_{e,t}(1 - q_e)$. The number of assignments that u never returns is $\sum_{t\in[T]} \Pr[u \text{ never comes back} \mid I] = \sum_{t\in[T]} \sum_{e\in E_u} x_{e,t} q_e \Pr[C_e \geq T - t + 1]$. Hence, we have $\sum_{t\in[T]} \sum_{e\in E_u} (x_{e,t}(1-q_e) + x_{e,t} q_e \Pr[C_e \geq T-t+1]) \leq \Delta_u$, and (2) holds.

The third constraint (3) is satisfied because the expected number of assignments made for an online vertex v is at most $p_{v,t} b_v$. Therefore, the solution x defined from offline optimal algorithms is feasible to the LP. □

Note that the LP (Off) is a non-trivial extension of those in the previous papers (Dickerson et al. 2021; Nanda et al. 2020). The constraints on rejections can be naturally expressed as $\sum_{t\in[T]} \sum_{e\in E_u} x_{e,t}(1 - q_e) \leq \Delta_u$, similarly to the offline LP in (Nanda et al. 2020). However, this is not enough to show our result, and we need a stronger formulation (2) of the constraints.

Example 1. *Let us consider the following instance. Let* $U = \{u\}$, $V = \{v_1, v_2, v_3\}$ *and* $E = \{(u,v) \mid v \in V\}$. *Let also* $T = 3$, $\Delta = 1$ *and* $\varepsilon \ll 1$. *Assume that, at time* t, *only a vertex* v_t *can arrive, whose probability is* p_t. *For our notational convenience, we identify an edge* (u, v_t) *with* t. *Let* $p_1 = p_2 = 1$, $p_3 = \varepsilon$, $q_1 = q_2 = 1/2$, $q_3 = 1$, $w_1 = 4/9$, $w_2 = 6/9$, *and* $w_3 = 4/(9\varepsilon)$. *We set* $\Pr[C_t = 1] = 1/2$, $\Pr[C_t = 2] = 1/2$, $\Pr[C_t = 3] = 0$ *for all* t.

We demonstrate the calculation of the expected profit for an adaptive online algorithm. Consider an adaptive algorithm that we always assign an arriving vertex to u *if possible. Then, at time* 1, *we obtain a profit* $\frac{4}{9} \cdot \frac{1}{2} = \frac{2}{9}$ *in expectation. The probability that* u *is available at time* 2 *is* $q_1 \Pr[C_t = 1] = \frac{1}{4}$ *because it is equal to the probability that* u *accepts* v_1 *and* C_1 *takes* 1. *Thus the probability that* v_2 *is assigned to* u *at time* 2 *is* $\Pr[(u \text{ is available at time } 2) \wedge (v_2 \text{ arrives at time } 2)] = \frac{1}{4}$. *Hence the expected profit at*

time 2 is $\frac{6}{9} \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{1}{12}$. *By a similar discussion, the expected profit at time 3 is* $\frac{5}{36}$. *Therefore, the expected profit in total is* $\frac{2}{9} + \frac{1}{12} + \frac{5}{36} = \frac{4}{9}$.

The offline optimal value can be calculated as follows. When we know v_3 *arrives at time 3 in advance, the best strategy is that we discard* v_1 *and* v_2 *but assign* v_3 *to obtain a profit* $\frac{4}{9\varepsilon}$. *On the other hand, suppose that* v_3 *does not arrive and we know it. We describe that the expected profit when we assign* v_1 *at time 1 is at most* $\frac{11}{36}$. *We obtain the expected profit* $q_1 w_1 = \frac{2}{9}$ *from* v_1. *Since* v_2 *is the last vertex to arrive, we should assign* v_2 *if* u *is available at time 2. The probability that* u *is available at time 2 is* $q_1 \Pr[C_t = 1] = \frac{1}{4}$ *because it is equal to the probability that* u *accepts* v_1 *and* C_1 *takes 1. Thus the probability that* v_2 *is assigned to* u *at time 2 is* $\Pr[(u \text{ is available at time } 2) \wedge (v_2 \text{ arrives at time } 2)] = \frac{1}{4}$. *Hence the expected profit at time 2 is* $\frac{6}{9} \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{1}{12}$. *Therefore, the expected profit in total is* $\frac{2}{9} + \frac{1}{12} = \frac{11}{36}$. *On the other hand, the expected profit obtained when we discard* v_1 *is* $\frac{1}{3}$, *and thus we see that it is better to discard* v_1. *Since the probability that* v_3 *arrives is* ε, *the offline optimal value is* $\varepsilon \cdot \frac{4}{9\varepsilon} + (1-\varepsilon) \cdot \frac{3}{9} = \frac{7-3\varepsilon}{9}$.

Let us see a corresponding feasible solution x *to the offline LP. We denote* $x_{uv_t,t} = x_t$ *for* $t = 1, 2, 3$ *for simplicity. Then as in the proof of Lemma 1,* x_t *corresponds to the probability that* v_t *is assigned to* u. *Since* v_1 *is not chosen in any offline optimal algorithm,* x_1 *is 0. Moreover,* x_2 *is* $\varepsilon \cdot 0 + (1-\varepsilon) \cdot 1 = 1 - \varepsilon$ *since we choose* v_2 *if* v_3 *does not arrive. Finally,* x_3 *is* $\varepsilon \cdot 1 + (1-\varepsilon) \cdot 0 = \varepsilon$. *The LP objective value for this solution* x *is* $\frac{2}{9}x_1 + \frac{3}{9}x_2 + \frac{4}{9\varepsilon}x_3 = \frac{7-3\varepsilon}{9}$.

2.2 Competitive Ratio

We evaluate the performance of an online algorithm by a competitive ratio. Let $\mathrm{ALG}(I)$ be the expected profit of an online algorithm ALG when the input sequence is I. We say that an online algorithm is *α-competitive* if $\mathbb{E}_{I \sim \mathcal{I}}[\mathrm{ALG}(I)] \geq \alpha \mathbb{E}_{I \sim \mathcal{I}}[\mathrm{OPT}(I)]$ for any instance.

3 Proposed Algorithm

In this section, we present our algorithm, and then analyze the competitive ratio in subsequent subsections.

The overview of our proposed algorithm is described as follows. We first find an optimal solution x^* to LP (Off). Then we use $x^*_{e,t}$ to determine a probability that v comes at time t and we assign u to v where $e = (u, v)$. That is, we choose a set S of at most b_v vertices in U so that $\Pr[u \in S, v \text{ arrives at time } t] = x^*_{e,t}$. Then, for each $u \in S$, we compute the expected profit earned at and after t by assigning u to v, and, if it is larger than the one by not assigning, then the algorithm tries to assign u to v.

Our algorithm is summarized in Algorithm 1. We now explain how to implement each step in more detail.

Finding a Set of Offline Vertices We first explain how to design a probabilistic distribution to find a set of offline vertices when v arrives at time t. We note that it is easy when $b_v = 1$, as we can just choose a vertex u in U with probability $x^*_{uv,t}/p_{v,t}$. However, when $b_v \geq 2$, an independently random choice of offline vertices may violate the feasibility constraint. To avoid it, we construct a probability distribution over feasible vertex sets and choose a feasible vertex set according to the distribution.

Let $\mathcal{S}_v = \{S \subseteq E_v \mid |S| \leq b_v\}$. For $S \in \mathcal{S}_v$, its characteristic vector is a vector $\chi_S \in \{0,1\}^{E_v}$ such that $(\chi_S)_e = 1$ if and only if $e \in S$. Consider the convex hull P_v of all characteristic vectors χ_S ($S \in \mathcal{S}_v$). Then the convex hull coincides with the polytope $\{y \in [0,1]^{E_v} \mid \sum_{e \in E_v} y_{uv} \leq b_v\}$. In addition, every vertex in P_v is an integral vector, which corresponds to a characteristic vector χ_S for some $S \in \mathcal{S}_v$.

For an optimal solution x^* of LP (Off), define $y^{v,t} = (x^*_{e,t}/p_{v,t})_{e \in E_v}$. Then we can see that $y^{v,t} \in P_v$ since x^* satisfies (3). It follows from well-known Carathéodory theorem that $y^{v,t}$ can be decomposed as a convex combination of at most $|E_v| + 1$ vertices in P_v. Since every vertex in P_v corresponds to a characteristic vector of a feasible set, there exist $S^v_k \in \mathcal{S}$ and $\lambda^{v,t}_k$ ($k = 1, \ldots, |E_v| + 1$) such that

$$y^{v,t} = \textstyle\sum_{k=1}^{|E_v|+1} \lambda^{v,t}_k \chi_{S^v_k}, \tag{5}$$

where $\sum_k \lambda^{v,t}_k = 1$ and $\lambda^{v,t}_k \geq 0$ for any $k \in [|E_v| + 1]$.

We regard $\lambda^{v,t}_k$ ($k = 1, \ldots, |E_v| + 1$) as a probability distribution over $\mathcal{S}_v$. That is, we choose a set $S_k \in \mathcal{S}_v$ with probability $\lambda^{v,t}_k$. Then the probability that u is chosen when v arrives at t is $\sum_{k: u \in S^v_k} \lambda^{v,t}_k = y^{v,t}_{uv} = \frac{x^*_{uv,t}}{p_{v,t}}$.

We note that $\lambda^{v,t}_k$'s can be obtained in polynomial time by the constructive proof of Carathéodory theorem. See the full version for the detailed description.

Remark 1. *In the above, we use only the fact that a subset of a feasible set is also feasible. Thus our algorithm would work even for more general constraints. We note, however, that it is required to solve linear programming problem over the convex hull* P_v *with constraints* (1), (2), *and* (4). *This implies that, to run our algorithm in polynomial time, we need to describe* P_v *efficiently. An example is a matroid constraint, that is, each online vertex* v *has a matroid on the ground set* U *and* v *chooses an independent set of the matroid. We note that a cardinality constraint is a special case of a matroid constraint.*

Assigning Offline Vertices We next describe how to decide whether we assign each online vertex $u \in S^v_k$ to v.

Let $R^d_{u,t}$ be the expected profit that u earns at and after time t, when u has a remaining budget d. By definition, $R^0_{u,t} = 0$. Also, for our notational convenience, we assume that $R^d_{u,t} = 0$ for all $d > T$. If we assigned an online vertex v to u with remaining budget $d > 1$ at time t, then the expected profit that u earns at and after t would be

$$Q^d_{e,t} := q_e \left(w_e + \textstyle\sum_{\ell=1}^{T-t} \Pr[C_e = \ell] R^d_{u,t+\ell} \right) + (1 - q_e) R^{d-1}_{u,t+1}.$$

Otherwise, the expected profit will be $R^d_{u,t+1}$.

In our algorithm, for each vertex $u \in S^v_k$, we compare $Q^d_{uv,t}$ and $R^d_{u,t+1}$. If the former is larger, then this means that assigning u to v makes larger profit than not assigning, and thus we assign u to v. Otherwise, i.e., if assigning u to v does not make enough profit, then we do not as-

Algorithm 1: Proposed online algorithm

Solve LP (Off) to obtain an optimal solution x^*
for $t = 1, \dots, T$ **do**
  Let v be a vertex that arrives at time t (if none, skip)
  Choose S_k^v with probability $\lambda_k^{v,t}$ by (5)
  for $u \in S_k^v$ **do**
    Let d be the remaining budget of u
    if $Q_{uv,t}^d \geq R_{u,t+1}^d$ and u is available **then**
      Assign u to v
    else
      Do nothing for u

sign u to v. Thus the expected profit for a vertex $u \in S_k^v$ is $\max\left\{Q_{uv,t}^d, R_{u,t+1}^d\right\}$.

Since the probability that v arrives at time t and u is chosen by v is $x_{e,t}^*$ for each $e = (u, v) \in E_v$ and $t \in [T]$, $R_{u,t}^d$ can be represented recursively by

$$R_{u,t}^d = \sum_{e \in E_u} x_{e,t}^* \max\left\{Q_{e,t}^d, R_{u,t+1}^d\right\} + \left(1 - \sum_{e \in E_u} x_{e,t}^*\right) R_{u,t+1}^d. \tag{6}$$

Our algorithm needs to compute $R_{u,t}^d$ for each d, $u \in U$ and $t \in [T]$. This can be done efficiently in advance by dynamic programming with the above recursive equation (6).

3.1 Modifying Instance

We first observe that the total expected profit of the algorithm is equal to $\sum_{u \in U} R_{u,1}^{\Delta_u}$. Since LP (Off) gives an upper bound on $\mathbb{E}_{I \sim \mathcal{I}}[\mathrm{OPT}(I)]$ by Lemma 1, we aim to determine the ratio between $\sum_{u \in U} R_{u,1}^{\Delta_u}$ and the LP optimal value $\sum_{u \in U} \sum_{e \in E_u} \sum_t w_e q_e x_{e,t}^*$. To this end, we fix a vertex u in U, and we evaluate the ratio α_u between $R_{u,1}^{\Delta_u}$ and $\sum_{e \in E_u} \sum_t w_e q_e x_{e,t}^*$. Then the competitive ratio of Algorithm 1 is at least $\min_u \alpha_u$.

To obtain a lower bound of $R_{u,1}^{\Delta_u}$, we modify a given instance to a simpler one with LP optimal solution x^*. We will show that the expected profit for the modified instance gives a lower bound on that for the original instance.

The new instance is defined as follows. We define $G' = (U', V'; E')$ where $U' = \{u\}$, $V' = \{v_t \mid t \in [T]\}$, and $E' = \{(u, v_t) \mid t \in [T]\}$. In this instance, we have only one offline vertex u, and, at each $t \in [T]$, only one vertex v_t may arrive with edge (u, v_t). We set the parameters[4] for v_t in the new instance as follows:

- the probability of arrival: $p_t' = \sum_{e \in E_u} x_{e,t}^*$;
- the probability of acceptance: $q_t' = \sum_{e \in E_u} x_{e,t}^* q_e / p_t'$ if $p_t' > 0$, and $q_t' = 0$ otherwise;
- the profit: $w_t' = \sum_{e \in E_u} x_{e,t}^* q_e w_e / (p_t' q_t')$ if $p_t' q_t' > 0$ and 0 otherwise;
- the distribution of occupation time (denoted by C_t):

$$\Pr[C_t = \ell] = \frac{\sum_{e \in E_u} x_{e,t}^* q_e \Pr[C_e = \ell]}{p_t' q_t'} \quad (\ell \in [T])$$

if $p_t', q_t' > 0$, and otherwise, $\Pr[C_t = T] = 1$;

- the capacity: $b_t' = 1$.

The budget Δ_u of u is set the same value as in the original instance. Since x^* is a feasible solution to LP (Off), it holds that $p_t' \leq 1$ and $q_t' \leq 1$ by (4), and moreover,

$$\sum_{\ell=1}^{T} \Pr[C_t = \ell] = \frac{\sum_{e \in E_u} x_{e,t}^* q_e \sum_\ell \Pr[C_e = \ell]}{p_t' q_t'} = 1.$$

Note also that $\sum_{t \in [T]} w_t' q_t' p_t' = \sum_{e \in E_u} \sum_t w_e q_e x_{e,t}^*$.

Let us execute Algorithm 1 for the modified instance, where we use p_t' instead of LP optimal solution x^*. Let $\tilde{R}_t^d$ be the expected profit that u earns on and after t when u has a remaining budget d at time t. Similarly to (6) for $R_{u,t}^d$, it holds that

$$\tilde{R}_t^d = p_t' \max\left\{Q_t^d, \tilde{R}_{t+1}^d\right\} + (1 - p_t')\tilde{R}_{t+1}^d, \tag{7}$$

where

$$Q_t^d = q_t' \left(w_t' + \sum_{\ell=1}^{T-t} \Pr[C_t = \ell] \tilde{R}_{t+\ell}^d\right) + (1 - q_t')\tilde{R}_{t+1}^{d-1}. \tag{8}$$

We show that the modification does not increase the profit.

Lemma 2. *$R_{u,t}^d \geq \tilde{R}_t^d$ for all $d \in [\Delta_u] \cup \{0\}$ and $t \in [T]$.*

By this lemma, it suffices to lower-bound $\tilde{R}_1^{\Delta_u}$ to obtain a lower bound on $R_{u,1}^{\Delta_u}$ for the original instance.

3.2 Analysis for the Unlimited Rejection Case

In this section, we assume that the number of allowed rejections is unlimited, that is, $\Delta_u = +\infty$ for $u \in U$. This means that we can ignore the constraint (2) in LP (Off). The main result of this section is to prove that Algorithm 1 is $1/2$-competitive for this case. For that purpose, we first modify the instance as in the previous section, and we will bound $\tilde{R}_1^{\Delta_u}$ from below.

We observe that, when executing the algorithm for the modified instance, the remaining budget d of the vertex u is infinite in the whole process. For simplicity, we denote $R_t = \tilde{R}_t^\infty$ and $Q_t = Q_t^\infty$. Then, by (8), it holds that

$$Q_t = q_t' \left(w_t' + \sum_{\ell=1}^{T-t} \Pr[C_t = \ell] R_{t+\ell}\right) + (1 - q_t')R_{t+1}.$$

Moreover, it follows from (7) that

$$R_t = \max\left\{p_t' Q_t + (1 - p_t')R_{t+1}, R_{t+1}\right\}. \tag{9}$$

We note that $R_T = p_T' q_T' w_T'$. We can make the first term in (9) simpler as $B_t w_t' + A_t R_{t+1} + B_t \sum_{\ell=2}^{T-t} \Pr[C_t = \ell] R_{t+\ell}$, where $B_t = p_t' q_t'$ and $A_t = p_t' q_t' \Pr[C_t = 1] + p_t'(1 - q_t') + (1 - p_t')$ for each $t \in [T]$. We note that $A_t = p_t' q_t' \Pr[C_t = 1] + 1 - p_t' q_t' = 1 - B_t \Pr[C_t \geq 2]$.

A lower bound on R_1 is obtained by minimizing R_1 subject to the condition (9). The minimization problem can be

[4] For ease of notation, we use simpler subscripts, e.g., p_t' instead of $p_{uv_t,t}$, as we have only one online vertex at time t.

formulated as a linear programming problem as below. Note that we do not need to solve the LP, but use it for analysis.[5]

$$\begin{aligned}
\min \quad & R_1 \\
\text{s.t.} \quad & R_t \geq B_t w'_t + A_t R_{t+1} \\
& \qquad + B_t \textstyle\sum_{\ell=2}^{T-t} \Pr[C_t = \ell] R_{t+\ell} \quad (t \in [T-1]) \\
& R_t \geq R_{t+1} \quad (t \in [T-1]) \\
& R_T \geq B_T w'_T \\
& R_t \geq 0 \quad (t \in [T]),
\end{aligned} \tag{10}$$

Then the optimal value of the above LP gives a lower bound of R_1. The dual of LP (10) is given by

$$\begin{aligned}
\max \quad & \textstyle\sum_{t=1}^{T} B_t w'_t \alpha_t \\
\text{s.t.} \quad & \alpha_1 + \beta_1 \leq 1 \\
& \alpha_2 + \beta_2 \leq A_1 \alpha_1 + \beta_1 \\
& \alpha_t + \beta_t \leq \textstyle\sum_{\ell=1}^{t-2} \alpha_\ell B_\ell \Pr[C_\ell = t - \ell] \\
& \qquad + A_{t-1} \alpha_{t-1} + \beta_{t-1} \quad (3 \leq t \leq T-1) \\
& \alpha_T \leq \textstyle\sum_{\ell=1}^{T-2} \alpha_\ell B_\ell \Pr[C_\ell = T - \ell] \\
& \qquad + A_{T-1} \alpha_{T-1} + \beta_{T-1} \\
& \alpha_t \geq 0 \quad (t \in [T]) \\
& \beta_t \geq 0 \quad (t \in [T-1]).
\end{aligned} \tag{11}$$

To show a lower bound on LP (10), we construct a feasible solution to the dual LP (11). Let $\gamma \leq 1/2$. We set $\alpha_t = \gamma$ for $t \in [T]$, $\beta_1 = 1 - \gamma$, $\beta_2 = \beta_1 - \gamma + A_1 \gamma$, and $\beta_t = \beta_{t-1} - \gamma + A_{t-1}\gamma + \sum_{\ell=1}^{t-2} B_\ell \Pr[C_\ell = t - \ell]\gamma$ $(3 \leq t \leq T-1)$.

Lemma 3. *For $0 \leq \gamma \leq 1/2$, α_t and β_t defined as above are feasible to* (11)*.*

Therefore, α_t and β_t defined as above are feasible to (11). The objective value for this solution is $\sum_t B_t w'_t \gamma = \gamma \sum_t p'_t q'_t w'_t$. It follows from the LP duality theorem that the optimal value of LP (10) is at least $\gamma \sum_t p'_t q'_t w'_t$. This is maximized when $\gamma = 1/2$, and in this case, R_1 is lower-bounded by $\frac{1}{2} \sum_t p'_t q'_t w'_t$. Since $\sum_t p'_t q'_t w'_t = \sum_t \sum_{e \in E_u} w_e q_e x^*_{e,t}$, we obtain $R_1 \geq \frac{1}{2} \sum_t \sum_{e \in E_u} w_e q_e x^*_{e,t}$.

Summarizing, we have $\sum_{u \in U} R_1^{\Delta_u} \geq \frac{1}{2} \mathbb{E}[\mathrm{OPT}(I)]$ by Lemma 1, which implies the following theorem.

Theorem 1. *Algorithm 1 is $1/2$-competitive for the problem with unlimited rejections.*

3.3 Analysis for the Limited Rejection Case

In this subsection, we prove that Algorithm 1 is $\Delta/(3\Delta - 1)$-competitive for the general case with $\Delta = \max_{u \in U} \Delta_u$. As in the analysis for the unlimited rejection case (Section 3.2), we present a lower bound on $\tilde{R}^d_t$.

We first show the following lemma.

Lemma 4. *$\tilde{R}^{d-1}_t \geq \frac{d-1}{d} \tilde{R}^d_t$ for each $d \in [\Delta_u]$ and $t \in [T]$.*

By (7) with the above lemma, we have the following relationships:

$$\tilde{R}^d_t \geq \max \left\{ p_t \hat{Q}^d_t + (1 - p_t) \tilde{R}^d_{t+1}, \tilde{R}^d_{t+1} \right\}, \tag{12}$$

where $\hat{Q}^d_t = q_t (w_t + \sum_{\ell \geq 1} \Pr[C_t = \ell] \tilde{R}^d_{t+\ell}) + \frac{d-1}{d}(1 - q_t) \tilde{R}^d_{t+1}$.

Recall that it suffices to give a lower bound on $\tilde{R}_1^{\Delta_u}$ to obtain the competitive ratio of Algorithm 1. By (12), we can construct a linear program to bound $\tilde{R}_1^{\Delta_u}$ in a similar way to the previous subsection. See the full version for the details.

Theorem 2. *Algorithm 1 is a $\Delta/(3\Delta - 1)$-competitive algorithm, where $\Delta = \max_{u \in U} \Delta_u$.*

We also show that our algorithm can be $\frac{1}{2}\left(1 - \frac{1}{e^2}\right)$-competitive in the KIID model with non-reusable agents.

Theorem 3. *There is a $(\frac{1}{2 - 1/\Delta}\left(1 - \frac{1}{e^{2-1/\Delta}}\right))$-competitive algorithm for the problem in the KIID model with non-reusable resources, where $\Delta = \max_{u \in U} \Delta_u$.*

Note that $\frac{1}{2-1/\Delta}\left(1 - \frac{1}{e^{2-1/\Delta}}\right) \geq \frac{1}{2}\left(1 - \frac{1}{e^2}\right) > \frac{1}{e}$. Thus, our algorithm has a better competitive ratio than the $\frac{1}{e}$-competitive algorithm by Nanda et al. (2020).

4 Experiments

In this section, we present experimental results to evaluate the performance of Algorithm 1. We describe the experimental environment in the full version.

Instance Setting We focus on the following four settings (a)–(d); see the full version for the other settings. The setting (a) is the KIID model with non-reusable agents (i.e., $\Pr[C_e = T] = 1$ for all $e \in E$) and $\Delta_u < T$ $(\forall u \in U)$. This coincides with that of Nanda et al. (2020). The settings (b)–(d) are the KAD model with reusable agents. In the setting (b), which is that of Dickerson et al. (2021), offline vertices always accept assignments (i.e., $q_e = 1$ for all $e \in E$), while they may reject in (c) and (d). We set $\Delta_u < T$ $(\forall u \in U)$ in (c) and $\Delta_u = +\infty$ in (d). For each setting, we generate one instance and 1000 input sequences, and focus on the average profits and runtime for evaluation.

Baseline Algorithms We compare our algorithm with the following baseline algorithms: (1) Random chooses offline vertices uniformly at random. (2) Greedy assigns at most b_v available offline vertices u to v in the order of $w_{(u,v)} q_{(u,v)}$. (3) NAdap* is a simple extension of NAdap proposed by Nanda et al. (2020). NAdap* solves LP (`Off`) and to assign a set of offline vertices with probability $\lambda_k^{v,t}$. When $b_v = 1$ for all v, NAdap* coincides with NAdap in the setting (a), and is also used in experiments in (Dickerson et al. 2021). In our experiments, we do not evaluate the algorithm by Dickerson et al. (2021) due to its impractical assumption.

4.1 Synthetic Dataset

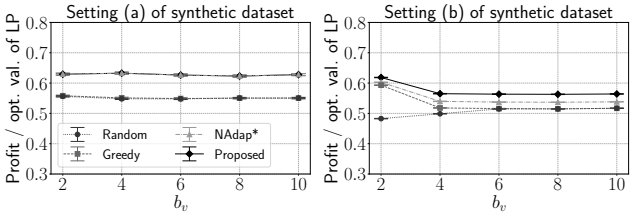
We generate synthetic datasets similarly to (Nanda et al. 2020) as follows. We set $|U| = 30$, $|V| = 100$, and $T = 200$. For each $u \in U$ and $v \in V$, an edge (u, v) exists in E with probability 0.1. For each $e \in E$, we set $q_e \sim U(0.5, 1)$ (uniform distribution) and $w_e \sim U(0, 1)$, respectively. For each $u \in U$ and $e \in E_u$, the distribution of C_e for reusability is defined as a binomial distribution $B(20, \eta_u)$, where $\eta_u \sim U(0, 1)$. For the settings (a) and (c),

[5] Such LP is called a factor-revealing LP.

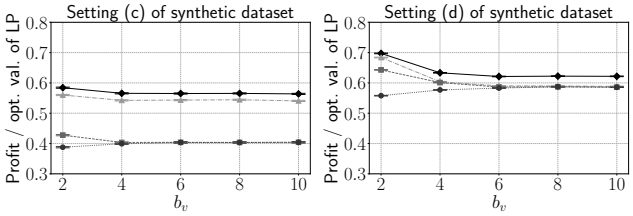
Δ_u is drawn uniformly at random from $\{1, 2, 3\}$ for each $u \in U$. We set the same b_v for all v from $\{2, 4, 6, 8, 10\}$.

Figure 1 plots the ratios between average profits and the LP optimal values. We see that our algorithm (the black solid line) performs the best in any cases. Moreover, it obtains more than half of the LP optimal value, while Random and Greedy sometimes fail. NAdap* performs almost the same as us in (a), but is worse in (b)–(d).

All the algorithms run within less than 1 second for processing online vertices. Our algorithm needs additionally around 6 seconds on average because of the preprocess of solving LP (`Off`) and computing the table of $R^d_{u,t}$'s. For the detailed results, see the full version.



(a) KIID without reusability (b) KAD without rejection



(c) KAD with $\Delta_u \leq 3$ (d) KAD with $\Delta_u = +\infty$

Figure 1: The rates of average profits for synthetic datasets.

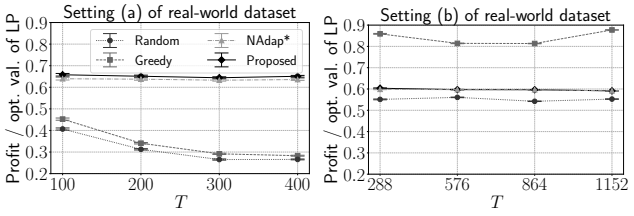
4.2 Real-world Dataset

We evaluate the performance for instances generated from the New York City yellow cabs dataset[6], which is used also in existing work such as (Dickerson et al. 2021; Nanda et al. 2020). Using the data in January 2013 we set up a bipartite graph with $|U| = 30$ and $|V| = 100$ and parameters similarly to (Nanda et al. 2020). The detailed set-up is described in the full version. We set $T = 100k$ ($k \in [4]$) in setting (a) and $T = 288k$ ($k \in [4]$) in others.

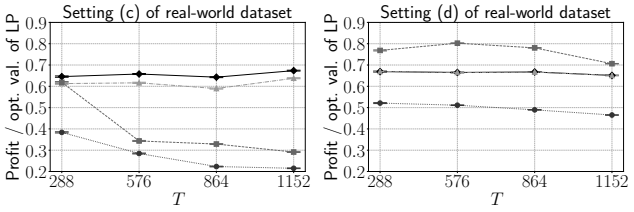
Figure 2 shows the ratios between average profits and the LP optimal values. Note that our algorithm earns more than half of the LP optimal value in any cases. In settings (a) and (c), Random and Greedy cannot obtain even half of the LP optimal value for a large T. NAdap* performs as well as ours in settings except (c), in which it performs worse than our algorithm. Our algorithm may be too careful in (b) and (d), in which there is no need to care about the rejection constraint, and has less performance. On the other hand, since offline vertices easily become unavailable in (a) and (c), our algorithm, which considers a long-term effect of the current assignment, performs the best.

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

All the algorithms run in less than 1 second for processing even 1152 online vertices. The preprocess in our algorithm completes in 400 seconds. Since the preprocess is done before arrival of online vertices, our algorithm makes decision as fast as others. For the detailed results, see the full version.



(a) KIID without reusability (b) KAD without rejection



(c) KAD with $\Delta_u \leq 3$ (d) KAD with $\Delta_u = +\infty$

Figure 2: The rates of average profits for real-world datasets.

5 Conclusion

In this paper, we studied the online task assignment problem with reusable resources in the KAD model, which generalizes the online bipartite matching in the KAD model. Our problem incorporates practical conditions arising in applications such as ridesharing and crowdsourcing. We proposed an online algorithm that is $1/2$-competitive for the unlimited rejection case, which is tight, and $\Delta/(3\Delta - 1)$-competitive for the general case. Practical usefulness of our algorithm is confirmed by numerical experiments.

For future work, as solving LP (`Off`) is time-consuming, it would be interesting to obtain a constant competitive ratio without solving the LP. One direction is to use an approximate solution to the LP in our algorithm. Another future work is to consider a model that, when a task is rejected, it is allowed to re-assign some other agents.

Acknowledgments

We would like to thank the AAAI anonymous reviewers for valuable comments to improve the paper. HS was supported by JSPS KAKENHI Grant Numbers JP17K12646, JP21K17708, and JP21H03397, Japan. SI was supported by JST, ACT-I, Grant Number JPMJPR18U5, Japan. TF was supported by JSPS KAKENHI Grant Numbers JP20H05965, JP21K11759, and JP21H03397, Japan. NK was supported by JSPS KAKENHI Grant Numbers JP20H05795, JP18H05291, and JP21H03397, Japan. KK was supported by JSPS KAKENHI Grant Number JP18H05291, Japan.

References

Alaei, S.; Hajiaghayi, M.; and Liaghat, V. 2012. Online Prophet-Inequality Matching with Applications to Ad Allocation. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC)*, 18–35.

Anagnostopoulos, A.; Castillo, C.; Fazzone, A.; Leonardi, S.; and Terzi, E. 2018. Algorithms for Hiring and Outsourcing in the Online Labor Market. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 1109–1118.

Assadi, S.; Hsu, J.; and Jabbari, S. 2015. Online Assignment of Heterogeneous Tasks in Crowdsourcing Markets. In *Proceedings of the 3rd AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 12–21.

Birnbaum, B.; and Mathieu, C. 2008. On-Line Bipartite Matching Made Simple. *SIGACT News*, 39(1): 80–87.

Brubach, B.; Sankararaman, K. A.; Srinivasan, A.; and Xu, P. 2016. New Algorithms, Better Bounds, and a Novel Model for Online Stochastic Matching. In *Proceedings of the 24th Annual European Symposium on Algorithms (ESA)*, volume 57, 24:1–24:16.

Devanur, N. R.; and Hayes, T. P. 2009. The Adwords Problem: Online Keyword Matching with Budgeted Bidders under Random Permutations. In *Proceedings of the 10th ACM Conference on Electronic Commerce (EC)*, 71–78.

Dickerson, J. P.; Sankararaman, K. A.; Sarpatwar, K. K.; Srinivasan, A.; Wu, K.-L.; and Xu, P. 2019. Online Resource Allocation with Matching Constraints. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1681–1689.

Dickerson, J. P.; Sankararaman, K. A.; Srinivasan, A.; and Xu, P. 2021. Allocation Problems in Ride-Sharing Platforms: Online Matching with Offline Reusable Resources. *ACM Transactions on Economics and Computation*, 9(3).

Dong, Z.; Das, S.; Fowler, P.; and Ho, C.-J. 2021. Efficient Nonmyopic Online Allocation of Scarce Reusable Resources. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 447–455.

Haeupler, B.; Mirrokni, V. S.; and Zadimoghaddam, M. 2011. Online Stochastic Weighted Matching: Improved Approximation Algorithms. In *Proceedings of the 7th International Workshop on Internet and Network Economics (WINE)*, 170–181.

Ho, C.-J.; and Vaughan, J. W. 2012. Online Task Assignment in Crowdsourcing Markets. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, 45–51.

Karp, R. M.; Vazirani, U. V.; and Vazirani, V. V. 1990. An Optimal Algorithm for On-Line Bipartite Matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, 352–358.

Krengel, U.; and Sucheston, L. 1977. Semiamarts and finite values. *Bulletin of the American Mathematical Society*, 83(4): 745–747.

Krengel, U.; and Sucheston, L. 1978. On semiamarts, amarts, and processes with finite value. *Probability on Banach spaces*, 4: 197–266.

Lowalekar, M.; Varakantham, P.; and Jaillet, P. 2020. Competitive Ratios for Online Multi-Capacity Ridesharing. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 771–779.

Manshadi, V. H.; Gharan, S. O.; and Saberi, A. 2012. Online Stochastic Matching: Online Actions Based on Offline Statistics. *Mathematics of Operations Research*, 37(4): 559–573.

Mehta, A. 2013. Online Matching and Ad Allocation. *Foundations and Trends in Theoretical Computer Science*, 8(4): 265–368.

Mehta, A.; and Panigrahi, D. 2012. Online matching with stochastic rewards. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, 728–737.

Mehta, A.; Saberi, A.; Vazirani, U.; and Vazirani, V. 2007. AdWords and Generalized Online Matching. *Journal of the ACM*, 54(5): 22:1–22:19.

Mehta, A.; Waggoner, B.; and Zadimoghaddam, M. 2015. Online Stochastic Matching with Unequal Probabilities. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1388–1404.

Nanda, V.; Xu, P.; Sankararaman, K. A.; Dickerson, J.; and Srinivasan, A. 2020. Balancing the Tradeoff between Profit and Fairness in Rideshare Platforms during High-Demand Hours. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 2210–2217.

Xu, P.; Srinivasan, A.; Sarpatwar, K. K.; and Wu, K.-L. 2017. Budgeted Online Assignment in Crowdsourcing Markets: Theory and Practice. In *Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1763–1765.