Congestion-Aware Matching and Learning for Service Platforms

Xu Sun
University of Miami Business School, Coral Gables, Florida 33146, USA, xxs767@miami.edu

Shiwei Chai
Warrington College of Business, University of Florida, Gainesville, FL 32611, USA, s.chai@ufl.edu

Jingtong Zhao
School of Economics, Renmin University of China, Beijing, China, zhaojingtong@ruc.edu.cn

Problem definition: We study dynamic matching in a service platform represented by a multi-class, multi-server queueing system. Matching is in the form of job-server assignments, and the service platform lacks knowledge of job-server-specific mean rewards. The objective is to minimize regret, defined as the difference between the cumulative payoff over a time horizon and the maximum payoff possible, with complete knowledge of all system parameters and stochasticity absent. **Methodology/results:** We propose two matching and learning algorithms: one close in spirit to the barrier method in optimization and the other using Lyapunov optimization. Both algorithms estimate job-server assignment rewards using a bandit learning subroutine. When the total service capacity exceeds the demand, both algorithms achieve sub-linear regret and maintain queue stability. Numerical experiments with synthetic and real-world data validate the effectiveness of our algorithms. **Managerial implications:** Our analysis reveals that, despite not being explicitly modeled as part of the managerial objective, queueing delays can impede learning, ultimately reducing platform profitability. Therefore, reducing service delays can be crucial not only to achieving high service levels but also to maximizing rewards.

Key words: service platforms; reward maximization; matching; online learning; multi-armed bandit; queueing

1. Introduction

Online service platforms have emerged across various domains, connecting customers (jobs) with service providers (servers) through technology. These platforms offer improved service access, reduced search costs, and shorter service delays. Among these service platforms, many share the following characteristics, which directly motivate this work: First, jobs are naturally segmented (and thus diverse) and served by servers with overlapping capabilities. Second, services are appointment-based, requiring job-server assignments to be performed upon arrival, and since servers have limited capacity, an assigned job may have to wait before the service can actually commence. Third, different assignments yield different rewards, but the platform does not know how good a match is at the outset, necessitating learning about match quality based on job and server characteristics.

The primary motivation for this work is the growth of digital healthcare platforms in recent years. These platforms offer patients convenient online consultations conducted through photos or phone calls for minor diseases.

Compared to traditional offline consultations, online healthcare consultations are often more affordable and can be conducted from the patients' homes, improving patient satisfaction and reducing hospital congestion while allowing offline medical resources to be reserved for patients with more serious medical conditions. Platforms like HaoDf and WeiDoctor assign a physician for each patient, allowing patients to accept an assignment instead of selecting a specific one. Specifically, patients provide personal information, including their gender, age, and a brief overview of their illness and medical history, and then the platform immediately assigns a physician to the patient, displaying the name, title, and affiliated hospital of the physician. This feature is particularly useful for patients who lack knowledge about available physicians and do not have the time to conduct a thorough search. To ensure user satisfaction, these platforms often guarantee that the assigned physician will respond within one hour. If the response time exceeds the specified limit, patients are typically eligible for a refund. Currently, these platforms prioritize assigning patients to physicians based solely on the physicians' ability to respond quickly in order to avoid patient wait times. This, however, overlooks patients' personalized needs, resulting in patients sometimes being assigned a less suitable physician, potentially compromising clinical outcomes.

A key challenge facing the aforementioned systems lies in maximizing system payoffs by effectively assigning jobs to servers while respecting capacity constraints in the absence of complete knowledge about reward parameters. This task involves, among other things, balancing exploration and exploitation. Accurate learning of reward parameters is essential for high-quality matches, but learning efficiency depends on the matching approach. In addition, the inherent stochasticity of job arrivals leads to complex queueing dynamics that must be considered. Ignoring this randomness, as will be explained later on, can overload servers, causing excessive job backlogs, delays in reward feedback, and a large number of unattended jobs at the end of the planning horizon. In this paper, we present a simplified model that abstracts the problem described above. The model represents a service platform as a matching queue with I job types and J servers. Jobs are assigned to servers upon arrival and processed on a first-come, first-served basis. Completion of each job generates a reward following a Bernoulli distribution, dependent on job type and server. While the platform knows the job types and their respective arrival rates (the assumption regarding known arrival rates can be relaxed for our second proposed algorithm), it lacks knowledge of the mean rewards, necessitating the need to learn about the mean matching rewards through the observation of rewards when jobs complete services and exit the system.

Hsu et al. (2022) study a matching queue similar to ours, where "clients" arrive, each bringing in random numbers of tasks for processing. Based on the assigned server, task completions generate Bernoulli rewards with mean values that are unknown to the system operator. The system operator aims to maximize total payoff while learning the mean rewards, by dynamically assigning jobs to servers. Unlike our approach, where jobs are assigned first and then queued, in their study, tasks are queued first and then assigned. The concept of matching in their paper thus corresponds to "scheduling," enabling synchronization between job assignments and reward collection. In our scenario, matching is essentially "routing," causing delays between job assignment decisions and reward collection. As discussed in Hsu et al. (2022), eliminating server-side queues can be practical for certain applications like online

advertising and crowd-sourcing; however, it is not practically feasible for appointment-based services, such as digital healthcare platforms. In a broader context, our model and analysis hold relevance for service systems that require upfront knowledge of when and with whom the service will occur. In addition to being a crucial modeling distinction, the existence of server-side queues poses two significant technical challenges: (i) How can we analytically characterize the potential "learning slowdown" arising from the asynchrony between job assignments and reward collection? (ii) To what extent does this learning slowdown effect contribute to the overall regret?

We develop and provide performance guarantees for two algorithms that combine matching and learning to assign incoming jobs to appropriate servers while estimating unknown mean reward values to achieve maximum payoff and ensure stable queues. Both algorithms use upper-confidence-bound (UCB) estimates as surrogates for the unknown mean reward parameters, as in Auer et al. (2002). Each algorithm involves a design parameter that can be tweaked to balance reward maximization against service level. The design parameter influences the rate at which unknown reward parameters are learned and is critical to ensuring the algorithm's effectiveness.

- Our first algorithm is an adaptation of the one analyzed by Hsu et al. (2022), considering that queues are now on the server side instead of the job side. To address the issue of an undesirable feedback loop between queuing and learning, Hsu et al. (2022) propose a novel "utility-guided" learning and matching scheme, drawing upon an Oracle linear program that can be seen as an offline version of their problem. The "utility function" therein can effectively combat insufficient learning caused by excessive queueing by promoting fairness in task assignments. While our first algorithm also utilizes an Oracle linear program and includes a penalty term that, like the utility function considered in Hsu et al. (2022), promotes equitable job distribution, it admits an intuitive interpretation closer in spirit to the barrier method in optimization. To briefly explain, consider solving our Oracle linear program, assuming all reward parameters are known. Since the optimal solution of a linear program is typically located at a vertex, operationalizing the solution would often suggest that some servers would be operating at full utilization, causing queues at these servers to grow and leading to significant payoff losses in the end and poor service levels over time. To ensure non-binding capacity constraints, the aforementioned penalty term serves as a barrier, causing the objective value to drop drastically near the capacity boundaries. For this reason, we refer to this algorithm as "barrier-function-based" algorithm. Like the utility-guided algorithm in Hsu et al. (2022), our approach balances service payoff rate and level using a scalar variable, $1/V$. A higher value of V increases the payoff rate but results in more pending jobs and affects the learning rate due to delayed reward collection after job completion.
- Our second algorithm adjusts the (estimated) reward of each job-server pair by subtracting a queue-length term as a penalty that captures the congestion level at that server. It then assigns jobs to servers based on the highest adjusted reward. The idea of using queue lengths as "shadow prices" of server capacity constraints is not new, and it is proved to have near-optimal performance when the payoff vector is known (Tan and Srikant 2012). The magnitude of the penalty is determined by a design parameter, ϵ, which balances service payoff and level. Similar to the barrier-function-based algorithm's V, ϵ influences the learning speed of unknown reward parameters. Notably, a similar queue-based algorithm has been proposed as a benchmark policy in Hsu et al. (2022), where it has been

numerically tested to deliver suboptimal performance. Our extensive numerical investigations (including those presented in Section 7 and EC.3), on the other hand, do not seem to suggest the inferiority of our queue-based algorithm compared to the barrier-function-based algorithm. (We explain the disparity in results in §6.2.) These numerical findings are consistent with the analytical results, which indicate a slightly more competitive regret bound for the queue-based algorithm in terms of magnitude when compared to the barrier-function-based algorithm.

We view the contributions of this paper as twofold.

- We establish competitive regret bounds for two algorithms: one is a variant of the utility-guided algorithm proposed in Hsu et al. (2022), and the other utilizes queue lengths as "shadow prices" associated with capacity constraints. We define "regret" as the difference in total payoff accrued over the planning horizon compared to the maximum achievable payoff when all mean rewards are known and stochastic variability is absent. Our regret analysis, especially for the first algorithm, exploits some useful analytical tools and arguments developed in Hsu et al. (2022) and identify an important connection between the learning slowdown effect (due to server-side queues) at each server and the running maximum of its queue-length process. Consequently, our regret bounds reveal an additional term that accounts for the learning slowdown caused by longer queues, in addition to the three terms identified in Hsu et al. (2022). These three terms correspond to the payoff sacrificed to ensure stable queues, the loss of payoff caused by the platform's agnostic attitude towards mean rewards, and the payoff loss due to the job backlog at the end of the planning horizon. Since server-side queues are ubiquitous in human society (further details are provided in §6.1) and our approach to exposing and addressing the learning slowdown effect does not rely on the specific problem setup considered in this paper, the approach and managerial insights derived from our analysis are expected to be useful for conducting regret analysis in other service settings, particularly those lacking synchronization between a job assignment and its "reward" collection.
- We establish tight moment bounds for the running maxima of the queue-length processes. Such moment bounds are pivotal in establishing informative regret bounds for both of our proposed algorithms. As far as we are aware, convenient moment bounds for extreme values of queues are only available for $M/M/1$ or birth-and-death-type queues (Asmussen 1998, §3.1.), as well as systems that can be "dominated" by this type of queues (Gupta 2022, Lemma 3 and Theorem 2). For other types of queueing systems, while there is literature on the extreme values of queues, most papers focus on the asymptotic behavior of the running maximum, leaving us unable to draw on their results. To overcome this limitation, we establish useful moment bounds for extreme values of queues by leveraging the theory of left-skip-free random walks and establishing a series of stochastic ordering relations, each of which holds independent interest. Notably, since the "embedded chain" of an $M/G/1$ system can be viewed as a left-skip-free random walk, we believe that the new analytical tools we develop in this paper enhance the toolkit for investigating the transient behaviors of extreme values in systems that extend beyond the $M/M/1$ and systems alike.

Organization. The rest of the paper is organized as follows. Section 2 reviews the pertinent literature. In Section 3, we provide the problem formulation and set the stage for our analysis. Our two algorithms are introduced in Section 4. Section 5 presents analysis results of the two algorithms and establishes their performance guarantees. In Section 6, we offer further comments on some of the key modeling and analytical aspects of the study. We present the results of our numerical experiments in Section 7. Finally, we conclude in Section 8. Related proofs and some additional numerical investigations can be found in E-Companion.

2. Literature Review

Several two-sided queueing systems have emerged in various applications, such as public housing allocation and the sharing economy, leading to diverse matching models in the literature. For bipartite matching, Caldentey et al. (2009) consider a bipartite matching model, which has since been extended by Adan and Weiss (2012), Bušić et al. (2013), Adan and Weiss (2014), Adan et al. (2018). Afeche et al. (2022) explore optimal matching topologies between customer classes and servers. In another study, Hu and Zhou (2022) examine a stochastic matching problem with the aim of maximizing discounted match-dependent rewards. Multipartite matching has also been explored in the literature. Gurvich and Ward (2015) consider settings where items from multiple queues are matched instantly with the objective of minimizing system-wide holding costs, subject to compatibility constraints. Moyal and Perry (2017) investigate the instability of the matching model considered in Gurvich and Ward (2015). More recently, Kerimov et al. (2023) study a matching market where agents arrive randomly and can be matched in a multilateral fashion, producing heterogeneous matching values. Benchmarking the "hindsight optimality," the authors discover that the best optimality gap achievable at all times is inversely proportional to the "general position gap," which serves as a measure of the capacity slack, and they propose a simple periodic resolving policy that guarantees this optimality gap. In the sequel Kerimov et al. (2021), the same authors demonstrate that certain greedy policies can attain optimal all-time regret scaling when restricting attention to two-way matching networks. Furthermore, they illustrate how a static priority can be devised based on the network structure, resulting in constant regret, through a novel construction of a Lyapunov function. In a subsequent study by Gupta (2022), a modified form of greediness called the "smoothed general position gap assumption" is introduced. This assumption enables the attainment of constant regret even in networks beyond the two-way matching scenario. Importantly, the approach presented in Gupta (2022) accommodates both online and offline resources, making it applicable to a broader range of real-world applications.

This work contributes to the research on matching for one-sided queueing systems. Existing papers in this field mainly focus on scheduling optimization under compatibility constraints to minimize congestion-related costs. For example, in a recent study, Özkan and Ward (2020) analyze the problem of maximizing the number of matches in a parallel server model with a compatibility graph, motivated by ride-hailing systems. In contrast, our approach does not rely on external compatibility constraints but instead uses mean rewards to determine the suitability of matches. Several studies have considered match-dependent rewards in one-sided queues, motivated by

problems related to organ allocation (Ding et al. 2021) and housing allocation (Arnosti and Shi 2020). While we also consider match-dependent rewards, the quality of different matches is unknown to the system a priori, necessitating integrating learning with job assignment decisions, a key feature of our work.

Our work is part of a broader research area that combines learning and optimization for resource allocation problems in service settings with limited capacity. Massoulié and Xu (2018) address the problem of assigning inspections to accurately recover labels while maintaining system stability. Levi et al. (2019) investigate scheduling in service environments with uncertain job attributes and trade-offs between testing and processing. Krishnasamy et al. (2021) investigate a queuing system with multiple servers where jobs enter a queue for service with the goal of finding algorithms that learn the unknown service probabilities and minimize queue regret, and they show that queue regret can scale logarithmically. Chen et al. (2020) develop an online learning framework for dynamic pricing and capacity sizing in single-server queues. Their framework is designed specifically for a single-server queue and utilizes stochastic gradient descent techniques. Jia et al. (2022) propose learning algorithms for adaptive pricing to maximize revenue in the presence of finite reusable resources. In their paper, the decision maker does not know how arrival and service rates depend on posted prices and, hence, sets prices adaptively using past observations to maximize revenue. Zhong et al. (2022) study a scheduling problem in a multi-class many server queueing system with abandonment, with the goal of minimizing the expected abandonment and holding costs by learning the best scheduling policy when the model parameters are initially unknown. They propose an explore-then-exploit-style algorithm, which they show to be asymptotically optimal in a proper sense. While these studies focus on decisions like scheduling, pricing, or resource allocation, our work distinguishes itself by emphasizing reward maximization in matching queues while ensuring queue stability.

Several studies have examined the joint matching and learning problems in various domains. Johari et al. (2021) investigate a service platform with heterogeneous workers and limited job supply, optimizing steady-state payoff accumulation while learning user types. The work most closely related to ours is Hsu et al. (2022), where the authors investigate a similar matching problem. In their setting, jobs ("clients") have randomly generated service requirements ("tasks") that can be distributed across multiple servers, and they propose a utility-guided online learning and matching algorithm. Their setting is later extended by Kim and Vojnovic (2021) to allow for bi-linear rewards. Liu et al. (2020) consider a problem in which M types of jobs must be dispatched to N servers while adhering to capacity, fairness, and resource budget constraints. One key distinction of the present work, relative to the aforementioned papers, is that our model requires jobs to be assigned to a server upon their arrivals and wait in a queue to be served, whereas their models allow jobs to be first backlogged in queues and then sent to different servers at different times. In other words, our model includes server-side queues, resulting in delays in collecting rewards that slow down the learning process.

3. Model

Notations. In this paper, we use boldface notation to represent vectors or matrices, with subscripts indicating individual elements. For example, $\mathbf{q} \in \mathbb{R}^n$ represents an n-dimensional vector with q_i as its ith element. Similarly,

r_{ij} denotes the element at the ith row and jth column of matrix $\mathbf{r}$. The indicator function on set $\mathcal{A}$ is denoted as $\mathbf{1}_{\mathcal{A}}(\cdot)$. We use $[\cdot]^+$ and $[\cdot]^-$ to represent the positive and negative parts, respectively, defined as $\max(\cdot, 0)$ and $-\min(\cdot, 0)$.

Consider an online service platform providing service to I types of jobs indexed by $i \in \mathcal{I} := \{1, \ldots, I\}$ with J servers indexed by $j \in \mathcal{J} := \{1, \ldots, J\}$. Time is slotted, and the planning horizon has a finite length of T. The arrival of newly arriving jobs of type i over consecutive time slots follows a common distribution with a mean λ_i. Job arrivals are assumed to be independent across types and time steps. Servers are fully flexible in the sense that they can serve any types of jobs, and the service time is assumed to be 1. This assumption is reasonable for applications like digital healthcare platforms, where service providers typically offer fixed-length consultations. The decision maker can choose this time window as the length of a time slot. Let $\lambda = \sum_{i=1}^{I} \lambda_i$, and we assume $\lambda < J$. The realized reward for matching a type i job with server j follows a Bernoulli distribution with a mean $r_{ij} \in [0, 1]$. We summarize the expected matching rewards in the matrix $\mathbf{r} = [r_{ij}]$. We assume that $\min_{i,j} r_{ij} > 0$, and for technical reasons, we assume that the platform knows the value of $\min_{i,j} r_{ij}$ is lower bounded by a positive number r_*. This lower truncation of reward estimates is assumed to address technical considerations and can be set arbitrarily small. Our extensive numerical experiments indicate that the algorithms remain effective even without this truncation.

At the start of each time period, jobs arrive and the platform must make an immediate irreversible matching decision, assigning these new jobs to one of the J servers. Jobs waiting to be served by a server are processed in a first-come-first-served order. An admissible matching policy Π maps the current system state (including observed reward realizations by time t) to the probabilities $p_{ij}(t)$ of assigning a type i job to server j at time t. We denote the actual number of type i jobs assigned to server j in time slot t under matching policy Π as $\Gamma_{ij}(t)$.

Let $Q_{ij}(t)$ represent the number of type i jobs waiting in queue j at the end of time slot t. We also define $Q_j(t)$ as the number of jobs awaiting service by server j at the end of time slot t, where $Q_j(t) = \sum_i Q_{ij}(t)$. Since waiting costs are not explicitly considered and all jobs have the same service time, once a job is routed to a queue, it is treated the same as existing jobs in that queue. Hence, we focus on Q_j, which can be described by the following dynamics:

$$Q_j(t) = \max\left\{ Q_j(t-1) + \sum_{i=1}^{I} \Gamma_{ij}(t) - 1, 0 \right\}, \tag{1}$$

where we assume $Q_j(0) = 0$ for each j to represent an empty system at the beginning. The assumption that the system is initially empty serves to streamline the mathematical analysis. However, it can be relaxed to allow for strictly positive queue contents at the beginning without fundamentally impacting our regret analysis. Furthermore, let $B_{ij}(t)$ denote the number of type i jobs served by server j in time slot t. Since each server can process only one job at a time, we have:

$$\sum_{i=1}^{I} B_{ij}(t) = \min\left\{ Q_j(t-1) + \sum_{i=1}^{I} \Gamma_{ij}(t), 1 \right\}. \tag{2}$$

The platform has knowledge of the job types and arrival rates, denoted by λ_i, but lacks information about the expected matching rewards captured in the matrix $\boldsymbol{r}$. Therefore, the platform aims to learn these reward parameters

through observed feedback while maximizing the total payoff over the planning horizon T. Formally, the platform seeks a matching and learning scheme Π that maximizes the objective:

$$\max_{\Pi} R^{\Pi}(T) := \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}\mathbb{E}[B_{ij}(t)r_{ij}]. \tag{3}$$

Here, $R^{\Pi}(T)$ represents the expected total payoff accumulated over the time horizon. Note that any job that remains in the queue at the end of the horizon does not generate any reward, so there is an implicit congestion penalty in the objective function.

In our model, we assume known job types and arrival rates (the assumption of knowing the arrival rates can be dropped for the proposed queue-based algorithm), which sets it apart from the model in Hsu et al. (2022). This assumption is motivated by our application in digital healthcare platforms with access to abundant data, allowing patient classification based on factors like gender, age, and severity. Historical data enables accurate estimation of arrival rates, which tend to remain stable over the long term. In contrast, Hsu et al. (2022) assigns tasks to servers to learn client types gradually. However, our model represents specific appointment requests, unlike task-based queues. Implementing such queues would be impractical as patients need specific appointment times. In addition to causing delays between job assignment decisions and reward collection, this modeling distinction poses unique challenges for analysis compared to Hsu et al. (2022).

Oracle linear programming problem: In an ideal scenario with complete knowledge of the mean reward matrix $\boldsymbol{r}$ and without considering stochastic variability, the task of maximizing rewards can be formulated as a static planning problem. The corresponding linear program is given by:

$$\begin{aligned} \max_{p_{ij}\geq 0} \ & \sum_{i=1}^{I}\sum_{j=1}^{J}\lambda_i p_{ij} r_{ij} \\ \text{subject to } & \sum_{i=1}^{I}\lambda_i p_{ij} \leq 1 \text{ for all } j \\ & \sum_{j=1}^{J} p_{ij} = 1 \text{ for all } i \end{aligned} \tag{4}$$

Here, the decision variables p_{ij} represent the long-run fraction of type i jobs assigned to server j. The first set of constraints ensures that the expected number of jobs assigned to a server does not exceed its processing capacity. The second set of constraints ensures that all job types are assigned, leaving no jobs unassigned. The optimal solution and objective value of this program are denoted as $\boldsymbol{p}^* := [p^*_{ij}]$ and R^*, respectively.

The dual problem of the linear program (4) can be obtained straightforwardly. Let $\boldsymbol{\alpha} := [\alpha_j]$ represent the dual variables corresponding to the first set of constraints in (4), and $\boldsymbol{\beta} := [\beta_i]$ the dual variables for the second set of constraints. The dual problem can be formulated as follows:

$$\min_{\alpha_j,\beta_i} \sum_{j=1}^{J}\alpha_j + \sum_{i=1}^{I}\beta_i$$

$$\text{subject to } \lambda_i \alpha_j + \beta_i \geq \lambda_i r_{ij} \text{ for all } i, j$$

$$\alpha_j \geq 0 \text{ for all } j$$

The following result establishes useful properties of the optimal dual variables, which will be useful in the regret analysis for our first proposed algorithm.

LEMMA 1. *Let $\beta^* := [\beta_i^*]$ denote the optimal dual variables associated with the I equality constraints. Then under the condition $\lambda < J$, we have that $\beta_i^* > 0$ for all $i = 1, \ldots, I$.*

Lemma 1 reveals that the optimal dual variables associated with the equality constraints are strictly positive, suggesting that the platform has no incentive to leave any job unassigned, even if it is allowed to do so. Therefore, we can modify all of the equality constraints to be "less than or equal to" without affecting the optimal objective value of the primal problem. To provide some intuition, consider a hypothetical scenario where rejecting a portion of jobs is optimal in the primal solution. Given that the total capacity exceeds the demand, redirecting the rejected jobs to underutilized servers would lead to a strictly higher objective value, contradicting the assumption of optimality.

We define the *regret* of any policy Π as:

$$Reg^{\Pi}(T) := R^* T - R^{\Pi}(T), \tag{5}$$

where R^*T serves as an upper bound for $R^{\Pi}(T)$. Ideally, we aim for an algorithm with sub-linear regret, which means that the upper bound on $Reg^{\Pi}(T)$ can be limited by a sub-linear function of the time horizon T. Astute readers may have observed that we have not yet provided a detailed description of the distributions or distributional properties of job arrivals beyond their means in our model. The choice of distributional assumption for job arrivals does impact the performance analysis of our proposed algorithms. Consequently, we present two distinct sets of assumptions regarding job arrival processes. The first set of assumptions relates to our first algorithm, while the second set of assumptions pertains to our second algorithm.

ASSUMPTION 1. *The number of newly arriving jobs of each type over consecutive time slots follows a Poisson distribution.*

ASSUMPTION 2. *The distribution dictating the number of new arrivals of each type per time slot has a finite support, so that the total number of arrivals per slot is upper-bounded by a constant M.*

All our analytical results will be stated under Assumption 1 *or* 2.

4. Proposed Algorithms

In this section, we present two algorithms: the barrier-function-based algorithm and the queue-based algorithm.

4.1. Barrier-Function-Based Algorithm

We propose a barrier-function-based algorithm inspired by Hsu et al. (2022). The algorithm consists of three steps, outlined in Algorithm 1. In the first step, we use past observations to calculate the UCB estimates of the rewards. These estimates, denoted as $\bar{r}_{ij}(t-1)$, are based on the average payoff of serving $h_{ij}(t-1)$ type i jobs by server j until time $t-1$. The UCB estimates at time t are obtained using (6). The second step involves solving an optimization problem defined by (7). This problem aims to maximize the total payoff, considering both the current UCB estimates of the rewards and a penalty term. The penalty term, $\log(1-\sum_{i=1}^{I}\lambda_i p_{ij}(t))$, ensures that no server operates at full capacity constantly, promoting more balanced job assignments and preventing queues from growing excessively. In the last step, jobs are assigned according to the optimal solution of the optimization problem, represented as $\boldsymbol{p}(t) := [p_{ij}(t)]$. We also observe rewards from the current period, which are used to update our reward estimates.

One commonality between our barrier-function-based algorithm and the utility-guided algorithm in Hsu et al. (2022) is a logarithmic term serving to promote a notion of fairness. Specifically, in Hsu et al. (2022), the logarithmic utility function is added to ensure fairness among clients, allowing clients even with very low payoff estimates to receive some service, potentially resolving ineffective learning caused by large backlogs of tasks whose matching values are yet to be learned. The logarithmic term in our barrier-function-based algorithm serves to prevent the solution from approaching the capacity boundaries, promoting a more equitable allocation of work among the servers. We use a logarithmic barrier function because it is the most common barrier function while facilitating the mathematical analysis. Given its nature, we expect other barrier functions to do the trick as well.

The choice of V in (7) strikes a balance between reward maximization and maintaining fairness in job assignment across the servers. The insight can be gained more effectively when all reward parameters are known, in which case one can look at the following optimization problem:

$$\begin{aligned} \max_{p_{ij}} \ & \sum_{j=1}^{J}\left\{\frac{1}{V}\log\left(1-\sum_{i=1}^{I}\lambda_i p_{ij}\right)+\sum_{i=1}^{I}\lambda_i p_{ij} r_{ij}\right\} \\ \text{subject to} \ & \sum_{j=1}^{J} p_{ij} = 1 \text{ for all } i \\ & p_{ij} \geq 0 \text{ for all } i,j \end{aligned} \tag{8}$$

Let $\tilde{\boldsymbol{p}} := [\tilde{p}_{ij}]$ be the optimal solution to the optimization problem (8). By utilizing $\tilde{\boldsymbol{p}}$ in a similar manner to Algorithm 1, one can perform probabilistic job assignments. The optimization problem (8) can be seen as a series of optimization problems, indexed by V, where the objective is to maximize rewards. As V approaches infinity, the performance of probabilistic routing using $\tilde{\boldsymbol{p}}$ is expected to approach the performance achieved by the linear program (4) in terms of reward maximization. This intuition can be formalized using the following proposition.

Proposition 1. *For $\tilde{\boldsymbol{p}}$ as defined above, we have that*

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\lambda_i(p^*_{ij}-\tilde{p}_{ij})r_{ij} \leq \frac{J}{V}.$$

Algorithm 1: Barrier-Function-Based Algorithm

Data: $\lambda_i, \forall i = 1, 2, ..., I$ and V

For every time slot $t = 1, ..., T$:

Step 1: Form truncated UCB mean reward estimates

for $i = 1, 2, ..., I$ **do**
  for $j = 1, 2, ..., J$ **do**
    if $h_{ij}(t-1) = 0$ **then**
      $r_{ij}(t) = 1.$
    end
    else
      Set

$$r_{ij}(t) = \max\left\{\min\left\{\bar{r}_{ij}(t-1) + \sqrt{\frac{2\log(t-1)}{h_{ij}(t-1)}}, 1\right\}, r_*\right\}. \tag{6}$$

    end
  end
end

Step 2: Solve $p_{ij}(t)$ for the optimization problem

$$\begin{aligned} \max_{p_{ij}(t)} \ & \sum_{j=1}^{J}\left\{\frac{1}{V}\log\left(1 - \sum_{i=1}^{I}\lambda_i p_{ij}(t)\right) + \sum_{i=1}^{I}\lambda_i p_{ij}(t) r_{ij}(t)\right\} \\ \text{subject to } & \sum_{j=1}^{J} p_{ij}(t) = 1 \text{ for all } i \\ & p_{ij}(t) \geq 0 \text{ for all } i, j \end{aligned} \tag{7}$$

Step 3: Assign jobs and obtain noisy reward feedback

for $i = 1, 2, ..., I$ **do**
  Assign each type i job that arrives at time t to the queue at server j with probability $p_{ij}(t)$.
end

for $j = 1, 2, ..., J$ **do**
  if *the queue at server j is not empty* **then**
    Observe the type of the first job in the queue i^*, and the reward $X_{i^*j} \sim Bern(r_{i^*j})$. Update
    $h_{i^*j}(t) = h_{i^*j}(t-1) + 1$ and $\bar{r}_{i^*j}(t) = (h_{i^*j}(t-1)\bar{r}_{i^*j}(t-1) + X_{i^*j})/h_{i^*j}(t)$.
  end
end

Proposition 1 demonstrates that introducing the logarithmic barrier to ensure queue stability leads to a loss in payoff of no more than J/V per unit time compared to the offline static problem. Furthermore, as V approaches infinity, each $\tilde{\beta}_i$, representing the optimal dual variable associated with the I equality constraints in (8), is expected to approach β_i.

4.2. Queue-Based Algorithm

At a high level, the proposed queue-based algorithm combines estimated rewards and queue lengths to assess the relative appeal of each server when assigning jobs. To be more precise, the queue lengths are now utilized as "shadow prices" associated with the inherent capacity constraints. The complete description of the queue-based algorithm is provided in Algorithm 2. In this algorithm, $k(i, \mathbf{Q}(t-1), \mathbf{r}(t))$ represents the smallest index of servers, with a maximum value of $r_{ij}(t) - \epsilon Q_j(t-1)$ at time t, for each i.

Algorithm 2: Queue-Based Algorithm

For every time slot $t = 1, ..., T$:

Step 1: Form truncated UCB mean reward estimates

for $i = 1, 2, ..., I$ **do**
  for $j = 1, 2, ..., J$ **do**
    if $h_{ij}(t-1) = 0$ **then**
      $r_{ij}(t) = 1$.
    end
    else
      Set

$$r_{ij}(t) = \max\left\{\min\left\{\bar{r}_{ij}(t-1) + \sqrt{\frac{2\log(t-1)}{h_{ij}(t-1)}}, 1\right\}, r_*\right\}.$$

    end
  end
end

Step 2: Assign jobs and obtain noisy reward feedback

for $i = 1, 2, ..., I$ **do**
  Assign each type i job that arrives at time t to the queue at server

$$k(i, \mathbf{Q}(t-1), \mathbf{r}(t)) := \min\left(\arg\max_j \quad r_{ij}(t) - \epsilon Q_j(t-1)\right), \tag{9}$$

end

for $j = 1, 2, ..., J$ **do**
  if *the queue at server j is not empty* **then**
    Observe the type of the first job in the queue i^*, and the reward $X_{i^*j} \sim Bern(r_{i^*j})$. Update
    $h_{i^*j}(t) = h_{i^*j}(t-1) + 1$ and $\bar{r}_{i^*j}(t) = (h_{i^*j}(t-1)\bar{r}_{i^*j}(t-1) + X_{i^*j})/h_{i^*j}(t)$.
  end
end

A similar queue-based control policy has been proposed and numerically studied in Hsu et al. (2022). Their study finds that a queue-based policy can lead to suboptimal outcomes, which can be attributed to the "vicious cycle" that arises between queuing and learning when server-side queues are allowed. Our regret bound for our queue-based

algorithm does not appear to contain any indication of inferior performance (in terms of magnitude) compared to our first algorithm. This theoretical finding is further substantiated by numerical evidence, which we will present later, demonstrating the highly favorable performance of our queue-based algorithm.

The parameter ϵ, which adjusts the magnitude of the penalty in the algorithm, helps to strike a balance between reward maximization and fairness in work allocation. In the event that all reward parameters r_{ij} are known, Algorithm 2 simplifies to a routing rule that assigns type i jobs to server

$$k(i, \mathbf{Q}(t-1)) := \min \left(\arg\max_j \quad r_{ij} - \epsilon Q_j(t-1) \right). \tag{10}$$

When $\epsilon = 0$, the rule assigns all type i jobs to the server with the highest r_{ij} without considering congestion. Conversely, as ϵ approaches infinity, the rule resembles the "join-the-shortest-queue" (JSQ) rule, which aims to distribute jobs equitably among all servers. Thus, the rule (10) strikes a balance between these two goals. The proposition below formalizes this intuition by providing performance guarantees for a system with complete knowledge of reward parameters and operating under the routing rule (10).

Proposition 2. *Suppose the platform has full knowledge of all the reward parameters r_{ij} and operates under the routing rule* (10)*. Let $B_1 := M^2 - 2\lambda + J$. Then under Assumption 2, the following statements are true:*

(i) The J-dimensional process $\boldsymbol{Q}$ has a steady-state distribution $\boldsymbol{Q}(\infty)$ and

$$\sum_{j=1}^{J} \mathbb{E}[Q_j(\infty)] \le \frac{J}{2(J-\lambda)} \left(\frac{2\lambda}{\epsilon} + B_1 \right).$$

(ii) For all $t \ge 1$,

$$\sum_{j=1}^{J} \mathbb{E}[Q_j(t)] \le J \left(\frac{1}{\epsilon} + M \right) + \frac{M^2 - 2\lambda J + J^2}{2(J-\lambda)}.$$

(iii) For any fixed T,

$$\widehat{Reg}^{QB}(T) := R^* T - R^{QB}(T) \le \frac{B_1 \epsilon}{2} T + \sum_{j=1}^{J} \mathbb{E}[Q_j(T)].$$

The proof of part (i) of the proposition utilizes Lyapunov drift analysis and Foster's theorem, providing insights into the steady-state moments of queues under load-balancing algorithms. This result could be of independent interest in the context of the literature on establishing steady-state moments for queues under load-balancing algorithms, as the rule (10) extends the well-known JSQ rule. For example, the proof of steady-state moments for JSQ, Eryilmaz and Srikant (2012) draws upon the establishment of a state-space collapse result. However, proving an appropriate state-space collapse result under rule (10) seems very difficult, if not impossible. Our proof leverages the mechanism of rule (10) in one-step drift analysis, eliminating the need for a state-space collapse result. Part (ii) of the proposition focuses on the transient moments of queue lengths. Its proof builds on a key observation: as long as the total number of jobs in the system exceeds some threshold, no server can be idle under rule (10). Consequently, the J servers collectively form a "super server" that works J times faster than individual servers, behaving like a

single-server queue when the total number of jobs passes that threshold. Combining parts (i) and (ii) reveals that all queues are stable and their lengths are at most of the order $1/\epsilon$ under rule (10). Part (iii) indicates that the loss of payoff relative to the offline static problem over a time horizon of length T is on the order of $\sqrt{T}$ when $\epsilon = 1/\sqrt{T}$. The proof employs the "drift-plus-penalty" method, which has been employed extensively in related literature for performance analysis in stochastic systems; see, e.g., Neely (2010).

5. Performance Analysis

This section explores the stability of queue lengths and provides a regret bound for both algorithms.

5.1. Analysis of Algorithm 1

Throughout this subsection, we choose V large enough so that $V \geq (1/r_*)(J/(J-\lambda))$ and $\tilde{\beta}_i \geq \beta_i/2$ for all i. Here we recall that $\tilde{\beta}_i$ and β_i correspond to the optimal dual variables associated with (8) and (4), respectively. The first question that naturally arises is whether the system is stable under Algorithm 1. This is addressed by the following theorem.

THEOREM 1. *Assume the platform employs Algorithm 1 and Assumption 1 holds. Then the following statements hold:*

(i) For every $t \geq 1$, $\mathbb{E}[Q_j(t)] \leq V + 1$ for every j.

(ii) For any fixed time horizon T,

$$\mathbb{E}[Q_j^{\max}(T)] \leq V \log T + V + 1 \tag{11}$$

for every j, where $Q_j^{\max}(T)$ denotes the running maximum of Q_j up to T.

Theorem 1 has two parts. Part (i) shows that using the barrier-function-based algorithm ensures system stability. Part (ii) establishes moment bounds for the running maximum of each queue when the system operates under Algorithm 1. This result is useful for establishing the regret bound for Algorithm 1. Specifically, the bound in (11) represents the moment bound for the running maximum of a discrete-time $GI/D/1$ queue (fed by a number of new arrivals during each time period, following a Poisson distribution with a mean of $1 - 1/V$, and having a deterministic service time of 1), which stochastically dominates each Q_j under the proposed algorithm. We will elaborate on the theoretical value of this result in §6.3.

As mentioned in Asmussen (1998), the literature on extreme value theory for queues is quite limited, with existing papers primarily focusing on asymptotic extreme-value limit theorems. These papers include Cohen (1968), Iglehart (1972), Serfozo (1988); see also Berger and Whitt (1995) for an insightful discussion on engineering approximations based on these limit results. Essentially, these limit theorems promote approximations of the form:

$$Q^{\max}(t) \approx \gamma\,(\log t + Z) \quad \text{for} \quad \text{sufficiently large } t, \tag{12}$$

where $Q^{\max}$ represents the running maximum process of Q, a generic queueing process, and Z follows a Gumbel distribution. Moreover, γ is a constant that can be approximated as $\mathbb{E}[Q(\infty)]$ (Berger and Whitt 1995). By comparing

the approximation in (12) with the right-hand side of (11) and considering part (i) of Theorem 1, it becomes evident that $V \log T + V + 1$, when viewed as the moment bound for the $GI/D/1$ queue discussed earlier, is fairly tight. To establish a tight moment bound for the running maximum, we leverage the theory of left-skip-free random walks and prove a series of stochastic ordering relations.

Having verified that all queues are stable, we now turn to establishing the regret bound for the barrier-function-based algorithm. Our starting point is to note that

$$Reg^{\text{BB}}(T) \leq \mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}\lambda_i(p_{ij}^* - p_{ij}(t))r_{ij}\right] + \mathbb{E}\left[\sum_{j=1}^{J}Q_j(T)\right]. \tag{13}$$

The first term on the right-hand side can be understood as the payoff gap between the payoff achieved by the algorithm and the benchmark value R^*T if all jobs were served at the end of time T. The second term on the right can be viewed as a correction term, capturing the loss of payoff due to jobs in the queue at the end of the considered time horizon. For the purpose of establishing the regret bound, it is convenient to further decompose the first term on the right-hand side of (13) to get

$$Reg^{\text{BB}}(T) \leq T\sum_{i=1}^{I}\sum_{j=1}^{J}\lambda_i(p_{ij}^* - \tilde{p}_{ij})r_{ij} + \mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}\lambda_i(\tilde{p}_{ij} - p_{ij}(t))r_{ij}\right] + \mathbb{E}\left[\sum_{j=1}^{J}Q_j(T)\right]. \tag{14}$$

From (14) one can see that a regret bound can be obtained if each term on the right-hand side can be properly bounded.

Proposition 3. *Suppose the platform operates under Algorithm 1. Then under Assumption 1, we have that*

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{j=1}^{J}\lambda_i(\tilde{p}_{ij} - p_{ij}(t))r_{ij}\right] \leq 2K\sqrt{2\log T}\left(2IJ + 2J\sqrt{IT} + \sum_{i,j}\mathbb{E}\left[Q_{ij}^{\max}(T)\right]\right) + \frac{14}{3}KJ\lambda, \tag{15}$$

where K is some positive constant dependent on the model primitives.

Proposition 3 involves the running maxima of queue lengths. To put this result into perspective, we remark that the notion of the extreme values of queues does not appear in the regret bound derived in Hsu et al. (2022) or Kim and Vojnovic (2021). This discrepancy arises due to the presence of server-side queues, which introduce delays between the actions of matching and the collection of rewards. Hsu et al. (2022) eliminate server-side queues by requiring jobs to queue before being dispatched, leading to perfect synchronization between matching and reward collection. Notably, the regret analysis conducted by Hsu et al. (2022) capitalizes on this "instant feedback" mechanism, which, in combination with a clever martingale argument they develop, yields an informative regret bound. However, our proof cannot leverage this mechanism. As a result, we divide the loss in payoff related to parameter learning into two components. The first component corresponds to the loss resulting from learning unknown reward parameters and can be analyzed similarly to Hsu et al. (2022). To address the second component, we employ a careful sample-path analysis. This, in conjunction with summation by parts, allows us to relate the

delay in receiving reward feedback (and hence learning slowdown) to the running maximum of the queue length. In summary, while our analysis shares some similarities with the regret analysis conducted in Hsu et al. (2022) for the utility-guided algorithm, there is a significant departure due to a fundamental difference in modeling. The novel linkage we discover between learning slowdown and extreme values of queues is expected to have broader relevance in other contexts as well (see §6.1 for more details).

The main result pertaining to Algorithm 1 is Theorem 2, which follows as a direct consequence of (14), Propositions 1 and 3, and Theorem 1.

THEOREM 2. *Assume the platform employs Algorithm 1 and Assumption 1 holds. Then, for any fixed T, we have that $Reg^{BB}(T)$ is less than or equal to*

$$\underbrace{\frac{JT}{V}}_{\textit{first term}} + \underbrace{\left[2JK\sqrt{2\log T}\left(2I+2\sqrt{IT}\right)+\frac{14}{3}KJ\lambda\right]}_{\textit{second term}} + \underbrace{2JK\sqrt{2\log T}\left(V\log T+V+1\right)}_{\textit{third term}} + \underbrace{J(V+1)}_{\textit{fourth term}}.$$

In the regret bound, there are four dominant terms coming from four different sources. The first term, JT/V, is the price we pay for keeping the queues stable. The second term is the price paid to learn the unknown mean rewards. The third term comes from the learning slowdown due to longer queues. The last term, $J(V+1)$, is due to the payoff loss caused by the backlogged jobs at the end of the planning horizon. Since a large V favors allocating jobs to servers with higher rewards but also promotes longer queues, the regret bound exposes the inherent tension between the goal of learning the unknown reward parameters and reward maximization. Finally, to see that the barrier-function-based algorithm achieves a sub-linear regret, it suffices to let $V=\sqrt{T}$, yielding a regret bound in the order of $\sqrt{T(\log T)^3}$.

5.2. Analysis of Algorithm 2

We now provide performance guarantees for the queue-based algorithm under the assumption that job arrival processes meet the conditions specified in Assumption 2. Note that the use of this algorithm implicitly assumes that all reward parameters are unknown.

THEOREM 3. *Suppose the platform operates under Algorithm 2 and Assumption 2 holds. Then the following statements are true:*

(i) For all $t \geq 1$,

$$\sum_{j=1}^{J}\mathbb{E}\left[Q_j(t)\right] \leq J\left(\frac{1}{\epsilon}+M\right)+\frac{M^2-2\lambda J+J^2}{2(J-\lambda)}.$$

(ii) Let $Q_\Sigma^{\max}(t)$ denote the running maximum of $Q_\Sigma := \sum_j Q_j$ up to time t, then

$$\mathbb{E}\left[Q_\Sigma^{\max}(T)\right] \leq J(1/\epsilon+M)+J(M-1)\left(\frac{Jq}{J-\lambda}\log T+\frac{Jq}{J-\lambda}+1\right),$$

with q being some constant depending on model primitives.

(iii) For any fixed T, $Reg^{QB}(T)$ is less than or equal to

$$\underbrace{\frac{\epsilon B_1 T}{2}}_{\text{first term}} + \underbrace{\left[4J\sqrt{2\log T}(I+\sqrt{IT}) + \frac{14}{3}J\lambda\right]}_{\text{second term}} + \underbrace{\left(2IJ\sqrt{2\log T}\right)\mathbb{E}\left[Q_\Sigma^{\max}(T)\right]}_{\text{third term}} + \underbrace{\sum_{j=1}^{J}\mathbb{E}[Q_j(T)]}_{\text{fourth term}},$$

where $B_1 = M^2 - 2\lambda + J$ as defined in Proposition 2.

Part (i) of Theorem 3 shows that all queues are stable, with an order of magnitude of $1/\epsilon$. Part (ii) of the theorem implies that the running maximum of the total number of jobs in the system under Algorithm 2 is bounded by the sum of two major components, one in the order of $1/\epsilon$ and one of order $\log T$. More importantly, part (iii) of the theorem reveals four major terms in the regret function. The first term, $\epsilon B_1 T/2$, is the price we pay for keeping the queues stable. The second term, which is of order $\sqrt{T\log T}$, corresponds to the price paid in order to learn the unknown reward parameters. The third term, which is of order $\sqrt{\log T}/\epsilon$ (consider ϵ such that $O(1/\epsilon) \gg O(\log T)$), represents the learning slowdown effect. The last term, linked to expected queue lengths, is of order $1/\epsilon$ by part (i) of the theorem and captures the payoff loss due to jobs backlogged in the queues at the end of the planning horizon.

A major challenge in proving Theorem 3 arises from its part (ii). When using *probabilistic routing* as implemented in Algorithm 1, it is convenient to construct an "upper-bounding" process for each queue without considering information about other queues. With queue-based routing, as in (9), the evolution of each queue is inherently correlated and heavily influenced by the behavior of other queues. Consequently, it is not possible to construct an "upper-bounding" process for each queue, disregarding what has occurred in other queues. To remedy this, our proof utilizes a coupling argument. Specifically, as a key step in the analysis, we employ the aggregate queue of a decentralized parallel service system to bound the queue of the corresponding centralized system. This approach allows us to overcome the inherent correlations and dependencies between individual queues under the queue-based algorithm, ultimately leading to an informative regret bound as shown in part (iii) of the theorem.

Finally, to see that the queue-based algorithm indeed achieves a sub-linear regret, it suffices to let $\epsilon = 1/\sqrt{T}$, yielding a regret bound of order $\sqrt{T\log T}$, which is slightly better (in magnitude) than that for the barrier-function-based algorithm.

6. Discussion

In this section, we provide further comments regarding the real-world scenarios in which our analysis and insights could potentially be applied. We also discuss the potential limitations of certain model assumptions and their possible relaxation, as well as the theoretical value of the moment bounds derived for the running maxima of the queue-length processes.

6.1. Job-Side vs. Server-Side Queues

Server-side queues in our study introduce a delay in receiving reward feedback, as rewards are realized only after job completion. In contrast, the paper by Hsu et al. (2022) eliminates server-side queues by allowing job assignments to be made after servers become available, creating immediate reward feedback. This arrangement is useful in applications like online advertising and crowd-sourcing, as discussed in Hsu et al. (2022). However, eliminating server-side queues is not practical for appointment-based services, including digital healthcare platforms. These systems require upfront knowledge of when and with whom the service will occur.

Our analysis holds relevance in other systems where jobs need to be routed to an appropriate server upon entry. One such example is the manuscript review system, where the jobs are manuscript submissions and the servers are domain experts (such as associate editors). In this context, rewards signify the suitability of assigning a specific domain expert to handle a manuscript, and matching corresponds to manuscript assignments by the chief or handling editor. For one thing, it is crucial for the journal to consider the current workload of each domain expert to avoid overburdening any individual. Additionally, the journal may lack prior knowledge of each domain expert's specific manuscript expertise, particularly for newly appointed editorial board members. Consequently, there is a need for the journal to gradually acquire knowledge about the strengths and expertise of these domain experts over time. In this regard, our model captures, to a reasonable extent, the adaptive process that enables a journal to optimize its manuscript assignments, ensuring that each submission is reviewed by the most suitable expert while also maintaining a reasonably fair distribution of the workload among the domain experts.

6.2. Model Ingredients

The assumption that the service rates are identical and set to one is arguably restrictive. However, both our proposed algorithms can be easily adapted to handle scenarios with server-dependent service rates. To incorporate server-dependent service rates into the barrier-function-based algorithm, it suffices to appropriately modify the convex optimization problem (7). As for the queue-based algorithm, essentially no changes are required. Theoretical results can likely be extended to accommodate server-dependent service rates as well, given that the majority of our proofs exhibit little to no reliance on the current service rate assumption (with perhaps the only exception being the proof of moments for the running maximum of each queue, which utilizes the theory of left-skip-free random walks).

Our paper assumes prior knowledge of job types, which is reasonable for applications like digital healthcare platforms. Hsu et al. (2022), on the other hand, allows for on-the-fly learning of unknown client types based on reward feedback from the assigned tasks. In many contexts, job types may not be perfectly observed but can be inferred from observed features, as explained in Singh et al. (2022). A middle ground between assuming perfect knowledge of job types and assuming complete type uncertainty is, thus, to assume that types can be inferred from job features. This idea is embraced in the recent study Kim and Vojnovic (2021), where the authors consider multi-class, multi-server queueing systems with stochastic bi-linear rewards and propose a scheduling algorithm incorporating a linear bandit subroutine to estimate assignment rewards.

In Hsu et al. (2022), the authors propose a queue-based algorithm that requires each client to give out one task to a server in every round, regardless of whether the algorithm has already identified the most profitable server(s). Meanwhile, since the platform lacks information about the client types, the learning process needs to be restarted for each new client, even if a client from the same class has previously arrived and left. As a result, it could happen that as the algorithm gains confidence in the superiority of one server over another for a client, the client is already close to leaving the system after completing most of its tasks. This situation, in our view, explains the significant loss in payoff as observed in Hsu et al. (2022) under their queue-based algorithm. To see why our queue-based algorithm continues to perform effectively, it is helpful to view each job type in our context as "permanent clients," as it allows for increasingly accurate estimates of server profitability over time. In summary, we attribute the difference in observed performance of the queue-based algorithm between Hsu et al. (2022) and our paper to the disparity in assumptions regarding knowledge of job types.

To further verify the foregoing intuition, we explore, in §EC.3.2 of the e-companion, a scenario where we also introduce the concept of "clients" segmented into I distinct types. These clients enter the system, stay for a random duration, and then depart. However, unlike in Hsu et al. (2022), where each client brings a batch of tasks upon arrival, we assume that each client in our system generates jobs at a constant rate, resulting in a "stationary" input stream. Since the platform does not observe the exact type of a new client, it needs to learn the client's type through the jobs it generates. Our numerical study reveals that as the "sojourn times" of clients decrease, the performance of our queue-based algorithm deteriorates, aligning with our expectations.

6.3. The Value of the Moment Bounds for Extreme Values of Queues

As illustrated in the influential survey paper by Asmussen (1998), for a wide range of queueing models, analyzing the running maximum of a queue often boils down to studying the maximum over a busy cycle, leveraging the regenerative structure of the system. For birth-and-death queues, which are discrete-time analogues of $M/M/1$ systems, it is straightforward to show that the cycle maximum follows a geometric distribution, allowing for explicit expressions. This observation is ingeniously exploited in Gupta (2022), where the author establishes that a random walk with bounded step sizes and desired negative drift can be dominated by a "birth-and-death queue" with a properly inflated step size. This result has sweeping power, enabling the bounding of the running maximum of a queue, modeled as a reflected random walk with finite step sizes, by the running maximum of a "birth-and-death queue" with a larger identical step size. However, directly applying this result to our specific scenario is not possible, as the underlying random walks associated with our queues do not satisfy the finite-size requirement when job arrivals from each type follow a Poisson distribution. To address this, we tap into the theory of left-skip-free random walks, noting that the queues in our setting can decrease in length by at most one. With a series of established stochastic ordering relations, we derive informative moment bounds for the extreme values of queues in our context. Furthermore, considering that the "embedded chain" of an $M/G/1$ system can be interpreted as a left-skip-free random walk (with reflection), we anticipate the new analytical techniques we introduce in this paper to expand the

set of tools available for studying the transient behavior of extreme values in systems that go beyond the scope of $M/M/1$ and similar systems.

7. Numerical Experiments

We conduct numerical experiments to showcase the effectiveness of our proposed algorithms. We start by introducing the benchmark and then demonstrate that our algorithms achieve regret values in simulation that align with our theoretical guarantees and outperform the benchmark. We also explore how different model and design parameters impact the performance of our algorithms. Additionally, we examine scenarios where the reward-generating process or arrival process deviates from our assumptions to evaluate the robustness of our algorithm. Finally, we apply our algorithm to real-world data extracted from a large Chinese digital healthcare platform. In our experiments, we use the plain greedy algorithm as our benchmark. To ensure fairness, the greedy algorithm also estimates unknown rewards using the UCB approach, following the same procedure as Algorithm 2, except for substituting line 15 with:

"Assign each type i job that arrives at time t to the queue at server $\operatorname{argmax}_j r_{ij}(t)$."

In §EC.3.1 of the e-companion, we explore alternative two-stage algorithms that involve exploration in the first stage and exploitation in the second stage. We utilize these explore-and-exploit-style algorithms as alternative benchmarks.

7.1. Simulation Results

In the simulation experiments, we set $I = 2$ and $J = 6$, and generate arrivals from a Poisson distribution with $\lambda = [2, 3]$. Additionally, we set $T = 1000$, $V = 1000$ in the barrier-function-based algorithm, and $\epsilon = 0.01$ in the queue-based algorithm. The values of V and ϵ will be varied later on. At the beginning of the experiment, we randomly generate mean rewards for each possible assignment from $Unif(0, 1)$.

The left plot in Figure 1 shows the cumulative reward, while the right plot displays the average regret of each algorithm. The orange solid line represents the greedy algorithm, the yellow dotted line represents the queue-based algorithm, and the blue dashed line represents the barrier-function-based algorithm. Both of our proposed algorithms perform exceptionally well and outperform the greedy algorithm, as observed in the left plot. In fact, the cumulative rewards of our algorithms approach the upper bound represented by the purple circled line. Moving to the right plot, we observe that the average regret of the greedy algorithm remains significant and does not decrease properly with time. In contrast, the average regret of our proposed algorithms decreases rapidly initially and gradually approaches zero. The purple circled line corresponds to the function $y = \sqrt{\log(t)}/\sqrt{t}$. It is worth noting that the average regret of our proposed algorithms closely aligns with this line, which is consistent with our theoretical results.

Apart from examining the reward and regret, we also analyze other aspects of the system, such as queue lengths at different servers under different algorithms. Due to the presence of $J = 6$ servers in our example, we focus on

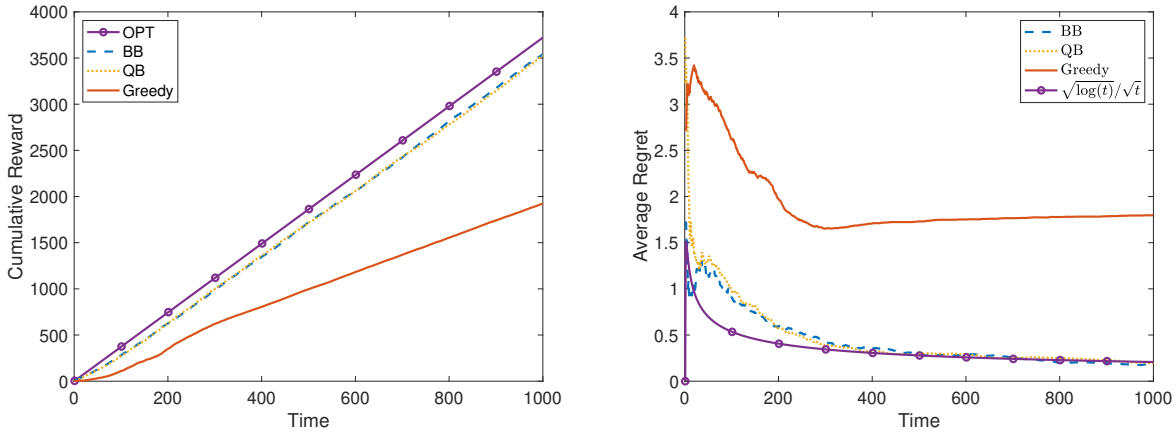

Figure 1 Cumulative reward (left) and average regret (right) of different algorithms

a few representative servers to maintain clarity in the figures. Figure 2 showcases the queue lengths at different servers for the greedy, queue-based, and barrier-function-based algorithms. The orange solid line represents the greedy algorithm, the yellow dotted line represents the queue-based algorithm, and the blue dashed line represents the barrier-function-based algorithm. The left plot illustrates the queue length at server 6, which has the highest matching reward for both job types. We observe a significant increase in the queue length under the greedy algorithm. This is expected as the greedy algorithm, after an initial learning period, forms a relatively accurate estimate of each pairing reward and assigns all jobs to server 6, the most profitable server. In contrast, the queue length remains relatively stable under both of our proposed algorithms, avoiding a sharp increase in queue length. This demonstrates that neither of our proposed algorithms assigns all jobs to server 6, even after learning its high profitability, as their objective is to avoid concentrating all jobs on one server. The right plot displays the queue length at server 5. As server 5 has the lowest reward for type 1 jobs and a relatively low reward for type 2 jobs, it is considered a less desirable server. For the greedy algorithm, after an initial learning period, it becomes more and more evident that server 5 is not the most profitable server for either job type and stops assigning any jobs to it. This is evident from the sharp decrease in queue length starting around time 170. In contrast, both of our proposed algorithms still assign some jobs to server 5, even though they have also learned that it may not be the best server, in order to balance the queues in the system.

In Table 1, we present the average queue length at each server and the average waiting time for each job type. As shown in the left part of the table, our proposed algorithms result in a much more balanced distribution of the average queue length compared to the greedy algorithm. Using Little's law, we can infer that the average waiting time for jobs under our proposed algorithms will be smaller than that under the greedy algorithm. This is supported by the right part of Table 1.

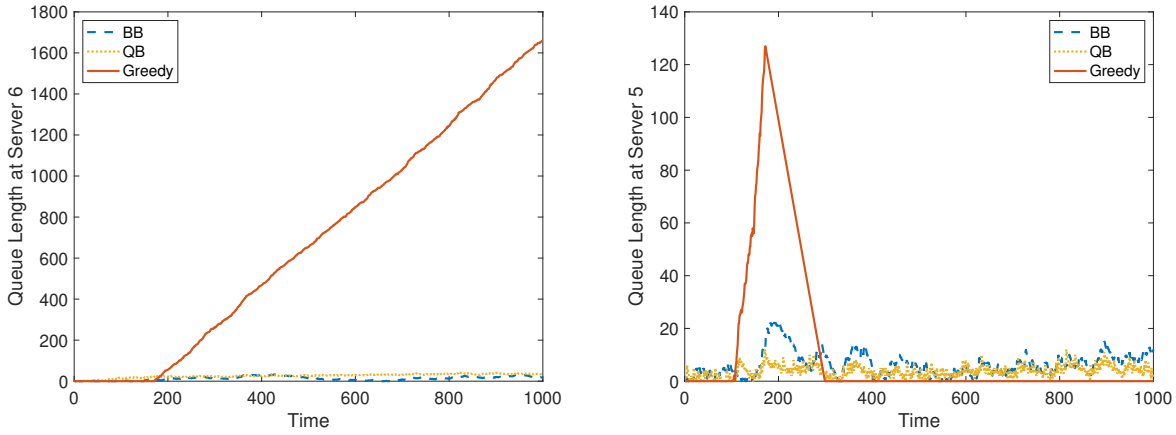

Figure 2 Queue length at server 6 (left) and server 5 (right)

	Average Queue Length						Average Waiting Time	
	Server 1	Server 2	Server 3	Server 4	Server 5	Server 6	Type 1 Jobs	Type 2 Jobs
Greedy	14.386	28.798	1.253	341.975	11.735	686.537	165.703	187.765
BB	5.215	7.968	0.116	21.366	6.408	13.385	13.327	9.062
QB	1.545	14.178	0.226	23.723	3.841	26.250	13.106	14.214

Table 1 Average queue length of each server and average waiting time of each type of jobs under different algorithms

7.2. Sensitivity Analysis

The performance of queueing systems is influenced by the level of congestion or traffic intensity, denoted as $\rho = \lambda/J$. Traffic intensity reflects the overall busyness of the system. Figure 3 shows the performance of different algorithms across various traffic intensities. We conducted four experiments, keeping ϵ, V, and $\mathbf{r}$ constant while adjusting the arrival rates of job types. The plots represent traffic intensities of $\rho = 0.3333$, 0.5, 0.6667, and 0.9967 corresponding to arrival rates of $[1, 1]$, $[1, 2]$, $[2, 2]$, and $[2.99, 2.99]$ respectively. In the top-left plot, the greedy algorithm performs well when the traffic intensity is low, and servers are mostly idle. However, as traffic intensity increases, the performance gap between the greedy algorithm and OPT widens, while the gap between our algorithms and OPT slightly narrows. In the bottom-right plot, where traffic intensity exceeds 99%, the gap becomes negligible. This finding aligns with the intuition that relying on the most profitable servers is sufficient when the system is uncongested. However, as traffic intensity approaches 1, it becomes crucial to utilize every server to prevent excessive queue growth, irrespective of specific server rewards. This objective is precisely achieved by our proposed algorithms.

We next analyze the impact of the design parameter V on the performance of the barrier-function-based algorithm, keeping $\lambda = [2, 3]$ constant. The left plot of Figure 4 shows the cumulative reward obtained by the algorithm for three V values: 100, 1000, and 10000, along with the upper bound. Surprisingly, cumulative rewards are similar across different V values. However, Table 2 reveals an increasing trend in the average queue length, particularly at server 6, as V increases. Server 6, having the highest reward for both job types, experiences longer queues with larger

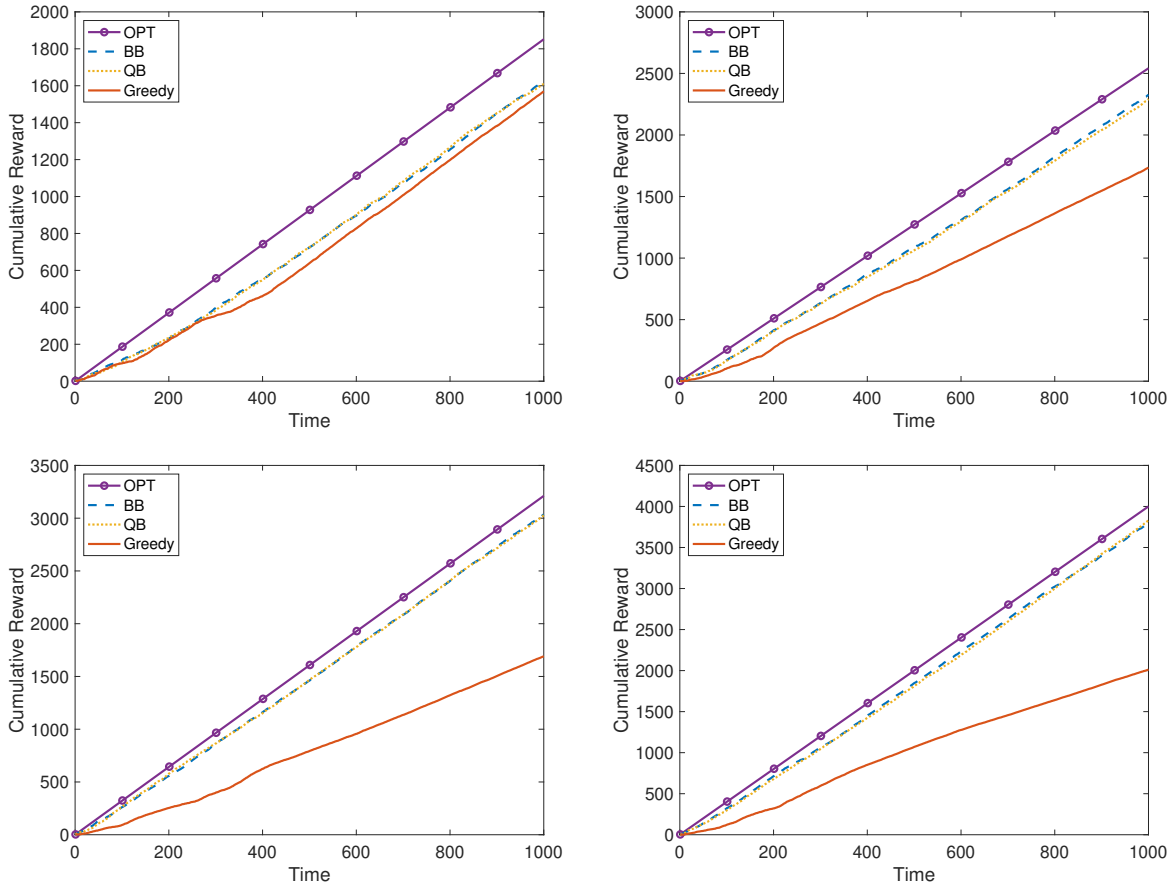

Figure 3 Cumulative rewards in systems with arrival rates $[1, 1]$ (top left), $[1, 2]$ (top right), $[2, 2]$ (bottom left), $[2.99, 2.99]$ (bottom right)

V values. Additionally, the average waiting time for both job types also increases with V, as shown in the right side of Table 2. These observations are expected since larger V reduces the penalty term in the algorithm, prioritizing rewards. Consequently, more jobs are assigned to the most profitable server, leading to longer waiting times. In practice, platforms serving real customers should consider a trade-off between prompt service and assigning jobs to the most desirable server. The choice of V can be based on which objective holds greater value for customers. For example, a digital healthcare platform emphasizing timely responses might opt for a smaller V value.

	Average Queue Length						Average Waiting Time	
	Server 1	Server 2	Server 3	Server 4	Server 5	Server 6	Type 1 Jobs	Type 2 Jobs
$V = 10000$	11.023	8.9550	0.080	20.066	4.585	17.610	15.133	10.209
$V = 1000$	8.152	6.680	0.029	18.223	8.742	14.660	12.706	10.157
$V = 100$	4.227	4.655	0.060	11.974	3.996	7.492	7.554	5.375

Table 2 Average queue length of each server and average waiting time of each type of jobs for different choices of V under the barrier-function-based algorithm

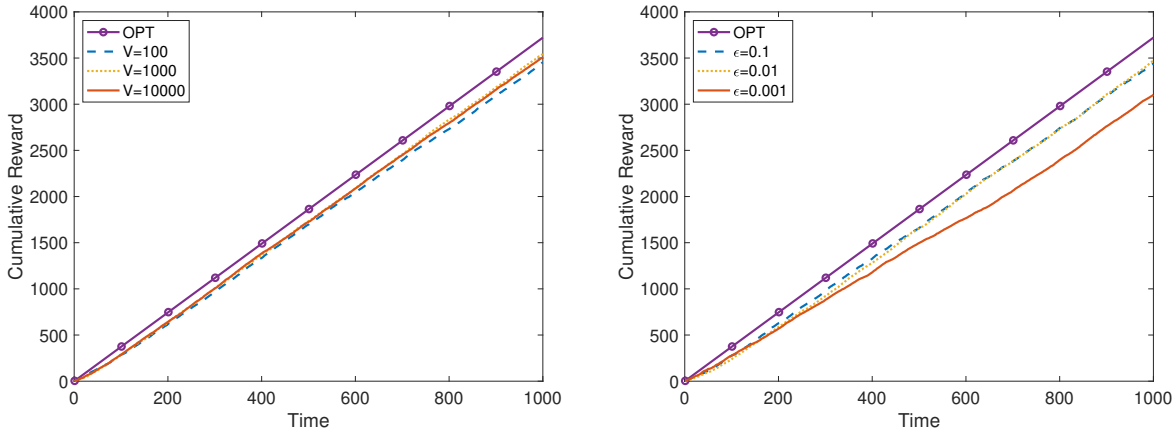

Figure 4 Cumulative reward for different choices of V under the barrier-function-based algorithm (left) and different choices of ϵ under the queue-based algorithm (right)

We observe a similar trend with the design parameter ϵ in the queue-based algorithm. As ϵ decreases, the average waiting time and queue length increase, as shown in Table 3. However, the right plot of Figure 4 indicates that when using $\epsilon = 0.001$, the performance is slightly worse because the queue-based algorithm behaves more like the greedy algorithm.

	Average Queue Length						Average Waiting Time	
	Server 1	Server 2	Server 3	Server 4	Server 5	Server 6	Type 1 Jobs	Type 2 Jobs
$\epsilon = 0.001$	3.585	78.152	0.191	151.593	3.030	161.694	67.838	81.140
$\epsilon = 0.01$	2.677	11.699	0.295	23.774	3.725	26.823	13.580	13.918
$\epsilon = 0.1$	1.221	2.939	0.495	3.001	2.143	3.898	2.289	3.047

Table 3 Average queue length of each server and average waiting time of each type of jobs for different choices of ϵ under the queue-based algorithm

7.3. Robustness Tests

To assess the robustness of our algorithms, we will investigate scenarios where some of our assumptions are not met. In the previous experiments, we have seen that the queue-based algorithm still performs well, even when arrival distributions are Poisson, which deviates from Assumption 2. Meanwhile, even though the performance guarantees of the barrier-function-based algorithm are established under the assumption of arrivals following a Poisson process, we expect that it will still perform well in practice even if this assumption does not hold. To verify this, we conducted two additional experiments, shown in Figure 5, where the arrival process deviates from the Poisson distribution. In the left plot, the number of arrivals follows a geometric distribution that satisfies neither Assumption 1 nor Assumption 2. In the right plot, the number of arrivals follows a binomial distribution with a success probability of 0.1. Despite these deviations, both of our proposed algorithms still perform very well, as long as the expected

number of arrivals for the two types of jobs is kept at $[2, 3]$ and all other parameters remain the same as in our main experiment.

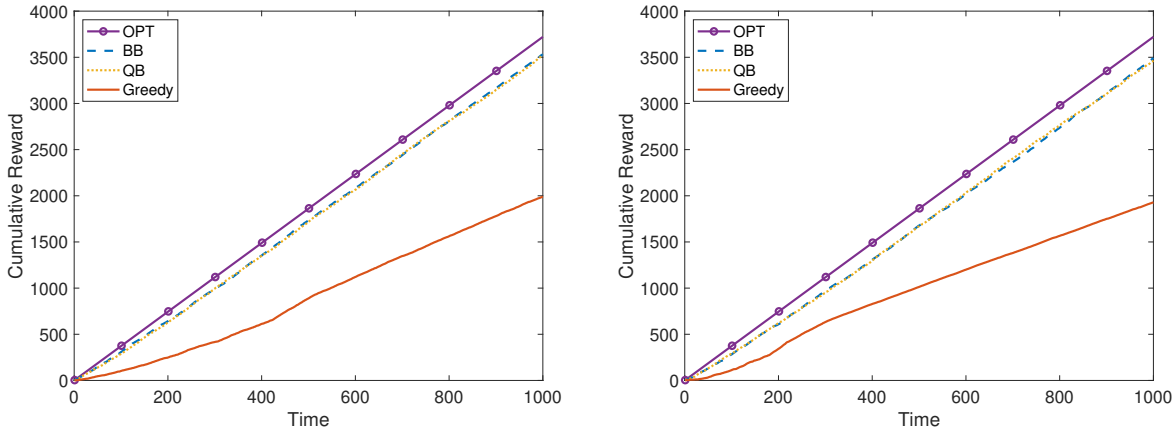

Figure 5 Cumulative rewards with geometric arrivals (left), and binomial arrivals (right)

We also investigate the impact of reward distribution on the performance of our proposed algorithms. To do this, we replace the assumption that rewards follow a Bernoulli distribution with a model in which the realized reward is the mean reward plus a random noise term drawn from $Unif(-\vartheta, \vartheta)$. We then apply the $\max(\cdot, 0)$ and $\min(\cdot, 1)$ functions to ensure the realized reward remains within the range of $[0, 1]$. In Figure 6, we present results for two different values of ϑ: $\vartheta = 0.1$ in the left plot, and $\vartheta = 0.5$ in the right plot. We observe that the performance of our proposed algorithms appears to decline slightly as the variance of the reward increases. Nevertheless, both algorithms remain very close to the upper bound and outperform the greedy algorithm by a significant margin.

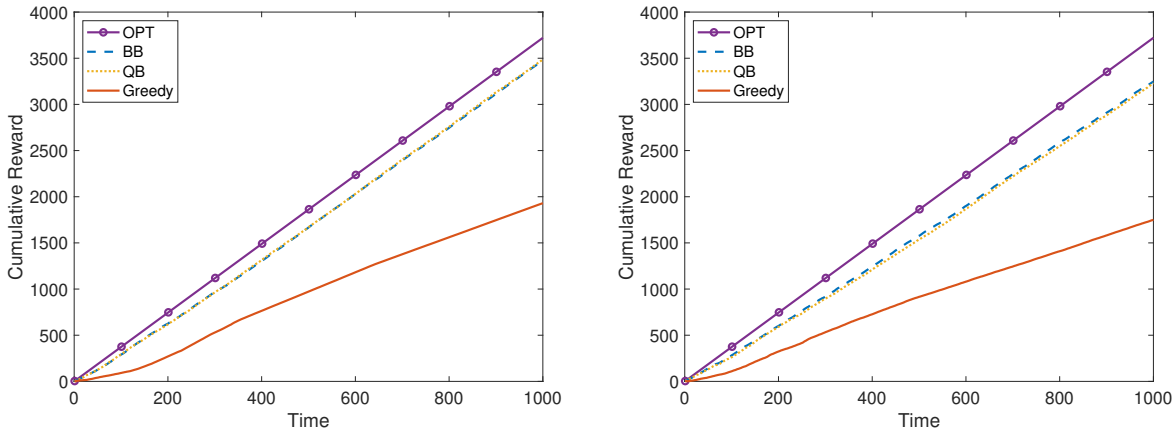

Figure 6 Cumulative rewards for the cases where the realized reward is equal to the mean reward plus a noise term from $Unif(-\vartheta, \vartheta)$ with $\vartheta = 0.1$ (left), and $\vartheta = 0.5$ (right)

7.4. Real-World Data

Finally, we report on the performance of our proposed algorithms using real-world data that we gathered from one of the largest digital healthcare platforms in China. Specifically, we collected data on 12 cardiologists who offer online consultation services via phone calls. The prices charged by these physicians for a 10-minute phone call are shown in the left plot of Figure 7, and range from 30 CNY ($4.5) to 270 CNY ($40.5). The physicians also have recommendation scores computed by the platform based on factors such as their title, the hospital where they work, and reviews from former patients. These scores, which range from 3.1 to 5, are summarized in the right plot of Figure 7. Since we aim to capture not only the price charged but also some aspect of patient satisfaction in the objective, we incorporate this score into the definition of the reward.

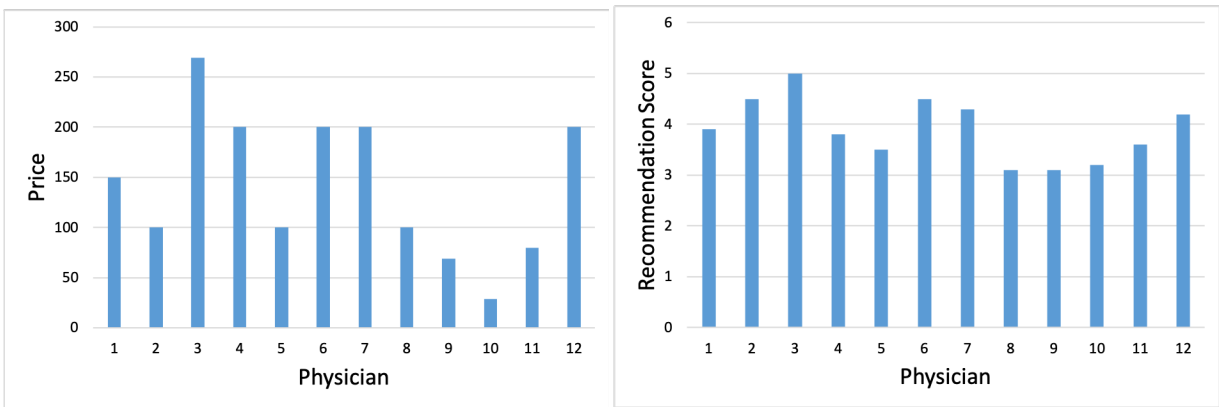

Figure 7 Price charged for a 10-minute phone call consultation (left), and recommendation score (right) of each physician

Due to limited data on the patients, we will broadly categorize them into two groups: those with severe diseases and those with minor ailments. As patients come from the cardiology department, many of them will require operations eventually, and the online consultation is only part of the initial diagnosis. We will refer to these patients as the "severe" type, while patients that can be treated with medication will be referred to as the "minor" type. Based on the percentage of each group, we have set the arrival rates for "minor" and "severe" patients to be [7, 3], respectively. Typically, the average consultation time for patients with minor ailments is less than 8 minutes, while for patients with severe diseases, it can exceed 15 minutes. As our model assumes fixed service time, we will incorporate this difference into the reward. Specifically, we define the mean reward of assigning a patient of type i to physician j as follows:

$$r_{ij} = \begin{cases} price_j \cdot \frac{score_j}{5} \cdot 0.8 & \text{if } i \text{ is the "minor" type} \\ price_j \cdot \frac{score_j}{5} \cdot 1.5 & \text{if } i \text{ is the "severe" type} \end{cases},$$

and the realized reward is equal to the mean reward plus a random noise term drawn from $Unif(-5, 5)$.

Since the range of mean rewards varies between 14 and 404, we make adjustments in our algorithms to appropriately penalize congestion. In the barrier-function-based algorithm, we set the value of V to 50, while in the

queue-based algorithm, we set the value of ϵ to 5. The findings are illustrated in Figure 8, revealing that our proposed algorithms in the real-world scenario perform exceptionally well, especially when compared to the greedy algorithm.

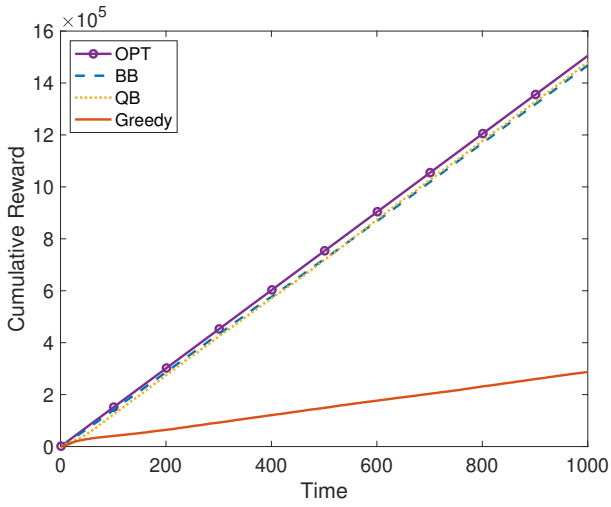

Figure 8 Cumulative rewards using real-world data

8. Concluding Remarks

We investigated the problem of matching jobs to servers in a queueing system, where stochastic rewards are earned upon job completion and the mean rewards depend on job types and servers. To maximize the total mean reward over a given time horizon and ensure queue stability, we proposed and evaluated the performance of two algorithms that combine matching and learning. Both algorithms estimate job-server pair rewards using a UCB-based learning procedure that balances exploration and exploitation. We showed that both proposed algorithms exhibit sub-linear regret with respect to the planning time horizon. The regret bounds explicitly capture the learning slowdown effect caused by job delays and highlight the dual role of queueing delays in degrading service levels and hindering system profitability. We conducted several numerical experiments using synthetic, randomly generated data or a real-world dataset to demonstrate the effectiveness and robustness of our proposed algorithms compared to meaningful benchmarks.

There are several avenues for future research. First, our current model assumes full server flexibility, which may not hold in some applications. Investigating how to adapt our proposed algorithms to compatibility constraints is thus a worthwhile pursuit. Second, mean rewards may be determined by job and server features using a bilinear model in some applications. Incorporating such features could lead to interesting data-driven decision problems as direct extensions of our current model. Third, while our paper and related works have focused on UCB-based learning, other learning procedures such as Thompson sampling may offer better performance in practice for multi-armed bandit problems. Thus, incorporating other learning procedures into future research could result in algorithms with equal or better performance.

References

Adan I, Bušić A, Mairesse J, Weiss G (2018) Reversibility and further properties of FCFS infinite bipartite matching. *Mathematics of Operations Research* 43(2):598–621.

Adan I, Weiss G (2012) Exact FCFS matching rates for two infinite multitype sequences. *Operations Research* 60(2):475–489.

Adan I, Weiss G (2014) A skill based parallel service system under FCFS-ALIS—steady state, overloads, and abandonments. *Stochastic Systems* 4(1):250–299.

Afeche P, Caldentey R, Gupta V (2022) On the optimal design of a bipartite matching queueing system. *Operations Research* 70(1):363–401.

Alfa AS (2016) *Applied Discrete-Time Queues* (Springer).

Arnosti N, Shi P (2020) Design of lotteries and wait-lists for affordable housing allocation. *Management Science* 66(6):2291–2307.

Asmussen S (1998) Extreme value theory for queues via cycle maxima. *Extremes* 1(2):137–168.

Auer P, Cesa-Bianchi N, Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47(2):235–256.

Berger AW, Whitt W (1995) Maximum values in queueing processes. *Probability in the Engineering and Informational Sciences* 9(3):375–409.

Boyd S, Boyd SP, Vandenberghe L (2004) *Convex Optimization* (Cambridge University Press).

Brown M, Peköz EA, Ross SM (2010) Some results for skip-free random walk. *Probability in the Engineering and Informational Sciences* 24(4):491–507.

Bušić A, Gupta V, Mairesse J (2013) Stability of the bipartite matching model. *Advances in Applied Probability* 45(2):351–378.

Caldentey R, Kaplan EH, Weiss G (2009) FCFS infinite bipartite matching of servers and customers. *Advances in Applied Probability* 41(3):695–730.

Chen X, Liu Y, Hong G (2020) An online learning approach to dynamic pricing and capacity sizing in service systems. *arXiv preprint arXiv:2009.02911* .

Cohen J (1968) Extreme value distribution for the M/G/1 and the G/M/1 queueing systems. *Annales de l'I.H.P. Probabilités et statistiques* 4(1):83–98.

Ding Y, McCormick ST, Nagarajan M (2021) A fluid model for one-sided bipartite matching queues with match-dependent rewards. *Operations Research* 69(4):1256–1281.

Eryilmaz A, Srikant R (2012) Asymptotically tight steady-state queue length bounds implied by drift conditions. *Queueing Systems* 72:311–359.

Gupta V (2022) Greedy algorithm for multiway matching with bounded regret. *Operations Research* .

Gurvich I, Ward A (2015) On the dynamic control of matching queues. *Stochastic Systems* 4(2):479–523.

Hsu WK, Xu J, Lin X, Bell MR (2022) Integrated online learning and adaptive control in queueing systems with uncertain payoffs. *Operations Research* 70(2):1166–1181.

Hu M, Zhou Y (2022) Dynamic type matching. *Manufacturing & Service Operations Management* 24(1):125–142.

Iglehart DL (1972) Extreme values in the GI/G/1 queue. *The Annals of Mathematical Statistics* 627–635.

Jia H, Shi C, Shen S (2022) Online learning and pricing for service systems with reusable resources. *Operations Research* .

Johari R, Kamble V, Kanoria Y (2021) Matching while learning. *Operations Research* 69(2):655–681.

Kerimov S, Ashlagi I, Gurvich I (2021) On the optimality of greedy policies in dynamic matching. *Available at SSRN* .

Kerimov S, Ashlagi I, Gurvich I (2023) Dynamic matching: Characterizing and achieving constant regret. *Management Science* .

Kim Jh, Vojnovic M (2021) Scheduling servers with stochastic bilinear rewards. *arXiv preprint arXiv:2112.06362* .

Krishnasamy S, Sen R, Johari R, Shakkottai S (2021) Learning unknown service rates in queues: A multiarmed bandit approach. *Operations Research* 69(1):315–330.

Lattimore T, Szepesvári C (2020) *Bandit Algorithms* (Cambridge University Press).

Levi R, Magnanti T, Shaposhnik Y (2019) Scheduling with testing. *Management Science* 65(2):776–793.

Liu X, Li B, Shi P, Ying L (2020) Pond: Pessimistic-optimistic online dispatching. *arXiv preprint arXiv:2010.09995* .

Massoulié L, Xu K (2018) On the capacity of information processing systems. *Operations Research* 66(2):568–586.

Moyal P, Perry O (2017) On the instability of matching queues. *The Annals of Applied Probability* 27(6):3385–3434.

Neely MJ (2010) Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks* 3(1):1–211.

Özkan E, Ward AR (2020) Dynamic matching for real-time ride sharing. *Stochastic Systems* 10(1):29–70.

Serfozo RF (1988) Extreme values of queue lengths in M/G/1 and GI/M/1 systems. *Mathematics of Operations Research* 13(2):349–357.

Singh S, Gurvich I, Van Mieghem JA (2022) Feature-based priority queuing. *Available at SSRN 3731865* .

Tan B, Srikant R (2012) Online advertisement, optimization and stochastic networks. *IEEE Transactions on Automatic Control* 57(11):2854–2868.

Xu K (2022) Drift method - from stochastic networks to machine learning. URL: `https://web.stanford.edu/~kuangxu/papers/driftmethod.pdf`. Last visited on 2023/03/09.

Zhong Y, Birge JR, Ward A (2022) Learning the scheduling policy in time-varying multiclass many server queues with abandonment. *Available at SSRN* .

E-Companion

EC.1. Proofs Related to Algorithm 1

Proof of Lemma 1. Let $\boldsymbol{\alpha}^* := [\alpha_j^*]$ denote the optimal dual variables associated with the J inequality constraints. Given $\lambda < J$, the J inequality constraints in (4) cannot be binding at the same time. Therefore, there exists some j' such that $\sum_i \lambda_i p_{i,j'}^* < 1$. Thus, in view of the complementary slackness conditions, we conclude that $\alpha_{j'}^* = 0$. Thus,

$$\beta_i^* \geq \lambda_i r_{ij'} - \lambda_i \alpha_{j'}^* = \lambda_i r_{ij'} \geq \lambda_i r_* > 0 \quad \text{for all} \quad i = 1, \ldots, I,$$

completing the proof. □

Proof of Proposition 1. Consider the optimization problem (8) and recall that $\tilde{\boldsymbol{\beta}} := [\tilde{\beta}_i]$ denotes the set of optimal dual variables of the I equality constraints in (8). For notational convenience, define

$$g_0(\boldsymbol{\pi}) := \sum_{i,j} \lambda_i \pi_{ij} r_{ij} \quad \text{and} \quad g_j(\boldsymbol{\pi}) = 1 - \sum_i \lambda_i \pi_{ij} \quad \text{for} \quad j = 1, \ldots, J.$$

Also, let $\ell_i(\boldsymbol{\pi}) := 1 - \sum_j \pi_{ij}$ for $i = 1, \ldots, I$. Then $\boldsymbol{\pi} = \tilde{\boldsymbol{p}}$ if

$$\nabla g_0(\boldsymbol{\pi}) + \sum_{j=1}^{J} \frac{1}{V g_j(\boldsymbol{\pi})} \nabla g_j(\boldsymbol{\pi}) + \sum_{i=1}^{I} \tilde{\beta}_i \nabla \ell_i(\boldsymbol{\pi}) = 0.$$

Therefore, $\tilde{\boldsymbol{p}}$ maximizes the Lagrangian:

$$L(\boldsymbol{\pi}, \tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\beta}}) := g_0(\boldsymbol{\pi}) + \sum_{j=1}^{J} \tilde{\alpha}_j g_j(\boldsymbol{\pi}) + \sum_{i=1}^{I} \tilde{\beta}_i \ell_i(\boldsymbol{\pi})$$

where we have defined $\tilde{\alpha}_j := 1/(V g_j(\tilde{\boldsymbol{p}}))$. By the well-known saddle-point interpretation of Lagrange duality for convex program (Boyd et al. 2004, §5.4), we have that

$$R^* \leq L(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\beta}}) = g_0(\tilde{\boldsymbol{p}}) + \frac{J}{V}.$$

This completes the proof. □

Proof of Theorem 1. Towards proving part (i), consider $\boldsymbol{p}(t)$, which solves the (convex) optimization problem (7). Let $\boldsymbol{\beta}(t) := [\beta_i(t)]$ denote the optimal dual variables of the I equality constraints in (7). A direct application of Karush–Kuhn–Tucker (KKT) optimality conditions (Boyd et al. 2004, §5.5.3) yields

$$-\frac{1}{V} \frac{\lambda_i}{1 - \sum_i \lambda_i p_{ij}(t)} + \lambda_i r_{ij}(t) - \beta_i(t) = 0 \quad \text{for all} \quad (i, j). \tag{EC.1}$$

Note that $\sum_j (1 - \sum_i \lambda_i p_{ij}(t)) = J - \lambda > 0$, where the inequality follows from our model assumption on system load. It follows that there must exist some j_0 such that

$$1 - \sum_i \lambda_i p_{ij_0}(t) \geq \frac{J - \lambda}{J}.$$

Combining the preceding inequality with (EC.1) gives us

$$\beta_i(t) = \lambda_i r_{ij_0}(t) - \frac{1}{V}\frac{\lambda_i}{1-\sum_i \lambda_i p_{ij_0}(t)} \geq \lambda_i r_{ij_0}(t) - \frac{1}{V}\frac{\lambda_i J}{J-\lambda} \geq \lambda_i r_* - \frac{1}{V}\frac{\lambda_i J}{J-\lambda} \quad \text{for all} \quad i,$$

where the last inequality is due to the lower truncation in (6). Because r_* is strictly positive, we conclude that $\beta_i(t) \geq 0$ for all $V \geq (1/r_*)(J/J-\lambda)$. This, in view of (EC.1), implies that

$$1 - \sum_i \lambda_i p_{ij}(t) = \frac{1}{V}\frac{\lambda_i}{\lambda_i r_{ij} - \beta_i(t)} \geq \frac{1}{V r_{ij}} \geq \frac{1}{V} \tag{EC.2}$$

To proceed, we make the following two observations: First, because the algorithm employs probabilistic routing at every time step, each server will receive a number of jobs that follow a Poisson distribution with mean $\sum_i \lambda_i p_{ij}(t)$ due to the thinning and superposition properties of the Poisson distribution. Second, if a batch of jobs received by a server over a time period is considered a "job," and the number of jobs in the batch is viewed as "service requirement" of that "job," then the queue length at each server will exhibit the same dynamics as the workload of a Geo/G/1 queue facing Bernoulli arrival with "success probability" 1 and Poisson service times with time-dependent mean given by $\sum_i \lambda_i p_{ij}(t)$. These two observations, together with a standard coupling argument similar to the one used in the proof of Theorem 1 in Hsu et al. (2022) plus (EC.2), allow us to upper-bound $Q_j(t)$ by the steady-state workload of the following Geo/G/1 discrete-time queueing system—a job would arrive (with probability one) at every time slot, and the service requirement follows a Poisson distribution with mean $1-1/V$. Denote by W the steady-state workload of this queueing process. Using the formula for the mean number in the system of a Geo/G/1 queue (see, e.g., Eq. (5.71) in (Alfa 2016, Chapter 5)), we have that

$$\mathbb{E}[W] = \left(1-\frac{1}{V}\right)\left[\left(1-\frac{1}{V}\right) + \frac{(1-1/V)^2 + 1 - 1/V}{2/V}\right] \leq 1 + V.$$

Thus, for V large enough,

$$\mathbb{E}[Q_j(t)] \leq \mathbb{E}[W] \leq V + 1,$$

completing the proof of part (i).

To establish part (ii), we can use (EC.2) and the coupling argument presented in part (i) to construct a single-server queue, denoted as $\bar{Q}$, which stochastically dominates each queue Q_j. Specifically, the dynamics of $\bar{Q}$ are given by:

$$\bar{Q}(t) = \left[\bar{Q}(t-1) + \bar{A}(t) - 1\right]^+, \tag{EC.3}$$

where $\{\bar{A}(t); t \geq 1\}$ are independent and identically distributed (i.i.d.) Poisson random variables with mean $1-1/V$, and the initial queue length is zero. Therefore, to obtain a moment bound for the running maximum of Q_j, it is sufficient to derive a moment bound for the running maximum of $\bar{Q}$. We achieve this by utilizing the fact that the "busy periods" of $\bar{Q}$ form regenerate cycles. More specifically, we define a sequence of stopping times: $\tau(1) := 0$ and $\tau(k) := \inf\left\{t > \tau(k-1) : \bar{Q}(t) = 0\right\}$, where $\tau(k)$ is the first time $\bar{Q}$ returns to 0 since $\tau_j(k-1)$. Thus, $\mathcal{T}(k) := \{\tau(k), \tau(k)+1, \ldots, \tau(k+1)-1\}$ represents the k-th busy period, and the consecutive cycle

maxima are i.i.d. Additionally, the running maximum of $\bar{Q}$ over a time horizon of length T is no greater than the maximum of K cycle maxima, where K is the number of busy periods the queue has experienced over the time horizon. Since there can be at most T busy periods, we have that

$$\bar{Q}^{\max}(T) \leq \max_{k=1,\ldots,T} \bar{C}_k, \tag{EC.4}$$

where $\bar{C}_k$ is the defined as the cycle maximum of the k-th busy period of queue $\bar{Q}$.

We will henceforth adopt the symbol $\bar{C}$ as a general notation for the cycle maximum pertaining to a busy period. We intend to argue that

$$\bar{C} \leq_{s.t.} \hat{S} + 1, \tag{EC.5}$$

where $\hat{S}$ follows an exponential distribution with mean V. To that end, we first note that

$$\bar{C} \leq_{s.t.} S := \max_{t \geq 0} X_t, \tag{EC.6}$$

where $\{X_n\}$ is a random walk defined by

$$X(0) = 0 \quad \text{and} \quad X(t) = X(t-1) + \bar{A}(t) - 1 \quad \text{for} \quad t \geq 1,$$

where $\{\bar{A}(t)\}$ are as in (EC.3). Since $\{X_n\}$ is a left-skip-free random walk, we can apply Propositions 5 and 6 in Brown et al. (2010) to get

$$S \stackrel{d}{=} \sum_{i=1}^{N-1} Y_i,$$

where N follows a geometric distribution $\mathbb{P}(N = k) = (1 - \frac{1}{Vq})^{k-1} \frac{1}{Vq}$ with $q := \mathbb{P}(\bar{A}(1) = 0) = e^{-(1-1/V)}$ for $k \geq 1$. In addition, Y_i are i.i.d. with a probability mass function given by

$$\mathbb{P}(Y_i = k) = \frac{\mathbb{P}(\bar{A}(1) - 1 \geq k)}{\mathbb{E}[\bar{A}(1)] - 1 + q} = \frac{1}{q - 1/V} \sum_{i=k+1}^{\infty} \frac{(1-1/V)^i e^{-(1-1/V)}}{i!}$$

for $k \geq 1$. Let H_i be i.i.d. samples from the distribution $\mathbb{P}(H_i = k) = q(1-q)^{k-1}$ for $k \geq 1$. We claim that $Y_i \leq_{s.t.} H_i$. To verify the claim, we check the monotone likelihood ratio, i.e., for any $k \geq 1$

$$\begin{aligned}
\frac{\mathbb{P}(H_i = k+1)}{\mathbb{P}(Y_i = k+1)} \frac{\mathbb{P}(Y_i = k)}{\mathbb{P}(H_i = k)} =& (1-q) \frac{\sum_{i=k+1}^{\infty} (1-1/V)^i / i!}{\sum_{i=k+2}^{\infty} (1-1/V)^i / i!} \\
=& (1-q) \left(1 + \frac{(1-1/V)^{k+1}/(k+1)!}{\sum_{i=k+2}^{\infty} (1-1/V)^i / i!} \right) \\
\geq& (1-q) \left(1 + \frac{1}{\sum_{i=1}^{\infty} (1-1/V)^i / i!} \right) \\
=& (1-q) \left(1 + \frac{1}{1/q - 1} \right) = 1
\end{aligned}$$

This suggests that $\frac{\mathbb{P}(H_i=k_2)}{\mathbb{P}(Y_i=k_2)} \geq \frac{\mathbb{P}(H_i=k_1)}{\mathbb{P}(Y_i=k_1)}$ for all $k_2 > k_1$, satisfying the monotone likelihood ratio. Hence, $Y_i \leq_{s.t.} H_i$, which in turn implies that

$$S \leq_{s.t.} \tilde{S} := \sum_{i=1}^{N} H_i. \tag{EC.7}$$

Because both N and H_i follow geometric distributions, it is a well-known result that $\tilde{S}$ is also geometrically distributed with a probability mass function given by $\mathbb{P}(\tilde{S}=k) = (\frac{1}{V})(1-\frac{1}{V})^{k-1}$ for $k \geq 1$. It is also elementary to show that

$$\tilde{S} \leq_{s.t.} \hat{S} + 1, \tag{EC.8}$$

where $\hat{S}$ follows an exponential distribution with mean V. Combining (EC.6)–(EC.8) leads to (EC.5). Next, by (EC.4) and (EC.5), we know that

$$Q_j^{\max}(T) \leq_{s.t.} \bar{Q}^{\max}(T) \leq \max_{k=1,\ldots,T}(\hat{S}_k + 1) = \max_{k=1,\ldots,T} \hat{S}_k + 1, \tag{EC.9}$$

where $\hat{S}_k$ are i.i.d. random variables sharing the same distribution with $\hat{S}$. To examine the distribution of $\max_{k=1,\ldots,T} \hat{S}_k$, we have

$$\mathbb{P}(\max_{k=1,\ldots,T} \hat{S}_k - V\log T \leq x) = \mathbb{P}(\hat{S} \leq V\log T + x)^T = \left(1 - \frac{e^{-x/V}}{T}\right)^T \geq 1 - e^{-x/V},$$

from which it follows that $\max_{k=1,\ldots,T} \hat{S}_k - V\log T$ is stochastically dominated by an exponential random variable with rate $1/V$. Combining this with (EC.9) yields

$$\mathbb{E}[Q_j^{\max}(T)] \leq V\log T + V + 1.$$

□

Proof of Proposition 3. To get a handle on the first term on the right-hand side of (13), we denote matrices $\boldsymbol{m} := [m_{ij}]$ and $\boldsymbol{\pi} := [\pi_{ij}]$ and define the function

$$f(\boldsymbol{\pi}|\boldsymbol{m}) := \sum_{j=1}^{J} \left[\log\left(1 - \sum_{i=1}^{I} \lambda_i \pi_{ij}\right) + V \sum_{i=1}^{I} \lambda_i \pi_{ij} m_{ij}\right].$$

Note that $f(\boldsymbol{\pi}|\boldsymbol{r}(t))$ is the objective of (7) multiplied by V. Thus, $\boldsymbol{p}(t) := [p_{ij}(t)]$ is the maximizer of $f(\boldsymbol{\pi}|\boldsymbol{r}(t))$ subject to the constraints in (7). Also, let $\tilde{\boldsymbol{p}} := [\tilde{p}_{ij}]$ denote the maximizer of $f(\boldsymbol{\pi}|\boldsymbol{r})$ over the constraints in (7).

Using the concavity of f in each π_{ij}, we have that

$$f(\boldsymbol{\pi}|\boldsymbol{r}) - f(\tilde{\boldsymbol{p}}|\boldsymbol{r}) \leq \sum_{i,j} \frac{\partial f}{\partial \pi_{ij}}(\tilde{\boldsymbol{p}}|\boldsymbol{r})(\pi_{ij} - \tilde{p}_{ij}). \tag{EC.10}$$

Using the definition of f, letting $\boldsymbol{\pi} = \boldsymbol{p}(t)$ in (EC.10) and rearranging terms, we obtain

$$\sum_{i,j} \left(V\lambda_i r_{ij} - \frac{\lambda_i}{1 - \sum_i \lambda_i \tilde{p}_{ij}}\right)(\tilde{p}_{ij} - p_{ij}(t)) \leq f(\tilde{\boldsymbol{p}}|\boldsymbol{r}) - f(\boldsymbol{p}(t)|\boldsymbol{r}). \tag{EC.11}$$

On the other hand,

$$\begin{aligned} f(\tilde{\boldsymbol{p}}|\boldsymbol{r}) &= f(\tilde{\boldsymbol{p}}|\boldsymbol{r}(t)) + V\sum_{i,j}\lambda_i(r_{ij}-r_{ij}(t))\tilde{p}_{ij} \\ &\le f(\boldsymbol{p}(t)|\boldsymbol{r}(t)) + V\sum_{i,j}\lambda_i(r_{ij}-r_{ij}(t))\tilde{p}_{ij} \\ &= f(\boldsymbol{p}(t)|\boldsymbol{r}) + V\sum_{i,j}\lambda_i(r_{ij}(t)-r_{ij})p_{ij}(t) + V\sum_{i,j}\lambda_i(r_{ij}-r_{ij}(t))\tilde{p}_{ij}, \end{aligned} \tag{EC.12}$$

where the two equalities result from the definition of f, while the inequality follows from the optimality of $\boldsymbol{p}(t)$. Combining (EC.11) and (EC.12), we get

$$\sum_{i,j}\left(V\lambda_i r_{ij} - \frac{\lambda_i}{1-\sum_i\lambda_i\tilde{p}_{ij}}\right)(\tilde{p}_{ij}-p_{ij}(t)) \le V(\Psi_1(t)+\Psi_2(t)), \tag{EC.13}$$

where

$$\Psi_1(t) := \sum_{i,j}\lambda_i(r_{ij}(t)-r_{ij})p_{ij}(t) \quad \text{and} \quad \Psi_2(t) := \sum_{i,j}\lambda_i(r_{ij}-r_{ij}(t))\tilde{p}_{ij}.$$

For ease of flow, we present the following lemma, whose proof is deferred to after the proof of the current proposition.

Lemma EC.1. *There exists some $\kappa > 0$ dependent on the model primitives such that*

$$\frac{\lambda_i}{1-\sum_i\lambda_i\tilde{p}_{ij}} \le V\lambda_i r_{ij}(1-\kappa) \quad \textit{for all} \quad (i,j).$$

To proceed, we combine Lemma EC.1 with (EC.13) yields

$$\kappa\sum_{i,j}\lambda_i r_{ij}(\tilde{p}_{ij}-p_{ij}(t)) \le \Psi_1(t)+\Psi_2(t).$$

From the preceding inequality, it follows that

$$\sum_{t=1}^{T}\sum_{i,j}\lambda_i r_{i,j}(\tilde{p}_{ij}-p_{ij}(t)) \le K\left(\sum_{t=1}^{T}\Psi_1(t)+\sum_{t=1}^{T}\Psi_2(t)\right) \quad \text{for} \quad K := \kappa^{-1}. \tag{EC.14}$$

Below, we bound the two terms in the apprentices on the right-hand side separately.

Recall that $h_{ij}(t)$ is the number of jobs of type i assigned to server j up to time t. To avoid division by zero, we define $\hat{h}_{ij}(t) := \max\{h_{ij}(t),1\}$ in the definition of the event $F_{ij}(t)$ below. Specifically, for each (i,j) pair, define

$$F_{ij}(t) := \left\{\bar{r}_{ij}(t-1) - r_{ij} \le \sqrt{\frac{2\log(t-1)}{\hat{h}_{ij}(t-1)}}\right\}.$$

Let $Y_{ij}(t) := (r_{ij}(t)-r_{ij}) \le 1$. Then

$$\begin{aligned} \sum_{t=1}^{T}\Psi_1(t) &= \sum_{i,j}\sum_{t=1}^{T}Y_{ij}(t)\lambda_i p_{ij}(t) \\ &= \sum_{i,j}\sum_{t=1}^{T}Y_{ij}(t)\lambda_i p_{ij}(t)1_{F_{ij}(t)} + \sum_{i,j}\sum_{t=1}^{T}Y_{ij}(t)\lambda_i p_{ij}(t)1_{F_{ij}^c(t)} \\ &\le \sum_{i,j}\sum_{t=1}^{T}Y_{ij}(t)\lambda_i p_{ij}(t)1_{F_{ij}(t)} + \sum_{i,j}\sum_{t=1}^{T}\lambda_i p_{ij}(t)1_{F_{ij}^c(t)}. \end{aligned} \tag{EC.15}$$

We first bound the expectation of the first component on the right-hand side of (EC.15). For this purpose, we invoke the definition of $F_{ij}(t)$ to get

$$\sum_{t=1}^{T} Y_{ij}(t)\lambda_i p_{ij}(t) 1_{F_{ij}(t)} \le 2\sum_{t=1}^{T}\sqrt{\frac{2\log(t-1)}{\hat{h}_{ij}(t-1)}}\lambda_i p_{ij}(t) \le 2\sqrt{2\log T}\sum_{t=1}^{T}\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}}\lambda_i p_{ij}(t). \quad \text{(EC.16)}$$

Recall that $\Gamma_{ij}(t)$ is the number of type i jobs assigned to server j in slot t. We have that

$$\mathbb{E}\left[\sum_{t=1}^{T}\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}}\lambda_i p_{ij}(t)\right] = \mathbb{E}\left[\sum_{t=1}^{T}\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}}\Gamma_{ij}(t)\right]. \quad \text{(EC.17)}$$

Clearly,

$$\sum_{t=1}^{T}\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}}\Gamma_{ij}(t) = \sum_{t=1}^{T}\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}}B_{ij}(t) + \sum_{t=1}^{T}\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}}(\Gamma_{ij}(t) - B_{ij}(t)). \quad \text{(EC.18)}$$

Since

$$B_{ij}(t) = h_{ij}(t) - h_{ij}(t-1),$$

we have that

$$\begin{aligned}\sum_{t=1}^{T}\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}}B_{ij}(t) &= \sum_{t=1}^{T}\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}}(h_{ij}(t) - h_{ij}(t-1)) \\ &\le 2 + \int_{1}^{h_{ij}(T-1)}\frac{1}{\sqrt{x}}\mathrm{d}x \\ &\le 2 + 2\sqrt{h_{ij}(T-1)}.\end{aligned} \quad \text{(EC.19)}$$

Using Abel's summation formula (summation by parts) on the second term on the right-hand side of (EC.18), along with the identity $Q_{ij}(t) = \sum_{s=1}^{t}\Gamma_{ij}(s) - \sum_{s=1}^{t}B_{ij}(s)$, we obtain

$$\begin{aligned}&\sum_{t=1}^{T}\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}}(\Gamma_{ij}(t) - B_{ij}(t)) \\ =&Q_{ij}(T)\sqrt{\frac{1}{\hat{h}_{ij}(T-1)}} + \sum_{t=1}^{T-1}Q_{ij}(t)\left(\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}} - \sqrt{\frac{1}{\hat{h}_{ij}(t)}}\right) \\ \le&Q_{ij}^{\max}(T)\left[\sqrt{\frac{1}{\hat{h}_{ij}(T-1)}} + \sum_{t=1}^{T-1}\left(\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}} - \sqrt{\frac{1}{\hat{h}_{ij}(t)}}\right)\right] \\ \le&Q_{ij}^{\max}(T),\end{aligned} \quad \text{(EC.20)}$$

where we have defined $Q_{ij}^{\max}(t)$ as the running maximum of Q_{ij} up to time t. Substituting (EC.19)–(EC.20) into (EC.18), we get

$$\sum_{t=1}^{T}\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}}\Gamma_{ij}(t) \le 2 + 2\sqrt{h_{ij}(T-1)} + Q_{ij}^{\max}(T). \quad \text{(EC.21)}$$

Next, combining (EC.21) with (EC.16) and (EC.17) yields

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i,j}\sum_{t=1}^{T} Y_{ij}(t)\lambda_i p_{ij}(t) 1_{F_{ij}(t)}\right] \\
\leq& 2\sqrt{2\log T}\left(2IJ + 2\sum_j \mathbb{E}\left[\sqrt{I\sum_i h_{ij}(T-1)}\right] + \sum_{i,j}\mathbb{E}\left[Q_{ij}^{\max}(T)\right]\right), \\
\leq& 2\sqrt{2\log T}\left(2IJ + 2J\sqrt{IT} + J(V\log T + V + 1)\right),
\end{aligned} \tag{EC.22}$$

where the inequality is due to the Cauchy-Schwartz inequality whereas the second inequality follows from the fact that $h_j(T-1) := \sum_i h_{ij}(T-1) \leq T$.

We next bound the expectation of the second component on the right-hand side of (EC.15). By apply Chernoff-Hoeffding Inequality (see, e.g., Ex. 7.1 in (Lattimore and Szepesvári 2020, Chapter 7), we have that

$$\mathbb{E}\left[\sum_{i,j}\sum_{t=1}^{T}\lambda_i p_{ij}(t) 1_{F_{ij}^c(t)}\right] \leq \sum_{i,j}\lambda_i \sum_{t=1}^{T}\mathbb{P}(F_{ij}^c(t)) \leq \sum_{i,j}\lambda_i\left(1 + \sum_{t=1}^{T-1}\frac{1}{t^4}\right) \leq \frac{7}{3}J\lambda. \tag{EC.23}$$

Finally, combining (EC.15) with (EC.22)–(EC.23) gives us

$$\mathbb{E}\left[\sum_{t=1}^{T}\Psi_1(t)\right] \leq 2\sqrt{2\log T}\left(2IJ + 2J\sqrt{IT} + J(V\log T + V + 1)\right) + \frac{7}{3}J\lambda. \tag{EC.24}$$

Now, for each (i,j) pair, define

$$G_{ij}(t) := \left\{r_{ij} - \bar{r}_{ij}(t-1) \leq \sqrt{\frac{2\log(t-1)}{\hat{h}_{ij}(t-1)}}\right\}.$$

Then

$$\begin{aligned}
\sum_{t=1}^{T}\Psi_2(t) &= \sum_{i,j}\sum_{t=1}^{T}\lambda_i(r_{ij} - r_{ij}(t))\tilde{p}_{ij} \\
&= \sum_{i,j}\sum_{t=1}^{T}\lambda_i(r_{ij} - r_{ij}(t))\tilde{p}_{ij}1_{G_{ij}(t)} + \sum_{i,j}\sum_{t=1}^{T}\lambda_i(r_{ij} - r_{ij}(t))\tilde{p}_{ij}1_{G_{ij}^c(t)} \\
&\leq \sum_{i,j}\sum_{t=1}^{T}\lambda_i\tilde{p}_{ij}1_{G_{ij}^c(t)},
\end{aligned} \tag{EC.25}$$

where the inequality uses the fact that $r_{ij} < r_{ij}(t)$ on $G_{ij}(t)$ for each (i,j) pair. Taking expectations on both sides of (EC.25) and applying Chernoff-Hoeffding Inequality yields

$$\mathbb{E}\left[\sum_{t=1}^{T}\Psi_2(t)\right] \leq \sum_{i,j}\lambda_i\sum_{t=1}^{T}\mathbb{P}(G_{ij}^c(t)) \leq \sum_{i,j}\lambda_i\left(1 + \sum_{t=1}^{T-1}\frac{1}{t^4}\right) \leq \frac{7}{3}J\lambda. \tag{EC.26}$$

Combining (EC.14), (EC.24) and (EC.26), we reach (15), hence completing the proof. □

Proof of Lemma EC.1. Consider $\tilde{\boldsymbol{p}}$, which solves the optimization problem (8), and recall that $\tilde{\boldsymbol{\beta}}$ denote the optimal dual variables of the equality constraints in (8). By the KKT conditions,

$$-\frac{1}{V}\frac{\lambda_i}{1-\sum_i \lambda_i \tilde{p}_{ij}} + \lambda_i r_{ij} - \tilde{\beta}_i = 0 \quad \text{for all} \quad (i,j). \tag{EC.27}$$

This, along with (EC.27), implies that

$$\frac{\lambda_i}{1-\sum_i \lambda_i \tilde{p}_{ij}} = V(\lambda_i r_{ij} - \tilde{\beta}_i) = V\lambda_i r_{ij}(1 - \tilde{\beta}_i/(\lambda_i r_{ij})).$$

Thus, by letting $\kappa := \min_{i,j} \tilde{\beta}_i/(\lambda_i r_{ij})$, we have that

$$\frac{\lambda_i}{1-\sum_i \lambda_i \tilde{p}_{ij}} \le V\lambda_i r_{ij}(1-\kappa).$$

This completes the proof. □

EC.2. Proofs Related to Algorithm 2

Proof of Proposition 2. Towards proving part (i), define the Lyapunov function $V(\mathbf{x}) := \sum_i x_i^2$, and let $k^{-1}(j;\mathbf{Q}(t)) := \{i \in \mathcal{I} : k(i,\mathbf{Q}(t)) = j\}$ denote all the types i such that $k(i,\mathbf{Q}(t)) = j$. Let $A_i(t)$ be the number of newly arriving jobs of type i in time slot t. Given $\mathbf{Q}(t)$, the one-step drift can be calculated as

$$\begin{aligned}
&\mathbb{E}\left[V(\mathbf{Q}(t+1))|\mathbf{Q}(t)\right] - V(\mathbf{Q}(t)) \\
\le&\mathbb{E}\left[\sum_{j=1}^{J}\left(Q_j(t) + \sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1) - 1\right)^2 \middle| \mathbf{Q}(t)\right] - \sum_{j=1}^{J} Q_j^2(t) \\
=&\sum_{j=1}^{J}\mathbb{E}\left[\left(\sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1) - 1\right)^2 \middle| \mathbf{Q}(t)\right] + 2Q_j(t)\mathbb{E}\left[\left(\sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1) - 1\right) \middle| \mathbf{Q}(t)\right].
\end{aligned}$$

First,

$$\begin{aligned}
&\sum_{j=1}^{J}\mathbb{E}\left[\left(\sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1) - 1\right)^2 \middle| \mathbf{Q}(t)\right] \\
=&\sum_{j=1}^{J}\mathbb{E}\left[\left(\sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1)\right)^2 - 2\sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1) + 1 \middle| \mathbf{Q}(t)\right] \\
\le&\mathbb{E}\left[\sum_{i=1}^{I} A_i(t+1)\right]^2 - 2\sum_{i=1}^{I}\lambda_i + J \\
=&M^2 - 2\lambda + J,
\end{aligned}$$

where the inequality follows from the general inequality $\sum_{i=1}^n x_i^2 \le (\sum_{i=1}^n x_i)^2$ and the independence between $\mathbf{Q}(t)$ and $A_i(t+1)$. Second,

$$\sum_{j=1}^{J} Q_j(t)\mathbb{E}\left[\left(\sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1) - 1\right) \middle| \mathbf{Q}(t)\right]$$

$$
\begin{aligned}
&= \sum_{j=1}^{J} Q_j(t) \left(\sum_{i \in k^{-1}(j;\mathbf{Q}(t))} \lambda_i - 1 \right) \\
&= \sum_{i=1}^{I} \sum_{j=1}^{J} Q_j(t) \lambda_i \mathbf{1}_{k^{-1}(j;\mathbf{Q}(t))}(i) - \sum_{j=1}^{J} Q_j(t) \\
&= \sum_{i=1}^{I} \sum_{j=1}^{J} \left(Q_j(t) - \frac{r_{ik(i,\mathbf{Q}(t))}}{\epsilon} + \frac{r_{ik(i,\mathbf{Q}(t))}}{\epsilon} \right) \lambda_i \mathbf{1}_{k^{-1}(j;\mathbf{Q}(t))}(i) - \sum_{j=1}^{J} Q_j(t) \\
&\leq \sum_{i=1}^{I} \min_j \left(Q_j(t) - \frac{r_{ij}}{\epsilon} \right) \lambda_i + \frac{1}{\epsilon} \sum_{i=1}^{I} \lambda_i - \sum_{j=1}^{J} Q_j(t) \\
&\leq \sum_{i=1}^{I} \lambda_i \frac{1}{J} \sum_{j=1}^{J} \left(Q_j(t) - \frac{r_{ij}}{\epsilon} \right) - \sum_{j=1}^{J} Q_j(t) + \frac{\lambda}{\epsilon} \\
&\leq - \frac{J - \lambda}{J} \sum_{j=1}^{J} Q_j(t) + \frac{\lambda}{\epsilon}.
\end{aligned}
$$

Combining the above inequalities, we have that

$$
\mathbb{E}\left[V(\mathbf{Q}(t+1))|\mathbf{Q}(t)\right] - V(\mathbf{Q}(t)) \leq -2\frac{J-\lambda}{J} \sum_{j=1}^{J} Q_j(t) + \frac{2\lambda}{\epsilon} + B_1,
$$

where $B_1 := M^2 - 2\lambda + J$. Part (i) of the proposition then follows from the preceding inequality and Foster-Lyapunov criterion (see e.g., Lemma 5.2.1 in Xu (2022)).

The key to proving part (ii) is to observe that, based on the routing rule, no two queues can differ by more than $1/\epsilon + M$ jobs at any given time. This means that whenever the total number of jobs in the system is greater than or equal to $J(1/\epsilon + M)$, no server will be idle. This observation, combined with a standard coupling argument allows us to establish the following stochastic ordering relations:

$$
\sum_{j=1}^{J} Q_j(t) \leq_{s.t.} J(1/\epsilon + M) + \tilde{Q}(t) \quad \text{for all} \quad t \geq 1, \tag{EC.28}
$$

where $\tilde{Q}$ satisfies the following dynamics

$$
\tilde{Q}(t) = \left[\tilde{Q}(t-1) + \sum_i A_i(t) - J \right]^+
$$

with $\tilde{Q}(0) = 0$. Standard stochastic ordering results for the $GI/G/1$ queue show that

$$
\mathbb{E}[\tilde{Q}(t)] \leq \mathbb{E}[\tilde{Q}(\infty)] \leq \frac{M^2 - 2\lambda J + J^2}{2(J-\lambda)} \tag{EC.29}
$$

for all $t \geq 1$, where the second inequality is derived by applying the steady-state drift bound (Xu 2022, Lemma 5.2.1) to a $GI/G/1$ queue with a quadratic Lyapunov function. Combining (EC.28) and (EC.29) establishes the claim in part (ii).

For part (iii), let us first rewrite the regret as

$$\widehat{Reg}^{\text{QB}}(T) = R^*T - \sum_{t=1}^{T}\mathbb{E}[\bar{R}(\mathbf{Q}(t-1))] + \sum_{t=1}^{T}\mathbb{E}[\bar{R}(\mathbf{Q}(t-1))] - R^{\text{QB}}(T), \tag{EC.30}$$

where $\bar{R}(\mathbf{Q}(t-1)) := \sum_{j=1}^{J}\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1))} A_i(t)r_{ij}$. Observe that

$$\sum_{t=1}^{T}\mathbb{E}[\bar{R}(\mathbf{Q}(t-1))] - R^{\text{QB}}(T) = \sum_{i=1}^{I}\sum_{j=1}^{J} r_{ij}\mathbb{E}[Q_{ij}(T)] \le \sum_{j=1}^{J}\mathbb{E}[Q_j(T)]. \tag{EC.31}$$

This inequality enables us to analyze $\sum_{t=1}^{T}\mathbb{E}[\bar{R}(\mathbf{Q}(t))]$ instead of $R^{\text{QB}}(T)$, given that we have obtained the moment bound from part (ii). To proceed, note that

$$\begin{aligned}
&\mathbb{E}[V(\mathbf{Q}(t+1))|\mathbf{Q}(t)] - V(\mathbf{Q}(t))\\
&= \mathbb{E}\left[\sum_{j=1}^{J}\left(\left[Q_j(t) + \sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1) - 1\right]^+\right)^2\middle|\mathbf{Q}(t)\right] - Q_j^2(t)\\
&\le \sum_{j=1}^{J}\mathbb{E}\left[\left(Q_j(t) + \sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1) - 1\right)^2\middle|\mathbf{Q}(t)\right] - Q_j^2(t)\\
&= \sum_{j=1}^{J} 2Q_j(t)\mathbb{E}\left[\left(\sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1) - 1\right)\middle|\mathbf{Q}(t)\right] + \sum_{j=1}^{J}\mathbb{E}\left[\left(\sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1) - 1\right)^2\middle|\mathbf{Q}(t)\right].
\end{aligned}$$

For the second term, recall that $\sum_{j=1}^{J}\mathbb{E}\left[\sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1) - 1\right]^2 \le B_1$.

Next, for the first term, we have that

$$\begin{aligned}
&\sum_{j=1}^{J} Q_j(t)\mathbb{E}\left(\sum_{i\in k^{-1}(j;\mathbf{Q}(t))} A_i(t+1) - 1\middle|\mathbf{Q}(t)\right)\\
=&\sum_{j=1}^{J} Q_j(t)\sum_{i\in k^{-1}(j;\mathbf{Q}(t))}\lambda_i - \sum_{j=1}^{J} Q_j(t)\\
=&\sum_{j=1}^{J} Q_j(t)\sum_{i\in k^{-1}(j;\mathbf{Q}(t))}\lambda_i - \sum_{j=1}^{J} Q_j(t) + \frac{1}{\epsilon}\sum_{j=1}^{J}\sum_{i\in k^{-1}(j;\mathbf{Q}(t))}\lambda_i r_{ij} - \frac{1}{\epsilon}\sum_{j=1}^{J}\sum_{i\in k^{-1}(j;\mathbf{Q}(t))}\lambda_i r_{ij}\\
=&\frac{\mathbb{E}[\bar{R}(\mathbf{Q}(t))|\mathbf{Q}(t)]}{\epsilon} - \frac{1}{\epsilon}\left[\sum_{j=1}^{J}\sum_{i\in k^{-1}(j;\mathbf{Q}(t))}\lambda_i\left(r_{ij} - \epsilon Q_j(t)\right)\right] - \sum_{j=1}^{J} Q_j(t)\\
\le&\frac{\mathbb{E}[\bar{R}(\mathbf{Q}(t))|\mathbf{Q}(t)]}{\epsilon} - \frac{1}{\epsilon}\left[\sum_{i=1}^{I}\sum_{j=1}^{J}\lambda_i p_{ij}^*\left(r_{ij} - \epsilon Q_j(t)\right)\right] - \sum_{j=1}^{J} Q_j(t) \qquad \text{(EC.32)}\\
=&-\frac{1}{\epsilon}\left(R^* - \mathbb{E}[\bar{R}(\mathbf{Q}(t))|\mathbf{Q}(t)]\right) - \sum_{j=1}^{J} Q_j(t)\left(1 - \sum_{i=1}^{I}\lambda_i p_{ij}^*\right)\\
\le&-\frac{1}{\epsilon}\left(R^* - \mathbb{E}[\bar{R}(\mathbf{Q}(t))|\mathbf{Q}(t)]\right),
\end{aligned}$$

where the first inequality follows from that queue-based algorithm achieves maximum of $r_{ij} - \epsilon Q_j$ for each type i from all feasible assignment rules, while the last inequality is due to $1 - \sum_{i=1}^{I} \lambda_i p_{ij}^* \geq 0$ for each j. Hence, we have that

$$\mathbb{E}[V(\mathbf{Q}(t+1))|\mathbf{Q}(t)] - V(\mathbf{Q}(t)) \leq -\frac{2}{\epsilon}\left(R^* - \mathbb{E}[\bar{R}(\mathbf{Q}(t))|\mathbf{Q}(t)]\right) + B_1 \quad \text{for} \quad t \geq 0.$$

Rearranging the preceding inequality yields

$$\frac{2}{\epsilon}\left(R^* - \mathbb{E}[\bar{R}(\mathbf{Q}(t))|\mathbf{Q}(t)]\right) \leq V(\mathbf{Q}(t)) - \mathbb{E}[V(\mathbf{Q}(t+1))|\mathbf{Q}(t)] + B_1 \quad \text{for} \quad t \geq 0$$

or, equivalently,

$$\frac{2}{\epsilon}\left(R^* - \mathbb{E}[\bar{R}(\mathbf{Q}(t-1))|\mathbf{Q}(t-1)]\right) \leq V(\mathbf{Q}(t-1)) - \mathbb{E}[V(\mathbf{Q}(t))|\mathbf{Q}(t-1)] + B_1 \quad \text{for} \quad t \geq 1.$$

Taking expectations on both sides and summing over $t = 1, \ldots, T$, we obtain

$$\frac{2}{\epsilon}\sum_{t=1}^{T}\mathbb{E}\left[R^* - \mathbb{E}[\bar{R}(\mathbf{Q}(t-1))|\mathbf{Q}(t-1)]\right] \leq \sum_{t=1}^{T}\mathbb{E}\left[V(\mathbf{Q}(t-1)) - \mathbb{E}\left[V(\mathbf{Q}(t))|\mathbf{Q}(t-1)\right]\right] + B_1 T,$$

which, after rearranging, yields

$$\sum_{t=1}^{T}\left(R^* - \mathbb{E}[\bar{R}(\mathbf{Q}(t-1))]\right) \leq \frac{B_1\epsilon}{2}T.$$

Combining the above arguments, together with (EC.30) and (EC.31) leads to the desired conclusion. □

Proof of Theorem 3. The proof of part (i) can be derived directly from part (ii) of Proposition 2 since the arguments do not rely on $r_{ij}(t)$.

For part (ii), we know from (EC.28) that $Q_\Sigma(t)$ is stochastically dominated by a constant $J(1/\epsilon + M)$ plus a GI/G/1 queue $\tilde{Q}$. The key to prove part (ii) is to first observe that $\tilde{Q}(t)$ is stochastically dominated by $\sum_j \tilde{Q}_j(t)$ where we define $(\tilde{Q}_j)_j$ as a decentralized queueing system where the jth queue has the dynamics:

$$\tilde{Q}_j(t) = [\tilde{Q}_j(t-1) + \tilde{A}_j(t) - 1]^+,$$

where $\tilde{A}_j(t) = (\sum_i A_i(t))\mathbf{1}_{\{j\}}(W(t))$ with $W(t)$ being a sequence of i.i.d. random variable having probability $\mathbb{P}(W(t) = j) = 1/J$ for $j = 1, \ldots, J$ and independent of everything else. Following the same arguments in the proof of Theorem 1, we know that the maximal value of $\tilde{Q}_j(t)$ in a busy cycle is stochastically dominated by $S = \sum_{i=1}^{N-1} Y_i$, where $Y_i \leq M - 1$ with probability 1 and N is geometrically distributed with mean $(Jq)/(J - \lambda)$ and $q = \mathbb{P}(\tilde{A}_j(1) = 0)$. More specifically,

$$\begin{aligned}
q &= \mathbb{P}\left(\left(\sum_i A_i(1)\right)\mathbf{1}_{\{j\}}(W(1)) = 0\right) = \mathbb{P}\left(\left(\sum_i A_i(1) = 0\right)\bigcup\left(\mathbf{1}_{\{j\}}(W(1)) = 0\right)\right) \\
&= 1 - \left(1 - \mathbb{P}\left(\sum_i A_i(1) = 0\right)\right)\left(1 - \mathbb{P}(\mathbf{1}_{\{j\}}(W(1)) = 0)\right) \\
&= 1 - \frac{1}{J}\left(1 - \mathbb{P}\left(\sum_i A_i(1) = 0\right)\right).
\end{aligned}$$

Hence, S is stochastically dominated by $(M-1)(\hat{S}+1)$ where $\hat{S}$ follows the exponential distribution with mean $(Jq)/(J-\lambda)$. Let $\hat{S}_k$ be a sequence of i.i.d. random variables sharing the same distribution with $\hat{S}$. Then $\tilde{Q}_j^{\max}(t) \leq_{s.t.} (M-1)(\max_{k\leq t}\hat{S}_k+1)$. Since

$$\mathbb{P}\left(\max_{k\leq t}\hat{S}_k - \frac{Jq}{J-\lambda}\log t \leq x\right) = \mathbb{P}\left(\hat{S}_t \leq \frac{Jq}{J-\lambda}\log t + x\right)^t \geq 1-e^{-\frac{J-\lambda}{Jq}x},$$

we know that $\max_{k\leq t}\hat{S}_k - \frac{Jq}{J-\lambda}\log t$ is stochastically dominated by an exponentially distributed random variable with mean $(Jq)/(J-\lambda)$. This leads to

$$\mathbb{E}[\tilde{Q}_j^{\max}(t)] \leq (M-1)\left(\frac{Jq}{J-\lambda}\log t + \frac{Jq}{J-\lambda}+1\right).$$

Furthermore, for each sample path, we have that

$$\max_{k\leq t}\sum_j \tilde{Q}_j(t) \leq \sum_j \max_{k\leq t}\tilde{Q}_j(t).$$

Taking expectations and combining aforementioned arguments, we get

$$\begin{aligned}\mathbb{E}[Q_\Sigma^{\max}(T)] &\leq J(1/\epsilon+M) + \sum_j \mathbb{E}\left[\tilde{Q}_j^{\max}(T)\right] \\ &\leq J(1/\epsilon+M) + J(M-1)\left(\frac{Jq}{J-\lambda}\log T + \frac{Jq}{J-\lambda}+1\right),\end{aligned}$$

finishing the proof of part (ii).

To prove part (iii), let $k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))$ denote all the types i such that $k(i,\mathbf{Q}(t-1),\mathbf{r}(t))=j$. Also, define

$$\bar{R}(\mathbf{Q}(t-1),\mathbf{r}(t)) := \sum_{j=1}^J \sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))} A_i(t) r_{ij},$$

which can be interpreted as the potential payoff gained during time slot t if all the jobs assigned at t were eventually completed. We first notice that

$$Reg^{\text{QB}}(T) = \sum_{t=1}^T \mathbb{E}[R^* - \bar{R}(\mathbf{Q}(t-1),\mathbf{r}(t))] + \sum_{t=1}^T \mathbb{E}[\bar{R}(\mathbf{Q}(t-1),\mathbf{r}(t))] - R^{QB}(T).$$

Similar to the previous arguments, we have that

$$\mathbb{E}\left[\sum_{t=1}^T \bar{R}(\mathbf{Q}(t-1),\mathbf{r}(t)) - R^{QB}(T)\right] \leq \sum_{j=1}^J \mathbb{E}[Q_j(T)]. \tag{EC.33}$$

We aim to bound the difference $\sum_{t=1}^T \mathbb{E}[R^* - \bar{R}(\mathbf{Q}(t-1),\mathbf{r}(t))]$. To this end, note that

$$\begin{aligned}&R^* - \mathbb{E}[\bar{R}(\mathbf{Q}(t-1),\mathbf{r}(t))] \\ =&\sum_{j=1}^J\sum_{i=1}^I \lambda_i p_{ij}^* r_{ij} - \mathbb{E}\left[\sum_{j=1}^J \sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))} \lambda_i r_{ij}\right]\end{aligned}$$

$$
\begin{aligned}
=&\mathbb{E}\left[\sum_{j=1}^{J}\sum_{i=1}^{I}\lambda_i p_{ij}^*\left(r_{ij}-r_{ij}(t)\right)\right]-\mathbb{E}\left[\sum_{j=1}^{J}\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}\lambda_i\left(r_{ij}-r_{ij}(t)\right)\right]\\
&+\mathbb{E}\left[\sum_{j=1}^{J}\left(\sum_{i=1}^{I}\lambda_i p_{ij}^* r_{ij}(t)-\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}\lambda_i r_{ij}(t)\right)\right]\\
=&\mathbb{E}[U(t)]+\mathbb{E}\left[\sum_{j=1}^{J}\left(\sum_{i=1}^{I}\lambda_i p_{ij}^* r_{ij}(t)-\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}\lambda_i r_{ij}(t)\right)\right],
\end{aligned}
$$

where we have defined

$$
U(t):=\sum_{j=1}^{J}\sum_{i=1}^{I}\lambda_i p_{ij}^*\left(r_{ij}-r_{ij}(t)\right)-\sum_{j=1}^{J}\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}\lambda_i\left(r_{ij}-r_{ij}(t)\right).
$$

By the design of our matching algorithm,

$$
\sum_{j=1}^{J}\sum_{i=1}^{I}\lambda_i p_{ij}^*\left(r_{ij}(t)-\epsilon Q_j(t-1)\right)\le\sum_{j=1}^{J}\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}\lambda_i\left(r_{ij}(t)-\epsilon Q_j(t-1)\right). \tag{EC.34}
$$

Thus,

$$
\begin{aligned}
&\sum_{j=1}^{J}\left(\sum_{i=1}^{I}\lambda_i p_{ij}^* r_{ij}(t)-\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}\lambda_i r_{ij}(t)\right)\\
=&\sum_{j=1}^{J}\left(\sum_{i=1}^{I}\lambda_i p_{ij}^*\left(r_{ij}(t)-\epsilon Q_j(t-1)+\epsilon Q_j(t-1)\right)-\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}\lambda_i\left(r_{ij}(t)-\epsilon Q_j(t-1)+\epsilon Q_j(t-1)\right)\right)\\
\le&\epsilon\left[\sum_{j=1}^{J}\sum_{i=1}^{I}\lambda_i p_{ij}^* Q_j(t-1)-\sum_{j=1}^{J}\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}\lambda_i Q_j(t-1)\right],
\end{aligned}
$$

where the inequality follows from (EC.34). Hence,

$$
\begin{aligned}
&\frac{1}{\epsilon}\mathbb{E}[R^*-\bar{R}(\mathbf{Q}(t-1),\mathbf{r}(t))|\mathbf{Q}(t-1),\mathbf{r}(t)]+\frac{1}{2}\left[\mathbb{E}[V(\mathbf{Q}(t))|\mathbf{Q}(t-1),\mathbf{r}(t)]-V(\mathbf{Q}(t-1))\right]\\
\le&\frac{1}{\epsilon}U(t)+\mathbb{E}\left[\sum_{j=1}^{J}\sum_{i=1}^{I}\lambda_i p_{ij}^* Q_j(t-1)-\sum_{j=1}^{J}Q_j(t-1)\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}\lambda_i\middle|\mathbf{Q}(t-1),\mathbf{r}(t)\right]\\
&+\frac{1}{2}\sum_{j=1}^{J}\mathbb{E}\left[\left(\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}A_i(t)-1\right)^2\middle|\mathbf{Q}(t-1),\mathbf{r}(t)\right]\\
&+\sum_{j=1}^{J}Q_j(t-1)\mathbb{E}\left[\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}A_i(t)-1\middle|\mathbf{Q}(t-1),\mathbf{r}(t)\right]\\
\le&\frac{1}{\epsilon}U(t)+\frac{1}{2}B_1+\sum_{j=1}^{J}\left(\sum_{i=1}^{I}\lambda_i p_{ij}^*-1\right)Q_j(t-1)\\
\le&\frac{1}{\epsilon}U(t)+\frac{1}{2}B_1,
\end{aligned}
$$

where B_1 is defined in the proof of Proposition 2 and the last inequality is due to $\sum_{i=1}^{I} \lambda_i p_{ij}^* - 1 \le 0$ for each j. Adding both sides over $t = 1, \ldots, T$ and taking expectations, we obtain

$$\sum_{t=1}^{T} \mathbb{E}[R^* - \bar{R}(\mathbf{Q}(t-1), \mathbf{r}(t))] \le \sum_{t=1}^{T} \mathbb{E}[U(t)] + \frac{\epsilon}{2} B_1 T. \tag{EC.35}$$

We next bound $\mathbb{E}[U(t)]$. To that end, let $F_{ij}(t) := \left\{ r_{ij} - \bar{r}_{ij}(t-1) \le \sqrt{\frac{2\log(t-1)}{\hat{h}_{ij}(t-1)}} \right\}$. We have that

$$\begin{aligned}
&\sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i=1}^{I} \lambda_i p_{ij}^* \left(r_{ij} - r_{ij}(t) \right) \\
=& \sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i=1}^{I} \lambda_i p_{ij}^* (r_{ij} - r_{ij}(t)) \mathbf{1}_{F_{ij}(t)} + \sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i=1}^{I} \lambda_i p_{ij}^* (r_{ij} - r_{ij}(t)) \mathbf{1}_{F_{ij}^c(t)} \\
\le& \sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i=1}^{I} \lambda_i p_{ij}^* \mathbf{1}_{F_{ij}^c(t)} \\
\le& \sum_{t=1}^{T} \sum_{j=1}^{J} \lambda \mathbf{1}_{F_{ij}^c(t)}.
\end{aligned}$$

Taking expectations on both sides and applying Chernoff-Hoeffding Inequality, we have that

$$\mathbb{E}\left[\sum_{t=1}^{T} \sum_{j=1}^{J} \lambda \mathbf{1}_{F_{ij}^c(t)} \right] \le \frac{7}{3} J \lambda.$$

Next, let $G_{ij}(t) := \left\{ \bar{r}_{ij}(t-1) - r_{ij} \le \sqrt{\frac{2\log(t-1)}{\hat{h}_{ij}(t-1)}} \right\}$ We have that

$$\begin{aligned}
&\sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i \in k^{-1}(j; \mathbf{Q}(t-1), \mathbf{r}(t))} \lambda_i \left(r_{ij}(t) - r_{ij} \right) \\
=& \sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i \in k^{-1}(j; \mathbf{Q}(t-1), \mathbf{r}(t))} \lambda_i \left(r_{ij}(t) - r_{ij} \right) \mathbf{1}_{G_{ij}(t)} + \sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i \in k^{-1}(j; \mathbf{Q}(t-1), \mathbf{r}(t))} \lambda_i \left(r_{ij}(t) - r_{ij} \right) \mathbf{1}_{G_{ij}^c(t)}
\end{aligned}$$

For the first term, we have that

$$\sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i \in k^{-1}(j; \mathbf{Q}(t-1), \mathbf{r}(t))} \lambda_i \left(r_{ij}(t) - r_{ij} \right) \mathbf{1}_{G_{ij}(t)} \le 2 \sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i \in k^{-1}(j; \mathbf{Q}(t-1), \mathbf{r}(t))} \lambda_i \sqrt{\frac{2\log(t-1)}{\hat{h}_{ij}(t-1)}}.$$

Recall that $\Gamma_{ij}(t)$ denotes the number of type i jobs assigned to server j during time slot t. Noting that $\Gamma_{ij}(t) = \mathbf{1}_{k^{-1}(j; \mathbf{Q}(t-1), \mathbf{r}(t))}(i) A_i(t)$ and using the independence between $A_i(t)$ and $\mathbf{Q}(t-1)$, we can deduce that

$$\begin{aligned}
2\mathbb{E}\left[\sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i \in k^{-1}(j; \mathbf{Q}(t-1), \mathbf{r}(t))} \lambda_i \sqrt{\frac{2\log(t-1)}{\hat{h}_{ij}(t-1)}} \right] &= 2\mathbb{E}\left[\sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i \in k^{-1}(j; \mathbf{Q}(t-1), \mathbf{r}(t))} A_i(t) \sqrt{\frac{2\log(t-1)}{\hat{h}_{ij}(t-1)}} \right] \\
&= 2\mathbb{E}\left[\sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{i=1}^{I} \Gamma_{ij}(t) \sqrt{\frac{2\log(t-1)}{\hat{h}_{ij}(t-1)}} \right].
\end{aligned}$$

At this stage, we can employ an argument similar to that in the Proof of Proposition 3 to get

$$\mathbb{E}\left[\sum_{t=1}^{T}\Gamma_{ij}(t)\sqrt{\frac{1}{\hat{h}_{ij}(t-1)}}\right]\leq 2+2\mathbb{E}\left[\sqrt{h_{ij}(T-1)}\right]+\mathbb{E}[Q_{ij}^{\max}(T)].$$

It follows that

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{j=1}^{J}\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}\lambda_i\left(r_{ij}(t)-r_{ij}\right)\mathbf{1}_{G_{ij}(t)}\right]\leq 2\sqrt{2\log T}\left(2IJ+2J\sqrt{IT}+\sum_{i,j}\mathbb{E}[Q_{ij}^{\max}(T)]\right).$$

Similarly, using the tower property and Chernoff-Hoeffding Inequality, we have that

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{j=1}^{J}\sum_{i\in k^{-1}(j;\mathbf{Q}(t-1),\mathbf{r}(t))}\lambda_i\left(r_{ij}(t)-r_{ij}\right)\mathbf{1}_{G_{ij}^c(t)}\right]\leq\frac{7}{3}J\lambda.$$

Combining the above pieces, we attain an upper bound for function $U(t)$:

$$\begin{aligned}\sum_{t=1}^{T}\mathbb{E}[U(t)]&\leq 2\sqrt{2\log T}\left(2IJ+2J\sqrt{IT}+\sum_{i,j}\mathbb{E}[Q_{ij}^{\max}(T)]\right)+\frac{14}{3}J\lambda\\&\leq 2\sqrt{2\log T}\left(2IJ+2J\sqrt{IT}+IJ\mathbb{E}[Q_{\Sigma}^{\max}(T)]\right)+\frac{14}{3}J\lambda.\end{aligned}\tag{EC.36}$$

Combining (EC.33), (EC.35) and (EC.36) leads to the desired bound as stated in part (iii) of the theorem. □

EC.3. Additional Numerical Investigations

In this section, we present additional numerical results. We first explore different benchmarks and then focus on the queue-based algorithm, which has shown suboptimal performance in Hsu et al. (2022).

EC.3.1. Two-Stage Algorithms

As a way to resolve the exploration-exploitation trade-off, the UCB approach continuously updates estimates based on new information. As an alternative, the explore-then-exploit approach involves initial exploration followed by exploitation and is commonly used in online learning, giving rise to a two-stage process.

In the following experiments, we introduce two additional two-stage methods as benchmarks: the two-stage barrier-function-based algorithm (TSBB) and the two-stage queue-based algorithm (TSQB). The parameter settings remain the same as those depicted in Figure 1. In both TSBB and TSQB, the two-stage process begins with a predefined time step value T_0. Prior to T_0, assignments are made randomly. At time T_0, the algorithms utilize the samples collected before T_0 to estimate the reward parameters $\mathbf{r}$, denoted as $r_{ij}(T_0)$. Starting from time T_0+1, the TSBB algorithm makes assignments based on the probabilities obtained by solving (7) with the estimated rewards $r_{ij}(T_0)$ replacing $r_{ij}(t)$. On the other hand, the TSQB algorithm assigns jobs according to (9) using the estimated rewards $r_{ij}(T_0)$ instead of $r_{ij}(t)$.

In Zhong et al. (2022), the authors suggest that for a two-stage algorithm, the length of the learning phase should be set at $O(\log(T))$, due to the exponential decay of tail probability. Hence, we conduct experiments with different

T_0 values (25, 50, 100, and 200) to investigate their impact on performance. Figure EC.1 displays the results. In each plot, the orange solid line represents the greedy algorithm, the yellow dotted line represents the queue-based algorithm, the blue dashed line represents the barrier-function-based algorithm, and the purple circled line represents the upper bound. The TSBB algorithm is denoted by the green line with '+', and the TSQB algorithm is represented by the red dash-dotted line.

When T_0 is set to 25 or 50, the performance of the two-stage algorithms matches that of the barrier-function-based and queue-based algorithms. This suggests that the two-stage algorithms can accurately estimate rewards using samples from a brief period, and continuous updates, as in the UCB approach, offer minimal additional benefit. These results also validate the efficiency of our barrier-function-based and queue-based assignment rules when reliable reward estimates are available. However, as T_0 exceeds 100, the performance of the two-stage algorithms gradually deteriorates. This decline is primarily attributed to the exploration stage, where random assignments can lead to potential loss or regret. If the exploration stage is prolonged, the incurred loss becomes significant, resulting in overall poorer performance. Based on these observations, we conclude that the duration of the first stage significantly impacts the performance of two-stage methods. Choosing an appropriate duration is crucial, but determining the optimal length in practical applications is challenging, as an incorrect selection may result in substantial regret. It's important to note that a small T_0 can lead to unstable performance, as there is a higher chance of the first-stage estimates deviating from the true values. In contrast, the UCB approach tends to be more robust and reliable in such scenarios.

EC.3.2. Performance of the Queue-Based Algorithm With "Floating" Job Types

Next, we dig further into the performance of the queue-based algorithm. In Hsu et al. (2022), the authors numerically show that the queue-based algorithm can sometimes deliver suboptimal performance. A key difference between our model and theirs is that in our model, job types are known upon arrival, while in Hsu et al. (2022), job types are unknown. Our intuition is that this difference in modeling assumptions explains why the queue-based algorithm performs well in our model but not in theirs.

To verify our intuition, we consider the following scenario: The platform serves I types of clients and has J servers. Each time period, a client of type i arrives with probability μ_i and remains with the platform for a random duration. We assume that the client's duration follows a geometric distribution with parameter ν_i. While a type i client is present, the client will generate jobs at each time period according to a Poisson distribution with rate λ_i. Similar to Hsu et al. (2022), we assume that the platform does not know the type of each arriving client and hence

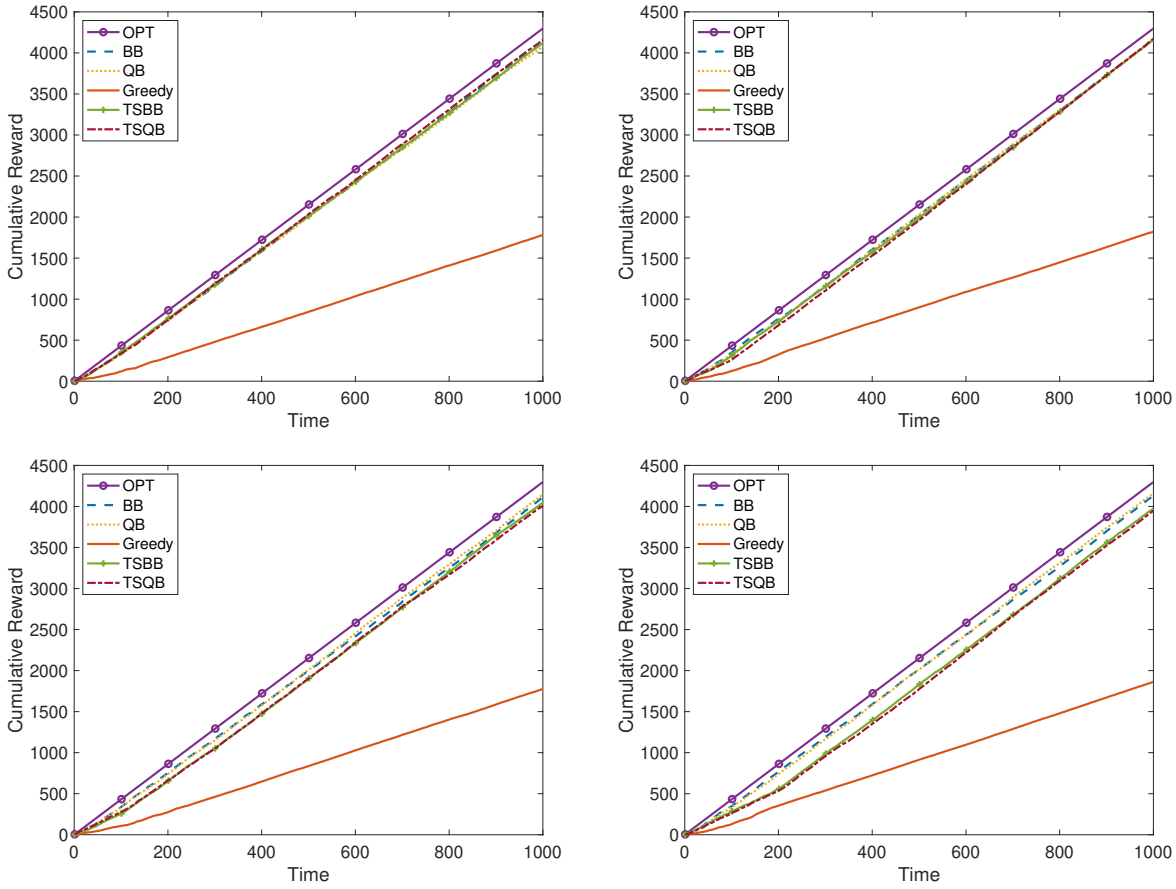

Figure EC.1 Cumulative rewards for two-stage algorithms, where the first stage has length $T_0 = 25$ (top left), 50 (top right), 100 (bottom left), 200 (bottom right)

needs to infer the type of the client. As a meaningful benchmark, the corresponding offline program is given by:

$$\begin{aligned} \max_{p_{ij}\geq 0} \ & \sum_{i=1}^{I}\sum_{j=1}^{J} \mu_i \frac{1}{\nu_i}\lambda_i p_{ij} r_{ij} \\ \text{subject to } & \sum_{i=1}^{I} \mu_i \frac{1}{\nu_i}\lambda_i p_{ij} \leq 1 \text{ for all } j \\ & \sum_{j=1}^{J} p_{ij} = 1 \text{ for all } i \end{aligned} \tag{EC.37}$$

In the following experiments, we focus on the performance of the queue-based algorithm under different parameter settings. We omit other benchmarks and modify the queue-based algorithm to learn the rewards from matching each client to the servers, using UCB estimates for assignments since the client types are unknown. The parameter settings are as follows: $I = 2$, $J = 6$, $\lambda = [2, 3]$, $T = 1000$, and $\epsilon = 0.1$. We simplify by setting $\mu_1 = \mu_2 = \nu_1 = \nu_2 = \nu$. Figure EC.2 illustrates the performance of the queue-based algorithm for three values of

ν: $\nu = 0.01$ (blue dashed line), $\nu = 0.1$ (yellow dotted line), and $\nu = 0.5$ (orange solid line). The linear program upper bound, represented by the purple circled line, remains the same in all cases.

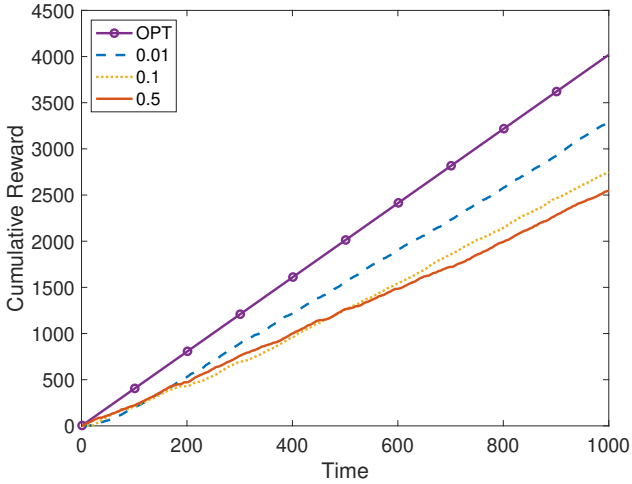

Figure EC.2 Cumulative rewards for the queue-based algorithm

From the results, we observe that the queue-based algorithm performs suboptimally in the current setting compared to previous experiments as it deviates further from the upper bound. However, we notice a slight improvement in performance as ν decreases. This improvement can be attributed to the longer duration of clients staying with the platform when ν is lower, leading to more opportunities for the platform to collect data and form better estimates of the rewards, hence improved assignment decisions. The inclusion of random client arrivals adds an extra layer of stochastic viability, which is likely to exacerbate the performance deterioration compared to the base case discussed in the main paper. This is because the number of clients present fluctuates over time, resulting in varying effective job arrival rates for each type. Consequently, there are instances when longer queues occur, further aggravating the learning slowdown effect. Overall, these findings suggest that when job types are known, pooling samples of the same type to estimate rewards enhances the learning process and enables the queue-based algorithm to achieve better performance.