Online Task Assignment with Controllable Processing Time

Ruoyu Wu , Wei Bao , Liming Ge
School of Computer Science, The University of Sydney
ruwu6940@uni.sydney.edu.au, {wei.bao, liming.ge}@sydney.edu.au

Abstract

We study a new online assignment problem, called the *Online Task Assignment with Controllable Processing Time*. In a bipartite graph, a set of online vertices (tasks) should be assigned to a set of offline vertices (machines) under the known adversarial distribution (KAD) assumption. We are the first to study controllable processing time in this scenario: There are multiple processing levels for each task and higher level brings larger utility but also larger processing delay. A machine can reject an assignment at the cost of a rejection penalty, taken from a pre-determined rejection budget. Different processing levels cause different penalties. We propose the Online Machine and Level Assignment (OMLA) Algorithm to simultaneously assign an offline machine and a processing level to each online task. We prove that OMLA achieves $1/2$-competitive ratio if each machine has unlimited rejection budget and $\Delta/(3\Delta - 1)$-competitive ratio if each machine has an initial rejection budget up to Δ. Interestingly, the competitive ratios do not change under different settings on the controllable processing time and we can conclude that OMLA is "insensitive" to the controllable processing time.

1 Introduction

In this paper, we study an online task assignment problem with controllable processing time. In this problem, we have a set of online vertices (tasks) and a set of offline vertices (machines). Online tasks arrive sequentially and each can be processed by a machine, but each task can only be processed by a subset of machines [Dickerson *et al.*, 2021; Sumita *et al.*, 2022]. We focus on the controllable processing time: Each task has multiple levels of processing time [Wang *et al.*, 2019; Shabtay and Steiner, 2007]. If a task is processed with a higher level, it obtains a higher reward, but needs to wait for a longer time. Each machine can only process one task at one time, and can only take another after the previous one is finished. The machines do not always accept task assignment, and they lose an amount of rejection budget every time they reject. The higher processing level of the rejected assignment is, the larger amount of the budget is taken. When a machine runs out of its rejection budget, it will be removed from the system immediately and permanently. In this paper, we consider the online tasks arriving under *known adversarial distributions* (KAD) [Dickerson *et al.*, 2021; Tong *et al.*, 2020]. The arrival probability of each task at each time is known ahead. The goal is to maximise the expected reward without the knowledge of future task arrivals (online setting).

Although a number of works have studied the online assignment problem, this paper is the first to consider controllable processing time. This is motivated by real-world scenarios in various fields, such as:

1. Task offloading in edge computing [Mahesh, 2020]. With the help of edge computing, end user devices can offload computing intensive tasks to edge computers, especially the ones processing machine learning (ML) models. Each user (task) has only a set of edge computers (machines) near them, so the tasks from this user can only be processed by these computers. When we assign an edge computer to a task, we can further control the processing time by implementing different ML models. A better processing quality requires a model with longer processing time, but a lightweight model (e.g., a pruned and sparsified model) finishes sooner, with lower accuracy.

2. Ride-sharing with tolls [Andani *et al.*, 2021]. Ride-sharing system assigns passenger requests (tasks) to available drivers (machines). Each request has an origin and can only be processed by drivers near this area. For each ride, we can choose to go through toll roads (high cost) for a shorter processing time, or to avoid toll for less cost.

3. Translation service [Taia, 2022]. Customers place orders (tasks) to the language service agencies (machines), and the agency can provide different degrees of service (e.g., one-off translation, translation and proofreading, etc.). Each translation request has a target language and can only be processed by translators with a certification of this language. The agency can choose to assign more time for a translation task, resulting in a higher processing quality; but the agency can also assign a shorter processing time to a task, so that more customer requests can be processed.

In many situations, because the arrival of tasks can only be known when they arrive, online algorithms are required. We are motivated to design such an online algorithm that can maximize the worst ratio against the offline optimal perfor-

mance (competitive ratio). In short, the online assignment problem studied in this paper has the following features:

A. Known adversarial distribution (KAD): the probability of the arrival of each task at each time is known in advance.

B. Reusable machines: the machine returns to the system after completing a task; the processing delay is drawn from known distributions.

C. Controllable processing time: each task can be processed with different levels; a higher processing level generates a higher reward but the expected delay is also higher.

D. Budgeted machine rejections: when a machine is assigned, it can reject the assignment with a penalty; rejecting a higher processing level task will cause higher penalty. When the budget runs out, the machine is permanently removed from the system.

The main contribution of this paper is designing an Online Machine and Level Assignment (OMLA) Algorithm for the above problem, especially with multiple processing levels. We prove that our algorithm achieves a $1/2$-competitive ratio when every machine has an infinite rejection budget, and a $\Delta/(3\Delta - 1)$-competitive ratio when each machine has a finite rejection budget, where Δ is the largest budget of machines at the beginning. The conclusion shows that regardless of the limited rejection budgets, the competitive ratio does not depend on the processing levels, indicating that OMLA is insensitive to controllable processing time.

Controllable processing time makes the problem studied in the paper more realistic but also introduces substantially more challenges to the online algorithm design and competitive analysis. Controllable processing time expands the searching space. Since each processing level causes different rewards, delays, and rejection budgets. The algorithm should balance these dimensions as a result of coupled objective and constraints. To tackle this challenge, in our online algorithm design, we first use the joint probabilities of choosing a machine and a level as decision variables to formulate an offline linear programming (LP). The optimal solution to the LP is then leveraged to calculate the activation value and the baseline value for each machine and level. These two values will determine our decision on the machine and level when we make decisions online. To bound the competitive ratio, we introduce a reference system where each task with L levels are reconstructed as L tasks with a single level. Then the performance of the reference system is employed as an intermediate value to bound the competitive ratio. Mathematical derivations demonstrate that multiple processing levels do not worsen the competitive ratio because the reference system uniformly bounds different processing levels, and thus the competitive ratio is insensitive to the controllable processing time.

2 Related Work

One category of works related to this paper is *Online Bipartite Matching*, where the system needs to assign offline machines to online tasks to maximize the utility [Mehta, 2013]. One subcategory of works focuses on the adversary arrival order [Karp *et al.*, 1990], and another subcategory assumes known adversarial distribution (KAD) [Lowalekar *et al.*, 2020; Alaei *et al.*, 1993] or known identical independent distributions (KIID) [Shabtay and Steiner, 2007], where task arrival follows known distributions. Motivated by real-world scenarios, [Dong *et al.*, 2021] and [Dickerson *et al.*, 2021] studied the case that machines are reusable, and [Jaillet and Lu, 2014; Mehta *et al.*, 2015; Andani *et al.*, 2021] studied the case that machines can reject task assignment. [Sumita *et al.*, 2022] studied both reusable machines and rejections. Other topics studied in this field include fairness for task assignment [Nanda *et al.*, 2020; Ma *et al.*, 2020], multi-unit demand (a task may need multiple machines to process) [Goyal *et al.*, 2020; Hosseini *et al.*, 2022], and multi-capacity agent (a machine can process multiple tasks) [Alonso-Mora *et al.*, 2017; Lowalekar *et al.*, 2021]. However, there is no existing work considering controllable processing time in the online bipartite matching problem. A majority part of [Sumita *et al.*, 2022] can be regarded as a special case of our work when controllable processing time is not considered. It gives a $1/2$-competitive algorithm when each offline machine can reject unlimited times, and a $\Delta/(3\Delta - 1)$-competitive algorithm when each machine can reject no more than Δ times. Interestingly, our proposed algorithm also gives the same competitive ratios, but with substantially more complicated designs and analyses. To this end, a key conclusion derived in our paper is that the competitive ratio is "insensitive" to the processing levels. Please note that another work [Hikima *et al.*, 2022] studied controllable reward and different arrival probabilities, where the assignment impacts reward and arrival probabilities, which is different from controllable processing time in nature. Online bipartite matching is leveraged to solve many real-world problems other than machine allocation, such as ride-sharing [Lowalekar *et al.*, 2020; Dickerson *et al.*, 2021; Nanda *et al.*, 2020], crowd-sourcing [Goyal *et al.*, 2020; Liu *et al.*, 2021; Hikima *et al.*, 2022] and AdWords [Mehta *et al.*, 2007]. Still none of the existing work considered controllable processing time.

Another category of works related to this paper is *Controllable Processing Time*. Controllable processing time is studied in the context of scheduling. We can reduce the processing time of a job at a cost of reduced processing reward [Shabtay and Steiner, 2007; Tafreshian *et al.*, 2020]. [Janiak and Kovalyov, 1996], [Chen *et al.*, 1997] and [He *et al.*, 2007] study single machine scheduling. [Alidaee and Ahmadian, 1993] and [Shabtay and Kaspi, 2006] study multiple parallel machine scheduling. [Cheng *et al.*, 1996] employs bipartite matching to analyze multiple parallel machine scheduling. The above works focus on the offline scheduling problem. [Lu *et al.*, 2017] and [Wang *et al.*, 2017] study online scheduling with controllable processing time. [Lu *et al.*, 2017] focuses on single machine scheduling and [Wang *et al.*, 2017] focuses on the flow shop scheduling. Controllable processing time is also analyzed in the context of stochastic lot-sizing problem. We can compress the production time with extra cost, so that a better performance of planning can be obtained [Tunc, 2021; Koca *et al.*, 2015; Koca *et al.*, 2018]. These works optimize the performance in an offline manner. To the best of our knowledge, no existing work considered controllable processing time for the bipartite matching problem.

3 Model

We present a formal description of our problem in this section. We have a bipartite graph $G = (U, V; E)$, where U is the set of machines and V is the set of repeatable tasks.[1] E is the set of edges indicating if a task $v \in V$ can be processed by a machine $u \in U$. For each task, we have L processing levels $\mathcal{L} = \{1, ..., L\}$, indicating the L quality levels. If task v is processed by machine u ($(u, v) \in E$) with processing level l, it will generate a reward of $r_{u,v,l}$. Without loss of generality (WLOG), we have $r_{u,v,l} < r_{u,v,l'}$ when $l < l'$ (larger processing level gives larger reward). The system runs on a finite time horizon $T \in \mathbb{N}^+$. Each processing level $l \in \mathcal{L}$ causes a random processing delay d_l, which presents the occupation time to process a task with level l. In other words, if a machine starts to process a task with level l, it becomes unavailable to any other tasks until time $t + d_l$. d_l is drawn from a known distribution D_l. We have $\mathbb{E}[d_l] < \mathbb{E}[d_{l'}]$ when $l < l'$ (larger processing level requires longer processing time). For the convenience, we denote the set of edges connected to task v by E_v for all task v ($E_v = \{(u, v)|(u, v) \in E\}$), and similarly E_u is set of edges connected to machine u.

At each time t, task v may arrive with probability $p_{v,t}$. With a probability of $1 - \sum_{v \in V} p_{v,t}$, no task arrives at t. The set of probability distributions $\{p_{v,t}\}_{v \in V, t \in [T]}$ is known in advance (time variant but independent across time). When a task v arrives, we immediately and irrevocably either assign one machine which is a neighbor to v and is available, or discard v. When we assign task v to machine u, we also specify a processing level l. When receiving the assignment of task v, machine u has two possible actions: with probability q_e, u accepts the assignment; with probability $1 - q_e$, u rejects the assignment ($e = (u, v)$). Suppose this assignment is specified with processing level l, these two actions have two different results. If u accepts this assignment, it immediately gets a reward $r_{u,v,l}$ and becomes unavailable for a random period d_l. A machine u has a limited rejection budget (initialized as Δ_u). If u rejects this assignment, a rejection-penalty θ_l is introduced. We assume that Δ_u and θ_l are integers. This penalty is taken from the remaining budget of u, denoted by δ_u. We denote $\theta = \max_{l \in \mathcal{L}} \theta_l$. When a machine u runs out of its remaining budget ($\delta_u \leq 0$), it is removed from the system immediately and permanently. If a machine u is removed from the system, it receives no more task assignments.

Each machine has a positive initial budget ($\Delta_u > 0$). Please note we allow $\Delta_u = \infty$ to indicate unlimited rejection. We also allow $\theta_l = 1$ to indicate homogeneous rejection penalty (to limit the number of rejections).

3.1 Solution Overview

Our objective is to maximize the sum reward. We focus on the online setting: we only know the arrival of a task when it arrives. We know the distribution of task arrival in advance and the distribution of occupation time (KAD).

We first construct a linear programming (LP) `Off` to get an optimal solution $\mathbf{x}^*$ and an upper bound of the offline optimal value, which is referred to as LP(`Off`). The optimal solution $\mathbf{x}^*$ is then employed to construct our online algorithm. In the meanwhile, the upper bound of the offline optimal value LP(`Off`) will be set as a benchmark to evaluate the online algorithm, so that we then evaluate the competitive ratio between the performance of online algorithm and the offline optimal value.

[1] $v \in V$ actually indicates a type of tasks. For presentation convenience, v is also called "a task" throughout this paper.

3.2 Offline Optimal Value and Competitive Ratio

We consider the offline optimization version of the original problem as the benchmark and define the competitive ratio. In the offline setting, the full task sequence I is known in advance. However, we do not know whether a machine will accept or reject an assignment until it happens. We only have the probability of acceptance q_e. Given a full task sequence I, if an offline algorithm maximizes the expected reward, it is an offline optimal algorithm for I. This maximized expected reward for I is denoted by OPT(I). The expected OPT(I) on every sequence I is $\mathbb{E}_{I \sim \mathcal{I}}[\text{OPT}(I)]$, which is referred to as the *offline optimal value*.

An online algorithm ALG is α-competitive if the expected reward obtained by ALG is at least α times the offline optimal value (ie., if $\mathbb{E}_{I \sim \mathcal{I}}[\text{ALG}(I)] \geq \alpha \mathbb{E}_{I \sim \mathcal{I}}[\text{OPT}(I)]$ for any $\mathcal{I}$).

3.3 Linear Programming

It is not straightforward to quantify the offline optimal value $\mathbb{E}_{I \sim \mathcal{I}}[\text{OPT}(I)]$. In what follows, we construct an offline LP to get the upper bound of the offline optimal value.

$$\max_{\substack{x_{e,l,t},\\ \forall e \in E,\ l \in \mathcal{L},\\ t \in [T]}} \sum_{t \in [T]} \sum_{e \in E} q_e \sum_{l \in \mathcal{L}} r_{u,v,l} x_{e,l,t}$$

$$\text{s.t.} \sum_{t' < t} \sum_{e \in E_u} q_e \sum_{l \in \mathcal{L}} x_{e,l,t'} \Pr\{d_l \geq t - t' + 1\} + \sum_{e \in E_u} q_e \sum_{l \in \mathcal{L}} x_{e,l,t} \leq 1, \quad (\forall u \in U, t \in [T]), \tag{1}$$

$$\sum_{t \in [T]} \sum_{e \in E_u} \sum_{l \in \mathcal{L}} x_{e,l,t} \big[\theta q_e \Pr\{d_l > T - t\} + (1 - q_e)\theta_l\big] \leq \Delta_u + \theta - 1, (\forall u \in U), \tag{2}$$

$$0 \leq \sum_{e \in E_v} \sum_{l \in \mathcal{L}} x_{e,l,t} \leq p_{v,t}, \quad (\forall v \in V, t \in [T]), \tag{3}$$

$$0 \leq \sum_{l \in \mathcal{L}} x_{e,l,t} \leq p_{v,t}, (\forall v \in V, e \in E_v, t \in [T]), \tag{4}$$

$$0 \leq \sum_{e \in E_u} \sum_{l \in \mathcal{L}} x_{e,l,t} \leq 1, \quad (\forall u \in U, t \in [T]), \tag{5}$$

This LP is referred to as `Off`. The optimal solution to `Off` is $\mathbf{x}^* \stackrel{\text{def}}{=} \{x^*_{e,l,t}\}$, and the optimal value for the objective function of `Off` is referred to as LP(`Off`). LP(`Off`) is the upper bound for $\mathbb{E}_{I \sim \mathcal{I}}[\text{OPT}(I)]$ (as shown in Lemma below). In addition, $\mathbf{x}^*$ is to be employed in the online algorithm.

In the following Lemma, we show that LP(`Off`) is a valid upper bound for the offline optimal value $\mathbb{E}_{I \sim \mathcal{I}}[\text{OPT}(I)]$.

Lemma 1. *(LP(`Off`) Upper Bound) LP(`Off`) $\geq \mathbb{E}_{I \sim \mathcal{I}}[OPT(I)]$*

Proof. Proofs of all Lemmas and Theorems are in the full version [Wu *et al.*, 2023]. □

3.4 Online Machine and Level Assignment (OMLA) Algorithm

In this section, we use the optimal solution to `Off` to construct our OMLA algorithm.

Design overview. In the online algorithm, we first decide the probability that we choose machine-level pair (u, l) when task v arrives at time t. Then we decide whether or not to assign machine u to task v with processing level l, by comparing the different expected rewards (of machine u) brought by different decisions. We present our online algorithm (OMLA) in Algorithm 1 and we discuss them line by line.

OMLA. Upon the arrival of task v, we first choose a machine u and a processing level l with probability $x^*_{e,l,t}/p_{v,t}$ (Line 4). Suppose u has a remaining rejection budget of δ at t. If u has run out of the rejection budget ($\delta \leq 0$) or is occupied by a previous task, we do not assign u or assign any other machine to v (Line 5). Then, we define the *baseline value* (R-value) of u at t as the expected sum reward of u at and after t without knowing the arrival at t, which is denoted by $R^{\delta}_{u,t}$; we define the *activation value* (Q-value) of u at t as the expected sum reward of u at and after t if u is assigned to v with processing level l, which is denoted by $Q^{\delta}_{e,l,t}$. More details on the derivations of baseline values and activation values will be given shortly. Baseline value and activation value will be compared to make a decision. We compare activation value at t and the baseline value at $t+1$, to decide if an active action at t (making the assignment) is beneficial. If the baseline value of u at $t+1$ is larger, we do not assign u and discard v; Otherwise, if the activation value is larger at t, we assign u to v with l (Line 6). When u accepts this assignment, it becomes occupied for a random time d_l (Line 8). When u rejects this assignment, a rejection penalty θ_l is taken from u's rejection budget δ (Line 9).

Calculation of Activation Values and Baseline Values

Because we focus on the KAD model, we can calculate each activation value and baseline value in advance (before we execute Algorithm 1). The calculation is presented in Algorithm 2. When u has a positive remaining budget ($\delta > 0$), the activation value $Q^{\delta}_{e,l,t}$ consists of two parts: ① With probability q_e ($e = (u, v)$), u accepts the assignment and immediately gets a reward $r_{u,v,l}$ (6). After a random occupation time d_l, u finishes this task, and its baseline value becomes $R^{\delta}_{u,t+d_l}$ at $t + d_l$ (Lines 18–19); ② With probability $1 - q_e$, u rejects the assignment and takes a rejection penalty θ_l on its remaining budget δ, and its baseline value becomes $R^{\delta-\theta_l}_{u,t+1}$ at $t+1$ (Lines 16–17). By the above two parts, we can calculate the activation value $Q^{\delta}_{e,l,t}$:

$$Q^{\delta}_{e,l,t} = q_e\Big(r_{u,v,l} + \sum_{d' \in [T-t]} \Pr\{d_l = d'\} R^{\delta}_{u,t+d'}\Big) + (1 - q_e) R^{\delta-\theta_l}_{u,t+1}, \quad (\delta > 0, t \in [T]). \tag{6}$$

Formula (6) is calculated in Line 20. When u has run out of the rejection budget ($\delta \leq 0$), it is removed from the market. We set $Q^{\delta}_{e,l,t} = 0$ if $\delta \leq 0$ or $t > T$ as boundary values. We choose the higher one between the activation value at t and the baseline value at $t+1$, so we have the expected reward of the chosen u as $\max\{Q^{\delta}_{e,l,t}, R^{\delta}_{u,t+1}\}$. Since the probability that v arrives at t and (u, l) is chosen is $x^*_{e,l,t}$, we can calculate each baseline value $R^{\delta}_{u,t}$ (Line 21) by

$$R^{\delta}_{u,t} = \sum_{e \in E_u} \sum_{l \in \mathcal{L}} x^*_{e,l,t} \max\{Q^{\delta}_{e,l,t}, R^{\delta}_{u,t+1}\} + \Big(1 - \sum_{e \in E_u} \sum_{l \in \mathcal{L}} x^*_{e,l,t}\Big) R^{\delta}_{u,t+1}, (\delta > 0, t \in [T]). \tag{7}$$

We set $R^{\delta}_{u,t} = 0$ if $\delta \leq 0$ or $t > T$ as boundary values.

In order to execute our online algorithm, we need to calculate each $Q^{\delta}_{e,l,t}$ and $R^{\delta}_{u,t}$ in advance. This can be done with

Algorithm 1 OMLA Algorithm

Input: U, V, E, $\{Q^{\delta}_{e,l,t}\}$, $\{R^{\delta}_{u,t}\}$, $\mathbf{x}^*$

1: **for** all $t \leftarrow 1$ to T **do**
2: **if** no task arrives **then** skip
3: **else** (v arrives)
4: choose pair (u, l) with probability $x^*_{e,l,t}/p_{v,t}$
5: **if** u is not occupied, u has a positive remaining budget $\delta_u > 0$ and $Q^{\delta}_{e,l,t} \geq R^{\delta}_{u,t+1}$ **then**
6: we assign (u, l) to v
7: **if** u accepts **then**
8: draw d_l from D_l, u gets occupied for d_l
9: **else** $\delta_u \leftarrow \delta_u - \theta_l$

Algorithm 2 Calculation of Activation and Baseline Values

Input: U, V, E, $\mathcal{L}$, T, $\{\theta_l\}$, $\{q_e\}$, $\{r_{u,v,l}\}$, $\{D_l\}$, $\{\Delta_u\}$

1: Solve LP(`Off`) to obtain $\mathbf{x}^*$
2: $\Delta \leftarrow \max \Delta_u$
3: **for** all (δ, u, v, l) that $(u, v) \in E, \delta > 0$ and $l \in \mathcal{L}$ **do**
4: $Q^{\delta}_{e,l,T} \leftarrow q_e r_{u,v,l}$
5: **for** all (δ, u) that $\delta > 0$ and $u \in U$ **do**
6: $a \leftarrow 0$
7: **for** all v that $(u, v) \in E_u$ **do**
8: $b \leftarrow 0$
9: **for** all $l \in \mathcal{L}$ **do** $b \leftarrow b + x^*_{e,l,T} q_e r_{u,v,l}$
10: $a \leftarrow a + b$
11: $R^{\delta}_{u,T} \leftarrow a$
12: **for** $t \leftarrow T - 1$ to 1 **do**
13: **for** (δ, u) that $\delta > 0$ and $u \in U$ **do**
14: **for** (v, l) that $(u, v) \in E_u$ and $l \in \mathcal{L}$ **do**
15: $a \leftarrow 0$
16: **if** $\delta \geq \theta_l$ **then** $b \leftarrow R^{\delta-\theta_l}_{u,t+1}$
17: **else** $b \leftarrow 0$
18: **for** $d \leftarrow 1$ to $T - t + 1$ **do**
19: $a \leftarrow a + R^{\delta}_{u,t+d} \Pr\{d_l = d\}$
20: $Q^{\delta}_{e,l,t} \leftarrow q_e(r_{u,v,l} + a) + (1 - q_e)b$
21: Calculate $R^{\delta}_{u,t}$ by (7)

Output: $\{Q^{\delta}_{e,l,t}\}$, $\{R^{\delta}_{u,t}\}$, $\mathbf{x}^*$

the initial condition at T (Lines 3–11)

$$\begin{cases} Q^{\delta}_{e,l,T} = q_e r_{u,v,l},\ \forall e \in E, l \in \mathcal{L}, \delta > 0, \\ R^{\delta}_{u,T} = \sum_{e \in E_u} \sum_{l \in \mathcal{L}} x^{*}_{e,l,T} q_e r_{u,v,l},\ \forall u \in U, \delta > 0. \end{cases} \tag{8}$$

4 Competitive Ratio Analysis

In this section, we analyze the competitive ratio of our online algorithm. We have already derived Lemma 1, where we find an upper bound LP($\texttt{Off}$) of the offline optimal value. Then we first introduce our reference system, which provides a lower bound of the original system (Lemma 2). In this reference system, each task with L levels are reconstructed as L tasks with a single level. After Lemma 2, we analyze the competitive ratio in two branches separately: ① each machine u has an infinite initial rejection budget $\Delta_u = \infty$ (the unlimited rejection case); ② each machine u has a finite rejection budget $\Delta_u < \infty$. For the unlimited rejection case, we first find a lower bound for the reference system (Lemma 3). Then we construct the *auxiliary inequality* for the unlimited rejection case (Lemma 4) by Lemmas 1–3. Then by Lemma 4, we prove that OMLA is $1/2$-competitive for the unlimited rejection case (Theorem 1). For the limited rejection case, we first find the *performance induction inequality* of the reference system (Lemma 5). With this inequality, we find a lower bound of the reference system (Lemma 6). Then we construct the auxiliary inequality for the limited rejection case (Lemma 7) by Lemmas 1, 2, and 6. By Lemma 7, we prove that OMLA is $\Delta/(3\Delta - 1)$-competitive for the limited rejection case, where $\Delta = \max_{u \in U} \Delta_u$ (Theorem 2).

4.1 Reference System

It is not straightforward to directly analyze the performance of the original system, so that we need to find an intermediate value to bound it. One key challenge of the original problem is introduced by different processing levels. With controllable processing time, L processing levels form one additional dimension. The reference system is to construct another bipartite matching system without this additional dimension, to provide a lower bound of the original tripartite matching system. We will also need to show: 1) the reference system is a valid system; and 2) the expected reward of this reference system is a valid lower bound for $R^{\Delta_u}_{u,1}$.

For each machine u, we construct a reference system for u as follows. The reference system has a bipartite graph $G'_u = (U'_u, V'_u; E'_u,)$, where U'_u contains only one machine u, V'_u is a set of non-repeatable tasks. At each t, one of L tasks may come, denoted by $v'_{u,l,t}$, with a probability $p'_{u,l,t}$. Each of $L \times T$ tasks in V'_u is different from each other. $v'_{u,l,t}$ can only be processed with processing level l. Each $v'_{u,l,t}$ has an edge to u. If task $v'_{u,l,t}$ arrives at t and u is available, we must choose u and we must decide whether to assign u to $v'_{u,l,t}$. Suppose u's remaining budget is δ at t. Similar to the baseline value and activation value (Section 3.4), we define the reference baseline value (resp. the reference activation value) as $\tilde{R}^{\delta}_{u,t}$ (resp., $\tilde{Q}^{\delta}_{u,l,t}$). The calculation of reference baseline values and reference activation values is similar to that of baseline values and activation values, and will be given shortly. We compare the reference activation value of u at t and the reference baseline value of u at $t+1$. The higher one indicates our choice: If the former one is larger, we assign u to $v'_{u,l,t}$; Otherwise, we discard $v'_{u,l,t}$. The probability that u accepts this assignment is $q'_{u,l,t}$. The reward of processing $v'_{u,l,t}$ is $r'_{u,l,t}$. The processing time d'_l is drawn from the known distribution D_l. The rejection penalty is θ_l, same as in the original system. The initial budget of u is Δ_u, same as in the original system. We set the parameters $p'_{u,l,t}$, $q'_{u,l,t}$, and $r'_{u,l,t}$ in the reference system as follows: The probability that $v'_{u,l,t}$ arrives at t: $p'_{u,l,t} = \sum_{e \in E_u} x^{*}_{e,l,t}$; The probability that u accepts the assignment of task $v'_{u,l,t}$: $q'_{u,l,t} = (\sum_{e \in E_u} q_e x^{*}_{e,l,t})/p'_{u,l,t}$ if $p'_{u,l,t} > 0$; otherwise $q'_{u,l,t} = 0$; The reward of processing $v'_{u,l,t}$ (with level l): $r'_{u,l,t} = (\sum_{e \in E_u} q_e r_{u,v,l} x^{*}_{e,l,t})/(p'_{u,l,t} q'_{u,l,t})$ if $p'_{u,l,t} q'_{u,l,t} > 0$, otherwise $r'_{u,l,t} = 0$. The distribution of occupation time of processing level l is $\Pr\{d'_l = d\} = \Pr\{d_l = d\}$.

With the above parameters, we can calculate $\tilde{Q}^{\delta}_{u,l,t}$ and $\tilde{R}^{\delta}_{u,t}$ by

$$\begin{aligned} \tilde{R}^{\delta}_{u,t} =& \sum_{l \in \mathcal{L}} p'_{u,l,t} \max\{\tilde{Q}^{\delta}_{u,l,t}, \tilde{R}^{\delta}_{u,t+1}\} \\ &+ \Big(1 - \sum_{l \in \mathcal{L}} p'_{u,l,t}\Big) \tilde{R}^{\delta}_{u,t+1},\ \ (\delta > 0, t \in [T]), \end{aligned} \tag{9}$$

$$\begin{aligned} \tilde{Q}^{\delta}_{u,l,t} =& q'_{u,l,t} \Big(r'_{u,l,t} + \sum_{d' \in [T-t]} \Pr\{d_l = d'\} \tilde{R}^{\delta}_{u,t+d'} \Big) \\ &+ (1 - q'_{u,l,t}) \tilde{R}^{\delta-\theta_l}_{u,t+1},\ \ (\delta > 0, t \in [T]). \end{aligned} \tag{10}$$

We set $\tilde{R}^{\delta}_{u,t} = 0$ and $\tilde{Q}^{\delta}_{u,l,t} = 0$ if $\delta \le 0$ or $t > T$. We do not need to calculate the specific value of $\tilde{Q}^{\delta}_{u,l,t}$ and $\tilde{R}^{\delta}_{u,t}$ (no computational complexity is introduced), as we only need these values in the analysis. The reference system for each u is a valid system, because each $p'_{u,l,t}$ and $q'_{u,l,t}$ is a valid probability value. From (5), we have $p'_{u,l,t} \le 1$ and $\sum_l p'_{u,l,t} \le 1$. From (1), we have $q'_{u,l,t} \le 1$. Therefore, $p'_{u,l,t}$ and $q'_{u,l,t}$ are valid probability values and the reference system for each machine u is a valid system.

In Lemma 2, we show that for each u, t, and δ, the performance of the reference system $\tilde{R}^{\delta}_{u,t}$ is a lower bound of the performance of u in the original system $R^{\delta}_{u,t}$.

Lemma 2. *(Reference System Bounds Original System)* $R^{\delta}_{u,t} \ge \tilde{R}^{\delta}_{u,t}$, $\forall \delta$ *and* t.

By Lemma 2, we can use the lower bound of $\tilde{R}^{\delta}_{u,t}$ as a valid lower bound for $R^{\delta}_{u,t}$. With this reference system, we first analyze the competitive ratio when each machine has an infinite initial rejection budget, then analyze the competitive ratio when each machine has an initial rejection budget no more than Δ, where $\Delta = \max_{u \in U} \Delta_u$.

4.2 Unlimited Rejection Case

In this section, we analyze the competitive ratio for the unlimited rejection case. In the unlimited rejection case, each u has

an infinite initial rejection budget $\Delta_u = \infty$. One straightforward way is to let Δ_u to be sufficiently large when we run Algorithms 1 and 2. A more efficient way is to replace $R^{\delta}_{u,t}$, $Q^{\delta}_{e,l,t}$, $\tilde{R}^{\delta}_{u,t}$, and $\tilde{Q}^{\delta}_{u,l,t}$ ($\forall\delta$) by $R_{u,t}$, $Q_{e,l,t}$, $\tilde{R}_{u,t}$, and $\tilde{Q}_{u,l,t}$ respectively as they are indifferent under different δ. The slightly modified Algorithms 1 and 2 are shown in the full version [Wu *et al.*, 2023].

The main result of this section is that Algorithm 1 is $1/2$-competitive for the unlimited rejection case (Theorem 1). To get this result, we first get a lower bound of the performance of the reference system (Lemma 3), then construct an auxiliary inequality (Lemma 4) to show that a lower bound of the competitive ratio can be obtained by the ratio between the lower bound of $\sum_u \tilde{R}_{u,1}$ and the offline optimal value LP($\mathtt{Off}$). By Lemma 4, we get the result in Theorem 1.

We first show that we have a lower bound for $\tilde{R}_{u,1}$ in Lemma 3. $\tilde{R}_{u,1}$ is u's expected sum reward at and after $t = 1$ (overall expected sum reward of u) in the reference system.

Lemma 3. *(Reference System Lower Bound) For each u, we have*

$$\tilde{R}_{u,1} \geq \frac{1}{2} \sum_{t\in[T]} \sum_{l\in\mathcal{L}} \sum_{e\in E_u} q_e r_{u,v,l} x^*_{e,l,t}. \tag{11}$$

To prove Lemma 3, the key step is to establish a dual LP to derive the bound. We also utilize the property that "the sum of maximum is no less than the maximum of sum". Next, we show the auxiliary inequality for the unlimited case by Lemmas 1–3.

Lemma 4. *(Auxiliary Inequality) For the original system, in the unlimited rejection case, we have*

$$\frac{\mathbb{E}_{I\sim\mathcal{I}}[ALG(I)]}{\mathbb{E}_{I\sim\mathcal{I}}[OPT(I)]} \geq \frac{\frac{1}{2}\sum_{u\in U}\sum_{t\in[T]}\sum_{l\in\mathcal{L}}\sum_{e\in E_u} q_e r_{u,v,l} x^*_{e,l,t}}{LP(\mathtt{Off})}. \tag{12}$$

Then we introduce Theorem 1. We prove that Algorithm 1 is $1/2$-competitive for the unlimited rejection case.

Theorem 1. *(Competitive Ratio of Unlimited Rejection) OMLA is $1/2$-competitive for the problem with unlimited rejection budget.*

4.3 Limited Rejection Case

In this section, we analyze the competitive ratio for the limited rejection case. Each u has a finite initial rejection budget $\Delta_u < \infty$. The main result of this subsection is that Algorithm 1 is $\Delta/(3\Delta-1)$-competitive for the limited rejection case, where $\Delta = \max_{u\in U} \Delta_u$ (Theorem 2). To get this result, we first present performance induction inequality (Lemma 5), which is used to get a lower bound of the performance of the reference system (Lemma 6). Then we construct an auxiliary inequality (Lemma 7) to show that a lower bound of the competitive ratio can be obtained by the ratio between the lower bound of $\sum_u \tilde{R}^{\Delta_u}_{u,1}$ and the offline optimal value LP($\mathtt{Off}$). Finally, by Lemma 7, we conclude the result in Theorem 2.

We first show an inequality on $\tilde{R}^{\delta-\theta_l}_{u,t}$ and $\tilde{R}^{\delta}_{u,t}$, which is used to get a lower bound of the performance of the reference system. Please note that this inequality is new for the limited rejection as we need to consider the remaining budget δ now.

Lemma 5. *(Performance Induction Inequality) For all δ, t and l, we have*

$$\tilde{R}^{\delta-\theta_l}_{u,t} \geq \frac{\delta-\theta_l}{\delta}\tilde{R}^{\delta}_{u,t}. \tag{13}$$

Then we show that we have a lower bound for $\tilde{R}^{\Delta_u}_{u,1}$.

Lemma 6. *(Reference System Lower Bound) For each u, we have*

$$\tilde{R}^{\Delta_u}_{u,1} \geq \frac{\Delta_u}{3\Delta_u - 1} \sum_{t\in[T]} \sum_{l\in\mathcal{L}} \sum_{e\in E_u} q_e r_{u,v,l} x^*_{e,l,t}. \tag{14}$$

The key step is to establish a dual LP to derive the bound. By Lemma 6, we can eliminate the influence of different rejection penalty values (non-homogeneous θ_l) of different levels from (6). In the proof, it is sufficient to derive a bound utilizing the dual LP, and the dual LP can eliminate the impact of non-homogeneous θ_l in (2).

Next, we show the auxiliary inequality for the limited rejection case by Lemmas 1, 2, and 6.

Lemma 7. *(Auxiliary Inequality) For the original system, under the limited rejection case, we have*

$$\frac{\mathbb{E}_{I\sim\mathcal{I}}[ALG(I)]}{\mathbb{E}_{I\sim\mathcal{I}}[OPT(I)]} \geq \frac{\sum_{u\in U}\frac{\Delta_u}{3\Delta_u-1}\sum_{t\in[T]}\sum_{l\in\mathcal{L}}\sum_{e\in E_u} q_e r_{u,v,l} x^*_{e,l,t}}{LP(\mathtt{Off})}. \tag{15}$$

Then we introduce Theorem 2. We prove that OMLA is $\Delta/(3\Delta-1)$-competitive for the limited rejection case, where $\Delta = \max_{u\in U} \Delta_u$.

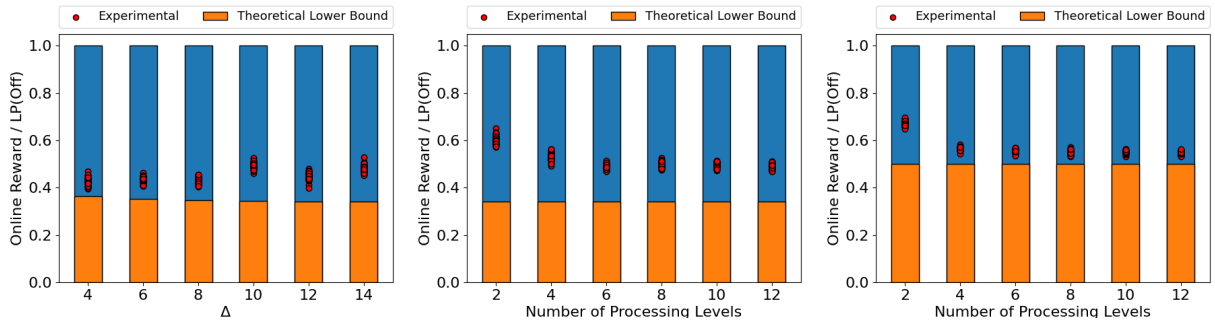
Theorem 2. *OMLA is a $\Delta/(3\Delta-1)$-competitive algorithm for the limited rejection case, where $\Delta = \max_{u\in U} \Delta_u$.*

5 Evaluation

5.1 Benchmarks and Synthetic Dataset

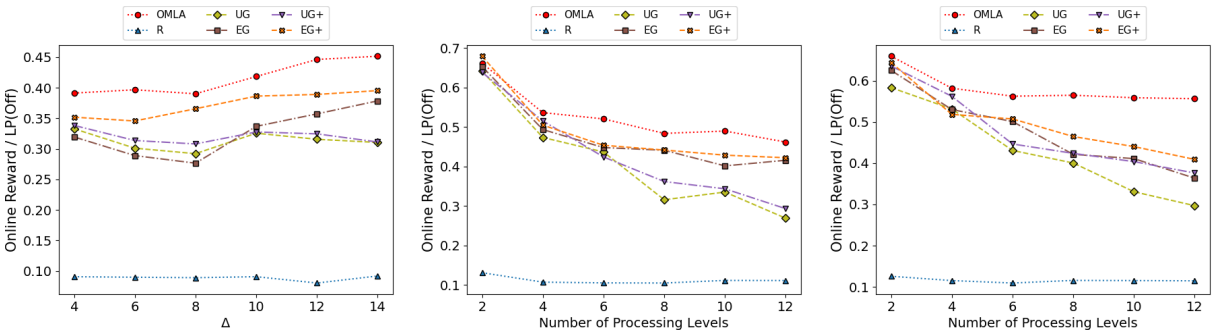
In this section, we evaluate OMLA against five benchmarks: 1) Random (R): the system randomly chooses a machine-level pair when an online task arrives. If the chosen machine is available, we assign this machine-level pair. Otherwise, we discard the task. 2) Utility Greedy (UG): when task v arrives, the system ranks all the machine-level pairs by $r_{u,v,l}$ and chooses the highest one available. 3) Efficiency Greedy (EG): with the expectation $\mathbb{E}[d_l]$ of the occupation time of processing level l calculated in advance, when task v arrives, the system ranks all the machine-level pair by $r_{u,v,l}/\mathbb{E}[d_l]$ and chooses the highest one available. 4) Utility Greedy + (UG+): when a task v arrives, we choose a machine u by [Sumita *et al.*, 2022], then choose the level l with the highest $r_{u,v,l}$. 5) Efficiency Greedy + (EG+): when a task v arrives, we choose a machine u by [Sumita *et al.*, 2022], then choose the level l with the highest $r_{u,v,l}/\mathbb{E}[d_l]$. Please note that that [Sumita *et al.*, 2022] did not consider processing level. We use the approach in [Sumita *et al.*, 2022] to choose machine and then use greedy method to choose level in UG+ and EG+.

We generate the synthetic data set in the experiment. (The approach was also adopted in [Sumita *et al.*, 2022].) We set $|U| = 10$, $|V| = 25$, and $T = 100$. For each u and v, an



(a) L=12 (b) Δ=20 (c) Δ=∞

Figure 1: Online Reward / LP(Off) of different L and Δ.



(a) L=12 (b) Δ=20 (c) Δ=∞

Figure 2: Online Reward / LP(Off) of different benchmarks with different L and Δ.

edge (u, v) exists in E with probability 0.1. For each $e \in E$, we set $q_e \sim U(0.5, 1)$ and $r_{u,v,l} \sim U(a \cdot l^{0.2}, a \cdot l^{0.4})$, where $a \sim U(0.5, 1)$. For each $l \in \mathcal{L}$ we set the distribution D_l as a binomial distribution $B(T, l^{1.2}/20)$. For settings (a), (b) in Figure 2 and (a) (b) in Figure 3, Δ_u is drawn uniformly from $[\Delta]$. We set the rejection penalty for level l as $\theta_l = l + 2$.

5.2 Results

In Figure 1, we investigate the ratio between the online performance of OMLA and LP(Off). We randomly generate a set of $\{p_{v,t}\}$, E and $\{q_e\}$ for each sub-figure. For each pair of Δ and L, we generate a set of $\{\Delta_u\}$ and $\{r_{u,v,l}\}$, then we run 50 rounds of experiment. In each round of experiment, we randomly generate 50 task sequences from $\{p_{v,t}\}$, and calculate the ratio between the averaged total reward of OMLA and LP(Off). The orange bars in Figure 1 represent the competitive ratio of OMLA for each pair of Δ and L. The red dots in Figure 1 show the ratio between the averaged online total reward and LP(Off). Figure 1 shows that the ratio between the averaged performance of OMLA is indeed higher than the theoretical lower bound of the competitive ratio. The results in Figure 1 verifies our conclusion on the competitive ratios.

In Figure 2, we compare the performance of OMLA with benchmarks. We randomly generate a set of $\{p_{v,t}\}$, E and $\{q_e\}$ for each sub-figure. For each pair of Δ and L, we generate a set of $\{\Delta_u\}$ and $\{r_{u,v,l}\}$. Then for each algorithm, we randomly generate 250 task sequences from $\{p_{v,t}\}$, and calculate the ratio between the averaged total reward with LP(Off). OMLA outperforms all of the benchmarks with each pair of Δ and L. In Figures 2(b) and 2(c), the performance of OMLA is slightly higher than the performance of UG+ and EG+ when the number of processing level is 2, but the performance gain becomes larger when there there are more processing levels. This demonstrates that OMLA is more advantageous for more processing levels as it is designed for joint assignment of machine and level. The results demonstrate that OMLA has the best performance with controllable processing time. OMLA provides both theoretical performance guarantees (competitive ratio) and the best average performance on the synthetic dataset.

6 Conclusion

In this paper, we investigate the online bipartite matching problem with controllable processing time. We design OMLA, an online algorithm to simultaneously assign an offline machine and a processing level to each online task. We prove that OMLA achieves $1/2$-competitive ratio if each machine has unlimited rejection budget and $\Delta/(3\Delta - 1)$-competitive ratio if each machine has an initial budget up to Δ. Furthermore, we conduct experiments on synthetic data sets, where the results demonstrate that OMLA outperforms benchmarks under a variety of environments.

References

[Alaei *et al.*, 1993] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. pages 18–35, 1993.

[Alidaee and Ahmadian, 1993] Bahram Alidaee and Ahmad Ahmadian. Two parallel machine sequencing problems involving controllable job processing times. *European Journal of Operational Research*, 70(3):335–341, 1993.

[Alonso-Mora *et al.*, 2017] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3):462–467, 2017.

[Andani *et al.*, 2021] I. Gusti Ayu Andani, Lissy La Paix Puello, and Karst Geurs. Modelling effects of changes in travel time and costs of toll road usage on choices for residential location, route and travel mode across population segments in the jakarta-bandung region, indonesia. *Transportation Research Part A: Policy and Practice*, 145:81–102, 2021.

[Chen *et al.*, 1997] Zhi-Long Chen, Qing Lu, and Guochun Tang. Single machine scheduling with discretely controllable processing times. *Operations Research Letters*, 21(2):69–76, 1997.

[Cheng *et al.*, 1996] T. C. E. Cheng, Z. L. Chen, and Chung-Lun Li. Parallel-machine scheduling with controllable processing times. *IIE transactions*, 28(2):177–180, 1996.

[Dickerson *et al.*, 2021] John P. Dickerson, Karthik A. Sankararaman, Aravind Srinivasan, and Pan Xu. Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. *ACM Transactions on Economics and Computation (TEAC)*, 9(3):1–17, 2021.

[Dong *et al.*, 2021] Zehao Dong, Sanmay Das, Patrick Fowler, and Chien-Ju Ho. Efficient nonmyopic online allocation of scarce reusable resources. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, page 447–455, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems.

[Goyal *et al.*, 2020] Vineet Goyal, Garud Iyengar, and Rajan Udwani. Asymptotically optimal competitive ratio for online allocation of reusable resources. *arXiv preprint arXiv:2002.02430*, 2020.

[He *et al.*, 2007] Yong He, Qi Wei, and T. C. E. Cheng. Single-machine scheduling with trade-off between number of tardy jobs and compression cost. *Journal of Scheduling*, 10(4):303–310, 2007.

[Hikima *et al.*, 2022] Yuya Hikima, Yasunori Akagi, Naoki Marumo, and Hideaki Kim. Online matching with controllable rewards and arrival probabilities. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 1825–1833. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

[Hosseini *et al.*, 2022] Hadi Hosseini, Zhiyi Huang, Ayumi Igarashi, and Nisarg Shah. Class fairness in online matching. *arXiv preprint arXiv:2203.03751*, 2022.

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

[Janiak and Kovalyov, 1996] Adam Janiak and Mikhail Y. Kovalyov. Single machine scheduling subject to deadlines and resource dependent processing times. *European Journal of Operational Research*, 94(2):284–291, 1996.

[Karp *et al.*, 1990] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 352–358, New York, NY, USA, 1990. Association for Computing Machinery.

[Koca *et al.*, 2015] Esra Koca, Hande Yaman, and M. Selim Aktürk. Stochastic lot sizing problem with controllable processing times. *Omega*, 53:1–10, 2015.

[Koca *et al.*, 2018] Esra Koca, Hande Yaman, and M. Selim Aktürk. Stochastic lot sizing problem with nervousness considerations. *Computers & Operations Research*, 94:23–37, 2018.

[Liu *et al.*, 2021] Hao Liu, Qiang Zhao, Yike Ma, and Feng Dai. Bipartite matching for crowd counting with point supervision. In *IJCAI*, pages 860–866, 2021.

[Lowalekar *et al.*, 2020] Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. Competitive ratios for online multi-capacity ridesharing. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, page 771–779, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems.

[Lowalekar *et al.*, 2021] Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. Zac: A zone path construction approach for effective real-time ridesharing. volume 29, pages 528–538, May 2021.

[Lu *et al.*, 2017] Chao Lu, Liang Gao, Xinyu Li, and Shengqiang Xiao. A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry. *Engineering Applications of Artificial Intelligence*, 57:61–79, 2017.

[Ma *et al.*, 2020] Will Ma, Pan Xu, and Yifan Xu. Group-level fairness maximization in online bipartite matching. *arXiv preprint arXiv:2011.13908*, 2020.

[Mahesh, 2020] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9:381–386, 2020.

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

[Mehta *et al.*, 2015] Aranyak Mehta, Bo Waggoner, and Morteza Zadimoghaddam. Online stochastic matching

with unequal probabilities. pages 1388–1404. SIAM, 2015.

[Mehta, 2013] Aranyak Mehta. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368, 2013.

[Nanda *et al.*, 2020] Vedant Nanda, Pan Xu, Karthik Abhinav Sankararaman, John Dickerson, and Aravind Srinivasan. Balancing the tradeoff between profit and fairness in rideshare platforms during high-demand hours. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):2210–2217, Apr. 2020.

[Shabtay and Kaspi, 2006] Dvir Shabtay and Moshe Kaspi. Parallel machine scheduling with a convex resource consumption function. *European Journal of Operational Research*, 173(1):92–107, 2006.

[Shabtay and Steiner, 2007] Dvir Shabtay and George Steiner. A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, 155(13):1643–1666, 2007.

[Sumita *et al.*, 2022] Hanna Sumita, Shinji Ito, Kei Takemura, Daisuke Hatano, Takuro Fukunaga, Naonori Kakimura, and Ken-ichi Kawarabayashi. Online task assignment problems with reusable resources. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5199–5207, 2022.

[Tafreshian *et al.*, 2020] Amirmahdi Tafreshian, Neda Masoud, and Yafeng Yin. Frontiers in service science: Ride matching for peer-to-peer ride sharing: A review and future directions. *Service Science*, 12(2-3):44–60, 2020.

[Taia, 2022] Taia. Select your translation delivery dates with ease. https://taia.io/features/translation-delivery-dates/, 2022. Accessed: 2023-01-13.

[Tong *et al.*, 2020] Yongxin Tong, Zimu Zhou, Yuxiang Zeng, Lei Chen, and Cyrus Shahabi. Spatial crowdsourcing: a survey. *The VLDB Journal*, 29(1):217–250, 2020.

[Tunc, 2021] Huseyin Tunc. A mixed integer programming formulation for the stochastic lot sizing problem with controllable processing times. *Computers & Operations Research*, 132:105302, 2021.

[Wang *et al.*, 2017] Du-Juan Wang, Feng Liu, and Yaochu Jin. A multi-objective evolutionary algorithm guided by directed search for dynamic scheduling. *Computers & Operations Research*, 79:279–290, 2017.

[Wang *et al.*, 2019] Zizhao Wang, Wei Bao, Dong Yuan, Liming Ge, Nguyen H. Tran, and Albert Y. Zomaya. See: Scheduling early exit for mobile dnn inference during service outage. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '19, page 279–288, New York, NY, USA, 2019. Association for Computing Machinery.

[Wu *et al.*, 2023] Ruoyu Wu, Wei Bao, and Liming Ge. Online task assignment with controllable processing time. *arXiv preprint arXiv:2305.04453*, 2023.