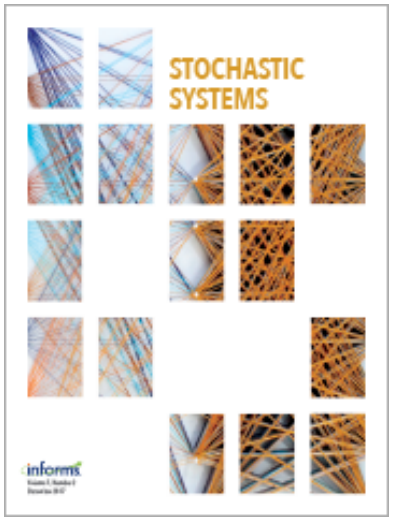


Stochastic Systems

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

Learning-Based Pricing and Matching for Two-Sided Queues

Zixian Yang, Lei Ying

To cite this article:
Zixian Yang, Lei Ying (2026) Learning-Based Pricing and Matching for Two-Sided Queues. Stochastic Systems 16(1):22-43. https://doi.org/10.1287/stsy.2024.0073

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

STOCHASTIC SYSTEMS
Vol. 16, No. 1, March 2026, pp. 22–43
ISSN 1946-5238 (online)

Learning-Based Pricing and Matching for Two-Sided Queues

Zixian Yang,[a,*] **Lei Ying**[a]

[a] Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan 48109
*Corresponding author
Contact: zixian@umich.edu, https://orcid.org/0000-0002-5710-683X (ZY); leiying@umich.edu, https://orcid.org/0000-0001-7955-9445 (LY)

Received: April 29, 2024
Revised: June 9, 2025
Accepted: November 17, 2025
Published Online in Articles in Advance: December 22, 2025

https://doi.org/10.1287/stsy.2024.0073

Abstract. We consider a dynamic system with multiple types of customers and servers. Each type of waiting customer or server joins a separate queue, forming a bipartite graph with customer-side queues and server-side queues. The platform can match the servers and customers and then the matched pairs leave the system. The platform will charge a customer a price according to their type when they arrive and will pay a server a price according to their type. The arrival rate of each queue is determined by the price according to some unknown demand or supply functions. Our goal is to design pricing and matching algorithms to maximize the profit of the platform with unknown demand and supply functions while keeping queue lengths of both customers and servers below a predetermined threshold. The difficulties of the problem include simultaneous learning and decision making and the tradeoff between maximizing profit and minimizing queue length. We use a longest-queue-first matching algorithm and propose a learning-based pricing algorithm, which combines gradient-free stochastic projected gradient ascent with bisection search. We prove that our proposed algorithm yields a sublinear regret $\tilde{O}(T^{5/6})$ and queue-length bound $\tilde{O}(T^{2/3})$, where T is the time horizon. We further establish a tradeoff between the regret bound and the queue-length bound.

Funding: This research was supported in part by the National Science Foundation [Grants 2112471, 2207548, 2228974, and 2240981].

Supplemental Material: The online appendix is available at https://doi.org/10.1287/stsy.2024.0073.

Keywords: queueing • two-sided • learning • pricing • matching

1. Introduction

We study pricing and matching in two-sided queueing systems with multiple types of customers and servers. Take ride sharing (Uber, Lyft, etc.) as an example. A ride order (request) can be viewed as a customer, and an available driver can be viewed as a server. A ride order can have multiple types. For example, we can view different numbers of passengers in the order as different customer types. We can also view different pick-up locations of the order as different customer types. A driver can also have multiple types. For example, we can view different vehicle capacities as different server types. We can also view different locations of the drivers as different server types. Each type of waiting ride order or driver joins a separate queue, forming a bipartite graph with customer-side (passenger-side) queues and server-side (driver-side) queues. We consider a discrete-time system. At each time slot, the ride-sharing platform will determine a price for each customer or server type. If a passenger accepts the price, they will be charged and join the queue. Similarly, if a driver accepts the price, they will be paid and join the queue. Therefore, the arrival rate of each type of ride order (or driver) is governed by the corresponding price according to some unknown demand (or supply) function. After that, the platform will match the drivers and ride orders in the queues if their types are compatible. For example, if the customer type is the number of passengers in the ride order and the server type is the capacity of the vehicle, then their types are compatible if the capacity of the vehicle is no less than the number of passengers in the ride order. The matched pairs then leave the system. The same process repeats in the next time slot. The profit of the platform is equal to the income from ride orders minus the cost paid to drivers. Assuming the demand and supply functions are unknown, the goal is to design pricing and matching algorithms to maximize the total profit of the platform over a finite time horizon while keeping the queue lengths of both customers and servers below a predetermined threshold.

Similar pricing and matching problems in two-sided queues have been studied in the literature (Caldentey et al. 2009; Adan and Weiss 2012; Gurvich and Ward 2015; Nguyen and Stolyar 2018; Chen and Hu 2020; Varma et al. 2020, 2023; Hu and Zhou 2022; Vaze and Nair 2022; Aveklouris et al. 2023, 2024). Caldentey et al. (2009), Adan and Weiss (2012), Hu and Zhou (2022), and Aveklouris et al. (2023, 2024) study matching problems in two-sided queueing systems with multiple types of demand and multiple types of supply. Nguyen and Stolyar (2018) study a two-sided queueing system with on-demand servers with the goal of designing adaptive server invitation scheme to minimize waiting times. Gurvich and Ward (2015) study a matching problem for multisided queues. However, these works do not consider pricing. Varma et al. (2020) and Chen and Hu (2020) study pricing and matching problems in two-sided queues, but they consider scenarios with strategic demand side and/or supply side, which is different from our setting, where customers and servers are not strategic, and their arrival rates are based on demand and supply functions. Vaze and Nair (2022) consider a two-sided queueing system where servers arrive at a fixed rate, and the arrival rate of customers is controlled by the price. They consider one-sided pricing and single-type customers and servers, whereas we consider two-sided pricing and matching and multiple customer and server types. The most relevant work in the literature is Varma et al. (2023), whose model is similar to ours. The differences between Varma et al. (2023) and ours include the following:

- The key difference is that they assume that the platform knows the demand and supply functions, whereas in our setting, the demand and supply functions are unknown.
- The objective in Varma et al. (2023) is the asymptotic long run expected profit of the platform subtracting a penalty that is linear in the queue length, whereas our objective is to maximize the total profit of the platform over a finite time horizon while keeping the queue lengths below a predetermined threshold during the entire time horizon.
- Varma et al. (2023) consider a large-scale regime in which they scale the arrival rates and study the profit loss in steady state. We focus on finite-time analysis without scaling the arrival rates.

There are some other related works about learning-based dynamic pricing and matching. Besbes and Zeevi (2015) study dynamic pricing with demand learning but they do not consider queueing. Chen and Shi (2024) study a dual-sourcing inventory system with demand learning, which is different from our two-sided queueing system. Tassiulas and Ephremides (1990) and Srikant and Ying (2014) study MaxWeight algorithms for scheduling in one-sided multiserver queueing systems. Our matching algorithm can also be seen as a MaxWeight algorithm but applied to a two-sided matching setting (Varma et al. 2023). Flaxman et al. (2004), Agarwal et al. (2010), Duchi et al. (2015), Shamir (2017), and Lattimore (2024) study zero-order gradient descent algorithms, some ideas from which are incorporated into parts of our pricing algorithm. There are other online learning algorithms in queueing systems (Chen et al. 2023, 2024; Yang et al. 2023), but they consider only one-sided queueing system.

The key challenge of our setting is that the demand and supply functions are unknown. Learning the demand and supply functions explicitly will not be adaptive if the demand or supply functions change over time. Therefore, we propose to learn and make pricing and matching decisions simultaneously. Although several candidate approaches exist, they are not suitable for our setting. One idea is to formulate the problem into a Markov decision process (MDP) and then use reinforcement learning (RL) to solve the MDP approximately. However, the state space of this MDP increases exponentially with the number of queues (i.e., customer and server types), so it is difficult to solve due to the curse of dimensionality. Another approach is to use bandit algorithms from Lipschitz bandits (Slivkins 2019). For example, one may discretize the price space and apply upper confidence bound (UCB)-based algorithms. However, such approaches fail to balance the matching rates and arrival rates on both sides, making it difficult to control the queue lengths. These algorithms also have a high computational complexity because the number of arms is large. Approaches from bandit convex optimization (Lattimore 2024) aim to solve convex optimization problems with bandit feedback. However, the control variables in our setting are prices and the profit function is not concave in terms of prices. The constraint set representing a set of balanced arrival rates is also not convex in terms of prices.

To deal with this challenge, we propose to use a longest-queue-first matching algorithm and a learning-based pricing algorithm, which combines gradient-free (zero-order) stochastic projected gradient ascent (Agarwal et al. 2010) with bisection search. We also borrow the idea of fluid pricing policy from Varma et al. (2023) to control the queue length by rejecting arrivals according to a predetermined queue-length threshold. In this paper, we prove that our proposed algorithm yields a sublinear regret $\tilde{O}(T^{5/6})$ when compared with a fluid-based baseline and guarantees an anytime queue-length bound $\tilde{O}(T^{2/3})$, where T is the time horizon. Furthermore, we establish a tradeoff between the regret bound and the queue-length bound: For any $\gamma \in (0, 2/3]$, if we keep the queue lengths of both customers and servers below $\tilde{\Theta}(T^{\gamma})$, then we have a regret upper bound $\tilde{O}(T^{1-\gamma/4})$. We prove a hardness result: The regret is at least $\Omega(T^{1-\gamma})$ to maintain the queue length below T^{γ} for a single-link system with demand and supply functions from a specific class, even when the functions are known.

2. Model

We consider a discrete-time system with two-sided queues: a customer side and a server side. There are multiple queues on each side, representing different types of customers and different types of servers. We can also view customers as demand and servers as supply. We model the system by a bipartite graph $G(\mathcal{I} \cup \mathcal{J}, \mathcal{E})$, where $\mathcal{I} = \{1, 2, \ldots, I\}$ is the set of customer types and $\mathcal{J} = \{1, 2, \ldots, J\}$ is the set of server types with $|\mathcal{I}| = I$ and $|\mathcal{J}| = J$. $\mathcal{E}$ is the set of all compatible links, which means that a type i customer can be served by a type j server if and only if $(i, j) \in \mathcal{E}$. Figure 1 is an example with three types of customers and two types of servers ($I = 3, J = 2$). At each time slot, there are arrivals of customers and servers. At time slot t, let $A_{\mathrm{c},i}(t)$ and $A_{\mathrm{s},j}(t)$ denote the number of arrivals of type i customers and type j servers, respectively. Assume that $A_{\mathrm{c},i}(t)$ and $A_{\mathrm{s},j}(t)$ follow independent Bernoulli distributions. Let $\lambda_i(t) := \mathbb{E}[A_{\mathrm{c},i}(t)]$ and $\mu_j(t) := \mathbb{E}[A_{\mathrm{s},j}(t)]$. At each time slot, after the arrival of customers and servers, the platform will match the customers and servers in the queues through compatible links. Once a customer is matched with a server, that is, the customer is served by the server, they will leave the system immediately. Let $Q_{\mathrm{c},i}(t)$ and $Q_{\mathrm{s},j}(t)$ denote the queue length of type i customers and that of type j servers, respectively, before arrivals and matching at time slot t. Assume the queues are all empty at $t = 1$.

There is also a price attached to each type of customer or server. Let $p_{\mathrm{c},i}(t)$ denote the price of the type i customer and $p_{\mathrm{s},j}(t)$ denote the price of the type j server at time slot t. At time slot t, when a customer of type i arrives, they will be charged $p_{\mathrm{c},i}(t)$ units of money by the platform. When a server of type j arrives, they will be paid $p_{\mathrm{s},j}(t)$ units of money by the platform. Note that the prices determine the arrival rates. Increasing the price of type i customers will reduce arrival rate $\lambda_i(t)$; increasing the price of type j servers will increase arrival rate $\mu_j(t)$. For a type i customer, the demand function is denoted by $F_i : [0,1] \to [p_{\mathrm{c},i,\min}, p_{\mathrm{c},i,\max}]$, where the input is an arrival rate $\lambda_i \in [0,1]$, the output is the price $p_{\mathrm{c},i} \in [p_{\mathrm{c},i,\min}, p_{\mathrm{c},i,\max}]$ corresponding to this arrival rate, and $p_{\mathrm{c},i,\min}$ and $p_{\mathrm{c},i,\max}$ are the minimum and maximum prices for type i customer, respectively. Similarly, for type j server, the supply function is denoted by $G_j : [0,1] \to [p_{\mathrm{s},j,\min}, p_{\mathrm{s},j,\max}]$, where the input is an arrival rate $\mu_j \in [0,1]$, the output is the price $p_{\mathrm{s},j} \in [p_{\mathrm{s},j,\min}, p_{\mathrm{s},j,\max}]$ corresponding to this arrival rate, and $p_{\mathrm{s},j,\min}$ and $p_{\mathrm{s},j,\max}$ are the minimum and maximum prices for a type j server, respectively. Then we have $p_{\mathrm{c},i}(t) = F_i(\lambda_i(t))$ and $p_{\mathrm{s},j}(t) = G_j(\mu_j(t))$. We make the following assumption on the demand and supply functions F_i and G_j.

Assumption 1. *For any customer type i, F_i is strictly decreasing, bijective, and L_{F_i}-Lipschitz. Let F_i^{-1} denote the inverse function of F_i. Assume that F_i^{-1} is $L_{F_i^{-1}}$-Lipschitz. For any server type j, G_j is strictly increasing, bijective, and L_{G_j}-Lipschitz. Let G_j^{-1} denote the inverse function of G_j. Assume that G_j^{-1} is $L_{G_j^{-1}}$-Lipschitz.*

From Assumption 1, we know $F_i(1) = p_{\mathrm{c},i,\min}, F_i(0) = p_{\mathrm{c},i,\max}$ for any customer type i and $G_j(0) = p_{\mathrm{s},j,\min}, G_j(1) = p_{\mathrm{s},j,\max}$ for any server type j.

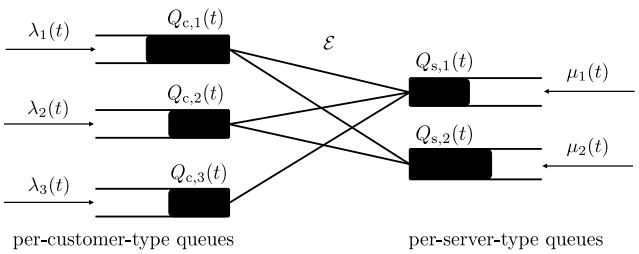
The sequence of events occurring in each time slot is shown in Figure 2. At the beginning of time slot t, we first observe the lengths of all queues $Q_{\mathrm{c},i}(t), Q_{\mathrm{s},j}(t)$ for all i, j. Then the platform runs a pricing algorithm to decide the prices $p_{\mathrm{c},i}(t), p_{\mathrm{s},j}(t)$, $\forall i, j$. The arrival rates $\lambda_i(t), \mu_j(t)$, $\forall i, j$ will be determined by these prices, and then the actual arrivals $A_{\mathrm{c},i}(t), A_{\mathrm{s},j}(t)$, $\forall i, j$ occur. Next, the platform runs a matching algorithm and then those matched customers and servers leave the system.

At time slot t, the platform makes a profit of

$$\sum_i A_{\mathrm{c},i}(t) p_{\mathrm{c},i}(t) - \sum_j A_{\mathrm{s},j}(t) p_{\mathrm{s},j}(t) = \sum_i A_{\mathrm{c},i}(t) F_i(\lambda_i(t)) - \sum_j A_{\mathrm{s},j}(t) G_j(\mu_j(t)).$$

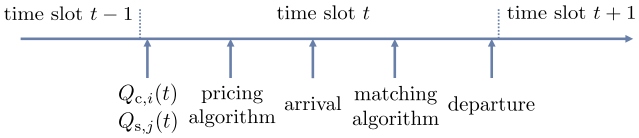
Our goal is to design an online pricing and matching algorithms to maximize the profit of the platform over T time slots without knowing the demand and supply functions while keeping queue lengths of both customers

Figure 1. Model



Note. An example with three types of customers and two types of servers.

Figure 2. Timeline in Each Time Slot



and servers below a predetermined threshold, that is, to maximize

$$\sum_{t=1}^{T}\left[\sum_i A_{c,i}(t)F_i(\lambda_i(t)) - \sum_j A_{s,j}(t)G_j(\mu_j(t))\right], \tag{1}$$

while keeping

$$\max_{t=1,\dots,T}\max\{\max_i Q_{c,i}(t),\ \max_j Q_{s,j}(t)\}$$

below a predetermined threshold. To quantify the performance of the online algorithm, we compare it with the following optimal solution to a fluid-based optimization problem (Varma et al. 2023):

$$\max_{\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{x}} \quad \sum_i \lambda_i F_i(\lambda_i) - \sum_j \mu_j G_j(\mu_j) \tag{2}$$

$$\text{s.t.} \quad \lambda_i = \sum_{j:(i,j)\in\mathcal{E}} x_{i,j}, \quad \text{for all } i\in\mathcal{I}, \tag{3}$$

$$\mu_j = \sum_{i:(i,j)\in\mathcal{E}} x_{i,j}, \quad \text{for all } j\in\mathcal{J}, \tag{4}$$

$$x_{i,j}\ge 0, \quad \text{for all } (i,j)\in\mathcal{E}, \tag{5}$$

$$\lambda_i,\mu_j\in[0,1], \quad \text{for all } i\in\mathcal{I},\ j\in\mathcal{J}, \tag{6}$$

where $\boldsymbol{x}:=(x_{i,j})_{(i,j)\in\mathcal{E}}$, $\boldsymbol{\lambda}:=(\lambda_i)_{i\in\mathcal{I}}$, and $\boldsymbol{\mu}:=(\mu_j)_{j\in\mathcal{J}}$. Similar to Varma et al. (2023), we make the following assumption.

Assumption 2. *For any customer type i, the revenue function $r_i(\lambda_i):=\lambda_i F_i(\lambda_i)$ is concave. For any server type j, the cost function $c_j(\mu_j):=\mu_j G_j(\mu_j)$ is convex.*

From Assumption 2, Objective (2) is concave. Note that the constraints are all affine. Hence, the baseline optimization problem (2)–(6) is concave, and there exists a solution that achieves the optimal value, denoted as $(\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*,\boldsymbol{x}^*)$.

Let $\lambda_i(t)$ and $\mu_j(t)$ be the arrival rates at time t under any policy. Then from (1), the expected profit under the policy is

$$\begin{aligned}&\sum_{t=1}^{T}\mathbb{E}\left[\sum_i A_{c,i}(t)F_i(\lambda_i(t)) - \sum_j A_{s,j}(t)G_j(\mu_j(t))\right]\\ &=\sum_{t=1}^{T}\mathbb{E}\left[\sum_i \lambda_i(t)F_i(\lambda_i(t)) - \sum_j \mu_j(t)G_j(\mu_j(t))\right].\end{aligned}$$

Let f^* denote the optimal value of the fluid optimization problem (2)–(6). For the profit maximization, we consider the following expected regret with respect to the fluid baseline:

$$\begin{aligned}\mathbb{E}[R(T)] &:= Tf^* - \sum_{t=1}^{T}\mathbb{E}\left[\sum_i A_{c,i}(t)F_i(\lambda_i(t)) - \sum_j A_{s,j}(t)G_j(\mu_j(t))\right]\\ &=\sum_{t=1}^{T}\left(\left(\sum_i \lambda_i^*F_i(\lambda_i^*) - \sum_j \mu_j^*G_j(\mu_j^*)\right) - \mathbb{E}\left[\sum_i \lambda_i(t)F_i(\lambda_i(t)) - \sum_j \mu_j(t)G_j(\mu_j(t))\right]\right),\end{aligned} \tag{7}$$

where $(\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*)$ is an optimal solution to the fluid optimization problem. The following lemma establishes the connection between the optimal expected profit and f^* over T time slots.

Lemma 1. *Let Assumption 1 and Assumption 2 hold. Then under any policy, we have*

$$\sum_{t=1}^{T}\mathbb{E}\left[\sum_i \lambda_i(t)F_i(\lambda_i(t)) - \sum_j \mu_j(t)G_j(\mu_j(t))\right] - Tf^* \le \sum_i (L_{F_i} + p_{\mathrm{c},i,\max})\mathbb{E}[Q_{\mathrm{c},i}(T+1)] + \sum_j (L_{G_j} + p_{\mathrm{s},j,\max})\mathbb{E}[Q_{\mathrm{s},j}(T+1)].$$

The proof of this lemma can be found in Appendix A. Lemma 1 means that the fluid baseline approximates the optimal profit, with an error at most proportional to the queue length at time T. This justifies the use of the regret definition (7). Specifically, let $\mathbb{E}[R^{\mathrm{opt}}(T)]$ denote the expected regret with respect to the policy that maximizes the expected profit. Then Lemma 1 implies that

$$\mathbb{E}[R^{\mathrm{opt}}(T)] - \mathbb{E}[R(T)] \le \sum_i (L_{F_i} + p_{\mathrm{c},i,\max})\mathbb{E}[Q_{\mathrm{c},i}(T+1)] + \sum_j (L_{G_j} + p_{\mathrm{s},j,\max})\mathbb{E}[Q_{\mathrm{s},j}(T+1)].$$

As long as the queue lengths $\mathbb{E}[Q_{\mathrm{c},i}(T+1)]$ and $\mathbb{E}[Q_{\mathrm{s},j}(T+1)]$ are orderwise smaller than $\mathbb{E}[R(T)]$, the regret defined in (7) approximates the "true" regret (with respect to the optimal expected profit). We consider policies that make the queue mean rate stable (Neely 2022); that is, under the policy, for all i, j,

$$\lim_{T\to\infty}\frac{1}{T}\mathbb{E}[Q_{\mathrm{c},i}(T)] = 0, \quad \lim_{T\to\infty}\frac{1}{T}\mathbb{E}[Q_{\mathrm{s},j}(T)] = 0.$$

Under any mean rate stable policy, using Lemma 1 and taking limit $T \to +\infty$, we obtain

$$\limsup_{T\to+\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\sum_i \lambda_i(t)F_i(\lambda_i(t)) - \sum_j \mu_j(t)G_j(\mu_j(t))\right] \le f^*,$$

which means that the optimal value of the fluid optimization problem is an upper bound on the asymptotic time-averaged expected profit under any mean rate stable policy. This further justifies the use of the fluid optimal value as a baseline to define regret.

We make another assumption on the problem.

Assumption 3. *There exists a known positive number $a_{\min} \in (0,1)$ such that there exists an optimal solution $\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*$ to the fluid optimization problem that satisfies $\lambda_i^* \ge a_{\min}$ and $\mu_j^* \ge a_{\min}$ for all i and j.*

Assumption 3 means that we need the optimal arrival rates to be strictly greater than zero. This assumption is not restrictive, because any queue with an optimal arrival rate of zero, indicating it is redundant, can be easily identified in practice and removed from the problem formulation. Note that $a_{\min} > 0$ can be any lower bound on the optimal arrival rates.

Assume that the functions F_i and G_j are *unknown*. The platform cannot directly control the arrival rates, but they can control the prices of the customers and the servers and the matching algorithm. The platform can also observe the number of arrivals and the queue lengths of all the queues. At each time slot, the platform needs to design a price for each queue (customer or server) and to make a decision on how the customers and servers are matched. The objective is to minimize the regret $\mathbb{E}[R(T)]$ while keeping the maximum queue length $\max_{t=1,\dots,T}\max\{\max_i Q_{\mathrm{c},i}(t), \max_j Q_{\mathrm{s},j}(t)\}$ below a predetermined threshold. This anytime queue length constraint is necessary in certain practical scenarios—for instance, when the system has a finite buffer size for each queue. Moreover, customers' experience is often affected by the anytime queue length because they typically expect a consistent level of service.

2.1. Notation

We use bold symbols to represent vectors in $\mathbb{R}^{|\mathcal{E}|}$ or $\mathbb{R}^I$ or $\mathbb{R}^J$, such as $\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}$. We use subscript i or j to indicate an element of a vector, for example, λ_i is the ith element of $\boldsymbol{\lambda}$. Note that we view $\boldsymbol{x} = (x_{i,j})_{(i,j)\in\mathcal{E}}$ as a vector in $\mathbb{R}^{|\mathcal{E}|}$ rather than a matrix, and the order of elements in $\boldsymbol{x}$ can be arbitrary as long as the order is fixed. We use subscript c to indicate the customer side and s for the server side. $g(T) = O(h(T))$ means that there exist positive constants C and T_0 such that $g(T) \le Ch(T)$ for all $T \ge T_0$; $g(T) = \Theta(h(T))$ means that there exist positive constants C_1, C_2 and T_0 such that $C_1 h(T) \le g(T) \le C_2 h(T)$ for all $T \ge T_0$; and $g(T) = \Omega(h(T))$ means that there exist positive constants C and T_0 such that $g(T) \ge Ch(T)$ for all $T \ge T_0$. We omit the base of logarithmic functions in $O(\cdot)$, $\Theta(\cdot)$, and $\Omega(\cdot)$, for example, $\Theta(\log(T))$, because the base does not affect the order. We use $\log^2(\cdot)$ to denote $(\log(\cdot))^2$. We use $\tilde{O}(\cdot)$, $\tilde{\Theta}(\cdot)$, and $\tilde{\Omega}(\cdot)$ to suppress polylog(T) factors.

3. Summary of Our Main Results

In this section, we provide an overview of the main results in this paper. We use a longest-queue-first matching algorithm, which is a discrete-time version of the matching algorithm in Varma et al. (2023). We propose a learning-based pricing algorithm, which combines gradient-free (zero-order) stochastic projected gradient ascent (Agarwal et al. 2010) with bisection search, illustrated in Section 4 in detail. The proposed pricing algorithm learns from samples and make pricing decisions simultaneously. Under the proposed matching and pricing algorithms, we establish the following profit-regret bound and queue-length bound:

$$\mathbb{E}[R(T)] = \tilde{O}\left(T^{\frac{5}{6}}\right), \qquad \max_{t=1,\dots,T} \max\{\max_i Q_{c,i}(t), \max_j Q_{s,j}(t)\} = \tilde{O}\left(T^{\frac{2}{3}}\right).$$

By changing the parameters of the proposed pricing algorithm, we can achieve the following tradeoff between regret and queue length: For any $\gamma \in [0, 2/3]$, there exists a set of parameters such that the algorithm can achieve:

$$\mathbb{E}[R(T)] = \tilde{O}\left(T^{1-\frac{\gamma}{4}}\right), \qquad \max_{t=1,\dots,T} \max\{\max_i Q_{c,i}(t), \max_j Q_{s,j}(t)\} = \tilde{O}(T^{\gamma}).$$

The above results are shown in Figure 3. As the allowable queue length increases, the regret bound that we can achieve becomes better. However, if we allow the queue length to increase over $\tilde{\Theta}(T^{2/3})$, the regret bound cannot be further improved and remains $\tilde{O}(T^{5/6})$.

3.1. Hardness Result

We show in the following lemma that in a single-link system, any policy that keeps the anytime queue length bounded by T^{γ} must incur a regret of at least $\Omega(T^{1-\gamma})$.

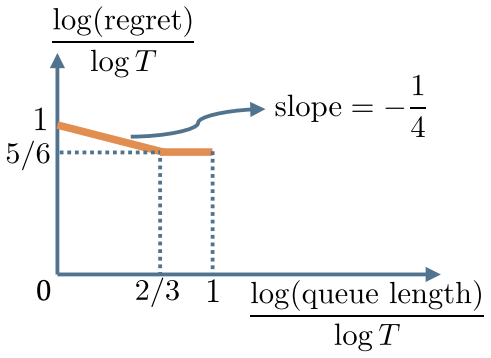
Lemma 2. *Consider a single-link system with one customer queue and one server queue with demand function $F(\cdot)$ and supply function $G(\cdot)$. Let $Q_c(t)$ and $Q_s(t)$ denote the customer-side queue length and server-side queue length, respectively. Consider the matching policy that matches all pairs whenever possible because there is no incentive to delay any match in a single-link system. There exists a class of problem instances that satisfies Assumption 1, Assumption 2, and Assumption 3, such that for any pricing policy satisfying $\max_{t=1,\dots,T}\max\{Q_c(t), Q_s(t)\} \le T^{\gamma}$, the expected regret satisfies $\mathbb{E}[R(T)] = \Omega(T^{1-\gamma})$ for any $\gamma \in [0, 1/2)$.*

The definition of the class of problem instances and the proof of Lemma 2 can be found in Appendix B. Our upper bound results under the proposed algorithm also hold for this problem class. Note that when the queue length threshold is a constant (i.e., $\gamma = 0$), the regret under our algorithm becomes linear, which is unavoidable by Lemma 2.

4. Algorithm

In this section, we present the proposed matching and pricing algorithms in detail. The matching algorithm is to match the customers on the demand-side queues with the servers on the supply-side queues, and the pricing algorithm is to learn and design the prices without knowing the demand and supply functions.

Figure 3. Tradeoff Between Regret and Queue Length



4.1. Matching Algorithm

We use a longest-queue-first matching algorithm as shown in Algorithm 1 and Figure 4, where we present the matching algorithm for one time slot, and it is the same for all time slots. In the algorithm, for each time slot, we iterate over each queue on both the customer and server sides. For each queue, we check whether there is a new arrival in the queue. If there is a new arrival (a customer or a server), the algorithm will match the arrival with one server (or customer) in the longest compatible queue on the other side, as shown in Figure 4. After matching, we decrease the queue length by one for both matched queues. Then we move on to the next queue and repeat the process.

Algorithm 1 (Matching Algorithm)

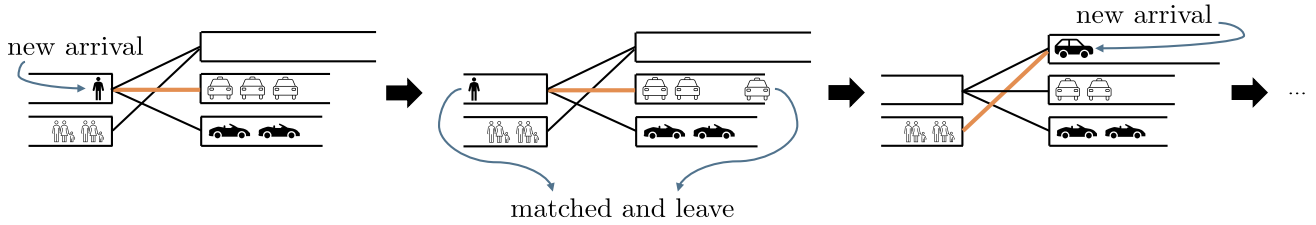
At each time slot t, without loss of generality, assume that the arrivals come in order $A_{c,1}(t), A_{c,2}(t), \dots, A_{c,I}(t)$, $A_{s,1}(t), A_{s,2}(t), \dots, A_{s,J}(t)$.
Initialize: $\tilde{Q}_{c,i} \leftarrow Q_{c,i}(t)$ for all i and $\tilde{Q}_{s,j} \leftarrow Q_{s,j}(t)$ for all j.
for $i = 1$ to I **do**
  Observe $A_{c,i}(t)$ and update $\tilde{Q}_{c,i} \leftarrow \tilde{Q}_{c,i} + A_{c,i}(t)$;
  Let $j_i^*(t) \in \arg\max_{j:(i,j)\in\mathcal{E}} \tilde{Q}_{s,j}$ with ties broken arbitrarily;
  if $A_{c,i}(t) = 1$ and $\sum_{j:(i,j)\in\mathcal{E}} \tilde{Q}_{s,j} > 0$ **then**
    Match the arrival $A_{c,i}(t)$ with a server in queue $j_i^*(t)$;
    Then update $\tilde{Q}_{c,i} \leftarrow \tilde{Q}_{c,i} - 1$ and $\tilde{Q}_{s,j_i^*(t)} \leftarrow \tilde{Q}_{s,j_i^*(t)} - 1$;
for $j = 1$ to J **do**
  Observe $A_{s,j}(t)$ and update $\tilde{Q}_{s,j} \leftarrow \tilde{Q}_{s,j} + A_{s,j}(t)$;
  Let $i_j^*(t) = \arg\max_{i:(i,j)\in\mathcal{E}} \tilde{Q}_{c,i}$ with ties broken arbitrarily;
  if $A_{s,j}(t) = 1$ and $\sum_{i:(i,j)\in\mathcal{E}} \tilde{Q}_{c,i} > 0$ **then**
    Match the arrival $A_{s,j}(t)$ with a customer in queue $i_j^*(t)$;
    Then update $\tilde{Q}_{s,j} \leftarrow \tilde{Q}_{s,j} - 1$ and $\tilde{Q}_{c,i_j^*(t)} \leftarrow \tilde{Q}_{c,i_j^*(t)} - 1$;

4.2. Pricing Algorithm

Because the functions F_i, G_j are unknown and the arrival rates cannot be directly controlled, we cannot simply solve the optimization problem (2)–(6) and obtain the prices. In this section, we propose to combine the techniques of gradient-free (zero-order) stochastic projected gradient descent (Agarwal et al. 2010) and bisection search to solve the problem.

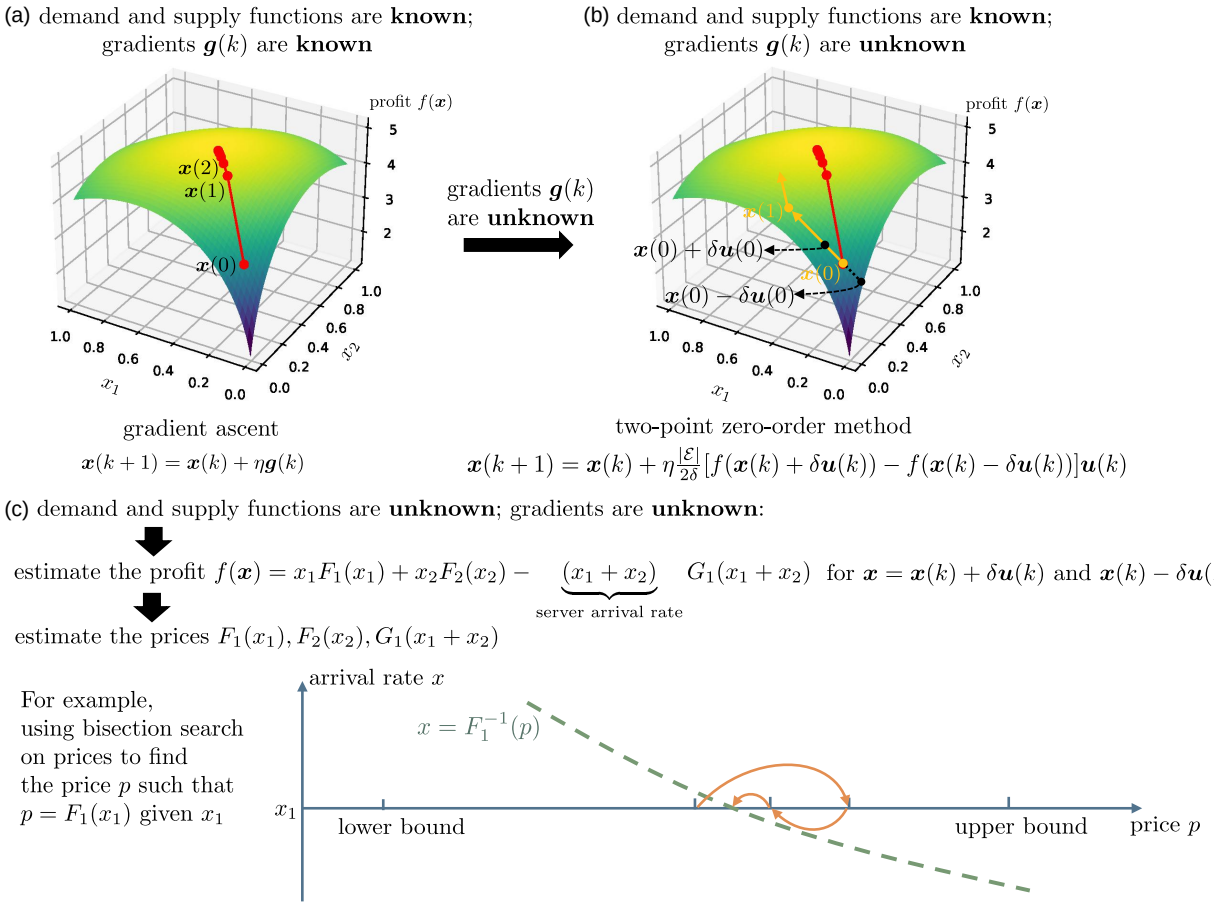
Before presenting the details of our pricing algorithm, we begin by providing a simple example to understand the idea of our algorithm. We consider two customer queues, which are connected to one server queue. Let x_1, x_2 denote the arrival rates of customer queue 1 and customer queue 2, respectively. Then the arrival rate of the server queue should be equal to $x_1 + x_2$ due to the balance constraints (3) and (4). Let $\boldsymbol{x}(k) := (x_1(k), x_2(k))$, where k denote the iteration. Suppose the demand and supply functions are known and the gradients $\boldsymbol{g}(k)$ are known; that is, we have first-order access to the profit function f. In this case, we can use the gradient ascent algorithm to solve the problem, as shown in Figure 5(a), where η is the step size. Suppose the demand and supply functions are known but the gradients are unknown; that is, we only have zero-order access to the profit function f. Then we can use the two-point zero-order method (Agarwal et al. 2010) to solve the problem, as shown in Figure 5(b). In the two point method, in each iteration k, we first sample a uniformly random direction $\boldsymbol{u}(k)$ ($\boldsymbol{u}(k)$ is a unit vector) and then calculate the profits using two points, $\boldsymbol{x}(k) + \delta\boldsymbol{u}(k)$ and $\boldsymbol{x}(k) - \delta\boldsymbol{u}(k)$, where δ is small number. Then we can estimate the gradient and update $\boldsymbol{x}(k)$ using the formula in Figure 5(b). However, if the demand and

Figure 4. Example of Longest-Queue-First Matching Algorithm



Notes. At each time slot, for each queue, if there is a new arrival, it will be matched with one server (or customer) in the longest compatible queue on the other side. Then the matched pairs leave the system and we move on to check the next queue and repeat the same process.

Figure 5. Pricing Algorithm

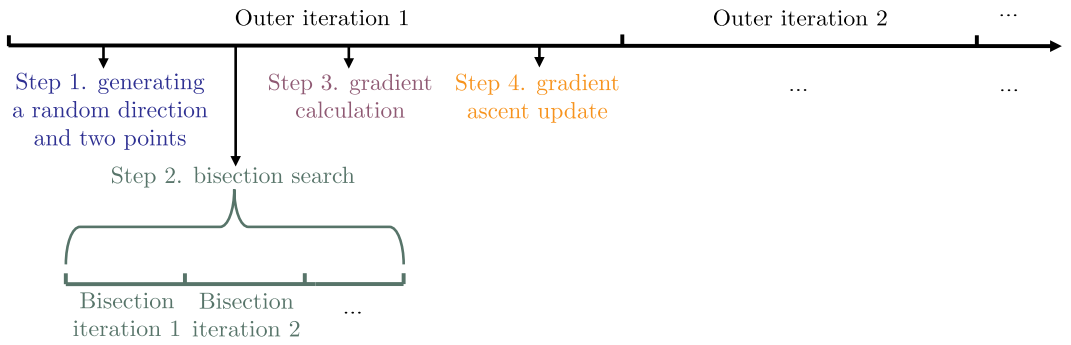


Notes. This is an example with two customer queues connected to one server queue. Let x_1 and x_2 denote the arrival rates of the two customer queues, respectively. The arrival rate of the server queue should be x_1+x_2 because of the balance constraint. Let $x(k):=(x_1(k),x_2(k))$, where k denotes the iteration.

supply functions and the gradients are all unknown, simply using the two-point method does not work because we even do not have zero-order access to the profit function f. As shown in Figure 5(c), for $x=x(k)+\delta u(k)$ or $x=x(k)-\delta u(k)$, to estimate the profit $f(x)$, we need to estimate the prices of all queues $F_1(x_1), F_2(x_2), G_1(x_1+x_2)$ because $f(x)=x_1F_1(x_1)+x_2F_2(x_2)-(x_1+x_2)G_1(x_1+x_2)$, where x_1+x_2 is the arrival rate of the server queue. Take $F_1(x_1)$ as an example. Estimating $F_1(x_1)$ is equivalent to finding the solution to the equation $x_1=F_1^{-1}(p)$ given x_1. We propose using bisection search on prices to estimate p such that $x_1=F_1^{-1}(p)$, as shown in Figure 5(c), where the horizontal axis is price p, the vertical axis is arrival rate x of the queue, and the green dashed curve is the inverse demand function $x=F_1^{-1}(p)$. We first have the initial knowledge that the price $F_1(x_1)$ will be between some lower bound and some upper bound, which forms our first search interval. Then we set the price of the queue to be the midpoint of the search interval and run the system for multiple time slots to calculate average number of arrivals per time slot. This is how we get an estimated value of $F_1^{-1}(p)$ for a chosen p. Then we determine the next search interval by comparing it with x_1, as shown in Figure 5(c). The bisection search stops when the accuracy is good enough. Note that during the process of running the system, we reject arrivals if the queue length is larger than or equal to a predetermined threshold, to control the queue length. After the bisection search, we obtain estimates of the profits $f(x(k)+\delta u(k))$ and $f(x(k)-\delta u(k))$. Then we can substitute these estimates into the two-point method in Figure 5(b) to get an estimate of the gradient and then update the arrival rate $x(k)$.

Now we present the details of the pricing algorithm in the general case. The structure of our pricing algorithm is shown in Figure 6. Before presenting the algorithm, we first define some notations and a shrunk feasible set. Let $\mathcal{E}_{c,i}:=\{j|(i,j)\in\mathcal{E}\}$ and $\mathcal{E}_{s,j}:=\{i|(i,j)\in\mathcal{E}\}$ denote the sets of all queues that are connected to customer queue i and server queue j, respectively. By substituting λ_i,μ_j and by Assumption 3, the optimization problem (2)–(6)

Figure 6. Structure of the Pricing Algorithm



can be equivalently rewritten as

$$\max_{\boldsymbol{x}} \; f(\boldsymbol{x}) := \sum_i \left(\sum_{j\in\mathcal{E}_{\mathrm{c},i}} x_{i,j}\right) F_i\left(\sum_{j\in\mathcal{E}_{\mathrm{c},i}} x_{i,j}\right) - \sum_j \left(\sum_{i\in\mathcal{E}_{\mathrm{s},j}} x_{i,j}\right) G_j\left(\sum_{i\in\mathcal{E}_{\mathrm{s},j}} x_{i,j}\right) \tag{8}$$

$$\text{s.t.} \sum_{j\in\mathcal{E}_{\mathrm{c},i}} x_{i,j} \in [a_{\min}, 1], \quad i = 1,\dots,I, \tag{9}$$

$$\sum_{i\in\mathcal{E}_{\mathrm{s},j}} x_{i,j} \in [a_{\min}, 1], \quad j = 1,\dots,J, \tag{10}$$

$$x_{i,j} \geq 0, \quad (i,j)\in\mathcal{E}. \tag{11}$$

Note that we view $\boldsymbol{x} := (x_{i,j})_{(i,j)\in\mathcal{E}}$ as a vector in $\mathbb{R}^{|\mathcal{E}|}$. Note that the problem (8)–(11) is also concave. Let $\mathcal{D}\subseteq\mathbb{R}^{|\mathcal{E}|}$ denote the feasible set of the problem (8)–(11), that is,

$$\mathcal{D} := \left\{ \boldsymbol{x} \,\middle|\, x_{i,j}\geq 0 \text{ for all } i,j, \sum_{j'\in\mathcal{E}_{\mathrm{c},i}} x_{i,j'} \in [a_{\min},1] \text{ for all } i, \sum_{i'\in\mathcal{E}_{\mathrm{s},j}} x_{i',j} \in [a_{\min},1] \text{ for all } j \right\}. \tag{12}$$

Let $N_{i,j} := \max\{|\mathcal{E}_{\mathrm{c},i}|, |\mathcal{E}_{\mathrm{s},j}|\}$ denote the maximum cardinality of the sets $\mathcal{E}_{\mathrm{c},i}$ and $\mathcal{E}_{\mathrm{s},j}$. Define a shrunk set of $\mathcal{D}$ with a parameter δ as follows:

$$\begin{aligned}\mathcal{D}' := \Bigg\{ \boldsymbol{x} \,\Bigg|\, & \text{for all } i,j, x_{i,j} - \frac{a_{\min}+1}{2N_{i,j}} \geq -\left(1-\frac{\delta}{r}\right)\frac{a_{\min}+1}{2N_{i,j}}, \\ & \sum_{j'\in\mathcal{E}_{\mathrm{c},i}} \left(x_{i,j'} - \frac{a_{\min}+1}{2N_{i,j'}}\right) \in \left[-\left(1-\frac{\delta}{r}\right)\left(\sum_{j'\in\mathcal{E}_{\mathrm{c},i}} \frac{a_{\min}+1}{2N_{i,j'}} - a_{\min}\right), \left(1-\frac{\delta}{r}\right)\left(1-\sum_{j'\in\mathcal{E}_{\mathrm{c},i}} \frac{a_{\min}+1}{2N_{i,j'}}\right)\right], \\ & \sum_{i'\in\mathcal{E}_{\mathrm{s},j}} \left(x_{i',j} - \frac{a_{\min}+1}{2N_{i',j}}\right) \in \left[-\left(1-\frac{\delta}{r}\right)\left(\sum_{i'\in\mathcal{E}_{\mathrm{s},j}} \frac{a_{\min}+1}{2N_{i',j}} - a_{\min}\right), \left(1-\frac{\delta}{r}\right)\left(1-\sum_{i'\in\mathcal{E}_{\mathrm{s},j}} \frac{a_{\min}+1}{2N_{i',j}}\right)\right] \Bigg\},\end{aligned} \tag{13}$$

where

$$\begin{aligned} r := \min_{i,j} \Bigg\{ & \frac{1+a_{\min}}{2N_{i,j}}, \frac{1}{|\mathcal{E}_{\mathrm{c},i}|}\left(1-\sum_{j'\in\mathcal{E}_{\mathrm{c},i}} \frac{a_{\min}+1}{2N_{i,j'}}\right), \frac{1}{|\mathcal{E}_{\mathrm{s},j}|}\left(1-\sum_{i'\in\mathcal{E}_{\mathrm{s},j}} \frac{a_{\min}+1}{2N_{i',j}}\right), \\ & \frac{1}{|\mathcal{E}_{\mathrm{c},i}|}\left(\sum_{j'\in\mathcal{E}_{\mathrm{c},i}} \frac{a_{\min}+1}{2N_{i,j'}} - a_{\min}\right), \frac{1}{|\mathcal{E}_{\mathrm{s},j}|}\left(\sum_{i'\in\mathcal{E}_{\mathrm{s},j}} \frac{a_{\min}+1}{2N_{i',j}} - a_{\min}\right) \Bigg\}, \end{aligned} \tag{14}$$

and $\delta\in(0,r)$. With the definition of $\mathcal{D}'$, we can show (Lemma 4 in the Online Appendix) that $\boldsymbol{x}+\delta\boldsymbol{u}\in\mathcal{D}$ for any $\boldsymbol{x}\in\mathcal{D}'$ and any vector $\boldsymbol{u}$ in the unit ball. To ensure that the definition of $\mathcal{D}'$ is valid, we need the following assumption.

Assumption 4. *Assume that* $\sum_{j\in\mathcal{E}_{c,i}} \frac{a_{\min}+1}{2N_{i,j}} - a_{\min} > 0$ *for all* i *and* $\sum_{i\in\mathcal{E}_{s,j}} \frac{a_{\min}+1}{2N_{i,j}} - a_{\min} > 0$ *for all* j.

As long as $a_{\min}$ is small enough, Assumption 4 holds. Because $N_{i,j} \geq |\mathcal{E}_{c,i}|$ and $a_{\min} < 1$, we have

$$1-\sum_{j\in\mathcal{E}_{c,i}} \frac{a_{\min}+1}{2N_{i,j}} \geq 1-\frac{a_{\min}+1}{2} > 0$$

for all i. Also, by Assumption 4, $\sum_{j\in\mathcal{E}_{c,i}}(a_{\min}+1)/(2N_{i,j}) - a_{\min} > 0$ for all i. Similarly, we have $1-\sum_{i\in\mathcal{E}_{s,j}}(a_{\min}+1)/(2N_{i,j}) > 0$ and $\sum_{i\in\mathcal{E}_{s,j}}(a_{\min}+1)/(2N_{i,j}) - a_{\min} > 0$ for all j. Hence, $r > 0$ and the definition of $\mathcal{D}'$ is valid. Let $\boldsymbol{x}_{\text{ctr}} := ((a_{\min}+1)/(2N_{i,j}))_{(i,j)\in\mathcal{E}}$. We can see that $\boldsymbol{x}_{\text{ctr}} \in \mathcal{D}'$, which can be used as the initial point for the pricing algorithm; $\boldsymbol{x}_{\text{ctr}}$ can be thought of as the "center" of the set $\mathcal{D}'$.

Algorithm 2 (Pricing Algorithm)

Initialize: Choose an exploration parameter $\delta \in (0, r)$. Choose a step size $\eta \in (0,1)$. Choose an accuracy parameter $\epsilon \in (0, 1/e)$. Choose a queue length threshold q^{th}. Choose $\boldsymbol{x}(1) = \boldsymbol{x}_{\text{ctr}} \in \mathcal{D}'$ as the initial point. Define $e_{c,i}$ and $e_{s,j}$ according to (C.1) and (C.2). Define $N := \lceil \beta \ln(1/\epsilon)/\epsilon^2 \rceil$ and $M := \lceil \log_2(1/\epsilon) \rceil$, where $\beta > 0$ is a constant.
Time step counter $t \leftarrow 1$;
Outer iteration counter $k \leftarrow 1$;
repeat
  // **Step 1. generate a random direction and two points**
  Choose a unit vector $\boldsymbol{u}(k) \in \mathbb{R}^{|\mathcal{E}|}$ uniformly at random, that is, $\|\boldsymbol{u}(k)\|_2 = 1$;
  Let $x^+_{i,j}(k) := (\boldsymbol{x}(k) + \delta \boldsymbol{u}(k))_{i,j}$ and $x^-_{i,j}(k) := (\boldsymbol{x}(k) - \delta \boldsymbol{u}(k))_{i,j}$;
  Let $\lambda^+_i(k) := \sum_{j\in\mathcal{E}_{c,i}} x^+_{i,j}(k)$, $\lambda^-_i(k) := \sum_{j\in\mathcal{E}_{c,i}} x^-_{i,j}(k)$, $\mu^+_j(k) := \sum_{i\in\mathcal{E}_{s,j}} x^+_{i,j}(k)$, $\mu^-_j(k) := \sum_{i\in\mathcal{E}_{s,j}} x^-_{i,j}(k)$;
  Let $\boldsymbol{\lambda}^+(k)$ be a vector of $\lambda^+_i(k), i = 1\ldots, I$. Define similarly $\boldsymbol{\lambda}^-(k)$, $\boldsymbol{\mu}^+(k)$, and $\boldsymbol{\mu}^-(k)$.
  // **Step 2. bisection search to approximate** $F_i(\lambda^+_i(k)), F_i(\lambda^-_i(k)), G_j(\mu^+_j(k)), G_j(\mu^-_j(k))$.
  if $k = 1$ **then**
    For all i, let $\underline{p}^+_{c,i}(k,1) = p_{c,i,\min}$, $\overline{p}^+_{c,i}(k,1) = p_{c,i,\max}$;
    For all j, let $\underline{p}^+_{s,j}(k,1) = p_{s,j,\min}$, $\overline{p}^+_{s,j}(k,1) = p_{s,j,\max}$;
    Let $\tilde{q}^{\text{th}} = +\infty$; // We do not reject arrivals for $k = 1$.
  else
    For all i, let $\underline{p}^+_{c,i}(k,1) = p^+_{c,i}(k-1,M) - e_{c,i}$, $\overline{p}^+_{c,i}(k,1) = p^+_{c,i}(k-1,M) + e_{c,i}$;
    For all j, let $\underline{p}^+_{s,j}(k,1) = p^+_{s,j}(k-1,M) - e_{s,j}$, $\overline{p}^+_{s,j}(k,1) = p^+_{s,j}(k-1,M) + e_{s,j}$;
    Let $\tilde{q}^{\text{th}} = q^{\text{th}}$;
  Let $\underline{\boldsymbol{p}}^+_c(k,1)$ be a vector of $\underline{p}^+_{c,i}(k,1), i = 1\ldots, I$. Define similarly $\bar{\boldsymbol{p}}^+_c(k,1)$, $\underline{\boldsymbol{p}}^+_s(k,1)$, $\bar{\boldsymbol{p}}^+_s(k,1)$.
  $t, \boldsymbol{p}^+_c(k,M), \boldsymbol{p}^+_s(k,M) = \text{Bisection}\left(\boldsymbol{\lambda}^+(k), \boldsymbol{\mu}^+(k), \underline{\boldsymbol{p}}^+_c(k,1), \overline{\boldsymbol{p}}^+_c(k,1), \underline{\boldsymbol{p}}^+_s(k,1), \overline{\boldsymbol{p}}^+_s(k,1), \tilde{q}^{\text{th}}, t, M, N, \epsilon\right)$;
  Do Line 9–18 for $\boldsymbol{\lambda}^-(k), \boldsymbol{\mu}^-(k)$. Denote the counterparts of the prices by $\underline{\boldsymbol{p}}^-_c(k,1)$, $\bar{\boldsymbol{p}}^-_c(k,1)$, $\boldsymbol{p}^-_c(k,M)$, $\underline{\boldsymbol{p}}^-_s(k,1)$, $\bar{\boldsymbol{p}}^-_s(k,1)$, $\boldsymbol{p}^-_s(k,M)$.
  // **Step 3. gradient calculation**
  Let $\hat{\boldsymbol{g}}(k) = \frac{|\mathcal{E}|}{2\delta}\left[\left(\sum_{i=1}^I \lambda^+_i(k) p^+_{c,i}(k,M) - \sum_{j=1}^J \mu^+_j(k) p^+_{s,j}(k,M)\right) - \left(\sum_{i=1}^I \lambda^-_i(k) p^-_{c,i}(k,M) - \sum_{j=1}^J \mu^-_j(k) p^-_{s,j}(k,M)\right)\right]\boldsymbol{u}(k)$;
  // **Step 4. gradient ascent update**
  Projected Gradient Ascent: $\boldsymbol{x}(k+1) = \Pi_{\mathcal{D}'}(\boldsymbol{x}(k) + \eta \hat{\boldsymbol{g}}(k))$;
  $k \leftarrow k+1$;
until $t > T$;

The details of the proposed pricing algorithm are shown in Algorithm 2 and Algorithm 3. In the notation in Algorithms 2 and 3, (k) denotes the kth outer iteration; (k,m) denotes the mth bisection iteration in the kth outer iteration. As shown in Algorithm 2, we first choose δ, η, ϵ, and q^{th}; $\delta \in (0, r)$ is an exploration parameter for estimating the gradient, and it is used to construct the set $\mathcal{D}'$. We choose $\boldsymbol{x}(1) = \boldsymbol{x}_{\text{ctr}} \in \mathcal{D}'$ as the initial point of the algorithm; $\eta \in (0,1)$ is the step size for the gradient ascent step; and $\epsilon \in (0, 1/e)$ is the accuracy for the bisection algorithm. The threshold $q^{\text{th}} > 0$ is used to control queue length. We also define $e_{c,i}$ and $e_{s,j}$, which can be later proved to be the upper bounds of the change of the prices in one outer iteration. Note that the parameters $\delta, \eta, \epsilon, q^{\text{th}}$ can be a function of total number of time steps T and can be optimized later to obtain a sublinear regret. In each outer iteration k, we have four steps.

- **Step 1:** We first generate a random direction, that is, a unit vector $\boldsymbol{u}(k) \in \mathbb{R}^{|\mathcal{E}|}$ that is uniformly sampled from the unit sphere, and then change $\boldsymbol{x}(k)$ in the direction of $\boldsymbol{u}(k)$ and also in the opposite direction of $\boldsymbol{u}(k)$, generating

$x^+_{i,j}(k)$ and $x^-_{i,j}(k)$, as shown in Line 6 of Algorithm 2. We can then calculate the arrival rates $\lambda^+_i(k), \mu^+_j(k)$ and $\lambda^-_i(k), \mu^-_j(k)$ as shown in Line 7 of Algorithm 2.

- **Step 2:** Next, we find the prices that approximate $F_i(\lambda^+_i(k)), G_j(\mu^+_j(k))$, and $F_i(\lambda^-_i(k)), G_j(\mu^-_j(k))$ using the bisection search shown in Algorithm 3, which will be illustrated in detail later in this section. Here, we specify the initial search intervals for the bisection search. If $k=1$, we start with search intervals $[p_{\mathrm{c},i,\min}, p_{\mathrm{c},i,\max}]$ and $[p_{\mathrm{s},j,\min}, p_{\mathrm{s},j,\max}]$, as shown in Lines 10 and 11 of Algorithm 2. Note that we do not reject arrivals in the first iteration because we want to quickly learn the price that can keep the queues balanced. If $k>1$, we use $p^+_{\mathrm{c},i}(k-1, M^+_{k-1})$, $p^+_{\mathrm{s},j}(k-1, M^+_{k-1}), p^-_{\mathrm{c},i}(k-1, M^-_{k-1})$, and $p^-_{\mathrm{s},j}(k-1, M^-_{k-1}), e_{\mathrm{c},i}, e_{\mathrm{s},j}$ to construct the lower and upper bounds for the bisection algorithm, as shown in Lines 14 and 15 of Algorithm 2, where $e_{\mathrm{c},i} = \Theta(\eta\epsilon/\delta + \eta + \delta + \epsilon)$ and $e_{\mathrm{s},j} = \Theta(\eta\epsilon/\delta + \eta + \delta + \epsilon)$, and the exact expressions can be found in Appendix C.
- **Step 3:** The bisection algorithm will output approximated prices $\boldsymbol{p}^+_{\mathrm{c}}(k, M^+_k)$, $\boldsymbol{p}^+_{\mathrm{s}}(k, M^+_k)$, $\boldsymbol{p}^-_{\mathrm{c}}(k, M^+_k)$, and $\boldsymbol{p}^-_{\mathrm{s}}(k, M^+_k)$ that correspond to $\boldsymbol{\lambda}^+(k)$, $\boldsymbol{\mu}^+(k)$, $\boldsymbol{\lambda}^-(k)$, and $\boldsymbol{\mu}^-(k)$, respectively, as shown in Lines 18 and 19 of Algorithm 2. Then we can use these arrival rates and prices to calculate an estimate $\hat{\boldsymbol{g}}(k)$ of the gradient of the objective function $f(\boldsymbol{x})$ in (8), as shown in Line 20 of Algorithm 2.
- **Step 4:** With this gradient estimate, we do a projected gradient ascent as shown in Line 21 of Algorithm 2. The algorithm stops until the number of time steps $t \geq T$.

Algorithm 3 (Bisection ($\boldsymbol{\lambda}^{+/-}(k), \boldsymbol{\mu}^{+/-}(k), \underline{\boldsymbol{p}}^{+/-}_{\mathrm{c}}(k,1), \overline{\boldsymbol{p}}^{+/-}_{\mathrm{c}}(k,1), \underline{\boldsymbol{p}}^{+/-}_{\mathrm{s}}(k,1), \overline{\boldsymbol{p}}^{+/-}_{\mathrm{s}}(k,1), \tilde{q}^{\mathrm{th}}, t, M, N, \epsilon$))

1 **for** $m = 1$ to M **do**
2 Let $p^{+/-}_{\mathrm{c},i}(k,m) = \frac{1}{2}(\underline{p}^{+/-}_{\mathrm{c},i}(k,m) + \overline{p}^{+/-}_{\mathrm{c},i}(k,m))$ for all i;
3 Let $p^{+/-}_{\mathrm{s},j}(k,m) = \frac{1}{2}(\underline{p}^{+/-}_{\mathrm{s},j}(k,m) + \overline{p}^{+/-}_{\mathrm{s},j}(k,m))$ for all j;
4 Let $n_{\mathrm{c},i}(k,m) = 0$ for all i and $n_{\mathrm{s},j}(k,m) = 0$ for all j;
5 **repeat**
6 **for** $i = 1$ to I **do**
7 **if** $Q_{\mathrm{c},i}(t) \geq \tilde{q}^{\mathrm{th}}$ **then** set price $p_{\mathrm{c},i,\max}$ for queue i;
8 **else** set price $p^{+/-}_{\mathrm{c},i}(k,m)$ for queue i and $n_{\mathrm{c},i}(k,m) \leftarrow n_{\mathrm{c},i}(k,m) + 1$;
9 **for** $j = 1$ to J **do**
10 **if** $Q_{\mathrm{s},j}(t) \geq \tilde{q}^{\mathrm{th}}$ **then** set price $p_{\mathrm{s},j,\min}$ for queue j;
11 **else** set price $p^{+/-}_{\mathrm{s},j}(k,m)$ for queue j and $n_{\mathrm{s},j}(k,m) \leftarrow n_{\mathrm{s},j}(k,m) + 1$;
12 Use the above set of prices to run one step of the system;
13 $t \leftarrow t+1$;
14 Terminate the algorithm when $t > T$;
15 **until** for all queues i, $n_{\mathrm{c},i}(k,m) \geq N$, that is, the price $p^{+/-}_{\mathrm{c},i}(k,m)$ is run for at least N times and for all queues j, $n_{\mathrm{s},j}(k,m) \geq N$, that is, the price $p^{+/-}_{\mathrm{s},j}(k,m)$ is run for at least N times;
16 Let $t^{+/-}_{\mathrm{c},i}(k,m,n)$ denote the time slot when the price $p^{+/-}_{\mathrm{c},i}(k,m)$ is run for the nth time for the customer-side queue i; let $t^{+/-}_{\mathrm{s},j}(k,m,n)$ denote the time slot when the price $p^{+/-}_{\mathrm{s},j}(k,m)$ is run for the n^{th} time for the server-side queue j;
17 **for** $i = 1$ to I **do**
18 Let $\hat{\lambda}^{+/-}_i(k,m) = (1/N)\sum_{n=1}^{N} A_{\mathrm{c},i}(t^{+/-}_{\mathrm{c},i}(k,m,n))$; // `sample average`
19 **if** $\hat{\lambda}^{+/-}_i(k,m) > \lambda^{+/-}_i(k)$ **then** $\underline{p}^{+/-}_{\mathrm{c},i}(k,m+1) = p^{+/-}_{\mathrm{c},i}(k,m)$, $\overline{p}^{+/-}_{\mathrm{c},i}(k,m+1) = \overline{p}^{+/-}_{\mathrm{c},i}(k,m)$;
20 **else** $\underline{p}^{+/-}_{\mathrm{c},i}(k,m+1) = \underline{p}^{+/-}_{\mathrm{c},i}(k,m)$, $\overline{p}^{+/-}_{\mathrm{c},i}(k,m+1) = p^{+/-}_{\mathrm{c},i}(k,m)$;
21 **for** $j = 1$ to J **do**
22 Let $\hat{\mu}^{+/-}_j(k,m) = (1/N)\sum_{n=1}^{N} A_{\mathrm{s},j}(t^{+/-}_{\mathrm{s},j}(k,m,n))$; // `sample average`
23 **if** $\hat{\mu}^{+/-}_j(k,m) > \mu^{+/-}_j(k)$ **then** $\underline{p}^{+/-}_{\mathrm{s},j}(k,m+1) = p^{+/-}_{\mathrm{s},j}(k,m)$, $\overline{p}^{+/-}_{\mathrm{s},j}(k,m+1) = p^{+/-}_{\mathrm{s},j}(k,m)$;
24 **else** $\underline{p}^{+/-}_{\mathrm{s},j}(k,m+1) = p^{+/-}_{\mathrm{s},j}(k,m)$, $\overline{p}^{+/-}_{\mathrm{s},j}(k,m+1) = \overline{p}^{+/-}_{\mathrm{s},j}(k,m)$;
25 Let $\boldsymbol{p}^{+/-}_{\mathrm{c}}(k,M)$ be a vector of $p^{+/-}_{\mathrm{c},i}(k,M), i = 1,\ldots,I$ and define similarly $\boldsymbol{p}^{+/-}_{\mathrm{s}}(k,M)$.
26 **return** t, $\boldsymbol{p}^{+/-}_{\mathrm{c}}(k,M)$, $\boldsymbol{p}^{+/-}_{\mathrm{s}}(k,M)$

The proposed bisection algorithm is shown in Algorithm 3, which is to find the prices $\boldsymbol{p}^{+/-}_{\mathrm{c}}, \boldsymbol{p}^{+/-}_{\mathrm{s}}$ such that $\lambda^{+/-}_i(k) \approx F_i^{-1}(p^{+/-}_{\mathrm{c},i})$ and $\mu^{+/-}_j(k) \approx G_j^{-1}(p^{+/-}_{\mathrm{s},j})$ for some given target arrival rates $\lambda^{+/-}_i(k), \mu^{+/-}_j(k)$ with initial intervals $[\underline{p}^{+/-}_{\mathrm{c},i}(k,1), \overline{p}^{+/-}_{\mathrm{c}_i}(k,1)]$ and $[\underline{p}^{+/-}_{\mathrm{s},j}(k,1), \overline{p}^{+/-}_{\mathrm{s},j}(k,1)]$ for all i, j. In each bisection iteration m, we first calculate the midpoints of the intervals, $p^{+/-}_{\mathrm{c},i}(k,m)$ and $p^{+/-}_{\mathrm{s},j}(k,m)$ for all i, j just like the standard bisection search, as shown in

Lines 2 and 3 of Algorithm 3. For each queue, if the queue length is less than $\tilde{q}^{\text{th}}$, we will set the price with $p_{\text{c},i}^{+/-}(k,m)$ or $p_{\text{s},j}^{+/-}(k,m)$ that was just calculated; otherwise, we will temporarily reject new arrivals into the queue to control the queue length by using the prices $p_{\text{c},i,\max}$ or $p_{\text{s},j,\min}$. We will use this set of prices to run the system for a number of time slots such that every price has at least N i.i.d. samples, as shown in Lines 4–15 of Algorithm 3. Next, with the samples we obtained, we can calculate an estimate of the arrival rate corresponding to the price we set for each queue, $\hat{\lambda}_i^{+/-}(k,m)$ or $\hat{\mu}_j^{+/-}(k,m)$, as shown in Lines 18 and 22 of Algorithm 3. Then we can update the lower and upper bounds of the searching intervals by comparing $\hat{\lambda}_i^{+/-}(k,m)$, $\hat{\mu}_j^{+/-}(k,m)$ with $\lambda_i^{+/-}(k,m)$, $\mu_j^{+/-}(k,m)$, as shown in Lines 19 and 20 and Lines 23 and 24 of Algorithm 3. We conduct $M = \lceil \log_2(1/\epsilon) \rceil$ bisection iterations to obtain sufficiently high accuracy.

5. Main Result

The main result of the paper is shown in Theorem 1.

Theorem 1. *Let Assumptions 1–4 hold. Under Algorithm 1, Algorithm 2, and Algorithm 3, if* $\epsilon < \delta$, $2e_{\text{c},i} \le p_{\text{c},i,\max} - p_{\text{c},i,\min}$, *and* $2e_{\text{s},j} \le p_{\text{s},j,\max} - p_{\text{s},j,\min}$, *we have*

$$\mathbb{E}[R(T)] = O\left(\frac{\log^4(1/\epsilon)}{\epsilon^4 q^{\text{th}}} + \frac{T}{q^{\text{th}}} + T^2\epsilon^{\frac{\beta}{2}+1} + T\epsilon^{\frac{\beta}{2}-1}\log(1/\epsilon) + \frac{T\max\left\{\frac{\log^2(1/\epsilon)}{\epsilon^2}, q^{\text{th}}\right\}(\eta+\delta)}{q^{\text{th}}} + \frac{T\epsilon}{\delta} + \frac{\log^2(1/\epsilon)}{\eta\epsilon^2}\right), \tag{15}$$

and the queue length

$$\max_{t=1,\dots,T} \max\left\{\max_i Q_{\text{c},i}(t), \max_j Q_{\text{s},j}(t)\right\} \le \max\left\{2\lceil\log_2(1/\epsilon)\rceil\left\lceil\frac{\beta\ln(1/\epsilon)}{\epsilon^2}\right\rceil, q^{\text{th}}\right\}. \tag{16}$$

Based on Theorem 1, we optimize the regret bound by setting the parameters $\epsilon, \eta, \delta, q^{\text{th}}$ to be a function of T. Assume that q^{th} and β are large enough. Then we can first focus on minimizing the order of the following term:

$$T\eta + T\delta + \frac{T\epsilon}{\delta} + \frac{\log^2(1/\epsilon)}{\eta\epsilon^2}. \tag{17}$$

If we ignore logarithmic order, the optimal solution is $\epsilon = T^{-1/3}$, $\eta = \delta = T^{-1/6}$, and the optimal order of (17) is $\tilde{O}(T^{5/6})$. By setting $\beta = 5$ and $q^{\text{th}} = \Theta(T^{2/3})$, we obtain $\mathbb{E}[R(T)] = \tilde{O}(T^{5/6})$. We summarize the above result as the following corollary.

Corollary 1. *Let Assumptions 1–4 hold. Under Algorithm 1, Algorithm 2, and Algorithm 3, with parameters* $\epsilon = T^{-1/3}$, $\eta = \delta = T^{-1/6}$, $q^{\text{th}} = T^{2/3}$, $\beta = 5$, *for sufficiently large* T *such that* $\delta < r$, $\epsilon < 1/e$, $2e_{\text{c},i} \le p_{\text{c},i,\max} - p_{\text{c},i,\min}$, *and* $2e_{\text{s},j} \le p_{\text{s},j,\max} - p_{\text{s},j,\min}$, *we can achieve a sublinear regret as*

$$\mathbb{E}[R(T)] = \tilde{O}(T^{\frac{5}{6}}),$$

and the queue length

$$\max_{t=1,\dots,T} \max\left\{\max_i Q_{\text{c},i}(t), \max_j Q_{\text{s},j}(t)\right\} = \tilde{O}(T^{\frac{2}{3}}).$$

From (15) and (16) in Theorem 1, we can see that there is a tradeoff between minimizing the regret and minimizing the queue length. Firstly, we should not set q^{th} to be greater than the order of $T^{2/3}$ because it will not benefit the regret bound orderwisely because $q^{\text{th}} = T^{2/3}$ is large enough to achieve the optimal order of (17). However, we can reduce q^{th} so that the queue-length bound can be improved, although we may suffer a larger regret. Suppose we want the queue-length bound to be the order of $\tilde{O}(T^{\gamma})$, where $\gamma < 2/3$. Then we need to set $q^{\text{th}} = \tilde{\Theta}(T^{\gamma})$ and $\epsilon \ge \tilde{\Theta}(T^{-\gamma/2})$ by (16). From (15), we need to set the parameters $\epsilon, \eta, \delta, \beta$ to optimize the regret bound. Assume β is large enough. Then optimizing the regret bound is equivalent to minimizing the order of the following term (ignoring logarithmic terms):

$$\frac{1}{\epsilon^4 T^{\gamma}} + T^{1-\gamma} + T\eta + T\delta + \frac{T\epsilon}{\delta} + \frac{1}{\eta\epsilon^2}. \tag{18}$$

The optimal solution is $\epsilon = T^{-\gamma/2}$, $\eta = \delta = T^{-\gamma/4}$, and the optimal order of (18) is $T^{1-\gamma/4}$. By setting $\beta = 4/\gamma - 1$, we obtain $\mathbb{E}[R(T)] = \tilde{O}(T^{1-\gamma/4})$. We summarize the above tradeoff as the following corollary.

Corollary 2. *Let Assumptions 1–4 hold. For any $\gamma \in (0, \frac{2}{3}]$, under Algorithm 1, Algorithm 2, and Algorithm 3, with parameters $\epsilon = T^{-\gamma/2}$, $\eta = \delta = T^{-\gamma/4}$, $q^{\text{th}} = T^{\gamma}$, $\beta = 4/\gamma - 1$, for sufficiently large T such that $\delta < r$, $\epsilon < 1/e$, $2e_{c,i} \le p_{c,i,\max} - p_{c,i,\min}$, and $2e_{s,j} \le p_{s,j,\max} - p_{s,j,\min}$, we can achieve a sublinear regret as*

$$\mathbb{E}[R(T)] = \tilde{O}(T^{1-\gamma/4}),$$

and the queue length

$$\max_{t=1,\dots,T} \max\left\{ \max_i Q_{c,i}(t), \max_j Q_{s,j}(t) \right\} = \tilde{O}(T^{\gamma}).$$

From Corollary 1 and Corollary 2, when the allowable queue length increases (up to $\tilde{\Theta}(T^{2/3})$), the regret bound that we can achieve becomes better. However, even if we allow the queue length to increase over $\tilde{\Theta}(T^{2/3})$, the regret bound cannot be improved and remains $\tilde{O}(T^{5/6})$.

6. Proof Roadmap

In this section, we present a proof roadmap for Theorem 1. The complete proof of Theorem 1 can be found in the Online Appendix. In the proof roadmap, we will consider the choice of parameters as in Corollary 1 to make the proof idea clear, that is, $\epsilon = T^{-1/3}$, $\eta = \delta = T^{-1/6}$, $q^{\text{th}} = T^{2/3}$, $\beta = 5$.

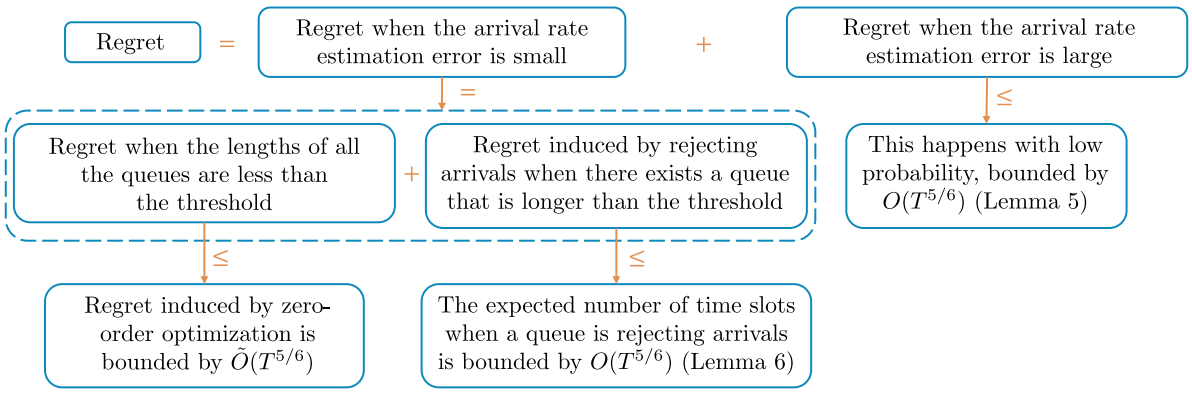
For the first outer iteration, we have no control of the queue length, so the queue length will possibly increase up to $MN = \tilde{\Theta}(T^{2/3})$. Also, because the queue length is controlled by the threshold $q^{\text{th}} = T^{2/3}$ after the first outer iteration, the queue-length bound is $\tilde{O}(T^{2/3})$.

For the regret bound, the proof roadmap is shown in Figure 7. We first define an event $\mathcal{C}$, which means that the estimations of all the arrival rates are ϵ-accurate. Then the regret can be divided into the regret when the event $\mathcal{C}$ occurs and the regret when the event $\mathcal{C}$ does not occur. We show in Lemma 5 in the Online Appendix that the complement of the event, $\mathcal{C}^c$, happens with low probability. Intuitively, because for each estimation we use $N = \tilde{O}(1/\epsilon^2)$ samples, we can obtain ϵ accuracy with high probability by Hoeffding's inequality. Note that Lemma 5 cannot be proved by simply applying Hoeffding's inequality, and we will discuss the challenge in Section 7.

Therefore, the regret when the event $\mathcal{C}$ does not happen can be bounded. For the regret when the event $\mathcal{C}$ happens, that is, the estimations are ϵ-accurate, we further divide it into two parts. One part is the regret when the lengths of all the queues are less than the threshold q^{th} so that there is no queue that is rejecting arrivals, and the other part is the regret induced by rejecting arrivals when there exists a queue whose length is greater than or equal to the threshold q^{th}. For the latter part, we can bound it by the expected number of time slots when a queue is rejecting arrivals. We show in Lemma 6 in the Online Appendix that this can be bounded by $O(T^{5/6})$. The proof is using Lyapunov drift analysis with a Lyapunov function $V_c(t) := \sum_i Q_{c,i}^2(t)$ for the customer side and $V_s(t) := \sum_j Q_{s,j}^2(t)$ for the server side. The challenge of the proof will be discussed in Section 7.

For the part of regret when there is no queue that is rejecting arrivals, regret is induced by zero-order optimization process. We show in the Online Appendix that this part of regret can be bounded by $\tilde{O}(T^{5/6})$. The intuition is as follows. If there is only one run of the system in each outer iteration, then from the literature of two-point method of zero-order optimization (Agarwal et al. 2010), the regret will be bounded by $O(\sqrt{K})$, where K is the

Figure 7. Proof Roadmap



number of outer iterations. In our algorithm, there are $\tilde{\Theta}(T^{2/3})$ runs of the systems in each outer iterations, and there are $\tilde{\Theta}(T^{1/3})$ outer iterations. Hence, the regret will be bounded by $\tilde{\Theta}(T^{2/3})O(\sqrt{T^{1/3}}) = \tilde{O}(T^{5/6})$. However, the proof needs additional steps because, even in the same outer iteration, the prices and arrival rates are changing. In fact, we need to connect the actual prices to the prices corresponding to $\boldsymbol{x}(k)$. Moreover, there can be dependence between the arrival rate estimation and the randomness of the optimization algorithm, which we will discuss in Section 7.

7. Challenges in the Proof

In this section, we discuss the challenges in the proof of Theorem 1 and the methods we use to overcome them.

The first challenge is to prove that the event $\mathcal{C}^c$ occurs with low probability (Lemma 5 in the Online Appendix). The event $\mathcal{C}$ is defined as

$$\mathcal{C} := \Bigg\{ \text{for all outer iteration } k = 1, \dots, \lceil T/(2MN) \rceil, \text{ all bisection iteration } m = 1, \dots, M, \text{ all } i, j,$$

$$\text{and all arrival rates}$$

$$\lambda_i(t^+_{c,i}(k,m,n)), \lambda_i(t^-_{c,i}(k,m,n)), \mu_j(t^+_{s,j}(k,m,n)), \mu_j(t^-_{s,j}(k,m,n)) \in [0,1],$$

$$\left| \frac{1}{N} \sum_{n=1}^{N} A_{c,i}(t^+_{c,i}(k,m,n)) - \lambda_i(t^+_{c,i}(k,m,n)) \right| < \epsilon,$$

$$\left| \frac{1}{N} \sum_{n=1}^{N} A_{c,i}(t^-_{c,i}(k,m,n)) - \lambda_i(t^-_{c,i}(k,m,n)) \right| < \epsilon,$$

$$\left| \frac{1}{N} \sum_{n=1}^{N} A_{s,j}(t^+_{s,j}(k,m,n)) - \mu_j(t^+_{s,j}(k,m,n)) \right| < \epsilon,$$

$$\left| \frac{1}{N} \sum_{n=1}^{N} A_{s,j}(t^-_{s,j}(k,m,n)) - \mu_j(t^-_{s,j}(k,m,n)) \right| < \epsilon \Bigg\},$$

where $t^+_{c,i}(k,m,n)$ is the time slot when the price $p^+_{c,i}(k,m)$ is run for the nth time for the customer-side queue i, and similarly, we define $t^-_{c,i}(k,m,n)$, $t^+_{s,j}(k,m,n)$, and $t^-_{s,j}(k,m,n)$. In the definition, we require that the accurate estimation holds for any arrival rates within [0, 1] because we want to ensure the independence between the event $\mathcal{C}$ and the randomness of $\boldsymbol{u}(k)$ in Algorithm 2, which we will discuss later in this section. To prove that the event $\mathcal{C}$ holds with high probability, simply using Hoeffding's inequality with the union bound does not work because arrival rates are continuous random variables within [0, 1]. Therefore, we use a discretization idea that is similar to the covering argument for linear bandits (Lattimore and Szepesvári 2020). We define a set $\mathcal{S}_{[0,1]}$, a discretized version of [0, 1] with resolution $\epsilon/2$, and rewrite the first inequality in the definition of $\mathcal{C}$ as $|(1/N)\sum_{n=1}^{N} \mathbb{1}\{U^+_{c,i}(k,m,n) \le \lambda\} - \lambda| < \epsilon$, where $U^+_{c,i}(k,m,n)$ is a random variable uniformly distributed in [0, 1] that is generated independently at the very beginning of the process, and similarly, we can rewrite other inequalities in the definition of $\mathcal{C}$. We can bound the quantization error and then use Hoeffding's inequality and the union bound over $\lambda \in \mathcal{S}_{[0,1]}$ to bound the probability of $\mathcal{C}^c$.

The second challenge is to prove the upper bound on the expected number of time slots when the length of one of the queues is greater than or equal to the threshold q^{th} (Lemma 6 in the Online Appendix). As mentioned in the proof roadmap in the previous section, we use Lyapunov drift analysis with quadratic Lyapunov functions. The challenge is how we make sure that the actual arrival rates are balanced or almost balanced. We know that $\boldsymbol{x}(k) \in \mathcal{D}'$, so the corresponding arrival rates are balanced. Because $\boldsymbol{x}^+(k)$ and $\boldsymbol{x}^-(k)$ are δ-close to $\boldsymbol{x}(k)$, their corresponding arrival rates are almost balanced. We want to prove that the actual arrival rates are close enough to the almost balanced arrival rates corresponding to $\boldsymbol{x}^+(k)$ or $\boldsymbol{x}^-(k)$. To show this, we use an important property that the true price corresponding to $\boldsymbol{x}^+(k)$ or $\boldsymbol{x}^-(k)$ is within the search interval of the bisection algorithm with high probability and an important fact that the width of the search interval is small enough for $k \ge 2$. Therefore, the actual arrival rates are also almost balanced for $k \ge 2$.

The third challenge is bounding the regret induced by zero-order optimization conditioned on ϵ-accurate arrival rate estimations (Lemma 7 in the Online Appendix). As mentioned in the proof roadmap in the previous

section, the challenge is that there could be dependence between the estimation of arrival rates and the randomness ($\boldsymbol{u}(k)$) in the optimization algorithm. By defining the event $\mathcal{C}$ such that the ϵ-accuracy is required for all arrival rates in [0, 1], we can show that the event $\mathcal{C}$ is independent of $\boldsymbol{u}(k)$. Note that if we change the definition of $\mathcal{C}$ such that only the estimations of the actual arrival rates are ϵ-accurate, this independence does not hold. This independence between $\mathcal{C}$ and $\boldsymbol{u}(k)$ is essential so that, conditioned on the event $\mathcal{C}$, we can use the proof idea of two-point method of zero-order optimization to bound the regret.

8. Numerical Results

In this section, we present numerical results of the proposed algorithm and compare its performance with that of the UCB algorithm with uniform discretization designed for Lipschitz bandits (Slivkins 2019).

We consider a system with three types of customers ($I = 3$) and three types of servers ($J = 3$). The compatibility graph is given by $\mathcal{E} = \{(1,1),(1,2),(1,3),(2,1),(2,2),(3,2),(3,3)\}$. The demand and supply functions are given by $F_i(\lambda_i) = 2(1-\lambda_i)$ for all $i = 1,2,3$ and $G_j(\mu_j) = 2\mu_j$ for all $j = 1,2,3$. We compare the following algorithms.

- Proposed algorithm: The parameters are set to $q^{\text{th}} = t^{2/3}$, $\beta = 1.0$, $\epsilon = 1.0 \times t^{-1/3}$, $\delta = 0.2 \times t^{-1/6}$, $\eta = 0.1 \times t^{-1/6}$, and $e_{c,i} = e_{s,j} = 8\max\{\epsilon, \delta, \eta\}$.
- Discretization-UCB: We first discretize the price space uniformly and then apply UCB algorithm as in Slivkins (2019). We gradually refine the discretization so that the discretization error decreases at the rate of $T^{-1/(I+J+2)}$, which theoretically achieves the optimal regret order for Lipschitz bandits according to Slivkins (2019).
- Discretization-UCB with $w > 0$: As mentioned in the introduction, the discretization-UCB algorithm fails to balance the matching rates and arrival rates, making it difficult to control the queue length. Therefore, we modify this algorithm by adding a penalty/bonus term such that the reward of pulling an arm at time t becomes

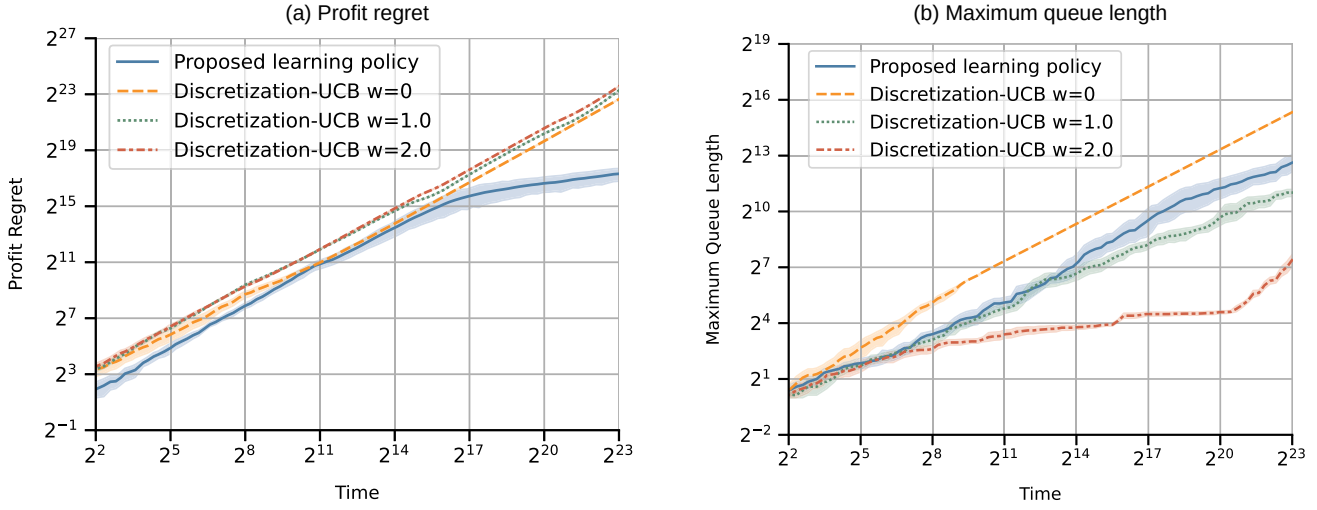
$$(\text{profit at time } t) - w\left(\sum_i Q_{c,i}(t+1) + \sum_j Q_{s,j}(t+1) - \sum_i Q_{c,i}(t) - \sum_j Q_{s,j}(t)\right),$$

where the additional term is the increase/decrease of the total queue length. Note that when $w = 0$, it reduces to the original discretization-UCB algorithm.

For all the algorithms, we set a queue length threshold of $q^{\text{th}} = t^{2/3}$ for fair comparison, which can be viewed as the buffer size of each queue. All the algorithms use the longest-queue-first matching policy.

Figure 8 presents the results of the profit regret and the maximum queue lengths. Note that both the x axis and y axis of the figures are in log scale. We can see that the proposed algorithm significantly outperforms the discretization-UCB algorithm ($w = 0$) in terms of both profit regret and maximum queue length. The maximum queue length under the discretization-UCB algorithm grows at the same rate as the threshold $t^{2/3}$, meaning that the queue lengths consistently hit the buffer size. This indicates that the discretization-UCB algorithm fails to control the queue lengths. The regret under the discretization-UCB algorithm grows almost linearly with t,

Figure 8. Comparison



Note. The shaded areas represent 95% confidence intervals, calculated from 10 runs.

because rejecting arrivals whenever the queue length hits the buffer size (threshold) incurs a constant regret, and the queue lengths hit the threshold frequently. The modified discretization-UCB algorithms with $w = 1.0$ and $w = 2.0$ achieve a better maximum queue length performance. However, their profit regret remains significantly worse than that of the proposed algorithm. The reason could be that the additional penalty/bonus affects the exploration of the arms (prices).

Sensitivity analysis of the proposed algorithm to its parameters, along with additional numerical results, can be found in the Online Appendix.

9. Conclusions

We studied the pricing and matching problem for two-sided queueing systems with unknown demand and supply functions. We proposed a novel learning-based pricing algorithm that combines zero-order stochastic projected gradient ascent with bisection search. We proved that the proposed algorithm yields a sublinear regret $\tilde{O}(T^{5/6})$ and guarantees an anytime queue-length bound $\tilde{O}(T^{2/3})$ and established a tradeoff between the regret bound and the queue-length bound: $\tilde{O}(T^{1-\gamma/4})$ versus $\tilde{O}(T^{\gamma})$ for $\gamma \in (0, 2/3]$. In addition to ride-sharing platforms, the proposed algorithm also has potential applications in online marketplaces and healthcare systems.

Appendix A. Proof of Lemma 1

Fix any policy. By the concavity in Assumption 2, we have

$$\begin{aligned}&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\sum_i \lambda_i(t)F_i(\lambda_i(t)) - \sum_j \mu_j(t)G_j(\mu_j(t))\right]\\ &\le \mathbb{E}\left[\sum_i \left(\frac{1}{T}\sum_{t=1}^{T}\lambda_i(t)\right)F_i\left(\frac{1}{T}\sum_{t=1}^{T}\lambda_i(t)\right) - \sum_j \left(\frac{1}{T}\sum_{t=1}^{T}\mu_j(t)\right)G_j\left(\frac{1}{T}\sum_{t=1}^{T}\mu_j(t)\right)\right].\end{aligned} \tag{A.1}$$

By Jensen's inequality, we have

$$\begin{aligned}&\mathbb{E}\left[\sum_i \left(\frac{1}{T}\sum_{t=1}^{T}\lambda_i(t)\right)F_i\left(\frac{1}{T}\sum_{t=1}^{T}\lambda_i(t)\right) - \sum_j \left(\frac{1}{T}\sum_{t=1}^{T}\mu_j(t)\right)G_j\left(\frac{1}{T}\sum_{t=1}^{T}\mu_j(t)\right)\right]\\ &\le \sum_i \left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)]\right)F_i\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)]\right) - \sum_j \left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu_j(t)]\right)G_j\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu_j(t)]\right).\end{aligned} \tag{A.2}$$

Combining (A.1) and (A.2), we have

$$\begin{aligned}&\sum_{t=1}^{T}\mathbb{E}\left[\sum_i \lambda_i(t)F_i(\lambda_i(t)) - \sum_j \mu_j(t)G_j(\mu_j(t))\right]\\ &\le T\left[\sum_i \left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)]\right)F_i\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)]\right) - \sum_j \left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu_j(t)]\right)G_j\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu_j(t)]\right)\right].\end{aligned} \tag{A.3}$$

Note that the dynamics of the queue length of type i customers is

$$Q_{c,i}(t+1) = Q_{c,i}(t) + A_{c,i}(t) - \sum_{j:(i,j)\in\mathcal{E}} X_{i,j}(t), \tag{A.4}$$

where $X_{i,j}(t)$ denotes the number of matches on link (i,j) at time t. Taking expectation on both sides, we have

$$\mathbb{E}[Q_{c,i}(t+1)] = \mathbb{E}[Q_{c,i}(t)] + \mathbb{E}[\lambda_i(t)] - \sum_{j:(i,j)\in\mathcal{E}} \mathbb{E}[X_{i,j}(t)].$$

Taking time average on both sides, we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[Q_{c,i}(t+1)] = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[Q_{c,i}(t)] + \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)] - \sum_{j:(i,j)\in\mathcal{E}}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[X_{i,j}(t)],$$

which implies that

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)] - \sum_{j:(i,j)\in\mathcal{E}}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[X_{i,j}(t)] &= \frac{1}{T}\sum_{t=1}^{T}(\mathbb{E}[Q_{c,i}(t+1)] - \mathbb{E}[Q_{c,i}(t)])\\ &= \frac{1}{T}\mathbb{E}[Q_{c,i}(T+1)].\end{aligned} \tag{A.5}$$

Similar argument holds for the server side and hence

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu_j(t)] - \sum_{i:(i,j)\in\mathcal{E}}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[X_{i,j}(t)] = \frac{1}{T}\mathbb{E}[Q_{s,j}(T+1)]. \tag{A.6}$$

From (A.5) and (A.6), we know that the tuple

$$\left(\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)] - \frac{1}{T}\mathbb{E}[Q_{c,i}(T+1)]\right)_{i\in\mathcal{I}}, \quad \left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu_j(t)] - \frac{1}{T}\mathbb{E}[Q_{s,j}(T+1)]\right)_{j\in\mathcal{J}}, \quad \left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[X_{i,j}(t)]\right)_{(i,j)\in\mathcal{E}}\right)$$

satisfies Constraints (3)–(6) and is therefore a feasible solution to the fluid optimization problem. Hence, the objective value of this feasible solution must be less than or equal to f^*, that is,

$$\begin{aligned}&\sum_i\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)] - \frac{1}{T}\mathbb{E}[Q_{c,i}(T+1)]\right)F_i\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)] - \frac{1}{T}\mathbb{E}[Q_{c,i}(T+1)]\right)\\ &-\sum_j\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu_j(t)] - \frac{1}{T}\mathbb{E}[Q_{s,j}(T+1)]\right)G_j\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu_j(t)] - \frac{1}{T}\mathbb{E}[Q_{s,j}(T+1)]\right)\le f^*.\end{aligned} \tag{A.7}$$

By the Lipschitz property in Assumption 1, we know that the function $\lambda_i F_i(\lambda_i)$ is Lipschitz in λ_i, and the function $\mu_j G_j(\mu_j)$ is Lipschitz in μ_j. This can be shown as below:

$$\begin{aligned}|(\lambda_i+\Delta)F_i(\lambda_i+\Delta) - \lambda_i F_i(\lambda_i)| &\le |\lambda_i(F_i(\lambda_i+\Delta) - F_i(\lambda_i))| + |\Delta F_i(\lambda_i+\Delta)|\\ &\le (L_{F_i} + p_{c,i,\max})|\Delta|.\end{aligned}$$

Similarly,

$$|(\mu_j+\Delta)G_j(\mu_j+\Delta) - \mu_j G_j(\mu_j)| \le (L_{G_j} + p_{s,j,\max})|\Delta|.$$

Hence, we have

$$\begin{aligned}&\sum_i\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)]\right)F_i\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)]\right) - \sum_j\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu_j(t)]\right)G_j\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu_j(t)]\right)\\ &\le \sum_i\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)] - \frac{1}{T}\mathbb{E}[Q_{c,i}(T+1)]\right)F_i\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\lambda_i(t)] - \frac{1}{T}\mathbb{E}[Q_{c,i}(T+1)]\right)\\ &\quad-\sum_j\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu_j(t)] - \frac{1}{T}\mathbb{E}[Q_{s,j}(T+1)]\right)G_j\left(\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu_j(t)] - \frac{1}{T}\mathbb{E}[Q_{s,j}(T+1)]\right)\\ &\quad+\sum_i(L_{F_i} + p_{c,i,\max})\frac{1}{T}\mathbb{E}[Q_{c,i}(T+1)] + \sum_j(L_{G_j} + p_{s,j,\max})\frac{1}{T}\mathbb{E}[Q_{s,j}(T+1)].\end{aligned} \tag{A.8}$$

Combining (A.3), (A.7), and (A.8), we have

$$\sum_{t=1}^{T}\mathbb{E}\left[\sum_i\lambda_i(t)F_i(\lambda_i(t)) - \sum_j\mu_j(t)G_j(\mu_j(t))\right] \le Tf* + \sum_i(L_{F_i} + p_{c,i,\max})\mathbb{E}[Q_{c,i}(T+1)] + \sum_j(L_{G_j} + p_{s,j,\max})\mathbb{E}[Q_{s,j}(T+1)].$$

Lemma 1 is proved.

Appendix B. Proof of Lemma 2

For the single-link system, we will omit the subscripts i, j in the notations because there is only one customer queue and one server queue. The fluid optimization problem (2)–(6) reduces to

$$\max_x xF(x) - xG(x) \quad \text{s.t.} \quad x\in[0,1]. \tag{B.1}$$

Let $f(x) := xF(x) - xG(x)$. Consider a class of problem instances (i.e., a class of functions F, G) satisfying Assumptions 1–3 for which there exists an optimal solution x^* to Problem (B.1) satisfying the following properties:

(A) $x^*\notin\{0,1\}$.

(B) There exist constants $\nu_r>0$, $\nu_c>0$, and $\nu'\in\partial r(x^*)\cap\partial c(x^*)$ such that for any $x\in[0,1]$,

$$\begin{aligned}r(x^*) - r(x) &\ge \nu'(x^*-x) + \nu_r|x-x^*|\\ c(x) - c(x^*) &\ge \nu'(x-x^*) + \nu_c|x-x^*|,\end{aligned}$$

where $r(x) := xF(x)$, $c(x) := xG(x)$, and $\partial r(\cdot)$ and $\partial c(\cdot)$ represent subdifferentials of functions r and c, respectively.

We will first show that this class of problem instances is not empty. Consider the following functions F and G:

$$F(x) := \begin{cases} 3-2x, & \text{if } x \in [0, \frac{1}{2}] \\ \frac{3}{2} - x + \frac{1}{2x}, & \text{if } x \in (\frac{1}{2}, 1] \end{cases} \qquad G(x) := \begin{cases} \frac{x}{2}, & \text{if } x \in [0, \frac{1}{2}] \\ x - \frac{1}{8x}, & \text{if } x \in (\frac{1}{2}, 1] \end{cases}. \tag{B.2}$$

We can show that this problem instance (B.2) satisfies the requirement. First note that both F and G are strictly decreasing, bijective, and Lipschitz in [0, 1]. Also, the inverse functions of F and G are also Lipschitz because the derivatives/subderivatives of F and G are strictly greater than zero. Hence, Assumption 1 holds. By definition, we obtain

$$r(x) = \begin{cases} 3x - 2x^2, & \text{if } x \in [0, \frac{1}{2}] \\ \frac{3x}{2} - x^2 + \frac{1}{2}, & \text{if } x \in (\frac{1}{2}, 1] \end{cases} \qquad c(x) = \begin{cases} \frac{x^2}{2}, & \text{if } x \in [0, \frac{1}{2}] \\ x^2 - \frac{1}{8}, & \text{if } x \in (\frac{1}{2}, 1] \end{cases}.$$

By definition, we can show that r is concave and c is convex. Hence, Assumption 2 holds. The subdifferentials of r and c are as follows:

$$\partial r(x) = \begin{cases} \{3-4x\}, & \text{if } x \in [0, \frac{1}{2}) \\ [\frac{1}{2}, 1], & \text{if } x = \frac{1}{2} \\ \{\frac{3}{2} - 2x\}, & \text{if } x \in (\frac{1}{2}, 1] \end{cases} \qquad \partial c(x) = \begin{cases} x, & \text{if } x \in [0, \frac{1}{2}) \\ [\frac{1}{2}, 1], & \text{if } x = \frac{1}{2} \\ 2x, & \text{if } x \in (\frac{1}{2}, 1] \end{cases}.$$

Also, we have

$$\partial(r-c)(x) = \begin{cases} \{3-5x\}, & \text{if } x \in [0, \frac{1}{2}) \\ [-\frac{1}{2}, \frac{1}{2}], & \text{if } x = \frac{1}{2} \\ \{\frac{3}{2} - 4x\}, & \text{if } x \in (\frac{1}{2}, 1] \end{cases}.$$

Because $0 \in \partial(r-c)(1/2)$, $x^* = 1/2$ is an optimal solution to Problem (B.1). Hence, Assumption 3 and Property (A) hold. Next, we will prove Property (B). Let $v_r = 1/4$, $v_c = 1/4$, and $v' = 3/4$. Note that $v' \in \partial r(x^*) \cap \partial c(x^*) = [1/2, 1]$. Consider two cases, $x \le x^* = 1/2$ and $x > x^* = 1/2$.

- $x \le x^* = 1/2$: For r, we have

$$r(x^*) - r(x) = 1 - 3x + 2x^2, \qquad v'(x^* - x) + v_r|x - x^*| = \frac{1}{2} - x.$$

Then

$$r(x^*) - r(x) - [v'(x^* - x) + v_r|x - x^*|] = \frac{1}{2} - 2x + 2x^2 = \left(\sqrt{2}x - \frac{1}{\sqrt{2}}\right)^2 \ge 0.$$

Hence, $r(x^*) - r(x) \ge v'(x^* - x) + v_r|x - x^*|$. For c, we have

$$c(x) - c(x^*) = \frac{1}{2}x^2 - \frac{1}{8}, \qquad v'(x - x^*) + v_c|x - x^*| = -\frac{1}{2}\left(\frac{1}{2} - x\right).$$

Then

$$c(x) - c(x^*) - [v'(x - x^*) + v_c|x - x^*|] = \frac{1}{2}x^2 - \frac{1}{2}x + \frac{1}{8} = \left(\frac{x}{\sqrt{2}} - \frac{1}{2\sqrt{2}}\right)^2 \ge 0.$$

Hence, $c(x) - c(x^*) \ge v'(x - x^*) + v_c|x - x^*|$.

- $x > x^* = 1/2$: For r, we have

$$r(x^*) - r(x) = \frac{1}{2} - \frac{3x}{2} + x^2, \qquad v'(x^* - x) + v_r|x - x^*| = -\frac{1}{2}\left(x - \frac{1}{2}\right).$$

Then

$$r(x^*) - r(x) - [v'(x^* - x) + v_r|x - x^*|] = \frac{1}{4} - x + x^2 = \left(x - \frac{1}{2}\right)^2 \ge 0.$$

Hence, $r(x^*) - r(x) \ge v'(x^* - x) + v_r|x - x^*|$. For c, we have

$$c(x) - c(x^*) = x^2 - \frac{1}{4}, \qquad v'(x - x^*) + v_c|x - x^*| = x - \frac{1}{2}.$$

Then

$$c(x) - c(x^*) - [v'(x - x^*) + v_c|x - x^*|] = x^2 - x + \frac{1}{4} = \left(x - \frac{1}{2}\right)^2 \ge 0.$$

Hence, $c(x) - c(x^*) \ge v'(x - x^*) + v_c|x - x^*|$.

Combining the above two cases, we have proved Property (B). Hence, this problem instance (B.2) belongs to the defined class of problem instances. Therefore, the defined class of problem instances is not empty.

Next, we will show that, for this class of problem instances, any policy that keeps the anytime maximum queue length bounded by T^γ must incur a regret of at least $\Omega(T^{1-\gamma})$. Fix any $\gamma \in [0, 1/2)$. Fix any policy such that $\max_{t=1,\dots,T}\max\{Q_c(t), Q_s(t)\} \le T^\gamma$. From the regret definition (7), we have

$$\begin{aligned}\mathbb{E}[R(T)] &= \sum_{t=1}^{T}(\mathbb{E}[x^*F(x^*) - \lambda(t)F(\lambda(t))] + \mathbb{E}[\mu(t)G(\mu(t)) - x^*G(x^*)])\\ &= \sum_{t=1}^{T}(\mathbb{E}[r(x^*) - r(\lambda(t))] + \mathbb{E}[c(\mu(t)) - c(x^*)]).\end{aligned} \tag{B.3}$$

From (B.3) and Property (B), we can obtain the following lower bound:

$$\begin{aligned}\mathbb{E}[R(T)] &\ge \sum_{t=1}^{T}(\mathbb{E}[\nu'(x^* - \lambda(t)) + \nu_r|\lambda(t) - x^*|] + \mathbb{E}[\nu'(\mu(t) - x^*) + \nu_c|\mu(t) - x^*|])\\ &= \sum_{t=1}^{T}\nu'\mathbb{E}[\mu(t) - \lambda(t)] + \sum_{t=1}^{T}\mathbb{E}[\nu_r|\lambda(t) - x^*| + \nu_c|\mu(t) - x^*|].\end{aligned} \tag{B.4}$$

From the queue dynamics, we have

$$Q_c(t+1) = Q_c(t) + A_c(t) - X(t).$$

Taking expectation on both sides, we have

$$\mathbb{E}[Q_c(t+1)] = \mathbb{E}[Q_c(t)] + \mathbb{E}[\lambda(t)] - \mathbb{E}[X(t)],$$

where $X(t)$ is the number of matches on the link at time t. Similarly, we have

$$\mathbb{E}[Q_s(t+1)] = \mathbb{E}[Q_s(t)] + \mathbb{E}[\mu(t)] - \mathbb{E}[X(t)].$$

Hence, we have

$$\mathbb{E}[Q_c(t+1)] - \mathbb{E}[Q_s(t+1)] = \mathbb{E}[Q_c(t)] - \mathbb{E}[Q_s(t)] + \mathbb{E}[\lambda(t) - \mu(t)].$$

Define $Q(t) := Q_c(t) - Q_s(t)$. Then, we have

$$\mathbb{E}[Q(t+1)] = \mathbb{E}[Q(t)] + \mathbb{E}[\lambda(t) - \mu(t)].$$

Summing both sides from $t = 1$ to $t = T$ and noting that $Q(1) = Q_c(1) - Q_s(1) = 0$, we obtain

$$\mathbb{E}[Q(T+1)] = \sum_{t=1}^{T}\mathbb{E}[\lambda(t) - \mu(t)]. \tag{B.5}$$

From (B.4) and (B.5), we obtain

$$\mathbb{E}[R(T)] \ge -|\nu'||\mathbb{E}[Q(T+1)]| + \sum_{t=1}^{T}\mathbb{E}[\nu_r|\lambda(t) - x^*| + \nu_c|\mu(t) - x^*|]. \tag{B.6}$$

By Lemma 3 in the Online Appendix, we know that either $Q_c(T+1) = 0$ or $Q_s(T+1) = 0$. Hence, $|\mathbb{E}[Q(T+1)]| \le \mathbb{E}[|Q(T+1)|] \le \mathbb{E}[\max\{Q_c(T+1), Q_s(T+1)\}] \le T^\gamma$ under the policy. Hence, (B.6) can be further bounded by

$$\begin{aligned}\mathbb{E}[R(T)] &\ge -|\nu'|T^\gamma + \sum_{t=1}^{T}\mathbb{E}[\nu_r|\lambda(t) - x^*| + \nu_c|\mu(t) - x^*|]\\ &\ge -|\nu'|T^\gamma + \min\{\nu_c, \nu_r\}\sum_{t=1}^{T}\mathbb{E}[|\lambda(t) - x^*| + |\mu(t) - x^*|].\end{aligned} \tag{B.7}$$

By the triangle inequality, the term $\sum_{t=1}^{T}\mathbb{E}[|\lambda(t) - x^*| + |\mu(t) - x^*|]$ in (B.7) can be bounded by

$$\sum_{t=1}^{T}\mathbb{E}[|\lambda(t) - x^*| + |\mu(t) - x^*|] \ge \frac{1}{2}\sum_{t=1}^{T}\mathbb{E}[|\lambda(t) - \mu(t)|] + \frac{1}{2}\sum_{t=1}^{T}\mathbb{E}[|\mu(t) - x^*| + |\lambda(t) - x^*|]. \tag{B.8}$$

Next, we will bound the term $\sum_{t=1}^{T}\mathbb{E}[|\lambda(t) - \mu(t)|]$ in (B.8) using the Lyapunov drift method. Consider the drift $\mathbb{E}[Q^2(t+1) - Q^2(t)]$. By the queue dynamics, we can obtain

$$\begin{aligned}\mathbb{E}[Q^2(t+1) - Q^2(t)] &= \mathbb{E}[(Q(t) + A_c(t) - A_s(t))^2 - Q^2(t)]\\ &= \mathbb{E}[(A_c(t) - A_s(t))^2] + 2\mathbb{E}[Q(t)(A_c(t) - A_s(t))]\\ &= \mathbb{E}[(A_c(t) - A_s(t))^2] + 2\mathbb{E}[Q(t)\mathbb{E}[A_c(t) - A_s(t)|Q(t), \lambda(t), \mu(t)]]\\ &= \mathbb{E}[(A_c(t) - A_s(t))^2] + 2\mathbb{E}[Q(t)(\lambda(t) - \mu(t))],\end{aligned}$$

where the third line uses the law of iterated expectation. Summing both sides from $t=1$ to $t=T$, we obtain

$$\begin{aligned}\mathbb{E}[Q^2(T+1)] &= \sum_{t=1}^{T}\mathbb{E}[(A_c(t)-A_s(t))^2] + 2\mathbb{E}\left[\sum_{t=1}^{T}Q(t)(\lambda(t)-\mu(t))\right] \\ &\geq \sum_{t=1}^{T}\mathbb{E}[(A_c(t)-A_s(t))^2] - 2\mathbb{E}\left[\max_{t'=1,\dots,T}|Q(t')|\sum_{t=1}^{T}|\lambda(t)-\mu(t)|\right]. \end{aligned} \tag{B.9}$$

For the variance term $\mathbb{E}[(A_c(t)-A_s(t))^2]$ in (B.9), we have

$$\begin{aligned}&\mathbb{E}[(A_c(t)-A_s(t))^2] \\ &= \mathbb{E}[A_c^2(t)] + \mathbb{E}[A_s^2(t)] - 2\mathbb{E}[A_c(t)A_s(t)] \\ &= \mathbb{E}[\mathbb{E}[A_c^2(t)|\lambda(t)]] + \mathbb{E}[\mathbb{E}[A_s^2(t)|\mu(t)]] - 2\mathbb{E}[\mathbb{E}[A_c(t)A_s(t)|\lambda(t),\mu(t)]] \\ &= \mathbb{E}[\lambda(t)+\mu(t)-2\lambda(t)\mu(t)],\end{aligned}$$

where the second equality uses the law of iterated expectation and the last equality uses the independence between $A_c(t)$ and $A_s(t)$ conditioned on $\lambda(t)$ and $\mu(t)$. Adding and subtracting x^* and $2x^*\mu(t)$, we obtain

$$\begin{aligned}\mathbb{E}[(A_c(t)-A_s(t))^2] &= \mathbb{E}[\lambda(t)-x^*+x^*+\mu(t)-2x^*\mu(t)+2(x^*-\lambda(t))\mu(t)] \\ &= \mathbb{E}[x^*+\mu(t)-2x^*\mu(t)] + \mathbb{E}[\lambda(t)-x^*+2(x^*-\lambda(t))\mu(t)] \\ &\geq \mathbb{E}[x^*+\mu(t)-2x^*\mu(t)] - 3\mathbb{E}[|\lambda(t)-x^*|].\end{aligned}$$

Adding and subtracting x^* and $2(x^*)^2$, we further obtain

$$\begin{aligned}\mathbb{E}[(A_c(t)-A_s(t))^2] &\geq x^*+x^*-2(x^*)^2+\mathbb{E}[\mu(t)-x^*+2x^*(x^*-\mu(t))] - 3\mathbb{E}[|\lambda(t)-x^*|] \\ &\geq 2x^*(1-x^*) - 3\mathbb{E}[|\mu(t)-x^*|] - 3\mathbb{E}[|\lambda(t)-x^*|].\end{aligned} \tag{B.10}$$

Hence, from (B.9) and (B.10), we have

$$\begin{aligned}\mathbb{E}[Q^2(T+1)] \geq &\sum_{t=1}^{T}2x^*(1-x^*) - 3\sum_{t=1}^{T}\mathbb{E}[|\mu(t)-x^*|+|\lambda(t)-x^*|] \\ &- 2\mathbb{E}\left[\max_{t'=1,\dots,T}|Q(t')|\sum_{t=1}^{T}|\lambda(t)-\mu(t)|\right].\end{aligned}$$

Rearranging terms and noting that $\max_{t=1,\dots,T}|Q(t)| \leq \max_{t=1,\dots,T}\max\{Q_c(t),Q_s(t)\} \leq T^\gamma$, we obtain

$$\mathbb{E}\left[\sum_{t=1}^{T}|\lambda(t)-\mu(t)|\right] \geq \frac{\sum_{t=1}^{T}2x^*(1-x^*)}{2T^\gamma} - \frac{3}{2T^\gamma}\sum_{t=1}^{T}\mathbb{E}[|\mu(t)-x^*|+|\lambda(t)-x^*|] - \frac{T^\gamma}{2}.$$

Because $x^* \notin \{0,1\}$ by Property (A), we have $2x^*(1-x^*)>0$. Hence, $\sum_{t=1}^{T}2x^*(1-x^*) = \Theta(T)$. Hence, we have

$$\begin{aligned}\mathbb{E}\left[\sum_{t=1}^{T}|\lambda(t)-\mu(t)|\right] &\geq \Theta(T^{1-\gamma}) - \frac{3}{2T^\gamma}\sum_{t=1}^{T}\mathbb{E}[|\mu(t)-x^*|+|\lambda(t)-x^*|] - \frac{T^\gamma}{2} \\ &= \Theta(T^{1-\gamma}) - \frac{3}{2T^\gamma}\sum_{t=1}^{T}\mathbb{E}[|\mu(t)-x^*|+|\lambda(t)-x^*|],\end{aligned} \tag{B.11}$$

where the last line holds since $\gamma<1/2$. From (B.8) and (B.11) and noting that $\mathbb{E}[\sum_{t=1}^{T}|\lambda(t)-\mu(t)|] \geq 0$, we have

$$\begin{aligned}\sum_{t=1}^{T}\mathbb{E}[|\lambda(t)-x^*|+|\mu(t)-x^*|] &\geq \max\left\{\Theta(T^{1-\gamma}) - \frac{3}{4T^\gamma}\sum_{t=1}^{T}\mathbb{E}[|\mu(t)-x^*|+|\lambda(t)-x^*|], 0\right\} \\ &\quad + \frac{1}{2}\sum_{t=1}^{T}\mathbb{E}[|\mu(t)-x^*|+|\lambda(t)-x^*|] \\ &= \max\left\{\Theta(T^{1-\gamma}) + \left(\frac{1}{2}-\frac{3}{4T^\gamma}\right)\sum_{t=1}^{T}\mathbb{E}[|\mu(t)-x^*|+|\lambda(t)-x^*|],\right. \\ &\quad \left.\frac{1}{2}\sum_{t=1}^{T}\mathbb{E}[|\mu(t)-x^*|+|\lambda(t)-x^*|]\right\} \\ &\geq \min_{y:0\leq y\leq 2T}\left\{\max\left\{\Theta(T^{1-\gamma}) + \left(\frac{1}{2}-\frac{3}{4T^\gamma}\right)y, \frac{y}{2}\right\}\right\},\end{aligned}$$

where in the last inequality we substitute $\sum_{t=1}^{T}\mathbb{E}[|\mu(t)-x^*|+|\lambda(t)-x^*|]$ with y and then take the minimum. If $\gamma>0$, then

the right-hand side will be of order $\min_{y:0\le y\le 2T}\{\max\{\Theta(T^{1-\gamma})+\Theta(y),\Theta(y)\}\}=\min_{y:0\le y\le 2T}\Theta(T^{1-\gamma}+y)=\Theta(T^{1-\gamma})$. If $\gamma=0$, then the right-hand side will be of order $\min_{y:0\le y\le 2T}\{\max\{\Theta(T)-\Theta(y),\Theta(y)\}\}=\Theta(T)$. Hence, we obtain

$$\sum_{t=1}^{T}\mathbb{E}[\,|\lambda(t)-x^*|+|\mu(t)-x^*|\,]\ge\Theta(T^{1-\gamma}). \tag{B.12}$$

From (B.7) and (B.12), we have

$$\begin{aligned}\mathbb{E}[R(T)]&\ge-|\nu'|T^{\gamma}+\min\{\nu_c,\nu_r\}\Theta(T^{1-\gamma})\\&=\Theta(T^{1-\gamma})-\Theta(T^{\gamma})=\Theta(T^{1-\gamma}),\end{aligned}$$

where the last equality is by $\gamma<1/2$. Lemma 2 is proved.

Appendix C. Exact Expressions of $e_{c,i}$ and $e_{s,j}$ in Algorithm 2

$$\begin{aligned}e_{c,i}&=\frac{2\eta\epsilon|\mathcal{E}|^{3/2}L_{F_i}}{\delta}\left[\sum_{i'=1}^{I}L_{F_{i'}}\left(1+L_{F_{i'}^{-1}}(p_{c,i',\max}-p_{c,i',\min})\right)+\sum_{j'=1}^{J}L_{G_{j'}}\left(1+L_{G_{j'}^{-1}}(p_{s,j',\max}-p_{s,j',\min})\right)\right]\\&\quad+2\epsilon L_{F_i}\left(1+L_{F_i^{-1}}(p_{c,i,\max}-p_{c,i,\min})\right)+\eta|\mathcal{E}|^{3/2}L_{F_i}\left(\sum_{i'=1}^{I}|\mathcal{E}_{c,i'}|(L_{F_{i'}}+p_{c,i',\max})+\sum_{j'=1}^{J}|\mathcal{E}_{s,j'}|(L_{G_{j'}}+p_{s,j',\max})\right)\\&\quad+2\delta|\mathcal{E}|^{1/2}L_{F_i}\\&=\Theta\left(\frac{\eta\epsilon}{\delta}+\eta+\delta+\epsilon\right),\end{aligned} \tag{C.1}$$

$$\begin{aligned}e_{s,j}&=\frac{2\eta\epsilon|\mathcal{E}|^{3/2}L_{G_j}}{\delta}\left[\sum_{i'=1}^{I}L_{F_{i'}}\left(1+L_{F_{i'}^{-1}}(p_{c,i',\max}-p_{c,i',\min})\right)+\sum_{j'=1}^{J}L_{G_{j'}}\left(1+L_{G_{j'}^{-1}}(p_{s,j',\max}-p_{s,j',\min})\right)\right]\\&\quad+2\epsilon L_{G_j}\left(1+L_{G_j^{-1}}(p_{s,j,\max}-p_{s,j,\min})\right)+\eta|\mathcal{E}|^{3/2}L_{G_j}\left(\sum_{i'=1}^{I}|\mathcal{E}_{c,i'}|(L_{F_{i'}}+p_{c,i',\max})+\sum_{j'=1}^{J}|\mathcal{E}_{s,j'}|(L_{G_{j'}}+p_{s,j',\max})\right)\\&\quad+2\delta|\mathcal{E}|^{1/2}L_{G_j}\\&=\Theta\left(\frac{\eta\epsilon}{\delta}+\eta+\delta+\epsilon\right).\end{aligned} \tag{C.2}$$

References

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

Agarwal A, Dekel O, Xiao L (2010) Optimal algorithms for online convex optimization with multi-point bandit feedback. Kalai AT, Mohri M, eds. *Proc. 23rd Annual Conf. Learn. Theory* (ACM, New York), 28–40.

Aveklouris A, Puha AL, Ward AR (2023) A fluid approximation for a matching model with general reneging distributions. *Queueing Systems* 106:199–238.

Aveklouris A, DeValve L, Stock M, Ward A (2024) Matching impatient and heterogeneous demand and supply. *Oper. Res.* 73(3):1637–1658.

Besbes O, Zeevi A (2015) On the (surprising) sufficiency of linear models for dynamic pricing with demand learning. *Management Sci.* 61(4):723–739.

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

Chen Y, Hu M (2020) Pricing and matching with forward-looking buyers and sellers. *Manufacturing Service Oper. Management* 22(4):717–734.

Chen B, Shi C (2024) Tailored base-surge policies in dual-sourcing inventory systems with demand learning. *Oper. Res.* 73(4):1723–1743.

Chen X, Liu Y, Hong G (2023) Online learning and optimization for queues with unknown demand curve and service distribution. Preprint, submitted March 6, https://arxiv.org/abs/2303.03399.

Chen X, Liu Y, Hong G (2024) An online learning approach to dynamic pricing and capacity sizing in service systems. *Oper. Res.* 72(6):2677–2697.

Duchi JC, Jordan MI, Wainwright MJ, Wibisono A (2015) Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Trans. Inform. Theory* 61(5):2788–2806.

Flaxman AD, Kalai AT, McMahan HB (2004) Online convex optimization in the bandit setting: Gradient descent without a gradient. Preprint, submitted August 2, https://arxiv.org/abs/cs/0408007.

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

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

Lattimore T (2024) Bandit convex optimisation. Preprint, submitted February 9, https://arxiv.org/abs/2402.06535.

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

Neely M (2022) *Stochastic Network Optimization with Application to Communication and Queueing Systems* (Springer, Cham, Switzerland).

Nguyen LM, Stolyar AL (2018) A queueing system with on-demand servers: Local stability of fluid limits. *Queueing Systems* 89:243–268.

Shamir O (2017) An optimal algorithm for bandit and zero-order convex optimization with two-point feedback. *J. Machine Learn. Res.* 18(52):1–11.

Slivkins A (2019) Introduction to multi-armed bandits. *Foundations Trends Machine Learn.* 12(1–2):1–286.

Srikant R, Ying L (2014) *Communication Networks: An Optimization, Control and Stochastic Networks Perspective* (Cambridge University Press, Cambridge, UK).

Tassiulas L, Ephremides A (1990) Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *Proc. 29th IEEE Conf. Decision Control* (IEEE, Piscataway, NJ), 2130–2132.

Varma SM, Castro F, Maguluri ST (2020) Near optimal control in ride hailing platforms with strategic servers. Preprint, submitted August 9, https://arxiv.org/abs/2008.03762.

Varma SM, Bumpensanti P, Maguluri ST, Wang H (2023) Dynamic pricing and matching for two-sided queues. *Oper. Res.* 71(1):83–100.

Vaze R, Nair J (2022) Non-asymptotic near optimal algorithms for two sided matchings. *Proc. 20th Internat. Sympos. Modeling Optim. Mobile Ad Hoc Wireless Networks* (IEEE, Piscataway, NJ), 17–24.

Yang Z, Srikant R, Ying L (2023) Learning while scheduling in multi-server systems with unknown statistics: Maxweight with discounted UCB. Ruiz F, Dy J, van de Meent J-W, eds. *Proc. Internat. Conf. Artificial Intelligence Statist.* (PMLR, New York), 4275–4312.