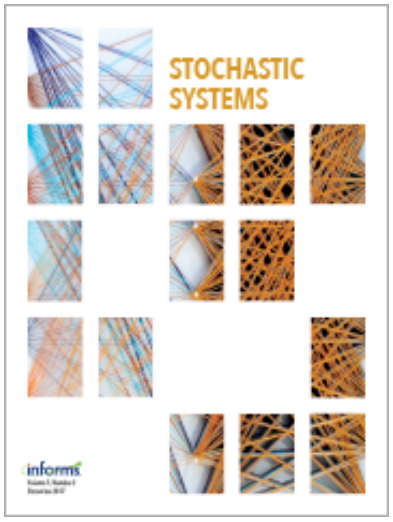
This article was downloaded by: [216.73.216.11] On: 04 June 2026, At: 17:34
Publisher: Institute for Operations Research and the Management Sciences (INFORMS)
INFORMS is located in Maryland, USA



Stochastic Systems

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Learning Payoffs While Routing in Skill-Based Queues

Sanne van Kempen, Jaron Sanders, Fiona Sloothaak, Maarten G. Wolf

To cite this article:
Sanne van Kempen, Jaron Sanders, Fiona Sloothaak, Maarten G. Wolf (2026) Learning Payoffs While Routing in Skill-Based Queues. Stochastic Systems

Published online in Articles in Advance 05 May 2026

. https://doi.org/10.1287/stsy.2024.0095

This work is licensed under a Creative Commons Attribution 4.0 International License. You are free to copy, distribute, transmit and adapt this work, but you must attribute this work as "*Stochastic Systems*. Copyright © 2026 The Author(s). https://doi.org/10.1287/stsy.2024.0095, used under a Creative Commons Attribution License: https://creativecommons.org/licenses/by/4.0/."

Copyright © 2026 The Author(s)

Please scroll down for article—it is on subsequent pages

With 12,500 members from nearly 90 countries, INFORMS is the largest international association of operations research (O.R.) and analytics professionals and students. INFORMS provides unique networking and learning opportunities for individual professionals, and organizations of all types and sizes, to better understand and use O.R. and analytics tools and methods to transform strategic visions and achieve better outcomes.
For more information on INFORMS, its publications, membership, or meetings visit http://www.informs.org

https://pubsonline.informs.org/journal/stsy

STOCHASTIC SYSTEMS
Articles in Advance, pp. 1–25
ISSN 1946-5238 (online)

Learning Payoffs While Routing in Skill-Based Queues

Sanne van Kempen,[a,*] **Jaron Sanders,**[a] **Fiona Sloothaak,**[a] **Maarten G. Wolf** [b]

[a] Department of Mathematics and Computer Science, Eindhoven University of Technology, 5612 AZ Eindhoven, Netherlands; [b] Koninklijke KPN N.V., 3072 AP Rotterdam, Netherlands

*Corresponding author

Contact: s.f.m.v.kempen@tue.nl, https://orcid.org/0009-0000-7263-2873 (SvK); jaron.sanders@tue.nl, https://orcid.org/0000-0003-0187-2065 (JS); f.sloothaak@tue.nl, https://orcid.org/0000-0002-1720-7070 (FS); maarten.wolf@kpn.com (MGW)

Received: December 13, 2024
Revised: July 11, 2025; December 10, 2025
Accepted: March 18, 2026
Published Online in Articles in Advance: May 5, 2026

https://doi.org/10.1287/stsy.2024.0095

Copyright: © 2026 The Author(s)

Abstract. Motivated by applications in service systems, we consider queueing systems where each customer must be handled by a server with the right skill set. We focus on optimizing the routing of customers to servers in order to maximize the total payoff of customer-server matches. In addition, customer-server-dependent payoff parameters are assumed to be unknown a priori. We construct a machine learning algorithm that adaptively learns the payoff parameters while maximizing the total payoff and prove that it achieves polylogarithmic regret. Moreover, we show that the algorithm is asymptotically optimal up to logarithmic terms by deriving a regret lower bound. The algorithm leverages the basic feasible solutions of a static linear program as the action space. The regret analysis overcomes the complex interplay between queueing and learning by analyzing the convergence of the queue length process to its stationary behavior. We also demonstrate the performance of the algorithm numerically and have included an experiment with time-varying parameters highlighting the potential of the algorithm in nonstatic environments.

Open Access Statement: This work is licensed under a Creative Commons Attribution 4.0 International License. You are free to copy, distribute, transmit and adapt this work, but you must attribute this work as "*Stochastic Systems*. Copyright © 2026 The Author(s). https://doi.org/10.1287/stsy.2024.0095, used under a Creative Commons Attribution License: https://creativecommons.org/licenses/by/4.0/."

Funding: This work is part of Valuable AI, a research collaboration between the Eindhoven University of Technology and the Koninklijke KPN N.V. Parts of this research have been funded by the IMPULS program of the Eindhoven Artificial Intelligence Systems Institute, and by Holland High Tech through the Top Consortium for Knowledge and Innovation program High Tech Systems and Materials, via the public–private partnership allowance scheme.

Supplemental Material: The online appendix is available at https://doi.org/10.1287/stsy.2024.0095.

Keywords: skill-based routing • queueing • adaptive control • multiarmed bandits • regret analysis

1. Introduction

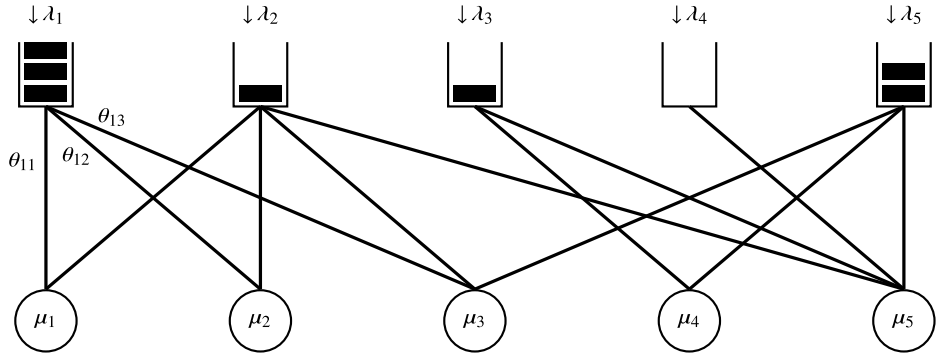
Service systems such as contact centers, computer networks, and manufacturing systems are widely used in practice (Koole and Mandelbaum 2002, Harchol-Balter 2013, Chen et al. 2020). Achieving the highest possible quality of service in such systems is consequently of general importance. However, actually doing so is typically quite challenging because of complex interactions within the system between, for example, customers and servers. Moreover, service provisioning is an intrinsically uncertain process.

In this paper, we develop a machine learning algorithm that can attain the highest possible performance in one such service system. Specifically, we focus on a finite skill-based queueing system with customer-server-dependent random payoffs. One such system is illustrated in Figure 1. We consider different customers to have different needs and different servers to be better at helping certain types of customers over others. In fact, some servers might even be unable to help some types of customers. We model these aspects by assuming that there are compatibility relations between customers and servers and by assuming that whenever a customer is served by a server, a random payoff is generated that depends on the specific customer-server pairing. Because the different servers are shared between the different customer types, and because they are limited in their number, the highest possible average reward can only be achieved by optimizing how one matches customers to servers.

Within queueing literature, a canonical tool for doing server allocation is routing policies. Routing policies decide which customer is served by which server, and at what time. However, routing policies typically do not take into account (i) compatibility relations or (ii) payoffs generated by different customer-server pairs. Furthermore, they usually do not consider (iii) uncertainty in and/or a lack of knowledge of model parameters (such as the average payoffs θ_{ij}).

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Figure 1. A Skill-Based Queueing System with Compatibility Lines and Customer-Server Dependent Payoffs



Note. Here, the λ_i denote arrival rates, the μ_j denote service rates, and the θ_{ij} indicate the average payoff generated upon service completion of a type-i customer at server j.

Our machine learning algorithm, Algorithm 1 in Section 4, does take into account aspects (i)–(iii). Letting $D_{ij}(t)$ refer to the number of departures of type-i customers at server j up to time t, and $\mathcal{L}$ to the set of compatibility lines, Algorithm 1 maximizes the long-term expected reward rate

$$\frac{1}{t}\sum_{(ij)\in\mathcal{L}}\theta_{ij}\mathbb{E}(D_{ij}(t)). \tag{1}$$

Algorithm 1 does so while (a) honoring the queueing dynamics and compatibility relations; (b) guaranteeing the stability of all queues; (c) being able to hone in on the true system parameters θ_{ij} when either misspecified or unknown a priori; and (d) dealing with the exploration-exploitation dilemma efficiently. These aspects (a)–(d) will be discussed in more detail in Section 1.4 below but may be summarized in one sentence as follows: we establish in Theorem 2 that Algorithm 1 achieves a polylogarithmic regret and in Theorem 1 that this is (asymptotically), up to logarithmic terms, the best possible within a certain class of stable policies.

The nature of the optimization problem that we study required us to combine techniques from different fields to develop Algorithm 1 and prove Theorems 1 and 2. Firstly, we were inspired by techniques for Multiarmed Bandit (MAB) problems from *machine learning* for the learning aspect. Secondly, in the regret analysis of Algorithm 1, we had to rely on *queueing theory* to establish the typical behavior of the queue length process under the dynamic routing policy. Finally, a key idea was to leverage *combinatorial optimization* to come up with "good" actions for Algorithm 1.

Let us briefly discuss these three facets in more detail:

We first discuss how queueing dynamics complicate decision making and how MAB techniques can be adapted to cope.

Contrary to the classical MAB setting (Lai and Robbins 1985), we have to deal with queueing dynamics and stability constraints that complicate the problem. Specifically, both the decision moments and the set of available routing decisions are subject to the underlying random queueing dynamics: we can only route a customer if there are a compatible customer and server available. Moreover, simply routing each customer to the server with the highest payoff might not result in a stable system in the long run.

To overcome this, we define episodes and only make a decision on the routing strategy at the start of each episode. The estimated reward of the chosen action is updated after each episode based on observed payoff samples. This way, similar to the MAB setting, the algorithm learns the average reward of each action adaptively. To face the exploration-exploitation trade-off efficiently, we use Upper Confidence Bounds (UCBs) for the payoff parameters (Auer et al. 2002).

We next describe how queueing theory is used to analyze Algorithm 1.

Each episode, Algorithm 1 chooses an action that determines a fixed routing policy for the duration of that episode. This means that the queue length process will show different behavior in each episode. Worse yet, at the moment that the routing policy changes, the queueing system will likely be far from its new equilibrium behavior. It then takes some time to adjust to the new environment. Still, in order to prove convergence results of Algorithm 1, we must show that sufficient payoff samples of all the different lines are collected. To this end, we

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

utilize known results of the stationary behavior and analyze the typical time of convergence of the queue length process to stationarity.

Lastly, we explain how combinatorial optimization allows one to identify "good" actions.

The optimal routing problem inherently poses combinatorial challenges. In particular, the optimal routing problem can be considered as a stochastic variant of the optimal transport problem (Bertsimas and Tsitsiklis 1997) with a finite set of possible solutions. We formulate an optimal transport linear program (LP) and exploit its structural properties in the construction of our algorithm: First, we characterize the basic feasible solutions of the LP in terms of routing rates. We then consider the different sets of routing rates as candidate routing policies in an MAB setting. In the analysis, we exploit the dual of the LP to decompose the regret.

1.1. Main Contributions

Our main contributions can be summarized as follows:

1.1.1. Regret Lower Bound. We present an asymptotic regret lower bound for a class of routing policies satisfying certain stability constraints in Theorem 1. The stability constraints ensure that the queue backlog does not grow infinitely large. In particular, we prove that any policy of this class must suffer $\Omega(\ln(t))$ regret to learn the payoff parameters.

In an MAB setting, a regret lower bound is often described in terms of a *suboptimality gap* for each suboptimal action. The gap measures the instant loss in reward by choosing this action over the optimal action (Lai and Robbins 1985, Burnetas and Katehakis 1996, Auer et al. 2002). A straightforward implementation of this approach proves to be difficult because in our model, some suboptimal routing decision might be necessary to maintain stability. Therefore, it is not trivial to identify the optimal action at each decision moment, nor to quantify the suboptimality gaps.

Our solution to this problem is as follows: we split the lines into an "optimal" and "suboptimal" set, based on the optimal solution of the optimal transport LP, and consequently quantify the suboptimality gaps using the dual. Burnetas and Kanavetas (2012) and Burnetas et al. (2017) use a similar method to obtain a regret lower bound in an MAB setting with cost constraints, although their work is limited to programs with only two equality constraints. The proof of Theorem 1 also uses a change-of-measure argument, a technique that is frequently used in the MAB setting (Lai and Robbins 1985; Burnetas and Katehakis 1996, 1997; Lattimore and Szepesvári 2020).

1.1.2. Adaptive Learning and Routing Algorithm. We present a machine learning algorithm that utilizes the basic feasible solutions of the optimal transport LP as actions in Algorithm 1—a method that, to the best of our knowledge, has not been explored before in this setting. We consider the related optimal transport LP, where the objective function (that includes the payoff parameter θ) is unknown. However, by the properties of the LP, the set of basic feasible solutions is finite, and the optimal solution is attained at one of these solutions (Bertsimas and Tsitsiklis 1997). Hence, we identify the set of basic feasible solutions as actions. Each solution uniquely defines a set of routing rates. Moreover, the objective function is approximated using UCB indices of the payoff parameters.

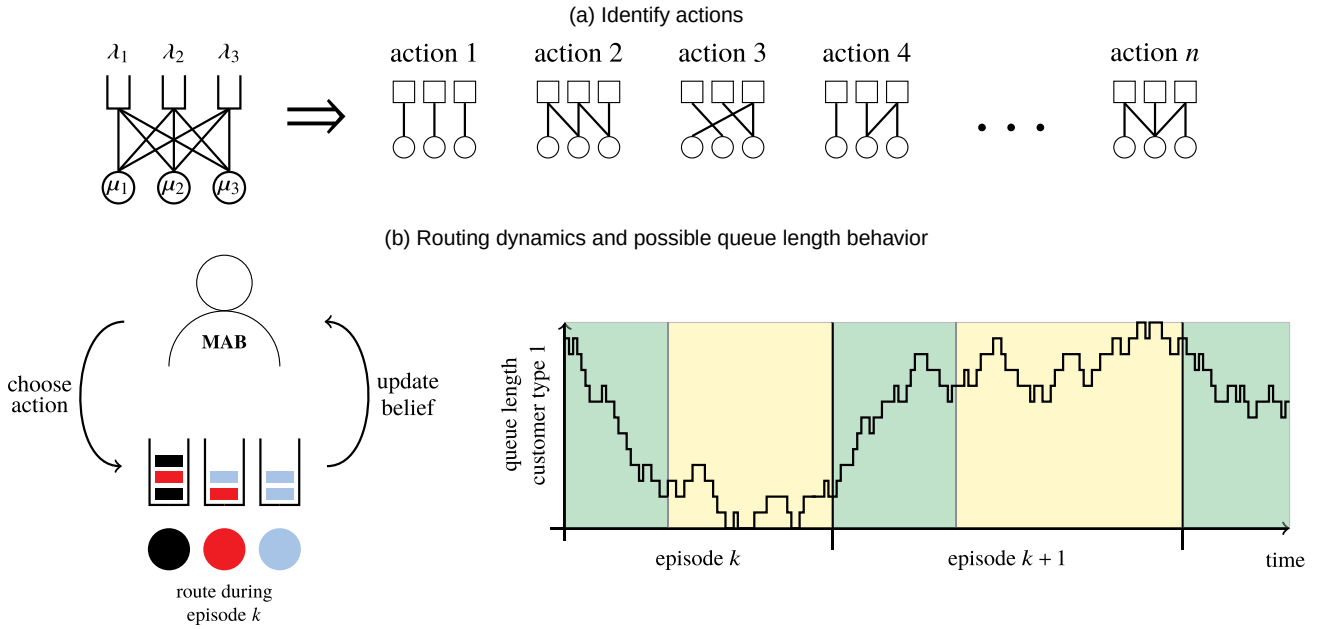
The learning is schematically depicted in Figure 2 and works as follows. At the start of each episode, the algorithm chooses the action with maximal (estimated) reward. During the episode, customers are routed according to the rates of the chosen action. In particular, customers are assigned a label corresponding to one of the compatible servers upon arrival (illustrated by colors in Figure 2). Each server serves customers with matching label in First-Come-First-Served (FCFS) order, and service completions generate payoffs. If, at the start of an episode, the chosen action differs from the action of the previous episode, the algorithm assigns new labels for all queueing customers sampled according to the routing rates of the new action.

For each line, we maintain a UCB index of its payoff parameter. At the end of the episode, the payoff samples are used to update the UCB indices. The reward of an action is then computed as the sum over all lines of the UCB index weighted by the routing rate of the action. We assume Bernoulli payoffs for simplicity.

1.1.3. Regret Upper Bound. We show that the asymptotic regret of our policy is $\mathcal{O}(\ln^{2\beta}(t))$ for any $\beta > 1$ in Theorem 2. This regret is only slightly worse than the $\mathcal{O}(\ln(t))$ regret of the benchmark UCB policy (Auer et al. 2002) in the classical MAB setting, even though our learning problem is more challenging. The regret analysis relies on concentration inequalities that bound the probability that the empirical average of payoff samples deviates far from its mean. However, it differs from the standard MAB techniques because the number of routing decisions and therefore the number of samples is subject to queueing dynamics. We deal with this issue by splitting each episode into a transient and a stationary phase and analyzing the convergence time of the queue length process

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Figure 2. Schematic Overview of Algorithm 1



to its stationary measure. We let the episodes grow in length such that the probability that the queue length process does not reach stationarity decreases sufficiently fast.

1.1.4. Numerical Experiments. Lastly, we analyze Algorithm 1 and its performance in several numerical experiments. We find that in our experiments, the average reward of Algorithm 1 converges quickly to the optimal reward and chooses the optimal action most of the time, even in a larger system with many different actions. The algorithm outperforms a benchmark greedy policy that myopically optimizes the instantaneous payoff without considering long-term effects. Even though the theoretical analysis of Algorithm 1 is for a static setting, we test its robustness against the change of a parameter value and find that the algorithm adapts adequately. This highlights the potential applicability of our algorithm beyond static environments.

1.2. Related Literature

The majority of literature on optimal routing in skill-based queues does not account for customer-server-dependent payoffs and focuses on waiting time minimization (Koole and Mandelbaum 2002; Gurvich and Whitt 2010; Adan and Weiss 2012, 2014; Visschers et al. 2012; Krishnasamy et al. 2018; Choudhury et al. 2021; Lee and Vojnovic 2021; Zhong et al. 2022; Yang et al. 2023). In this scenario, the $c\mu$ (or $c\mu/\theta$ in case of abandonments) routing policy that prioritizes customer types based on a ranking depending on the holding costs, is known to be asymptotically optimal (Zhong et al. 2022). Therefore, an intuitive approach for an adaptive learning policy is to route according to a variant of the $c\mu$ policy where the unknown parameters are replaced by empirical estimators. In our case, however, the objective in (1) is difficult to evaluate in full generality, because the departure process depends on the availability of servers and the state of the queues, which are random and fairly intractable. Even for simple routing policies like FCFS Assign-the-Longest-Idle-Server (ALIS), the expected number of type-i departures at server j is, to the best of our knowledge, unknown for finite time. Only a select few convergence results are known for specific queueing systems (Adan and Weiss 2012, 2014).

On top of the classical exploration-exploitation trade-off, learning in queueing systems requires one to deal with limited capacity and the interplay between queueing and learning (queueing degrades learning and vice versa). Some works decouple learning and queueing by splitting the time horizon into distinct exploration and exploitation phases (Johari et al. 2021, Zhong et al. 2022, Jia et al. 2024), whereas others directly implement a reinforcement learning algorithm (Liu et al. 2019, Choudhury et al. 2021) or apply Bayesian inference (Bimpikis and Markakis 2019, Shah et al. 2020). Our Algorithm 1, however, is an integrated learning and routing policy.

The focus in our work lies on online payoff maximization in skill-based queues. Payoff maximization problems with unknown utility functions are well studied in literature (Tan and Srikant 2012, Agarwal et al. 2013, Agrawal et al. 2014, Yu et al. 2017, Tan et al. 2020, Vera and Banerjee 2021). In the setting of queueing systems, there exists

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

a variety of algorithms based on different approaches. For example, Sun et al. (2023), Hsu et al. (2022), and Kim and Vojnovic (2021) propose utility guided algorithms based on a static LP. This method is an extension of the Lyapunov drift plus penalty reward, where the Lyapunov drift assures stability and the penalty is used for payoff maximization. Sun et al. (2023) obtain moment bounds for the maximal queue length in the system and an instance independent regret upper bound of order $\mathcal{O}(\sqrt{t\ln(t)})$. However, the algorithm suffers a linear loss in reward due to stability constraints. In our work, we instead split the regret into a queueing and a learning component.

Other algorithms for online payoff maximization in queueing systems are considered by Liu et al. (2020) and Steiger et al. (2022), who use Lyapunov drift analysis. A different method is used by Fu and Modiano (2022), who present a routing algorithm that combines the Join-the-Shortest-Queue (JSQ) routing scheme with a confidence ball learning algorithm (introduced by Dani et al. 2008) in the setting of optimization under bandit feedback. It is shown that, given a fixed horizon, the threshold parameter K in the JSQ-K algorithm can be tuned in such a way that polylogarithmic regret can be attained, but no guarantees are provided on the convergence of the routing rates to their optimal values. In our work, we consider an asymptotic result, and, therefore, our algorithm does not require the knowledge of the time horizon.

Lastly, primal-dual methods can be applied that make routing decisions based on estimates of the dual variables. Such methods are especially useful when there is uncertainty in the feasible region of the primal problem, making it impractical to solve directly. For example, Tan and Srikant (2012) present an algorithm in which current queue lengths serve as proxies for the dual variables in a setting with unknown arrival rates. Similarly, Wei et al. (2023) introduce an algorithm for two-sided matching with unknown arrival rates where a notion of customer shortage is used to ensure convergence to the optimal matching rates. In contrast, our method does not rely on estimating the feasible region. Instead, we exploit the structure of the known feasible region to learn the optimal routing rates.

The analysis of our learning algorithm relies on establishing convergence properties of an episodic queue length process to a stationary probability measure. Similar convergence properties have been used in Jia et al. (2024), and Besbes and Zeevi (2015). We let the episodes grow in duration so that convergence is achieved with increasing probability. Sanders et al. (2016), Comte et al. (2023), Jiang and Walrand (2010), and Liu et al. (2019) use similar concepts in different settings of optimal control in queueing networks.

The methodology of our learning algorithm bears most resemblance with the achievable region approach in Bertsimas (1995) and Dacre et al. (1999), where the goal is to solve an optimization problem with an unknown utility function using the feasible region spanned by a set of constraints. Bertsimas (1995) and Dacre et al. (1999) consider the optimal routing problem in a static setting with known parameters, whereas our policy integrates learning and optimization in an online fashion.

1.3. Outline

The remainder of this work is organized as follows. In Section 2, we discuss the model, optimal routing problem, and regret formulation. Next, in Section 3, we study the regret of routing policies and present an asymptotic regret lower bound for a class of routing policies. In Section 4, we present the adaptive learning algorithm and show that its asymptotic regret is of polylogarithmic order. Lastly, in Section 5, we present a numerical implementation of the algorithm.

2. Model Description

In this section, we describe the queueing system and routing policies in more detail. We analyze a related deterministic optimal transport LP and discuss its properties. Lastly, we present a definition of regret of a routing policy, which is based on the optimal transport LP.

2.1. Arrival and Service Process

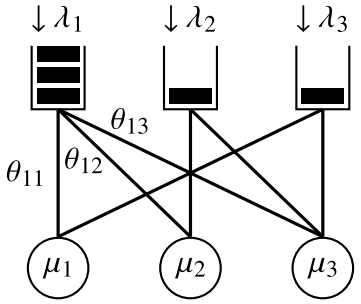
We consider a continuous-time queueing system with a fixed set of customer types $\mathcal{I} = \{1,\dots,I\}$, servers $\mathcal{J} = \{1,\dots,J\}$, and a set of lines connecting compatible customer types and servers $\mathcal{L} = \{(ij) : i \in \mathcal{I}, j \in \mathcal{J}\}$. Here, $I, J \in \mathbb{N}_{>0}$. Let $L = |\mathcal{L}| \in \mathbb{N}_{>0}$ denote the number of compatible lines. We assume that the bipartite graph of customer types, servers, and lines $\mathcal{G} = (\mathcal{I} \cup \mathcal{J}, \mathcal{L})$ is connected. To exclude trivial cases, we assume that $L > I + J - 1$. We denote the set of servers that are compatible with customer type i by $\mathcal{S}_i$, and, similarly, we denote the set of customer types that are compatible with server j by $\mathcal{C}_j$—that is,

$$\mathcal{S}_i = \{j \in \mathcal{J} : (ij) \in \mathcal{L}\}, \tag{2}$$

$$\mathcal{C}_j = \{i \in \mathcal{I} : (ij) \in \mathcal{L}\}. \tag{3}$$

An example of such a queueing network is given in Figure 3.

Figure 3. A Queueing Model with $I = 3$ Types of Customers and $J = 3$ Servers with Compatibility Lines



Note. Here, for example, $\mathcal{S}_1 = \{1,2,3\}$ and $\mathcal{C}_1 = \{1,3\}$.

For $i \in \mathcal{I}$, type-i customers are assumed to arrive according to a Poisson process with rate $\lambda_i > 0$. They wait in queue i until they are allocated to a compatible (and idle) server. We assume that there are no customers in the system at time $t = 0$. For $j \in \mathcal{J}$, service times of customers at server j are assumed exponentially distributed with rate $\mu_j > 0$. We also assume that service times are independent between servers and across customers. Note that the service time distribution only depends on the server and not the customer type. For convenience, we write $\lambda = (\lambda_1, \ldots, \lambda_I)$ and similarly $\mu = (\mu_1, \ldots, \mu_J)$. For stability, we assume that for any subset of customer types $\mathcal{A} \subseteq \mathcal{I}$,

$$\sum_{i \in \mathcal{A}} \lambda_i < \sum_{j \in \mathcal{S}_{\mathcal{A}}} \mu_j. \tag{4}$$

Here, $\mathcal{S}_{\mathcal{A}} = \bigcup_{i \in \mathcal{A}} \mathcal{S}_i$ denotes the set of servers that are compatible with any of the customer types $i \in \mathcal{A}$. By Adan and Weiss (2014, theorem 2.1), there exists a routing policy such that the joint queue length process under that policy is ergodic.

Upon the service completion of a type-i customer at server j, a random payoff Y_{ij} is obtained, sampled from a payoff distribution $P_{\theta_{ij}}$ with support $\mathcal{X} \subseteq \mathbb{N}_{\geq 0}$ and unknown and finite mean $\theta_{ij} \in \mathbb{R}_{\geq 0}$. Note that although all lines share the same payoff distribution, the mean payoff differs per line. We denote the vector of unknown mean payoffs as $\theta = (\theta_{ij})_{(ij) \in \mathcal{L}} \in \mathbb{R}^L_{\geq 0}$. As quantification of the difference between the payoff distribution for two lines (ij) and $(k\ell) \in \mathcal{L}$, we use the Kullback-Leibler (KL) divergence, given by

$$I(\theta_{ij}, \theta_{k\ell}) := \sum_{x \in \mathcal{X}} P_{\theta_{ij}}(x) \ln\left(\frac{P_{\theta_{ij}}(x)}{P_{\theta_{k\ell}}(x)}\right). \tag{5}$$

We assume that the payoff distribution is such that

$$0 < I(\theta_{ij}, \theta_{k\ell}) < \infty \quad \text{when} \quad \theta_{ij} < \theta_{k\ell}. \tag{6}$$

2.2. Routing Policies

A *routing policy* π determines which customer is served by which server at what time. We denote the probability measure and expectation with respect to routing policy π and payoff vector $\theta \in \mathbb{R}^L_{\geq 0}$ by $\mathbb{P}^\theta_\pi$ and $\mathbb{E}^\theta_\pi$, respectively. Moreover, we denote the total number of service completions—that is, departures—of type-i customers at server j up to time $t \geq 0$ by $D_{ij}(t)$.

We consider a class of routing policies that satisfy for $\varepsilon \geq 0$ and for any payoff vector $\theta \in \mathbb{R}^L_{\geq 0}$ the following conditions:

$$\lim_{t \to \infty} t\lambda_i - \sum_{j \in \mathcal{S}_i} \mathbb{E}^\theta_\pi(D_{ij}(t)) < \infty, \quad \forall i \in \mathcal{I}, \tag{7}$$

$$t(\mu_j - \varepsilon) - \sum_{i \in \mathcal{C}_j} \mathbb{E}^\theta_\pi(D_{ij}(t)) \geq 0, \quad \forall j \in \mathcal{J}, \quad \forall t \geq 0. \tag{8}$$

Constraint (7) is a notion of stability: we require that the difference between the expected number of arrivals and departures—that is, the expected queue length—of any customer type does not diverge. Note that the set of policies satisfying (7) is nonempty by Assumption (4). Constraint (8) requires that the expected utilization of server j is at most $(\mu_j - \varepsilon)/\mu_j$. If $\varepsilon > 0$, (8) prevents critically loaded servers.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

We consider the reward of a policy π with respect to payoff vector $\theta \in \mathbb{R}^L_{\geq 0}$ up to time $t \geq 0$ as the total average payoff—that is,

$$\sum_{(ij)\in\mathcal{L}} \theta_{ij}\mathbb{E}^{\theta}_{\pi}(D_{ij}(t)). \tag{9}$$

As discussed previously, (9) can be difficult to compute in full generality. Instead, we consider a static version of the optimal control problem which is discussed next.

2.3. Optimal Transport LP

For a payoff vector $\theta \in \mathbb{R}^L_{\geq 0}$ and $\varepsilon \geq 0$, let

$$\mathrm{LP}(\theta,\varepsilon): \quad \max_{x} \sum_{(ij)\in\mathcal{L}} \theta_{ij}x_{ij}, \tag{10a}$$

$$\text{s.t.} \sum_{j\in\mathcal{S}_i} x_{ij} = \lambda_i, \quad \forall i \in \mathcal{I}, \tag{10b}$$

$$\sum_{i\in\mathcal{C}_j} x_{ij} \leq \mu_j - \varepsilon, \quad \forall j \in \mathcal{J}, \tag{10c}$$

$$x_{ij} \geq 0, \quad \forall (ij) \in \mathcal{L}. \tag{10d}$$

The objective of $\mathrm{LP}(\theta,\varepsilon)$ is to maximize the average payoff per time unit. Each optimization variable $x_{ij} \in [0,\lambda_i]$ can be interpreted as the long-term rate of type-i customers that are routed to server j per time unit. Constraint (10b) requires that all customers are routed to some server, and Constraint (10c) states that the capacity of each server must not be exceeded. Here, $\varepsilon \geq 0$ is a fixed amount of slack at each server. The feasibility of $\mathrm{LP}(\theta,\varepsilon)$ depends on the value of ε, which will be discussed later. $\mathrm{LP}(\theta,\varepsilon)$ can be regarded as a variant of the assignment problem initially presented in Shapley and Shubik (1971), where the resources are divisible. Similar LP formulations of the static planning problem in the context of reward maximization in queueing models are used in Hsu et al. (2022), Fu and Modiano (2021, 2022), Tan and Srikant (2012), Tan et al. (2020), and Vera and Banerjee (2021).

We briefly discuss some fundamental properties of LPs. Consider an LP in standard form: $\max_x c'x$ s.t. $Ax = b$, $x \geq 0$ with $A \in \mathbb{R}^{m\times n}$. For $B \subseteq \{1,\dots,n\}$ with $|B| = m$, we denote by $\mathbf{B} = [A_{B(1)},\dots,A_{B(m)}] \in \mathbb{R}^{m\times m}$ the matrix formed by the columns $B(1),\dots,B(m)$. We call B a *basis* if $\mathbf{B}$ is invertible (or, equivalently, has full rank); see Bertsimas and Tsitsiklis (1997). In this case, the vector $x_B \in \mathbb{R}^n$ obtained by setting $(x_B)_{B(j)} = (\mathbf{B}^{-1}b)_j$ for $j = 1,\dots,m$ and $(x_B)_k = 0$ for $k \notin B$, is a *basic solution*. A basis (or equivalently a basic solution) is *feasible* if $\mathbf{B}^{-1}b \geq 0$ and *nondegenerate* if $(\mathbf{B}^{-1}b)_k \neq 0$ for all $k = 1,\dots,m$. Lastly, recall that if an LP has an optimal solution, then it has an optimal basic feasible solution (Bertsimas and Tsitsiklis 1997, theorem 2.7).

In the remainder of this work, we assume nondegeneracy—that is, we assume that all basic feasible solutions of $\mathrm{LP}(\theta,\varepsilon)$ and its dual are nondegenerate. We note that this assumption implies that each basic feasible solution of the primal (dual) implies a unique basic feasible solution of the dual (primal) (see, e.g., Bertsimas and Tsitsiklis 1997, exercise 4.12)).

2.3.1. Basic Feasible Solutions.

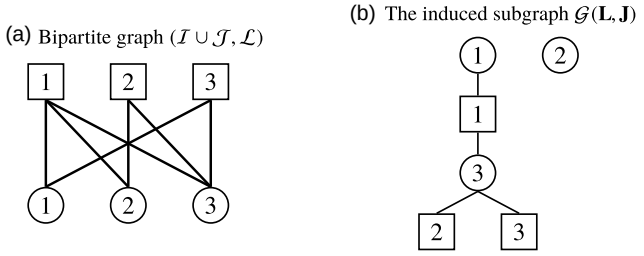
We will now characterize the basic feasible solutions of $\mathrm{LP}(\theta,\varepsilon)$ in (10). Because $\mathrm{LP}(\theta,\varepsilon)$ has a finite number of linear inequality constraints, the set of basic feasible solutions is finite (Bertsimas and Tsitsiklis 1997, corollary 2.1).

It is a known result that a basic feasible solution of $\mathrm{LP}(\theta,\varepsilon)$ induces a spanning forest on the bipartite graph $(\mathcal{I}\cup\mathcal{J},\mathcal{L})$ (Fu and Modiano 2022, proposition 2). We extend the result of Fu and Modiano (2022) by providing the explicit form of any basis (see Lemma 1 below) and any basic solution (see Lemma 2 below). The proofs are presented in Online Appendix C.1 and Online Appendix C.2, respectively.

The introduction of Lemmas 1 and 2 requires some more notation. For $\mathbf{L} \subseteq \mathcal{L}$ and $\mathbf{J} \subseteq \mathcal{J}$ we denote by $\mathcal{G}(\mathbf{L},\mathbf{J})$ the subgraph of $(\mathcal{I}\cup\mathcal{J},\mathcal{L})$ induced by $\mathbf{L}$ and $\mathbf{J}$. We say that $\mathcal{G}(\mathbf{L},\mathbf{J})$ is a *spanning forest* of $(\mathcal{I}\cup\mathcal{J},\mathcal{L})$ if (i) it is a union of trees—that is, it contains no cycles; (ii) each $v \in \mathcal{I}\cup\mathcal{J}$ is contained in $\mathcal{G}(\mathbf{L},\mathbf{J})$; and (iii) each tree of $\mathcal{G}(\mathbf{L},\mathbf{J})$ contains a unique root node $j \in \mathbf{J}$. In such a tree, parent and child nodes of servers are customer types and vice versa. We denote the subtree rooted in v, including v itself, by $\curlywedge(v)$. We let $\mathcal{C}(\curlywedge(v))$ denote the set of customer types contained in the subtree rooted in v—that is, $\mathcal{C}(\curlywedge(v)) = \curlywedge(v)\cap\mathcal{I}$. Similarly, $\mathcal{S}(\curlywedge(v)) = \curlywedge(v)\cap\mathcal{J}$. An example is shown in Figure 4.

We are now in a position to state Lemmas 1 and 2:

Figure 4. Consider the Queueing System in Figure 3 and Let $\mathbf{L} = \{(11), (13), (23), (33)\}$ and $\mathbf{J} = \{1, 2\}$



Note. $\mathcal{G}(\mathbf{L}, \mathbf{J})$ is then a spanning forest of $(\mathcal{I} \cup \mathcal{J}, \mathcal{L})$ consisting of two trees.

Lemma 1. *Let $\varepsilon \geq 0$, $\mathbf{L} \subseteq \mathcal{L}$, and $\mathbf{J} \subseteq \mathcal{J}$. $B = \mathbf{L} \cup \mathbf{J}$ is a basis of $\mathrm{LP}(\theta, \varepsilon)$ if and only if $\mathcal{G}(\mathbf{L}, \mathbf{J})$ is a spanning forest of $(\mathcal{I} \cup \mathcal{J}, \mathcal{L})$.*

Lemma 2. *Let $\varepsilon \geq 0$, $\mathbf{L} \subseteq \mathcal{L}$, and $\mathbf{J} \subseteq \mathcal{J}$. If $B = \mathbf{L} \cup \mathbf{J}$ is a basis of $\mathrm{LP}(\theta, \varepsilon)$, then its corresponding basic solution $x \in \mathbb{R}^L$ satisfies*

$$x_{ij} = \begin{cases} \sum_{k \in \mathcal{C}(\pi(i))} \lambda_k - \sum_{\ell \in \mathcal{S}(\pi(i))} (\mu_\ell - \varepsilon), & \text{if } i \text{ is a child of } j \text{ in } \mathcal{G}(\mathbf{L}, \mathbf{J}), \\ \sum_{\ell \in \mathcal{S}(\pi(j))} (\mu_\ell - \varepsilon) - \sum_{k \in \mathcal{C}(\pi(j))} \lambda_k, & \text{if } i \text{ is the parent of } j \text{ in } \mathcal{G}(\mathbf{L}, \mathbf{J}), \\ 0 & \text{if } (ij) \in \mathcal{L} \setminus \mathbf{L}. \end{cases} \tag{11}$$

2.3.2. Feasibility. Note that by (4), $\mathrm{LP}(\theta, 0)$ is feasible—that is, there exists at least one basic feasible solution. The feasibility region of $\mathrm{LP}(\theta, \varepsilon)$ in (10) depends on ε: the larger ε, the smaller the feasible region. Lemma 3 characterizes the range of ε such that the $\mathrm{LP}(\theta, \varepsilon)$ has the same set of feasible bases as $\mathrm{LP}(\theta, 0)$. The lemma is proven in Online Appendix C.3.

Lemma 3. *Let $B = \mathbf{L} \cup \mathbf{J}$ with $\mathbf{L} \subseteq \mathcal{L}$ and $\mathbf{J} \subseteq \mathcal{J}$ be a nondegenerate feasible basis of $\mathrm{LP}(\theta, 0)$. If*

$$0 \leq \varepsilon < \min_{j \in \mathcal{J}} \frac{\sum_{\ell \in \mathcal{S}(\pi(j))} \mu_\ell - \sum_{k \in \mathcal{C}(\pi(j))} \lambda_k}{|\mathcal{S}(\pi(j))|}, \tag{12}$$

then B is also a nondegenerate feasible basis of $\mathrm{LP}(\theta, \varepsilon)$.

2.4. Regret Formulation

We use the value of $\mathrm{LP}(\theta, \varepsilon)$ in (10) as the oracle reward to define the regret of a learning policy. Let $\varepsilon > 0$ satisfy (12), $\theta \in \mathbb{R}^L_{\geq 0}$ be a payoff vector, and $x^\theta \in \mathbb{R}^L_{\geq 0}$ be the optimal basic feasible solution of $\mathrm{LP}(\theta, \varepsilon)$ in (10). We define the (expected) *regret* of a routing policy π at time $t \geq 0$ as

$$R^\theta_\pi(t) := \sum_{(ij) \in \mathcal{L}} \theta_{ij} (t x^\theta_{ij} - \mathbb{E}^\theta_\pi(D_{ij}(t))). \tag{13}$$

Lemma 4 states that the asymptotic regret is nonnegative and is proven in Online Appendix C.4.

Lemma 4. *Let $\varepsilon \geq 0$ and let π satisfy (7) and (8). Then, $\lim_{t \to \infty} R^\theta_\pi(t)/t \geq 0$.*

3. Regret Lower Bound

We provide an asymptotic lower bound for the regret for routing policies satisfying Constraints (7) and (8) in Theorem 1. To this end, we first provide a regret decomposition in Lemma 5.

3.1. Regret Decomposition

Let $\varepsilon > 0$ satisfy (12), $\theta \in \mathbb{R}^L_{\geq 0}$ be a payoff vector, and $x^\theta \in \mathbb{R}^L_{\geq 0}$ be the unique optimal solution of $\mathrm{LP}(\theta, \varepsilon)$ in (10). We define the set of optimal lines $O(\theta)$ as the lines with nonzero load in the optimal solution $x^\theta \in \mathbb{R}^L$ and the set of suboptimal lines $O^c(\theta)$ as its complement:

$$O(\theta) = \{(ij) \in \mathcal{L} : x^\theta_{ij} > 0\}, \tag{14}$$

$$O^c(\theta) = \mathcal{L} \setminus O(\theta). \tag{15}$$

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

The dual problem (Bertsimas and Tsitsiklis 1997) of LP(θ,ε) is given by

$$\mathrm{D}(\theta,\varepsilon): \quad \min_{v,w} \ \sum_{i\in\mathcal{I}} \lambda_i v_i + \sum_{j\in\mathcal{J}} (\mu_j - \varepsilon) w_j, \tag{16a}$$

$$\text{s.t.} \ \ v_i + w_j - \theta_{ij} \geq 0, \quad \forall (ij) \in \mathcal{L}, \tag{16b}$$

$$v_i \in \mathbb{R}, \quad \forall i \in \mathcal{I}, \tag{16c}$$

$$w_j \geq 0, \quad \forall j \in \mathcal{J}. \tag{16d}$$

By strong duality, we have that D(θ,ε) is feasible because LP(θ,ε) is feasible. Moreover, x^θ and v^θ, w^θ are an optimal primal-dual pair if and only if they satisfy the complementary slackness conditions (Bertsimas and Tsitsiklis 1997, theorem 4.5), which are given by

$$x^\theta_{ij}(v^\theta_i + w^\theta_j - \theta_{ij}) = 0, \quad \forall (ij) \in \mathcal{L}, \tag{17}$$

$$\left(\mu_j - \varepsilon - \sum_{i\in\mathcal{C}_j} x^\theta_{ij}\right) w^\theta_j = 0, \quad \forall j \in \mathcal{J}. \tag{18}$$

We note that the optimal solutions of LP(θ,ε) and D(θ,ε) are unique due to the nondegeneracy assumption. Moreover, the nondegeneracy assumption implies strict complementarity (see, e.g., in Bertsimas and Tsitsiklis 1997, exercises 4.12 and 4.20c).

For a line $(ij) \in \mathcal{L}$, we define

$$\phi^\theta_{ij} = v^\theta_i + w^\theta_j - \theta_{ij}, \tag{19}$$

as its suboptimality gap. Here, $v^\theta \in \mathbb{R}^I$, $w^\theta \in \mathbb{R}^J_{\geq 0}$ is the optimal solution of D(θ,ε) in (16). Note that $\phi^\theta_{ij} = 0$ for any $(ij) \in O(\theta)$ by (17) and $\phi_{k\ell} > 0$ for any $(k\ell) \in O^c(\theta)$ by strict complementarity.

Lemma 5 presents a regret decomposition based on the line classification. The proof can be found in Online Appendix C.5.

Lemma 5. *Let $v^\theta \in \mathbb{R}^I$, $w^\theta \in \mathbb{R}^J_{\geq 0}$ be the optimal solution of* D(θ,ε) *in* (16). *Let π satisfy* (7) *and* (8). *Then, the regret in* (13) *can be expressed as*

$$R^\theta_\pi(t) = \underbrace{\sum_{(k\ell)\in O^c(\theta)} \phi^\theta_{k\ell} \mathbb{E}^\theta_\pi(D_{k\ell}(t))}_{\mathrm{I}} + \underbrace{\sum_{j\in\mathcal{J}} w^\theta_j \left(t(\mu_j - \varepsilon) - \sum_{i\in\mathcal{C}_j} \mathbb{E}^\theta_\pi(D_{ij}(t))\right)}_{\mathrm{II}} + \underbrace{\sum_{i\in\mathcal{I}} v^\theta_i \left(t\lambda_i - \sum_{j\in\mathcal{S}_i} \mathbb{E}^\theta_\pi(D_{ij}(t))\right)}_{\mathrm{III}}. \tag{20}$$

In Decomposition (20), term I represents the loss in reward due to customers being routed over suboptimal lines. The other two summations represent regret accumulated by deviating from the oracle routing rates. In particular, II measures regret from not utilizing the allowed capacity from servers, and III measures regret accumulated from customers that are still waiting in the queue or are in service at time t, because we do not obtain any payoff until a customer finishes service. Note that because $v^\theta_i \in \mathbb{R}$, the queue length can actually contribute to negative regret, because it might be beneficial to let customers wait until a high payoff server is available.

3.2. Regret Lower Bound

Before we can state the asymptotic regret lower bound, we need to introduce some further notation. Let $\theta \in \mathbb{R}^L_{\geq 0}$. We define $A(\theta,ij,a) \in \mathbb{R}^L_{\geq 0}$ element-wise by

$$A(\theta,ij,a)_{k\ell} = \begin{cases} a & \text{if } (k\ell) = (ij), \\ \theta_{k\ell} & \text{otherwise.} \end{cases} \tag{21}$$

In particular, $A(\theta,ij,a)$ represents a payoff vector that is (almost) equal to θ except for index (ij). For $(ij) \in O^c(\theta)$, we define

$$\Delta(\theta,ij) = \{a \geq 0 : (ij) \in O(A(\theta,ij,a))\}, \tag{22}$$

as the set of parameter values that make (ij) an optimal line without changing the payoff values of the other lines. Note that by construction, either $\Delta(\theta,ij) = \emptyset$, or $\Delta(\theta,ij) = (a_0,\infty)$ for some $a_0 \in \mathbb{R}_{\geq 0}$. We define

$$K(\theta,ij) = \inf\{I(\theta_{ij},a) : a \in \Delta(\theta,ij)\}, \tag{23}$$

as the minimum distance in terms of KL divergence between the parameter θ_{ij} and the set of parameter values that make (ij) an optimal line. Here, the KL divergence I is defined in (5).

Lemma 6 states that any suboptimal line could become optimal by changing its payoff value. It is proven in Online Appendix C.6.

Lemma 6. *For any $(ij) \in O^c(\theta)$, we have $\Delta(\theta, ij) \neq \emptyset$.*

Theorem 1 provides an asymptotic regret lower bound and is proven in Online Appendix A.

Theorem 1. *Let $\varepsilon > 0$ satisfy (12) and let π satisfy (7) and (8). Moreover, we assume that π is consistent—that is, that for any deterministic payoff vector $\xi \in \mathbb{R}^L_{\geq 0}$,*

$$\lim_{t\to\infty} \mathbb{P}^{\xi}_{\pi}(D_{ij}(t) < b\ln(t)) = 0, \qquad \forall b > 0, \qquad \forall (ij) \in O(\xi). \tag{24}$$

Then, for any $\theta \in \mathbb{R}^L_{\geq 0}$,

$$\lim_{t\to\infty} \frac{R^{\theta}_{\pi}(t)}{\ln(t)} \geq \sum_{(k\ell)\in O^c(\theta)} \frac{\phi^{\theta}_{k\ell}}{K(\theta, k\ell)} > 0. \tag{25}$$

The lower bound in (25) is a summation over all suboptimal lines of the suboptimality gap divided by the infimum KL distance as defined in (23). This means that a suboptimal line that is nearly optimal, in the sense that its suboptimality gap is small, leads to a large contribution to the regret, because it is difficult for a policy to distinguish between optimal and nearly optimal lines. Note that the lower bound is not redundant, as strict positivity is guaranteed.

Assumption (24) is a notion of consistency which is a common assumption in regret analysis in MAB literature (Lattimore and Szepesvári 2020), as it excludes policies that are specialized on a subset of problem instances. Intuitively speaking, it states that optimal lines are sufficiently explored. In particular, (24) requires that the number of departures of optimal lines eventually grows larger than logarithmic with probability one. In classical bandit literature, an assumption on the concentration of the expectation of the number of optimal decisions is often sufficient, because there is exactly one decision at each time step. In our case, however, the number of departures is random even for stationary policies. It remains an open question whether a similar regret lower bound can be proven for a class of policies satisfying a weaker constraint such as $\lim_{t\to\infty} \mathbb{E}^{\theta}_{\pi}(tx^{\theta}_{ij} - D_{ij}(t))/t^b = 0$ for any $b > 0$ and $(ij) \in \mathcal{L}$, in combination with Assumptions (7) and (8).

Theorem 1 states that the regret is $\Omega(\ln(t))$, which is the same order as the lower bound for classical MABs (Lai and Robbins 1985, theorem 2). The proof of Theorem 1 in Online Appendix A is structured in a similar way as those presented in Lai and Robbins (1985), Burnetas and Katehakis (1996, 1997), and Lattimore and Szepesvári (2020) and uses a change-of-measure argument. Because payoffs are random samples, each line is associated with a level of uncertainty that decreases with the number of obtained samples, which in our case corresponds to departures. In order to prove the theorem, we construct a second, "confusing" system which has the same parameters as the original model with the exception of the average payoff $\theta_{k\ell}$ for some line $(k\ell) \in \mathcal{L}$, which we set to a high value such that line $(k\ell)$ is an optimal line in the second model. We then prove that a consistent policy needs $\Omega(\ln(t))$ samples of payoffs of line $(k\ell)$ in order to distinguish between the two models.

Recall that Theorem 1 assumes that the payoff distribution is discrete (see the end of Section 2.1). We note that the proof can be adapted to include payoff distributions defined by a probability density function $f(\cdot, \theta_{ij})$: in this case, the KL divergence in (5) changes to

$$I(\theta_{ij}, \theta_{k\ell}) := \int_0^{\infty} f(x; \theta_{ij}) \ln\left(\frac{f(x; \theta_{ij})}{f(x; \theta_{k\ell})}\right) \mathrm{d}x. \tag{26}$$

Similarly, the Radon-Nikodym (RN) derivative in the proof of Theorem 1 in Online Appendix A changes to $\mathrm{d}\mathbb{P}^{\theta}_{\pi}/\mathrm{d}\mathbb{P}^{\theta'}_{\pi} = \prod_{i=1}^{m} f(X_i; \theta_{k\ell})/f(X_i; \theta'_{kl})$, and Equations (69), (72), and (73) change accordingly. Extension of our result to general probability measures requires careful construction of the RN derivative. This can be done in line with the discussion following Lattimore and Szepesvári (2020, proposition 4.8).

4. Adaptive Queue Routing Policy

In this section, we introduce a routing algorithm called Adaptive UCB Queue Routing Algorithm (UCB QR) and show that its asymptotic regret is close to the lower bound in Theorem 1. Specifically, Theorem 2 proves that its regret is polylogarithmic as time grows large.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

4.1. Description of the Algorithm

We consider a fixed network structure $\mathcal{I}, \mathcal{J}, \mathcal{L}, \lambda, \mu$, and $\varepsilon > 0$ satisfying (12). We assume that the payoff distribution of line $(ij) \in \mathcal{L}$ is Bernoulli distributed with parameter $\theta_{ij} \in [0,1]$.

We consider the basic feasible solutions of LP(θ, ε) in (10) as the different actions in an MAB setting. Because the set of basic feasible solutions of LP(θ, ε) is finite (see Section 2.3.1), the set of actions is finite and will be denoted by $\mathcal{A}$. Concretely, each action $a \in \mathcal{A}$ is associated with a basic feasible solution $x^a \in \mathbb{R}^L_{\geq 0}$ of LP(θ, ε). We call the action corresponding to the optimal basic feasible solution the optimal action, which is unique by the nondegeneracy assumption.

For $a \in \mathcal{A}$, we denote the set of lines with positive load by $\mathcal{L}^a \subseteq \mathcal{L}$—that is,

$$\mathcal{L}^a := \{(ij) \in \mathcal{L} : x^a_{ij} > 0\}. \tag{27}$$

The long-term average reward $r^a \in \mathbb{R}_{\geq 0}$ of action a will be denoted by

$$r^a := \sum_{(ij)\in\mathcal{L}^a} x^a_{ij}\theta_{ij}, \tag{28}$$

and its associated suboptimality gap by

$$d^a := \max_{\tilde{a}} r^{\tilde{a}} - r^a. \tag{29}$$

The UCB QR algorithm is given in Algorithm 1.

Algorithm 1 (Adaptive UCB Queue Routing Algorithm)

1: Initialize $k = 1$, for all $(ij) \in \mathcal{L}$ initialize $T_{ij}(0) = 0$, $\hat{\theta}_{ij}(0) = 0$, and $U_{ij}(0) = \infty$ and for all $a \in \mathcal{A}$ initialize $U^a(0) = \infty$.
2: **for** each episode $k = 1, 2, \ldots$ of length H_k **do**
3: Choose action $A_k = \arg\max_a U^a(k-1)$, with ties broken arbitrarily.
4: **if** $A_k \neq A_{k-1}$ **then**
5: Reallocate customers to virtual queues: for each waiting type-i customer, assign it to the virtual queue of server j with probability $x^{A_k}_{ij}/\lambda_i$.
6: For each $j \in \mathcal{J}$, order the customers in virtual queue j by their arrival time.
7: **end if**
8: **for** time $t \in [0, H_k]$ **do**
9: Serve customers according to Algorithm 2: FCFS RR(A_k).
10: **end for**
11: For each line $(ij) \in \mathcal{L}^{A_k}$, observe $N^k_{ij} \in \mathbb{N}_{\geq 0}$ payoff samples $(Y^\ell_{ij})_{\ell=1,\ldots,N^k_{ij}}$.
12: For each line $(ij) \in \mathcal{L}^{A_k}$, update

$$T_{ij}(k) = T_{ij}(k-1) + N^k_{ij}, \tag{30}$$

$$\hat{\theta}_{ij}(k) = \frac{\hat{\theta}_{ij}(k-1)T_{ij}(k-1) + \sum_{\ell=1}^{N^k_{ij}} Y^\ell_{ij}}{T_{ij}(k)}, \tag{31}$$

$$U_{ij}(k) = \begin{cases} \hat{\theta}_{ij}(k) + \sqrt{\dfrac{\ln(k)}{T_{ij}(k)}} & \text{if } T_{ij}(k) > 0, \\ \infty & \text{if } T_{ij}(k) = 0. \end{cases} \tag{32}$$

13: Update the UCB index of action A_k,

$$U^{A_k}(k) = \sum_{(ij)\in\mathcal{L}^{A_k}} x^{A_k}_{ij} U_{ij}(k). \tag{33}$$

14: **end for**

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Algorithm 2 (FCFS RR(a))

1: **for** each arrival of a type-i customer **do**
2: Assign customer to the virtual queue of server j with probability x_{ij}^a/λ_i.
3: If the server is idle, start service.
4: **end for**
5: **for** each service completion at server j **do**
6: Obtain a payoff $Y_{ij} \sim \text{Ber}(\theta_{ij})$, where i is the departing customer's type.
7: If the virtual queue of server j is nonempty, start service of the customer at the head of the queue.
8: **end for**

The learning algorithm works as follows. We maintain UCB indices for the average payoffs of lines $(ij) \in \mathcal{L}$ and use these to compute UCB indices for each action $a \in \mathcal{A}$. At the start of each episode, we choose the action $a \in \mathcal{A}$ with the highest UCB index. During the episode, we route customers according to the rates $x^a \in \mathbb{R}^L_{\geq 0}$ using the FCFS Random-Routing (FCFS RR) scheme in Algorithm 2. We use virtual queues to implement the FCFS RR scheme. For each departure of a type-i customer at server j, we obtain a payoff sample from a Bernoulli distribution with parameter $\theta_{ij} \in [0,1]$. At the end of each episode, we use the obtained payoff samples to update the UCB indices. Note that customers may be reallocated to a different virtual queue when a change in action occurs at episode transitions. Episode k has a predetermined length $H_k \in \mathbb{R}_{>0}$ that may depend on k.

For each line $(ij) \in \mathcal{L}$, we keep track of $T_{ij}(k)$, the total number of departures of type-i customers at server j up to episode number $k \in \mathbb{N}_{\geq 1}$. Similarly, we keep track of the empirical mean $\hat{\theta}_{ij}(k)$ of type-(ij) payoffs and a UCB estimator $U_{ij}(k)$. The counters and empirical means are both initialized at zero, whereas the UCB estimator is initialized at infinity as an incentive for the algorithm to obtain at least one sample from each line. The number of type-(ij) departures in episode k, N_{ij}^k, in line 11 is random due to the queueing dynamics.

We note that Algorithm 2 may lead to higher queueing delays when compared with other throughput optimal routing rules such as Serve-the-Longest-Queue. However, for the regret analysis, we require a routing rule for which the empirical routing rates $\mathbb{E}(D_{ij}(t))/t$ converge to the target rates x_{ij} as $t \to \infty$. To the best of our knowledge, not many such rules exist in literature, because routing rates and even stability conditions are generally difficult to determine for arbitrary routing rules for skill-based queues (Adan and Weiss 2012, Weiss 2021).

4.2. Small Example

We illustrate how the algorithm works on a small example. We consider the queueing system illustrated on the left in Figure 5 with $\varepsilon = 0.5$. It can be verified that (12) is satisfied. LP(θ, ε) in (10) has six nondegenerate basic feasible solutions which we label as actions in Figure 5.

The routing of customers using the virtual queues is shown in Figure 6.

4.3. Regret Upper Bound

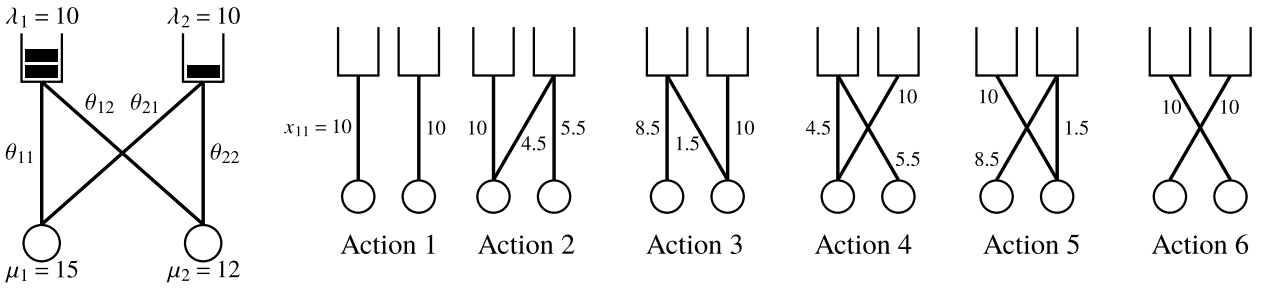
Theorem 2 analyzes the regret of Algorithm 1.

We denote the minimal positive rate on a line across any action as $x_{\min} := \min_{a \in \mathcal{A}} \min_{(ij) \in \mathcal{L}^a} x_{ij}^a$. Moreover, $\tilde{\lambda}_j^a := \sum_{i \in \mathcal{C}_j} x_{ij}^a$ denotes the total arrival rate of customers at server j under action a and

$$\rho_j^a := \tilde{\lambda}_j^a/\mu_j \in (0,1), \tag{34}$$

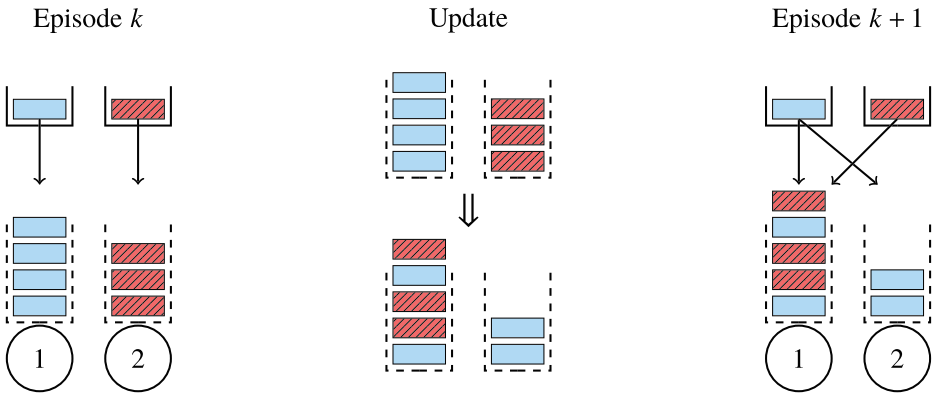
denotes the load of server j under action a, for each $j \in \mathcal{J}$ and $a \in \mathcal{A}$.

Figure 5. A Skill-Based Queueing System with $I = 2$ Customer Types, $J = 2$ Servers, and Compatibility Lines $\mathcal{L} = \{(11), (12), (21), (22)\}$, Along with the Six Different Actions



Note. For each action, the routing rates $\{x_{ij}\}_{(ij) \in \mathcal{L}}$ are illustrated next to the corresponding lines.

Figure 6. Visualization of Customer Routing in Two Consecutive Episodes of Algorithm 1 Where Action 1 Is Chosen in Episode k and Action 4 is Chosen in Episode $k+1$



Notes. Arriving customers are placed in virtual queues, depicted by dashed lines, according to the routing rates associated with the action. In particular, in episode k, type-i customers are routed to virtual queue i (left), whereas in episode $k+1$, a type-1 customer is routed to virtual queue 1 with probability (w.p.) 0.45 and to virtual queue 2 w.p. 0.55 (right). Just before episode $k+1$ starts (middle), waiting customers are redistributed over the virtual queues according to the routing rates of the new action.

Theorem 2. *Let $\varepsilon > 0$ satisfy (12), let $\beta \in \mathbb{R}_{>1}$, and let $\alpha \in \mathbb{R}_{\geq 1}$ satisfy*

$$\alpha \geq \max_{j\in\mathcal{J}} \max_{a\in\mathcal{A}} \left\{ \frac{3(\tilde{\lambda}_j^a + \mu_j)\ln(\rho_j^a) - 2\sqrt{(\mu_j - \tilde{\lambda}_j^a)^2 + 9\tilde{\lambda}_j^a \mu_j \ln^2(\rho_j^a)}}{2(\mu_j - \tilde{\lambda}_j^a)^2 \ln(\rho_j^a)}, 1 \right\}. \tag{35a}$$

Let $H_0 \in \mathbb{R}_{\geq 1}$ and the episode lengths H_k, $k = 1, 2, \ldots$, satisfy

$$H_k = \tau_k + H_0 \quad \text{with} \quad \tau_k = \alpha \ln^\beta(2Jk) \quad \text{and} \quad H_0 \geq \max\left\{ \frac{4}{x_{\min}}, 1 \right\}. \tag{36}$$

Lastly, let $\theta \in \mathbb{R}^L_{\geq 0}$ be a payoff vector. Then, Algorithm 1 satisfies (24). Moreover, the regret $R^\theta(t)$ of Algorithm 1 satisfies

$$\lim_{t\to\infty} \frac{R^\theta(t)}{\alpha \ln^{2\beta}(t)} \leq \sum_{(k\ell)\in O^c(\theta)} \phi^\theta_{k\ell}(\mu_\ell - \varepsilon)|\mathcal{A}| + \sum_{j\in\mathcal{J}} w^\theta_j(\mu_j - \varepsilon)|\mathcal{A}|, \tag{37}$$

where $w^\theta \in \mathbb{R}^J_{\geq 0}$ is the optimal dual variable of $\mathrm{D}(\theta, \varepsilon)$ in (16) and $\phi^\theta \in \mathbb{R}^L_{\geq 0}$ is defined in (19).

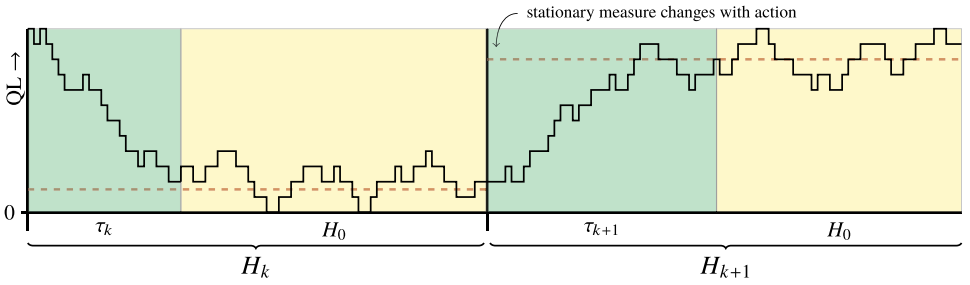
The regret upper bound in Theorem 2 provides a theoretical guarantee on the performance of Algorithm 1. It shows that the regret scales with rate at most $\alpha \ln^{2\beta}(t)$ as the time horizon t grows large. Note that this matches the regret lower bound in Theorem 1 up to logarithmic terms. From the right side of (37), we observe that the upper bound increases with the cardinality of the action set $|\mathcal{A}|$, this will be discussed in Section 4.5.2. Moreover, the regret for each suboptimal action is upper bounded by the instantaneous regret of suboptimal lines in the set $O^c(\theta)$ (recall (15)) multiplied by the maximal expected departure rate at server j, which is $\mu_j - \varepsilon$ by construction. For a suboptimal line $(ij) \in O^c(\theta)$, the instantaneous regret is expressed by the suboptimality gap $\phi_{ij} \in \mathbb{R}_{>0}$ in (19). The last term in (37) is related to the queueing regret: it represents the regret obtained at the start of optimal episodes when the queue length process is not necessarily close to its stationary behavior. We outline the proof of Theorem 2 in Section 4.4. The full proof is given in Online Appendix B.

4.3.1. Episode Length. In order to analyze the regret of the algorithm, we provide lower bounds on the number of departures for each line. The analysis is challenging because the routing scheme changes at the start of each episode; hence, the starting state of the queueing system is not necessarily close to its stationary behavior, as illustrated in Figure 7. This motivates the introduction of a *warmup time* τ_k in the construction of the episode length H_k in (36). The warmup time is followed by a period H_0 of fixed length.

In the analysis of the algorithm, we provide a lower bound on the probability of convergence of the queue length process to its stationary behavior within the warmup time. Constraint (35) guarantees that this bound is sharp enough. The warmup time τ_k is increasing in k, so that this probability converges to one at the right speed in terms of the episode number k.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Figure 7. Possible Realization of the Queue Length Process of the Virtual Queue of Server $j \in \mathcal{J}$ Under Algorithm 1 for Two Episodes



Notes. Episode k consists of a warmup time τ_k followed by a period of fixed length H_0. Every episode, the algorithm can choose a different action with its own stationary measure.

4.4. Proof Outline

We sketch the proof of Theorem 2, highlighting the key contributions. The proof can be split into four steps. We briefly explain these steps, followed by a more in-depth elaboration per step. Without loss of generality, we label the first action as the unique optimal action—that is, $\arg\max_{a\in\mathcal{A}} r^a = 1$.

Step 1. *Analyzing the number of departures in FCFS RR.* For the proof of Theorem 2, we need to show that Algorithm 1 learns the payoff parameters sufficiently fast. Because payoff samples are collected upon departures, we need to prove a lower bound on the number of departures in the queueing system. We do so via an intermediate step: we consider the scenario where the system is initially empty at the start of an episode, which represents a "worst-case" scenario. Next, we show that this worst-case queue length process reaches stationarity within warmup time τ_m with sufficiently high probability (Lemma 7).

Step 2. *Bounding the probability of choosing a suboptimal action.* We note that Algorithm 1 chooses a suboptimal action in row 3 only if the UCB index of the suboptimal action in (33) is at least as high as the UCB index of the optimal action. Broadly speaking, this can happen if (A) the index of the suboptimal action *overestimates* its true mean; or (B) the index of the optimal action *underestimates* its true mean. Compared with the classical MAB analysis in Lattimore and Szepesvári (2020), event (A) is more complicated to analyze in our model, because the number of payoff samples we obtain within one episode is stochastic rather than deterministic as in the classical MAB setting. We show that event (A) is unlikely by showing that on the one hand, overestimation based on "sufficient" samples is unlikely, and on the other hand, it is unlikely to obtain "insufficient" samples. Here, the term sufficient is carefully constructed, as discussed below. Event (B) is unlikely by construction of the UCB estimators. We bound the probability of event (A) in Lemmas 8 and 9 and event (B) in Lemma 10.

Step 3. *Bounding the number of suboptimal episodes.* We use the bounds from Step 2 to bound the number of episodes where Algorithm 1 chooses a suboptimal action in Lemma 11.

Step 4. *Bounding the regret of Algorithm 1.* Finally, we prove Theorem 2 in Online Appendix B using the Regret Decomposition (20). Term I is the main contributing factor because it measures the regret accumulated by using suboptimal lines. We bound this term using Lemma 11. Term II is bounded by analyzing the queue length behavior in episodes where the algorithm chooses the optimal action. Lastly, we bound term III using the properties of FCFS RR and the constraints in LP(θ, ε).

Let us next describe Steps 1–4 in more detail.

4.4.1. Step 1. Analyzing the Number of Departures in FCFS RR. We consider the queue length process $Q_j^{mA_m}(t)$, $t \in [0, H_m)$ of the virtual queue of server $j \in \mathcal{J}$ during episode $m \in \mathbb{N}_{\geq 1}$, where A_m denotes the action chosen by Algorithm 1 in row 3. Within an episode, Algorithm 1 routes customers according to the fixed FCFS RR policy in Algorithm 2. Because arrival processes are independent across customer types, we have by the Poisson splitting and merging properties (Çinlar and Agnew 1968) that the arrival process of $Q_j^{mA_m}(t)$ is a Poisson process with rate $\sum_{i\in\mathcal{I}} x_{ij}^{A_m}$. This implies that $Q_j^{mA_m}(t)$ behaves as an $M/M/1$ queueing system independently from other servers. Recall that x^{A_m} is a basic feasible solution of LP(θ, ε) in (10), so by Constraint (10c), we have $\sum_{i\in\mathcal{C}_j} x_{ij}^{A_m} \leq \mu_j - \varepsilon < \mu_j$. Therefore, $\rho_j^{A_m} = \sum_{i\in\mathcal{I}} x_{ij}^{A_m}/\mu_j < 1$, and hence, $Q_j^{mA_m}(t)$ is positive recurrent (Cohen 1982), and its stationary distribution is given by

$$p_j^{A_m}(n) := \lim_{t\to\infty} \mathbb{P}(Q_j^{mA_m}(t) = n) = (1 - \rho_j^{A_m})(\rho_j^{A_m})^n. \tag{38}$$

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

For each server, we associate two new queue length processes $\hat{Q}_j^{mA_m}(t)$ and $\underline{Q}_j^{mA_m}(t)$ where the initial value $\hat{Q}_j^{mA_m}(0)$ is sampled from the stationary measure $p_j^{A_m}$ and $\underline{Q}_j^{mA_m}(0) = 0$. We couple the processes $Q_j^{mA_m}(t)$, $\hat{Q}_j^{mA_m}(t)$, and $\underline{Q}_j^{mA_m}$ in the sense that the sampled arrival times and service completions (if possible) are the same for all processes, as illustrated in Figure 8.

Note that if $\hat{Q}_j^{mA_m}(t)$ and $\underline{Q}_j^{mA_m}(t)$ collide, then the departure process of $\underline{Q}_j^{mA_m}(t)$ is stationary afterward. Lemma 7 provides a bound on the hitting time of $\hat{Q}_j^{mA_m}(t)$ and $\underline{Q}_j^{mA_m}(t)$. The proof relies on hitting time analysis of an $M/M/1$ queueing system. Lemma 7 extends the result of Jia et al. (2024, proposition 4), in the sense that we provide a sharper bound by exploiting the known formula for the moment generating function of hitting times in an $M/M/1$ system. We also provide exact values for the constants in our result, which are not provided in Jia et al. (2024).

Lemma 7. *Let $\beta \in \mathbb{R}_{>1}$ and let $m \in \mathbb{N}_{\geq 1}$ satisfy $m \geq C_\beta$ with*

$$C_\beta := \frac{1}{2J} \exp\left(\beta^{\frac{1}{\beta-1}}\right). \tag{39}$$

For any $t \geq \tau_m$, where τ_m is defined as in (36), we have $\mathbb{P}(\underline{Q}_j^{mA_m}(t) = \hat{Q}_j^{mA_m}(t),\ \forall j \in \mathcal{J}) \geq 1 - 1/m^\beta$.

The proof of Lemma 7 is in Online Appendix C.7.

4.4.2. Step 2. Bounding the Probability of Choosing a Suboptimal Action.

4.4.2.1. Bound on Event (A). Note that the confidence bound in (33) decreases with the number of obtained samples (departures). This means that overestimation of the true mean is likely when the number of samples is small, but becomes less likely as the number of obtained samples increases. We split the analysis into two parts. First, Lemma 8 bounds the probability that Algorithm 1 has not obtained sufficient samples after $C_\beta + \ln^\beta(k)$ episodes. Here, we quantify sufficient by σ_{ijk}^a in (43) below.

To provide the lemma, we introduce some notation. Let

$$\overline{\theta}_{ij}[s] := \frac{1}{s} \sum_{\ell=1}^{s} Y_{ij}^\ell, \tag{40}$$

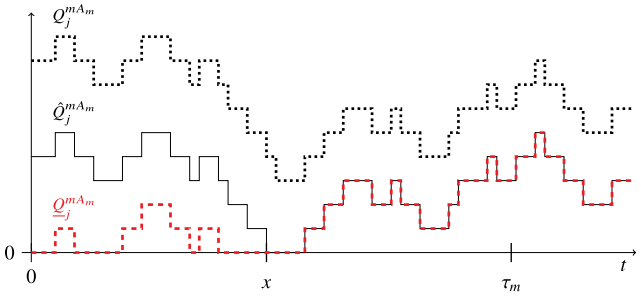
denote the empirical average payoff of type-(ij) departures based on s samples. Here, the $(Y_{ij}^\ell)_{\ell=1}^s$ denote independent and identically distributed random variables with distribution $\text{Ber}(\theta_{ij})$. We let $D_{ij}^{mA_m}$ denote the number of type-(ij) departures within episode $m \in \mathbb{N}_{\geq 1}$ *but after warmup time* τ_m, where A_m denotes the action chosen by Algorithm 1 in row 3. Moreover, for $a \in \mathcal{A}$ let

$$\overline{T}_{ij}^a[s] := \sum_{m=1}^{s} D_{ij}^{ma}, \tag{41}$$

denote the total number of type-(ij) departures (after warmup periods) after completing $s \in \mathbb{N}_{\geq 1}$ episodes where action a was chosen. We let $\overline{U}^a[s, m]$ denote the UCB index of action $a \in \mathcal{A}$ at episode $m \in \mathbb{N}_{\geq 1}$ if action a was chosen in $s \leq m$ episodes,

$$\overline{U}^a[s, m] := \sum_{(ij) \in \mathcal{L}^a} x_{ij}^a \left(\overline{\theta}_{ij}[\overline{T}_{ij}^a[s]] + \sqrt{\frac{\ln(m)}{\overline{T}_{ij}^a[s]}} \right). \tag{42}$$

Figure 8. A Possible Realization of the Queue Length Processes $Q_j^{mA_m}(t)$, $\hat{Q}_j^{mA_m}(t)$, and $\underline{Q}_j^{mA_m}(t)$



Notes. Customer arrivals lead to a unit increase and service completions to a unit decrease (unless the queue is empty). At hitting time x, the processes $\hat{Q}_j^{mA_m}(t)$ and $\underline{Q}_j^{mA_m}(t)$ collide and evolve identically afterward.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

We define for $k \in \mathbb{N}_{\geq 1}$, $a \in \mathcal{A}$, and $(ij) \in \mathcal{L}^a$ the quantities

$$u_k = \ln^{\beta}(k), \qquad \eta_k = \ln^{\frac{\beta+1}{2}}(k), \qquad \sigma^a_{ijk} = x^a_{ij} H_0(u_k - 2\eta_k). \tag{43}$$

We are interested in the event that the number of type-(ij) departures under action a after $\lceil C_\beta + u_k \rceil$ episodes is at least $\lceil \sigma^a_{ijk} \rceil$—that is,

$$E^a_k = \{\overline{T}^a_{ij}[\lceil C_\beta + u_k \rceil] \geq \lceil \sigma^a_{ijk} \rceil, \ \forall (ij) \in \mathcal{L}^a\}. \tag{44}$$

Lemma 8. *Let $\beta \in \mathbb{R}_{>1}$ and $a \in \mathcal{A}$. Then, $\lim_{k\to\infty} k\mathbb{P}((E^a_k)^c) = 0$.*

The proof of Lemma 8 in Online Appendix C.8 relies on Lemma 7. We use that the number of departures is at least as high as the number of departures in a coupled system that starts from the empty state at the start of the episode (the worst-case scenario). We then apply Lemma 7 to show that the worst-case process reaches stationarity with high probability. Next, we use Poisson merging and splitting properties (see, e.g., Çinlar and Agnew 1968) and Burke's Theorem (Cohen 1982, II.2.4 theorem 2.1) to obtain that the *stationary* departure process of the FCFS RR(a) policy in Algorithm 2 is a Poisson process with rate x^a_{ij}. Lastly, we use a Poisson tail bound (see Lemma C.3 in the Online Appendix) to obtain a lower bound on the number of departures of the stationary process.

For the second part of event (A), Lemma 9 shows that the probability of overestimating the true reward is small if the number of obtained samples (departures) is sufficiently large. The proof in Online Appendix C.9 relies on Hoeffding's inequality (Hoeffding 1963, theorem 2).

Lemma 9. *Let $\beta \in \mathbb{R}_{>1}$, $a \in \mathcal{A}$, and $\xi_k := \ln^{-\frac{\beta}{4}}(k)$, then for d^a in (29), $\lim_{k\to\infty} \sum_{s=\lceil C_\beta + u_k \rceil}^{k} \mathbb{P}(\overline{U}^a[s,k] - r^a \geq d^a - \xi_k, E^a_k) = 0$.*

The value σ^a_{ijk} in (43) is constructed precisely such that Lemma 8 and Lemma 9 hold simultaneously. The intuition behind this value is as follows: if we have observed u_k episodes with action a, then for approximately $u_k - \eta_k$ of those episodes, the number of departures can be bounded from below by the number of departures of the stationary process, which is Poisson distributed. From these episodes, we lose approximately η_k episodes where the Poisson tail bound fails (see Lemma C.3 in the Online Appendix). Hence, we collect payoff samples (departures) at a rate of approximately $u_k - 2\eta_k$, as reflected in the definition of σ^a_{ijk}.

4.4.2.2. Bound on Event (B).

For event (B), recall that the true mean of action $a \in \mathcal{A}$ is r^a as defined in (28). By construction, the UCB index in (33) includes a confidence bound (exploration bonus) which implies that the probability of event (B) is small. This is made precise in Lemma 10, which is proven in Online Appendix C.9. We let $\|\cdot\|_2$ denote the L^2-norm.

Lemma 10. *Let $\beta \in \mathbb{R}_{>1}$, $a \in \mathcal{A}$, $k \in \mathbb{N}_{\geq 1}$, and $0 < \xi < r^a$. Then, $\sum_{m=1}^{k} \mathbb{P}(U^a(m) < r^a - \xi) \leq \pi^2 \|\lambda\|_2^2/(12\xi^2)$.*

4.4.3. Step 3. Bounding the Number of Suboptimal Episodes.

We define $S^a(k)$ as the number of episodes where Algorithm 1 chooses action $a \in \mathcal{A}$ in Row 3 up to and including episode number k—that is,

$$S^a(k) := \sum_{m=1}^{k} \mathbb{1}\{A_m = a\}. \tag{45}$$

From Lemma 11, it follows that the expected number of suboptimal episodes grows asymptotically with rate at most $\ln^{\beta}(k)$ with the number of episodes k. The proof of Lemma 11 is given in Online Appendix C.11 and relies on Lemma 8, 9, and 10.

Lemma 11. *Let $\beta \in \mathbb{R}_{>1}$. For any suboptimal action $a \in \mathcal{A} \setminus \{1\}$ we have*

$$\lim_{k\to\infty} \frac{\mathbb{E}(S^a(k))}{\ln^{\beta}(k)} \leq 1. \tag{46}$$

4.4.4. Step 4. Bounding the Regret of Algorithm 1.

We bound the terms in the Regret Decomposition (20) individually. For term I, we use that suboptimal lines are only used in suboptimal episodes by construction. Hence, for $(k\ell) \in O^c(\theta)$, the expected number of type-$(k\ell)$ departures can be upper bounded by the maximal departure rate of server ℓ, which is $\mu_\ell - \varepsilon$ by construction, multiplied by the total number of suboptimal episodes. This, we bound using Lemma 11.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

For term II, we show that the queue length process in consecutive optimal episodes converges sufficiently fast to its stationary behavior. Specifically, we use the bound in Lemma 7: if $A_k = A_{k+1} = 1$ and the queue length processes reaches stationarity within the warmup period of episode k, then the process is stationary at the start of episode $k+1$, because the stationary measure does not change in this case.

Lastly, for term III, we use the fact that x^{A_m} satisfies (10c) to show that the virtual queues of all servers are positive recurrent under the FCFS RR(A_m) policy. In view of (10b), this implies that the queue of any customer type is positive recurrent as well. This means that the expected queue lengths remain finite, and hence, the contribution of term III to the regret vanishes on a logarithmic scale.

4.5. Discussion

4.5.1. Avoiding Carryover Effect Between Episodes. The reallocation in rows 5 and 6 avoids a delay in observations when a new episode starts: if the chosen action in episode k differs from the action that was chosen in the previous episode $k-1$, we reallocate all waiting customers to the virtual queues according to new rates of the action chosen in episode k. After the new reallocation, we order the customers in each virtual queue in order of their arrival times, so that the service order within the virtual queue is FCFS. This reallocation improves the learning efficiency because there is no "carryover" effect between episodes, unlike other learning algorithms in literature (Fu and Modiano 2022, Jia et al. 2024).

4.5.2. Transfer Learning. The rewards of the actions are related by the common unknown payoff parameter θ via (28). Algorithm 1 utilizes transfer learning in the sense that knowledge about the payoff parameter is shared between actions: specifically, the UCB index of a reward is computed using the UCB indices of the lines. This effect also becomes evident in the numerical analysis in Section 5.1.

Our regret bound in Theorem 2 is pessimistic: in analyzing the convergence of the UCB index U^a in (33), we only account for departures that are obtained in episodes where action a was chosen. However, we possibly observe type-(ij) departures as well in episodes where another action $\tilde{a}$ with $x^{\tilde{a}}_{ij} > 0$ was chosen. This insight can be used in future work to improve the bound in Lemma 8 and, in turn, improve the regret bound in Theorem 2. In particular, it might be possible to replace the cardinality of the action set $|\mathcal{A}|$ in (37) by the number of lines L, which is typically smaller. Such analysis methods are often used in the setting of linear bandits (Lattimore and Szepesvári 2020).

4.5.3. Parameter Values. To implement UCB QR, we need to set $\alpha, H_0 \in \mathbb{R}_{\geq 1}$, and $\beta \in \mathbb{R}_{>1}$. These parameters determine the episode length via (36). Smaller values of α, H_0, and β lead to shorter episodes, thereby increasing the learning rate. Indeed, the asymptotic regret bound in (37) is increasing in α and β.

Theorem 2 requires that $\alpha \in \mathbb{R}_{\geq 1}$ satisfies (35) and $H_0 \in \mathbb{R}_{\geq 1}$ satisfies (36). We briefly discuss how to deal with these constraints in practice. Because (35) involves a maximum taken over all actions, it can be tedious to compute when the action set $\mathcal{A}$ is large. However, we can provide an upper bound on (35) to ease evaluation. We show in Online Appendix C.12 that

$$\alpha \geq \max\left\{\frac{7\mu_{\max}}{\varepsilon^2}, 1\right\}, \tag{47}$$

satisfies Constraint (35), where $\mu_{\max} := \max_{j\in\mathcal{J}} \mu_j$. Observe that (47) only relies on the service rates and ε, and not on the action set, and is much simpler than (35).

To set H_0 according to (36), we need to determine $x_{\min} := \min_{a\in\mathcal{A}} \min_{(ij)\in\mathcal{L}^a} x^a_{ij}$. We show in Online Appendix C.12 that in case λ and μ are integer valued, we have $x_{\min} \geq 2^{-(I+J)/2}$. In that case,

$$H_0 = 2^{2+(I+J)/2}, \tag{48}$$

satisfies constraint (36). Because the right side of (48) only depends on the number of customer types I and servers J, this bound is easier to compute than finding the minimum nonzero value x^a_{ij} over all actions $a \in \mathcal{A}$. In case λ and μ are rational, a similar bound can be obtained after scaling the parameters.

4.5.4. Dependency on the Value of ε. We note that the required lower bound on α in (35) can be large when the slack parameter $\varepsilon > 0$ is small. In particular, if $\tilde{\lambda}^a_j = \mu_j - \varepsilon$ for some action a and server j, then the lower bound in (35) is $\Omega(1/\varepsilon^2)$. This is in line with (47). In this case, the regret of Algorithm 1 is $O(\alpha \ln^{2\beta}(t)) = O(\ln^{2\beta}(t)/\varepsilon^2)$ which diverges when $\varepsilon \downarrow 0$.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

In this paper we only consider policies satisfying (8). For fixed ε, Condition (8) excludes some stable policies. Indeed, one can construct a stable policy by replacing ε with $\varepsilon/2$ in (8). Hence, if ε is reduced, we allow for a larger class of stable policies. At the same time, the regret of Algorithm 1 increases as ε decreases. So there always exists a stable policy with an arbitrarily smaller regret when compared with Algorithm 1.

If one aims to relax Condition (8) to the set of all stable policies, then a more suitable benchmark for the regret would be $\mathrm{LP}(\theta,0)$ instead of $\mathrm{LP}(\theta,\varepsilon)$. The regret of Algorithm 1 would then also include a linear term εT because the difference between $\mathrm{LP}(\theta,0)$ and $\mathrm{LP}(\theta,\varepsilon)$ is $\Theta(\varepsilon)$. That is, in that case, $R^\theta(T) \sim \varepsilon T + \ln^{2\beta}(T)/\varepsilon^2$. Optimizing over all possible values of ε yields $R^\theta(T) \geq \tilde{\Omega}(T^{2/3})$, where the notation $\tilde{\Omega}$ omits polylogarithmic factors in T.

There thus exists a trade-off in choosing the value of ε in practice. A smaller ε increases the optimal value of $\mathrm{LP}(\theta,\varepsilon)$, but also increases the regret of Algorithm 1. More precisely, the reward obtained by Algorithm 1 in T time is proportional to $(\mathrm{LP}(\theta,0)-\varepsilon)T + \log^{2\beta}(T)/\varepsilon^2$. The value ε should be chosen carefully to maximize this reward.

We require the Lower Bound (35) in the mixing time analysis of an M/M/1 queue (see Lemma 7). Lower Bound (35) is constructed in such a way to bound a term in the proof of Lemma 7 (see (167) in the Online Appendix). However, this bound is quite loose. More careful analysis of the individual terms may give a lower value than (35). We show in Section 5.3 that Algorithm 1 still performs well in finite time, even when α is chosen smaller than the lower bound in (35).

Another method to improve the scaling $\Omega(1/\varepsilon^2)$ of the regret would be to use another routing rule. An alternative to Algorithm 2 that mixes the process faster may be beneficial. We leave this for future work.

4.5.5. Number of Actions. The number of actions grows exponentially with the number of queues I, servers J, and lines L in the queueing system. Concretely, the number of bases of $\mathrm{LP}(\theta,\varepsilon)$ in (10) is $\binom{L+J}{I+J}$; hence, the number of basic feasible solutions grows exponentially with the system's complexity.

Although the cardinality of the action set $|\mathcal{A}|$ appears in the Regret Upper Bound (37), this does not necessarily pose a problem in practical implementation of the algorithm. With slight modification, we can avoid the need to keep track of all basic feasible solutions $x^a, a \in \mathcal{A}$ of the LP and evaluate their estimators U^a at the start of each episode (Algorithm 1, row 3). Rather, we can solve $\mathrm{LP}(\theta,\varepsilon)$ in (10) directly and take its maximizer as the action for the next episode. Modern numerical solvers can efficiently solve LPs using, for example, the simplex method or interior-point methods (Bertsimas and Tsitsiklis 1997). If the LP solver correctly finds the optimal solution, the performance of the algorithm is not affected because the optimal basic feasible solution of $\mathrm{LP}(\theta,\varepsilon)$ is the same as the action with maximal value.

In case the number of customer types, servers, and lines of the queueing system are so large that the LP cannot be solved sufficiently fast, column generation (Desrosiers and Lübbecke 2005), or a branch-and-bound technique like Bender's decomposition (García-Muñoz et al. 2023), can be used to solve the LP efficiently.

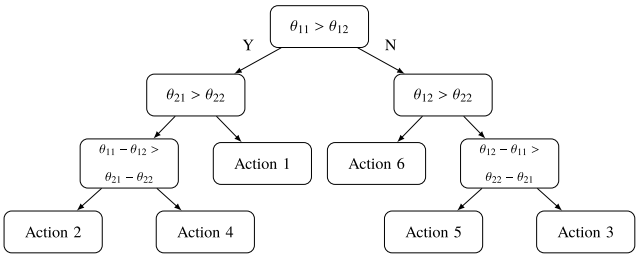
5. Numerical Results

In this section, we analyze the properties of Algorithm 1 (UCB QR) numerically. In Section 5.1, we consider a small skill-based queueing system. For this system, we analyze the regret of the UCB QR algorithm and compare its long-term average payoff rate with several benchmark routing policies. In Section 5.2, we analyze the robustness of the learning policy against changes in the true payoff parameter. In Section 5.3, we consider a larger queueing system and compare the performance of UCB QR against benchmark policies in several scenarios.

We consider four different benchmark policies which are described as follows.

- **Oracle:** This is almost the same as the UCB QR algorithm, but with row 3 replaced by $A_k = \arg\max_{\tilde{a}} r^{\tilde{a}}$. Note that this policy relies on the true payoff parameter θ. This policy represents the optimal upper bound for the long-term reward rate of any stabilizing policy.
- **FCFS ALIS:** If there are nonempty compatible queues at a service completion, assign the customer with maximal waiting time. If there are compatible servers idle upon a customer arrival, assign it to the server with maximal idle time. This policy is widely used in practice because both customers and servers experience a sense of fairness (Adan and Weiss 2014).
- **Greedy:** If there are nonempty compatible queues at a service completion of server j, assign the first-in-line customer from queue $\arg\max_{i\in\mathcal{C}_j} \theta_{ij}$. If there are compatible servers idle upon a type-i customer arrival, assign it to server $\arg\max_{j\in\mathcal{S}_i} \theta_{ij}$. This policy myopically optimizes the instantaneous payoff without considering long-term effects or stability constraints.

Figure 9. Optimality Conditions for Each of the Six Feasible Solutions of LP(θ, ε)



- **Random:** If there are nonempty compatible queues at a service completion, choose one of these queues uniformly at random and assign the first-in-line customer. If there are compatible servers idle at a customer arrival, assign it to one of these servers uniformly at random. This policy serves as a most naive baseline by making completely uninformed decisions.
- **Empirical $\hat{\theta}\mu$:** Lastly, we consider a variation of the classical $c\mu$ rule (Smith 1956), which is known to be optimal for holding cost minimization in a multicustomer single-server setting and has been well-studied in multicustomer multiserver settings (Xia et al. 2022, Long et al. 2024) and even in adaptive learning settings (Krishnasamy et al. 2018). Instead of holding costs, we balance the empirical average payoff $\hat{\theta}_{ij}$ with service speed μ_j. Concretely, if there are nonempty compatible queues at a service completion of server j, assign the first-in-line customer from queue $\arg\max_{i \in \mathcal{C}_j} \hat{\theta}_{ij}\mu_j$. If there are compatible servers idle upon a type-i customer arrival, assign it to server $\arg\max_{j \in \mathcal{S}_i} \hat{\theta}_{ij}\mu_j$.

5.1. Small Example

We consider the queueing system and action set as shown in Figure 5. The optimality of the actions depends on the payoff vector $\theta = \{\theta_{11}, \theta_{12}, \theta_{21}, \theta_{22}\}$ as illustrated in Figure 9.

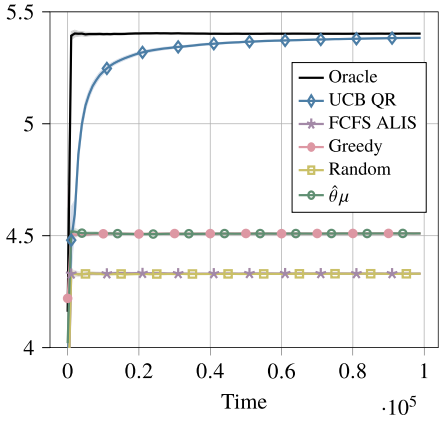
We set the true payoff vector as $\theta = \{0.4, 0.1, 0.3, 0.01\}$. This implies that action 2 is optimal and that the optimal long-term payoff rate is approximately 5.4. Moreover, line 12 is suboptimal because it has zero load in the optimal solution. The suboptimality gap of line 12 (recall (19)) is $\phi_{12} = 0.01$. The suboptimality gaps for each action in (29) are given by

$$d^1 = 1.305,\ d^2 = 0,\ d^3 = 1.755,\ d^4 = 0.055,\ d^5 = 1.84,\ d^6 = 1.405. \tag{49}$$

Recall that the UCB QR algorithm requires configuration parameters α, β, and H_0 which define the episode length according to (36). We set $\alpha = 364$, $\beta = 1.01$, and $H_0 = 10$. These parameters satisfy (35) and (36). All of our numerical results are based on 50 independent replications, and all plots show 95% confidence areas, although these intervals are too small to be visible in Figure 10.

We observe in Figure 10 that, as expected, the payoff rate of the UCB QR algorithm converges to the optimal payoff rate of the Oracle policy. The accumulated regret grows slowly over time, as shown in Figure 11. In the initial period up to approximately time 2,000, the UCB QR algorithm has not yet collected sufficient samples to

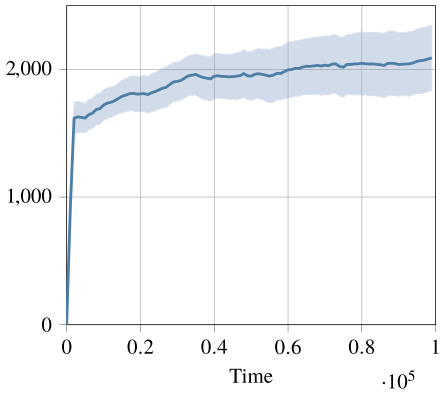
Figure 10. Running Average Payoff Rate over Time for All Policies



Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Figure 11. Regret Accumulation over Time of the UCB QR Algorithm



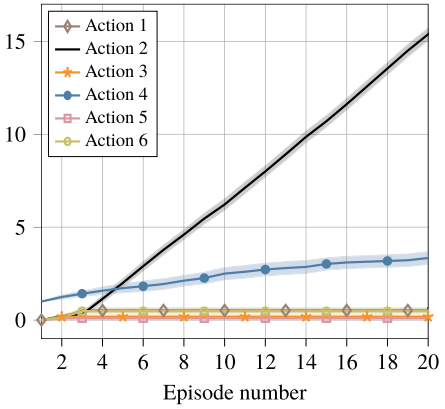
distinguish the actions. Therefore, it chooses actions uniformly at random, which corresponds to the first few episodes in Figure 12. In turn, the regret accumulation is high because actions with large suboptimality gaps are chosen. After the initial period, we observe in Figure 12 that the algorithm primarily struggles to differentiate the optimal action 2 and action 4. This leads to a sharp decrease in regret accumulation, because action 4 has a small suboptimality gap.

Moreover, we observe in Figure 12 that actions $a \in \{1,3,5,6\}$ are hardly ever chosen by the UCB QR algorithm. As opposed to a classical MAB, the rewards of the actions are related by the common unknown payoff parameter θ. Therefore, it suffices to only sample a subset of actions, as long as we observe sufficient payoff samples from all lines. By only using actions 2 and 4, the algorithm infers sufficient information about all four payoff parameters. In particular, in accordance with Algorithm 1, row 12, the indices U_{11}, U_{21}, and U_{22} are updated after episodes with action 2, whereas U_{11}, U_{12}, and U_{21} are updated after episodes with action 4. We note that our regret analysis in Section 4 does not exploit this feature of transfer learning in the construction of the regret upper bound. This remains an interesting direction for future research.

The FCFS ALIS, Greedy, Random, and $\hat{\theta}\mu$ policies maintain a consistent gap to the optimal payoff of the Oracle policy. The value of this gap depends on the true payoff parameter θ. Because the total load in the system is far from critical, the long-term routing rates under the FCFS ALIS policy are close to those of the Random policy. This leads to minor difference in total payoff rate with respect to the Random policy, as shown in Figure 10. In particular, the FCFS ALIS policy is indistinguishable from the Random policy in terms of reward. The Greedy policy obtains a slightly higher reward, making a greedy decision whenever possible. The empirical $\hat{\theta}\mu$ policy performs similarly to the Greedy policy in terms of payoff accumulation. We observe that the empirical estimators concentrate quickly because the $\hat{\theta}\mu$ policy shows stable behavior early on in Figure 10, without fluctuations in the average payoff rate. However, even provided with accurate parameter estimates, this policy does not achieve the optimal payoff rate, similar to the Oracle policy.

To summarize, we have shown for a small queueing system that the payoff rate of the UCB QR algorithm converges to the optimal rate and that it chooses the optimal action most of the time. Thereby, it outperforms our

Figure 12. Cumulative Count of Actions over the First 20 Episodes of the UCB QR Algorithm



Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

benchmark policies that (a) are agnostic to the payoff parameters or (b) rely on the payoff parameters but make suboptimal routing decisions.

5.2. Changing Parameters

Although our theoretical analysis of the UCB QR algorithm in Section 4 is for a static environment, we analyze its robustness against changes in the payoff parameter θ. Learning in nonstatic settings is an important challenge for practical applications where underlying system parameters may evolve over time due to changes in, for example, demand patterns, staffing, or external disruptions. Detection and adaptation to such changing conditions are crucial for maintaining high performance over time. Although a formal analysis of such scenarios is beyond the scope of this paper, we perform an initial exploration of learning in nonstatic settings in this section. Our numerical results suggest that UCB QR can correctly identify changes in the true system parameters.

In this experiment, we use the same payoff vector as in Section 5.1 initially. At one-third of the total runtime, we update the payoff parameter of line 12 to $\theta'_{12} = 0.5$. As a consequence, the optimal action under the new payoff parameter is action 6. We set $\alpha = H_0 = 10$ and $\beta = 1.01$. Our numerical results are based on 100 independent replications.

Observe in Figure 13 that upon the parameter change (at episode number 60, depicted by a vertical line), the UCB QR algorithm at first prefers action 4. This makes sense because action 4 is closest to the initially optimal action 2, while it also includes line 12 which is an optimal line under the updated parameter θ'. From approximately episode number 100 onward, the learning algorithm switches more to the optimal action 6. Hence, we see that the UCB QR algorithm suffers from a switchover period, but eventually changes to the optimal action. The convergence after the parameter change is slower than the initial convergence. We see this effect because the empirical estimators take into account the complete history. Therefore, after the parameter update, learning is hindered by samples obtained before the update. We expect that robustness against changing parameter values can be improved by decreasing the episode length or altering the empirical estimators in (31) by decreasing the weight of observations from the past. This remains open for future work.

5.3. Complex Queueing System

We consider the queueing system illustrated in Figure 14. This system is inspired by the operation of the call center of a real-world telecommunications company. We let $\varepsilon = 0.05$ such that (12) is satisfied. The number of basic feasible solutions of LP(θ, ε) in (10)—that is, the number of actions—is 88. The lower bound in (35) is 165,088. However, because a large α slows down learning, we set $\alpha = 10$. We set $\beta = 1.01$ and $H_0 = 10$. Our numerical results are based on 100 independent replications.

We note that the average runtime of one simulation for the UCB QR algorithm was approximately 90 seconds. This was comparable to the runtimes of the benchmark policies, which ranged from approximately 100 to 150 seconds.

We consider three different scenarios with different levels of difficulty for adaptive learning:

- **Initial parameters:** The first scenario is the system with initial parameters as shown in Figure 14.
- **Minimal payoff discrepancy:** In this scenario, all but one payoff parameters are equal. In particular, $\theta_{ij} = 0.5$ for all $(ij) \in \mathcal{L}$ except for $\theta_{55} = 0.6$. The learning policy must identify this minimal payoff gap.
- **Balanced arrival rates:** In the third scenario, the arrival rates are equal for all customers, namely, $\lambda_i = 42$ for all $i \in \mathcal{I}$. This value is chosen so that the total load of the system is high, namely, approximately 0.98.

Figure 13. Cumulative Count of Actions per Episode of the UCB QR Algorithm When Changing the True Payoff Parameter of Line 21 to $\theta'_{21} = 0.5$ at One-Third of the Total Runtime

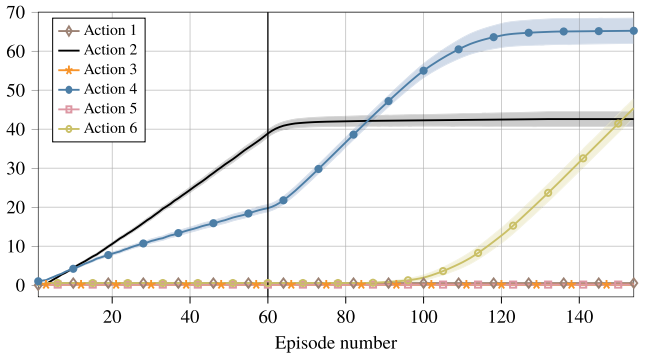
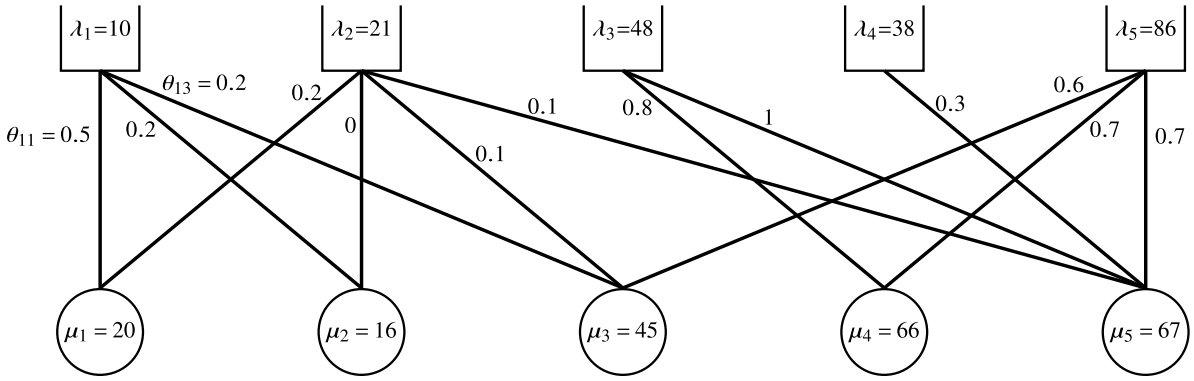


Figure 14. Big Queueing System with $I = 5$ Customer Types and $J = 5$ Servers



Note. The parameter values are shown in the figure.

We compare the UCB QR algorithm with the other benchmark policies as described above. The average number of customers in the system is provided in Table 1, and the payoff rate is illustrated in Figure 15.

Observe in Table 1 that the UCB QR and Oracle policies have a similar average number of customers in the system in all scenarios. The average queue lengths of the FCFS ALIS policy are much smaller than those of Oracle and the UCB QR algorithm. This is to be expected, because this policy inherently aims to decrease the queue length. However, the nonidling nature of this policy leads to a gap in payoff rate with respect to the Oracle policy, as can be seen in Figure 15. For the Greedy, Random, and $\hat{\theta}\mu$ policies, the average number of customers in the system is much higher than for the other policies. Despite the high average queue length, the average payoff of Greedy, Random, and $\hat{\theta}\mu$ in Figure 15 is comparable to that of FCFS ALIS. This is explained as follows: the policies obtain high reward by routing customer types with high average payoffs, whereas other customer types with overall lower average payoffs are not served at all. We find that for the initial parameters and minimal payoff discrepancy scenarios, especially the queue length of customer type 4 is large for these policies, because this customer type is only compatible with server 5, whereas the service capacity of server 5 is for a large part used by customer types 3 and 5. For this reason, the Greedy policy obtains an even lower average reward than the Random policy.

Observe also in Figure 15 that the convergence of the payoff rate of the UCB QR algorithm to the optimal payoff of the Oracle policy is faster than in the small queueing system considered in Section 5.1 (see Figure 10)—that is, the regret is smaller. This can be explained by the choice of α (10 versus 364). A smaller value of α implies that the episodes are shorter. Hence, per time interval, there are more decision moments where the policy can switch between actions, which speeds up learning.

The Greedy and $\hat{\theta}\mu$ policy obtain similar payoffs in the initial and balanced arrival rate scenario. In the minimal payoff discrepancy scenario, the $\hat{\theta}\mu$ policy obtains higher payoffs than the Greedy policy, only just outperforming the Random policy. Concentration of the estimators is sufficiently fast such that we do not see fluctuations in the average payoff rate of the $\hat{\theta}\mu$ policy.

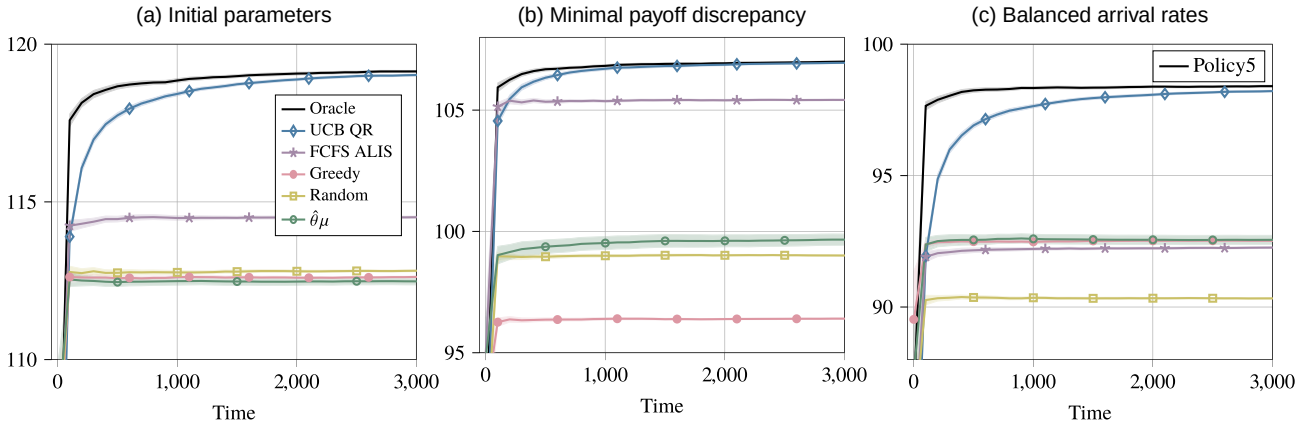
Comparing the different scenarios, we find that in the minimal payoff discrepancy scenario, the UCB QR algorithm chooses suboptimal actions more frequently than in the other scenarios. However, because the

Table 1. Mean ($\mathbb{E}$) and Standard Deviation (σ) of the Number of Customers in the System over the Entire Simulation Time for All Policies and Scenarios

	Scenario					
	Initial parameters		Min. payoff discrepancy		Balanced arrival rates	
Policy	$\mathbb{E}$	σ	$\mathbb{E}$	σ	$\mathbb{E}$	σ
Oracle	329	106	269	80	277	90
UCB QR	315	101	147	31	248	71
FCFS ALIS	9	0.3	12	0.8	12	0.8
Greedy	14,069	142	17,372	148	14,193	207
Random	8,491	138	8,483	130	9,447	130
$\hat{\theta}\mu$	14,253	806	10,552	1,869	12,931	1,841

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Figure 15. Running Average Payoff Rate Over Time for All Policies and Scenarios



Note. The legend is the same for all subplots.

suboptimality gaps of the suboptimal actions are small by construction, the average reward is still close to the optimal, as shown in Figure 15(b). Out of the three scenarios, the convergence is slowest in Figure 15(c). Our belief is that in this case, the algorithm needs approximately the same number of suboptimal episodes to learn the payoff parameters as in the initial scenario, whereas the suboptimality gaps are larger—that is, the cost of exploration is higher.

Lastly, we note that Algorithm 1, although it maintains system stability by construction, is not incentivized to minimize or balance the queue lengths or server loads. Possible extensions of the algorithm where customer waiting times, server loads, or fairness constraints are taken into account are interesting for future research. For example, LP(θ, ε) in (10) can be altered to include either waiting time constraints or a penalty factor proportional to the average queue length in the objective function. This would result in a different action space for the algorithm.

To summarize the findings of this section, we have shown that even in a complex queueing system with 88 different actions, the UCB QR algorithm converges quickly to the oracle reward, while maintaining reasonable queue lengths in different scenarios.

Acknowledgments

The authors thank Thomas van Vuren for his insightful comments and fruitful discussions. This work is part of *Valuable AI*, a research collaboration between the Eindhoven University of Technology and the Koninklijke KPN N.V.

References

Adan I, Weiss G (2012) Exact FCFS matching rates for two infinite multitype sequences. *Oper. Res.* 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.
Agarwal A, Foster DP, Hsu D, Kakade SM, Rakhlin A (2013) Stochastic convex optimization with bandit feedback. *SIAM J. Optim.* 23(1): 213–240.
Agrawal S, Wang Z, Ye Y (2014) A dynamic near-optimal algorithm for online linear programming. *Oper. Res.* 62(4):876–890.
Auer P, Cesa-Bianchi N, Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. *Machine Learn.* 47(2–3):235–256.
Bertsimas D (1995) The achievable region method in the optimal control of queueing systems; formulations, bounds and policies. *Queueing Systems* 21(3–4):337–389.
Bertsimas D, Tsitsiklis JN (1997) *Introduction to Linear Optimization* (Athena Scientific, Belmont, MA).
Besbes O, Zeevi A (2015) On the (surprising) sufficiency of linear models for dynamic pricing with demand learning. *Management Sci.* 61(4): 723–739.
Bimpikis K, Markakis MG (2019) Learning and hierarchies in service systems. *Management Sci.* 65(3):1268–1285.
Burnetas AN, Kanavetas O (2012) Adaptive policies for sequential sampling under incomplete information and a cost constraint. Daras N, ed. *Applications of Mathematics and Informatics in Military Science*, Springer Optimization and its Applications, vol. 71 (Springer, New York), 97–112.
Burnetas AN, Katehakis MN (1996) Optimal adaptive policies for sequential allocation problems. *Adv. Appl. Math.* 17(2):122–142.
Burnetas AN, Katehakis MN (1997) Optimal adaptive policies for Markov Decision Processes. *Math. Oper. Res.* 22(1):222–255.
Burnetas AN, Kanavetas O, Katehakis MN (2017) Asymptotically optimal multi-armed bandit policies under a cost constraint. *Probab. Engrg. Infom. Sci.* 31(3):284–310.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.

Chen J, Dong J, Shi P (2020) A survey on skill-based routing with applications to service operations management. *Queueing Systems* 96(1–2): 53–82.

Choudhury T, Joshi G, Wang W, Shakkottai S (2021) Job dispatching policies for queueing systems with unknown service rates. *MobiHoc'21 Proc. Twenty-Second Internat. Sympos. Theory Algorithmic Foundations Protocol Design Mobile Networks Mobile Comput.* (Association for Computing Machinery, New York), 181–190.

Çinlar E, Agnew R (1968) On the superposition of point processes. *J. Roy. Statist. Soc. Ser. B Methodological* 30:576–581.

Cohen JW (1982) *The Single Server Queue*. 2nd ed. (Elsevier, Amsterdam).

Comte C, Jonckheere M, Sanders J, Senen-Cerda A (2023) Score-aware policy-gradient methods and performance guarantees using local Lyapunov conditions: Applications to product-form stochastic networks and queueing systems. Preprint, submitted December 5, https://arxiv.org/abs/2312.02804v1.

Dacre M, Glazebrook K, Niño-Mora J (1999) The achievable region approach to the optimal control of stochastic systems. *J. Roy. Statist. Soc. Ser. B Statist. Methodology* 61(4):747–791.

Dani V, Hayes TP, Kakade SM (2008) Stochastic linear optimization under bandit feedback. *21st Annual Conf. Learn. Theory COLT 2008* (Omnipress, Madison, WI), 355–366.

Desrosiers J, Lübbecke ME (2005) A primer in column generation. Desaulniers G, Desrosiers J, Solomon MM, eds. *Column Generation* (Springer, Boston), 1–32.

Fu X, Modiano E (2021) Learning-NUM: Network Utility Maximization with unknown utility functions and queueing delay. *Proc. Internat. Sympos. Mobile Ad Hoc Networking Comput. MobiHoc* (Association for Computing Machinery, New York), 21–30.

Fu X, Modiano E (2022) Joint learning and control in stochastic queueing networks with unknown utilities. *Proc. ACM Measurement Anal. Comput. Systems* 6(3):58.

García-Muñoz F, Dávila S, Quezada F (2023) A Benders decomposition approach for solving a two-stage local energy market problem under uncertainty. *Appl. Energy* 329:120226.

Gurvich I, Whitt W (2010) Service-level differentiation in many-server service systems via queue-ratio routing. *Oper. Res.* 58(2):316–328.

Harchol-Balter M (2013) *Performance Modeling and Design of Computer Systems: Queueing Theory in Action* (Cambridge University Press, Cambridge, UK).

Hoeffding W (1963) Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.* 58(301):13–30.

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

Jia H, Shi C, Shen S (2024) Online learning and pricing for service systems with reusable resources. *Oper. Res.* 72(3):1203–1241.

Jiang L, Walrand J (2010) A distributed CSMA algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Trans. Networking* 18(3):960–972.

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

Kim J-h, Vojnovic M (2021) Scheduling servers with stochastic bilinear rewards. Preprint, submitted December 13, https://arxiv.org/abs/2112.06362v1.

Koole GM, Mandelbaum A (2002) Queueing models of call centers: An introduction. *Ann. Oper. Res.* 113(1–4):41–59.

Krishnasamy S, Arapostathis A, Johari R, Shakkottai S (2018) On learning the cμ rule in single and parallel server networks. *2018 56th Annual Allerton Conf. Commun. Control Comput. Allerton 2018* (IEEE, Piscataway, NJ), 153–154.

Lai TL, Robbins H (1985) Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.* 6(1):4–22.

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

Lee D, Vojnovic M (2021) Scheduling jobs with stochastic holding costs. *Adv. Neural Inform. Processing Systems* 23:19375–19384.

Liu B, Xie Q, Modiano E (2019) RL-QN: A reinforcement learning framework for optimal control of queueing systems. *2019 57th Annual Allerton Conf. Commun. Control Comput. Allerton 2019* (IEEE, Piscataway, NJ), 663–670.

Liu X, Li B, Shi P, Ying L (2020) POND: Pessimistic–Optimistic oNline Dispatching. Preprint, submitted October 20, https://arxiv.org/abs/2010.09995v1.

Long Z, Zhang H, Zhang J, Zhang ZG (2024) The generalized c/μ rule for queues with heterogeneous server pools. *Oper. Res.* 72(6): 2488–2506.

Sanders J, Borst SC, Van Leeuwaarden JS (2016) Online network optimization using product-form Markov processes. *IEEE Trans. Automatic Control* 61(7):1838–1853.

Shah V, Gulikers L, Massoulié L, Vojnović M (2020) Adaptive matching for expert systems with uncertain task types. *Oper. Res.* 68(5): 1403–1424.

Shapley LS, Shubik M (1971) The assignment game I: The core. *Internat. J. Game Theory* 1(1):111–130.

Smith WE (1956) Various optimizers for single-stage production. *Naval Res. Logist. Quart.* 3:59–66.

Steiger J, Li B, Lu N (2022) Learning from delayed semi-bandit feedback under strong fairness guarantees. *IEEE INFOCOM 2022 IEEE Conf. Comput. Commun.* (IEEE, Piscataway, NJ), 1379–1388.

Sun X, Zhao J, Chai S (2023) Congestion-aware matching and learning for service platforms. Preprint, submitted March 14, https://doi.org/10.2139/ssrn.5258944.

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

Tan X, Sun B, Leon-Garcia A, Wu Y, Tsang DH (2020) Mechanism design for online resource allocation: A unified approach. *SIGMETRICS'20 Abstracts 2020 SIGMETRICS/Performance Joint Internat. Conf. Measurement Modeling Comput. Systems* (Association for Computing Machinery, New York), 11–12.

Vera A, Banerjee S (2021) The Bayesian prophet: A low-regret framework for online decision making. *Management Sci.* 67(3):1368–1391.

Visschers J, Adan I, Weiss G (2012) A product form solution to a system with multi-type jobs and multi-type servers. *Queueing Systems* 70(3): 269–298.

Wei Y, Xu J, Yu SH (2023) Constant regret primal-dual policy for multi-way dynamic matching. *Performance Evaluation Rev.* 51(1):79–80.

Weiss G (2021) *Scheduling and Control of Queueing Networks* (Cambridge University Press, Cambridge, UK).

Xia L, Zhang ZG, Li QL (2022) A c/μ-rule for job assignment in heterogeneous group-server queues. *Production Oper. Management* 31(3): 1191–1215.

Yang Z, Srikant R, Ying L (2023) Learning while scheduling in multi-server systems with unknown statistics: MaxWeight with Discounted UCB. *Proc. Machine Learn. Res.* 206:4275–4312.

Yu H, Neely MJ, Wei X (2017) Online convex optimization with stochastic constraints. *Adv. Neural Inform. Processing Systems* 30:1429–1439.

Zhong Y, Birge JR, Ward A (2022) Learning the scheduling policy in time-varying multiclass many server queues with abandonment. Preprint, submitted April 21, https://doi.org/10.2139/ssrn.4090021.

Downloaded from informs.org by [216.73.216.11] on 04 June 2026, at 17:34 . For personal use only, all rights reserved.