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

OPERATIONS RESEARCH
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0030-364X, EISSN 1526-5463

Submitted to *Operations Research*

Online Learning and Optimization for Queues with Unknown Arrival Rate and Service Distribution

Xinyun Chen
School of Data Science, School of Management and Economics, The Chinese University of Hong Kong, Shenzhen, chenxinyun@cuhk.edu.cn

Guiyu Hong
College of Bussiness, Shanghai University of Finance and Economics, Shanghai, China, hongguiyu@sufe.edu.cn

Yunan Liu
Supply Chain Optimization Technology, Amazon, New York, NY, USA, yunanliu@amazon.com,
Industrial and Systems Engineering, North Carolina State University, Raleigh, NC, USA, yliu48@ncsu.edu

Abstract. We investigate an optimization problem in a queueing system where the service provider selects the optimal service fee p and service capacity μ to maximize the cumulative expected profit (the service revenue minus the capacity cost and delay penalty). The conventional *predict-then-optimize* (PTO) approach takes two steps: first, it estimates the model parameters (e.g., arrival rate and service-time distribution) from data; second, it optimizes a model taking these parameters as input. A major drawback of PTO is that its solution accuracy can often be highly sensitive to the parameter estimation errors because PTO is unable to effectively account for how these errors (step 1) will impact the solution quality of the downstream optimization (step 2). To remedy this issue, we develop an online learning framework that automatically incorporates the aforementioned parameter estimation errors in the optimization process; it is an end-to-end approach that can learn the optimal solution without needing to set up the parameter estimation as a separate step as in PTO. Effectiveness of our online learning approach is substantiated by (i) theoretical results including the algorithm convergence and analysis of the *regret* ("cost" to pay over time for the algorithm to learn the optimal policy), and (ii) engineering confirmation via simulation experiments of a variety of representative examples. We also provide careful comparisons between PTO and our online learning method.

Key words: online learning in queues; service systems; capacity planning; staffing; pricing in service systems

1. Introduction

The conventional performance analysis and optimization in queueing systems require the precise knowledge of certain distributional information of the arrival process and service times. For example, consider the $M/GI/1$ queue having Poisson arrivals and general service times, the expected steady-state workload $W(\lambda, \mu, c_s^2)$ is a function of the arrival rate λ, service rate μ and second moment or *squared coefficient of variation* (SCV) $c_s^2 \equiv \text{Var}(S)/\mathbb{E}[S]^2$ of the service time S. In particular, according to the famous Pollaczek–Khinchine (PK) formula (Pollaczek 1930), we have

$$\mathbb{E}[W(\lambda, \mu, c_s^2)] = \frac{\rho}{1-\rho}\frac{1+c_s^2}{2}, \qquad \text{with} \quad \rho \equiv \frac{\lambda}{\mu}. \tag{1}$$

One can never overstate the power of the PK formula because it has such a nice structure that insightfully ties the system performance to all model primitives λ, μ and c_s^2. Indeed, the PK formula has been predominantly used in practice and largely extended to several more general settings such as the $GI/GI/1$ queue with non-Poisson arrivals (Abate et al. 1993) and $M/GI/n$ queue with multiple servers (Cosmetatos 1976).

To optimize desired queueing performance, it is natural to follow the *predict-then-optimize* (PTO) approach, where "predict" means the estimation of required model parameters (e.g., λ, μ and c_s^2) from data (e.g., arrival times and service times) and "optimize" means the optimization of certain queueing decisions using formulas such as (1) with the predicted parameters treated as the true parameters. See panel (a) in Figure 1 for a flow chart of PTO. A potential issue of PTO is that the required queueing formulas can be highly sensitive to the estimation errors of the input parameters (e.g., λ and μ), especially when the system's congestion is critical. For example, when $c_s = \mu = 1$ and $\lambda = 0.99$, the PK formula (1) yields that $\mathbb{E}[W(\lambda, \mu, c_s^2)] = 99$. But a 0.5% increase of the demand rate λ will yield $\mathbb{E}[W(\lambda, \mu, c_s^2)] = 197$, resulting in a 99% relative error in the predicted workload. Consequently, the practical effectiveness of PTO heavily relies on the accuracy of the prediction step to provide near-perfect estimates of the input parameters. Without such precision, solution methods based on these convenient formulas may prove counterproductive or even fail to deliver the desired outcomes.

The performance shortcomings of PTO, particularly in heavy-traffic conditions, stem from its inability to adequately account for parameter estimation errors and the substantial impact these errors have on the quality of the resulting "optimized" decision variables. To help remedy this issue, we propose *an online learning framework that automatically incorporates the aforementioned parameter estimation errors in the solution prescription process; it is an end-to-end approach that can learn the optimal solution more directly from data, so that we no longer need to set up the parameter estimation as a separate stage as in PTO.* In this paper, we solve a pricing and capacity sizing problem in an $M/GI/1$ queue, where the service provider seeks the optimal service fee p and service rate μ so as to maximize the long-term profit, which is the revenue minus the staffing cost and the queueing penalty, namely,

$$\max_{\mu,p} \ \mathcal{P}(\mu, p) \equiv \lambda(p)p - h_0\mathbb{E}[W] - c(\mu), \tag{2}$$

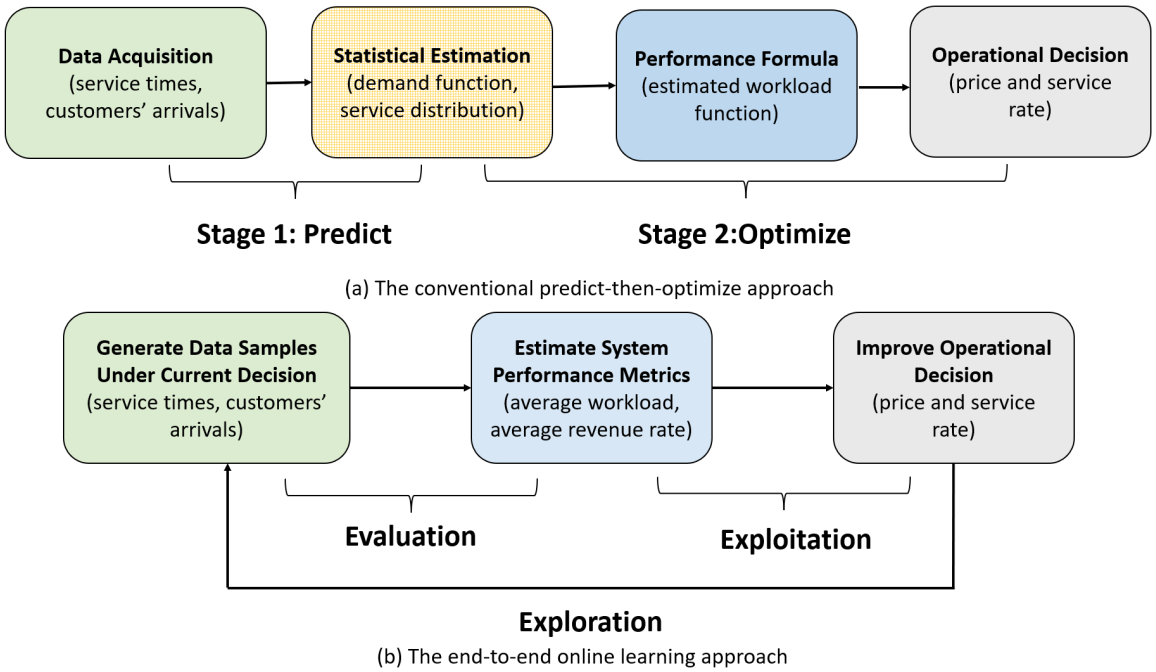


Figure 1 Schematic presentations for (a) the two-step conventional *predict-then-optimize* scheme and (b) the end-to-end *online learning* scheme.

where W is the system's steady-state workload, $c(\mu)$ is the cost for providing service capacity μ and h_0 is a holding cost per job per unit of time. Problems in this framework have a long history, see for example Kumar and Randhawa (2010), Lee and Ward (2014), Lee and Ward (2019), Maglaras and Zeevi (2003), Nair et al. (2016), Kim and Randhawa (2018), Chen et al. (2024) and the references therein. The major distinction is that in the present paper, we assume that neither the arrival rate $\lambda(p)$ (as a function of p) or the service-time distribution is explicitly available to the service provider. (As showed In Section 6.1.1, we will see that the PTO approach for solving Problem (2) indeed suffer from unaccountable estimation errors in the model parameters.)

Our online learning approach operates in successive cycles in which the service provider's operational decisions are being continuously evolved using newly generated data. Data here include customers' arrival and service times under the policy presently in use. See panel (b) in Figure 1 for an illustration of the online learning approach. In each iteration k, the service provider evaluates the current decision (μ_k, p_k) based on the newly generated data. Then, the decision is updated to (μ_{k+1}, p_{k+1}) according to the evaluation result (the exploitation step). In the next iteration, the service provider continues to operate the system under (μ_{k+1}, p_{k+1}) to generate more data (the exploration step). We call this algorithm *Learning in Queue with Unknown Arrival Rate* (LiQUAR).

1.1. Advantages and challenges.

First, the conventional queueing control problem builds heavily on formulas such as (1) and requires the precise knowledge of certain distributional information which may not always be readily available. For example, the acquisition of an accurate estimate of the function $\lambda(p)$ across the entire spectrum of the price p is not straightforward and can be both time consuming and costly. In contrast, the online learning approach eliminates the need for such prior information, excelling at "learning from scratch". Second, unlike the two-step PTO procedure, the online learning approach is an integrated method that inherently accounts for estimation errors in observed data during the decision-making process. This allows it to utilize data more effectively, leading to improved decisions that are more robust and effective. In contrast to PTO's "static" learning, where prediction and optimization are distinctly separate steps, LiQUAR employs a reactive learning approach, characterized by its continuous and dynamic interaction with data.

On the other hand, the development of online learning methodologies in queue systems is far from a straightforward extension of their use in other fields, as it must address the unique characteristics of queueing dynamics. First, when the control policy is updated at the beginning of a cycle, the previously established near steady-state dynamics are disrupted, and the system enters a transient phase. The dynamics during this period are endogenously influenced by the updated control policy, giving rise to the so-called *regret of nonstationarity*. Second, the convergence of decision iterations depends heavily on the statistical efficiency of the evaluation step and the specific properties of the queueing data. This introduces new challenges due to the distinctive nature of queueing dynamics. Unlike standard online learning settings (e.g. stochastic bandits), queueing data such as waiting times and queue lengths are often *biased, unbounded, and temporally correlated.* These unique features of queueing models present significant obstacles to the design and analysis of online learning methodologies, necessitating novel approaches that account for these complexities. Finally, our algorithm operates without requiring knowledge of the arrival rate function or the service-time distribution. This makes our research problem more challenging because we cannot take advantage of the detailed structure of the underlying model. Therefore, we are motivated to develop a conceptually simple, model-free learning framework in order to address the above-mentioned challenges.

1.2. Contributions and organization

Our paper makes the following contributions.

- We are the first to develop an online learning scheme for the $M/GI/1$ queue with unknown demand function and service-time distribution. The effectiveness of our algorithm stems from its well-integrated queueing features, encompassing both the overall algorithm design and the optimization of hyperparameters. For our online learning algorithm, we establish a regret bound of $O(\sqrt{T}\log(T))$. In comparison with the standard $O(\sqrt{T})$ regret for model-free *stochastic gradient descent* (SGD)

methods assuming unbiased and independent reward samples, our regret analysis exhibits an extra $\log(T)$ term which rises from the nonstationary queueing dynamics due to the policy updates. For the $M/M/1$ model, we derived a more detailed regret bound expressed explicitly as a function of the traffic intensity.

- At the heart of our regret analysis is to properly link the estimation errors from queueing data to the algorithm's hyperparameters and regret bound. For this purpose, we develop new results that establish useful statistical properties of data samples generated by a $M/G/1$ queue. Besides serving as building blocks for our regret analysis in the present paper, these results are of independent research interest and may be used to analyze the estimation errors of data in sequential decision making in ergodic queues. Hence, the theoretic analysis and construction of the gradient estimator may be extended to other queueing models which share similar ergodicity properties.
- Supplementing the theoretical results, we evaluate the practical effectiveness of our method by conducting comprehensive numerical experiments. In particular, our numerical results confirm that the online learning algorithm is efficient and robust to several model and algorithm parameters such as service distributions and updating step sizes; we also generalize our algorithm to the $GI/GI/1$ model. Next, we conduct a systematic analysis and experiments to compare LiQUAR to (i) PTO and (ii) gradient-based reinforcement learning methods.

Organization of the paper. In Section 2, we review the related literature. In Section 3, we introduce the model and its assumptions. In Section 4, we present LiQUAR and describe how the queueing data is processed in our algorithm. In Section 5, we conduct the convergence and regret analysis for LiQUAR. The key steps of our analysis form a quantitative explanation of how estimation errors in queueing data propagate through our algorithm flow and how they influence the quality of the LiQUAR solutions. We analyze the total regret by separately treating *regret of non-stationarity* - the part of regret stemming from transient system dynamics, *regret of suboptimality* - the part aroused by the errors due to suboptimal decsions, and *regret of finite difference* - the part originating from the need of estimation of gradient. In Section 5.3, we report a regret bound explicitly expressed as a function of the traffic intensity for $M/M/1$. In Section 6, we conduct numerical experiments to confirm the effectiveness and robustness of LiQUAR. In Sections 7 and 8, We compare LiQUAR to PTO and gradient-based reinforcement learning methods. We provide concluding remarks in Section 9. Technical proofs and supplementary results are given in the e-Companion.

2. Related Literature

The present paper is related to the following four streams of literature.

Pricing and capacity sizing in queues. There is rich literature on pricing and capacity sizing for service systems under various settings. Maglaras and Zeevi (2003) studies a static pricing and capacity sizing problem in a processor sharing queue motivated by internet applications; Kumar and Randhawa (2010) considers

a static pricing problem for a single-server system with nonlinear delay cost; Nair et al. (2016) studies static capacity sizing problem for $M/M/1$ and $M/M/k$ systems with network effect among customers; Kim and Randhawa (2018) considers a dynamic pricing problem in an $M/M/1$ queue. The specific problem that we consider here is related to Lee and Ward (2014), which considers joint static pricing and capacity sizing for $GI/GI/1$ queues with known demand. Later, they further extend their results to the $GI/GI/1+G$ model with customer abandonment in Lee and Ward (2019). Although the present work is motivated by the pricing and capacity sizing problem for service systems, unlike the above-cited works, we assume no knowledge of the demand rate and service distribution.

Demand Learning. Broder and Rusmevichientong (2012) considers a dynamic pricing problem for a single product with an unknown parametric demand curve and establishes an optimal minimax regret in the order of $O(\sqrt{T})$. Keskin and Zeevi (2014) investigates a pricing problem for a set of products with an unknown parameter of the underlying demand curve. Besbes and Zeevi (2015) studies demand learning using a linear curve as a local approximation of the demand curve and establishes a minimax regret in the order of $O(\sqrt{T})$. Later, Cheung et al. (2017) solves a dynamic pricing and demand learning problem with limited price experiments. We draw distinctions from these papers by studying a pricing and capacity sizing problem with demand learning in a queueing setting where our algorithm design and analysis need to take into account unique features of the queueing systems.

Machine learning in queueing systems Our paper is related to a small but booming literature on machine earning in queueing systems. Dai and Gluzman (2021) studies an actor-critic algorithm for queueing networks. Liu et al. (2019) and Shah et al. (2020) develop reinforcement learning techniques to treat the unboundedness of the state space of queueing systems. Raeis et al. (2021) applies deep deterministic policy gradient algorithms to a service rate control problem. Murthy et al. (2024) considers natural policy gradient for control problems in communication networks. Comte et al. (2023) develops a policy gradient method with known score functions of queueing systems. Krishnasamy et al. (2021) develops bandit methods for scheduling problems in a multi-server queue with unknown service rates. Zhong et al. (2024) proposes an online learning method to study a scheduling problem for a multiclass $M_t/M/N+M$ system with unknown service rates and abandonment rates. Chen et al. (2024) studies the joint pricing and capacity sizing problem for $GI/GI/1$ with known demand. See Walton and Xu (2021) for a review of the role of information and learning in queueing systems. Recent research has explored the application of deep learning methods to predict queueing performance: Baron et al. (2023) proposes a deep-learning-based steady-state predictor for the $GI/GI/1$ queue; Garyfallos et al. (2025a, 2024, 2025b) develop recurrent neural network models to predict transient performance in nonstationary queues. Our paper is most closely related to Jia et al. (2024) which studies a price-based revenue management problem in an $M/M/c$ queue with unknown demand and discrete price space, under a multi-armed bandit framework. Later, Jia et al. (2022) extends the results in Jia et al. (2024) to the problem setting with a continuous price space and considers linear demand functions.

Similar to Jia et al. (2024, 2022), we also study a queueing control problem with unknown demand and continuous decision variables. The major distinction is that in addition to maximizing the service profit as by Jia et al. (2024), the present paper also includes a *queueing penalty* in our optimization problem as a measurement of the quality of service (Kumar and Randhawa 2010, Lee and Ward 2014, 2019). However, this introduces new technical challenges in algorithm design and regret analysis, such as addressing the bias and autocorrelation inherent in queueing data. Besides, the present paper considers more general service distributions and demand functions.

Online learning with continuous decision-making has also been explored in inventory systems. For instance, Huh et al. (2009) proposed an SGD-based algorithm to optimize base-stock policies for inventory systems with positive lead times. Later, Zhang et al. (2020) developed a simulation-based algorithm for the same problem, achieving an optimal regret bound of $O(T^{1/2})$. More recently, Yuan et al. (2021) integrated stochastic gradient descent with bandit algorithms to address convexity challenges and optimize (s, S) policies. While our approach also employs gradient-based methods, the fundamental differences between queueing system dynamics and inventory models lead to distinct algorithm designs, particularly in the construction of gradient estimators, as well as in the theoretical analysis.

3. Model and Assumptions

We study an $M/GI/1$ queueing system having customer arrivals according to a Poisson process (the M), *independent and identically distributed* (I.I.D.) service times following a general distribution (the GI), and a single server that provides service following the *first-in-first-out* (FIFO) discipline. Each customer upon joining the queue is charged by the service provider a fee $p > 0$. The demand arrival rate (per time unit) depends on the service fee p and is denoted as $\lambda(p)$. To maintain a service rate μ, the service provider continuously incurs a staffing cost at a rate $c(\mu)$ per time unit.

For $\mu \in [\underline{\mu}, \bar{\mu}]$ and $p \in [\underline{p}, \bar{p}]$, we have $\lambda(p) \in [\underline{\lambda}, \bar{\lambda}] \equiv [\lambda(\bar{p}), \lambda(\underline{p})]$, and the service provider's goal is to determine the optimal service fee p^* and service capacity μ^* with the objective of maximizing the steady-state expected profit, or equivalently minimizing the objective function $f(\mu, p)$ as follows

$$\min_{(\mu,p)\in\mathcal{B}} f(\mu, p) \equiv h_0 \mathbb{E}[W_\infty(\mu, p)] + c(\mu) - p\lambda(p), \qquad \mathcal{B} \equiv [\underline{\mu}, \bar{\mu}] \times [\underline{p}, \bar{p}]. \tag{3}$$

Here $W_\infty(\mu, p)$ is the stationary workload process observed in continuous time under control parameter (μ, p).

REMARK 1 (STATIC PRICE AND SERVICE CAPACITY AS BENCHMARK). In this paper, we focus on optimizing a static service fee p and service rate μ, whose optimal values can be explicitly characterized using the P–K formula when model parameters are known. This choice allows us to better focus on studying the effect of parameter uncertainty on the performance of learning algorithms in queues, while keeping the benchmark analytically tractable. Although the theoretically optimal policy over a finite horizon would be

state- and time-dependent, analyzing such dynamic policies is substantially more complex and beyond the scope of this work. Moreover, static policies are widely adopted in practice, admit closed-form analysis, and are known to achieve robust performance in many settings (Kumar and Randhawa 2010, Elmachtoub and Shi 2023, Bergquist and Elmachtoub 2023).

In detail, under control parameter (μ, p), customers arrive according to a Poisson process with rate $\lambda(p)$. Let V_n be an I.I.D. sequence corresponding to customers' *workloads* under unit service rate (under service rate μ, customer n has service time V_n/μ). We have $\mathbb{E}[V_n] = 1$ so that the mean service time is $1/\mu$ under service rate μ. Denote by $N(t)$ the number of arrivals by time t. The total amount of workload brought by customers at time t is denoted by $J(t) = \sum_{k=1}^{N(t)} V_k$. Then the workload process $W(t)$ follows the *stochastic differential equation* (SDE)

$$dW(t) = dJ(t) - \mu \mathbf{1}\left(W(t) > 0\right) dt.$$

In particular, given the initial value of $W(0)$, we have

$$W(t) = R(t) - 0 \wedge \min_{0 \le s \le t} R(s), \quad R(t) \equiv W(0) + J(t) - \mu t.$$

The difference $(W(t) - R(t))/\mu$ is the total idle time of the server by time t. It is known in the literature (Asmussen 2003, Corollary 3.3, Chapter X) that under the stability condition $\lambda(p) < \mu$, the workload process $W(t)$ has a unique stationary distribution and we denote by $W_\infty(\mu, p)$ the stationary workload under parameter (μ, p).

We impose the following assumptions on the $M/GI/1$ system throughout the paper.

ASSUMPTION 1. (***Demand rate, staffing cost, and uniform stability***)

(a) *The arrival rate $\lambda(p)$ is continuously differentiable in the third order and non-increasing in p. Besides,*

$$C_1 < \lambda'(p) < C_2,$$

where

$$C_1 \equiv 2\max\left(g(\bar{\mu})\frac{\lambda''(p)}{\lambda'(p)}, g(\underline{\mu})\frac{\lambda''(p)}{\lambda'(p)}\right)\lambda(p) - \frac{4\lambda(p)(\underline{\mu} - \lambda(p))}{h_0 C}, \quad C_2 \equiv -\max\left(\sqrt{\frac{0 \vee (-\lambda''(p)(\bar{\mu} - \lambda(p)))}{2}}, \frac{p\lambda''(p)}{2}\right),$$

$g(\mu) = \frac{\mu}{\mu - \lambda(p)} - \frac{p(\mu - \lambda(p))}{h_0 C}$ *and* $C = (1 + c_s^2)/2$.

(b) *The staffing cost $c(\mu)$ is continuously differentiable in the third order, non-decreasing and convex in μ.*

(c) *The lower bounds $\underline{p}$ and $\underline{\mu}$ satisfy that $\lambda(\underline{p}) < \underline{\mu}$ so that the system is uniformly stable for all feasible choices of (μ, p).*

Although Condition (a) looks complicated, it essentially requires that the derivative of $\lambda(p)$ be not too large or too small. Condition (a) will be used to ensure that the objective function $f(\mu, p)$ is convex in the convergence analysis of our gradient-based online learning algorithm in Section 5.1. The two inequalities hold

for a variety of commonly used demand functions, including both convex functions and concave functions. Examples include (1) linear demand $\lambda(p) = a - bp$ with $0 < b < 4\underline{\lambda}(\underline{\mu} - \bar{\lambda})/h_0 C$; (2) quadratic demand $\lambda(p) = c - ap^2$ with $a, c > 0$, and $\frac{\bar{\mu}-c}{3\underline{p}^2} < a < \left(\frac{3(\underline{\mu}-\bar{\lambda})\underline{p}}{h_0 C} - \frac{\underline{\mu}}{\underline{\mu}-\bar{\lambda}}\right)\frac{\underline{\lambda}}{\bar{p}^2}$; (3) exponential demand $\lambda(p) = \exp(a - bp)$ with $0 < b < 2/\bar{p}$; (4) logit demand $\lambda(p) = M_0 \exp(a - bp)/(1 + \exp(a - bp))$ with $a - b\bar{p} < \log(1/2)$ and $0 < b < 2/\bar{p}$. See Section EC.4 for detailed discussions.

Condition (c) of Assumption 1 is commonly used in the literature of SGD methods for queueing models to ensure that the steady-state mean waiting time $\mathbb{E}[W_\infty(\mu, p)]$ is differentiable with respect to model parameters. See Chong and Ramadge (1993), Fu (1990), L'Ecuyer et al. (1994), L'Ecuyer and Glynn (1994), and also Theorem 3.2 of Glasserman (1992). In Section EC.5.2, we present an initial attempt to relax Assumption 1(c).

We do not require full knowledge of service and inter-arrival time distributions. But in order to bound the estimation error of the queueing data, we require the individual workload to be light-tailed. Specifically, we make the following assumptions on V_n.

ASSUMPTION 2. (***Light-tailed individual workload***) *There exists a sufficiently small constant $\eta > 0$ such that*

$$\mathbb{E}[\exp(\eta V_n)] < \infty.$$

In addition, there exist constants $0 < \theta < \eta/2\bar{\mu}$ and $\gamma_0 > 0$ such that

$$\phi_V(\theta) < \log\left(1 + \underline{\mu}\theta/\bar{\lambda}\right) - \gamma_0, \tag{4}$$

where $\phi_V(\theta) \equiv \log \mathbb{E}[\exp(\theta V_n)]$ is the cumulant generating functions of V_n.

Note that $\phi_V'(0) = 1$ as $\mathbb{E}[V_n] = 1$. Suppose ϕ_V is smooth around 0, then we have $\phi_V(\theta) = \theta + o(\theta)$ by Taylor's expansion. On the other hand, as $\underline{\mu} > \bar{\lambda}$ under Assumption 1, there exists $a > 0$ such that $\log\left(1 + \underline{\mu}\theta/\bar{\lambda}\right) = (1 + a)\theta + o(\theta)$. This implies that, we can choose θ small enough such that $\log\left(1 + \underline{\mu}\theta/\bar{\lambda}\right) - \phi_V(\theta) > \frac{a\theta}{2}$ and then we set $\gamma_0 = \frac{a\theta}{2}$. Hence, a sufficient condition that warrants (4) is to require that ϕ_V be smooth around 0, which is true for many distributions of V considered in common queueing models. Assumption 2 will be used in our proofs to establish ergodicity result.

4. Our Online Learning Algorithm

We first explain the main ideas in the design of LiQUAR and provide the algorithm outline in Section 4.1. The key step in our algorithm design is to construct a data-based gradient estimator, which is explained with details in Section 4.2. As a unique feature of service systems, there is a delay in data observation of individual workloads, i.e., they are revealed only after service completion. We also explain how to deal with this issue in Section 4.2. The design of algorithm hyperparameters in LiQUAR will be specified later in Section 5 based on the regret analysis results. In the rest of the paper, we use bold symbols for vectors and matrices.

4.1. Algorithm outline

The basic structure of LiQUAR follows the online learning scheme as illustrated in Figure 1. It interacts with the queueing system in continuous time and improves pricing and staffing policies iteratively. In each iteration $k \in \{1, 2, ...\}$, LiQUAR operates the queueing system according to control parameters $\bar{\boldsymbol{x}}_k \equiv (\bar{\mu}_k, \bar{p}_k)$ for a certain time period, and collects data generated by the queueing system during the period. At the end of an iteration, LiQUAR estimates the gradient of the objective function $\nabla f(\bar{\boldsymbol{x}}_k)$ based on the collected data and accordingly updates the control parameters. The updated control parameters will be used in the next iteration.

We use the *finite difference* (FD) method (Broadie et al. 2011) to construct our gradient estimator. Our main purpose is to make LiQUAR model-free and applicable to the settings where the demand function $\lambda(p)$ is unknown. To obtain the FD estimator of $\nabla f(\bar{\boldsymbol{x}}_k)$, LiQUAR splits total time of iteration k into two equally divided intervals (i.e., cycles) each with T_k time units. We index the two cycles by $2k-1$ and $2k$, in which the system is respectively operated under control parameters

$$\boldsymbol{x}_{2k-1} \equiv \bar{\boldsymbol{x}}_k - \delta_k \cdot \boldsymbol{Z}_k/2 \equiv (\mu_{2k-1}, p_{2k-1}) \quad \text{and} \quad \boldsymbol{x}_{2k} \equiv \bar{\boldsymbol{x}}_k + \delta_k \cdot \boldsymbol{Z}_k/2 \equiv (\mu_{2k}, p_{2k}), \tag{5}$$

where δ_k is a positive and small number and $\boldsymbol{Z}_k \in \mathbb{R}^2$ is a random vector independent of system dynamics such that $\mathbb{E}[\boldsymbol{Z}_k] = (1,1)^\top$. Using data collected in the two cycles, LiQUAR obtains estimates of the system performance $\hat{f}(\boldsymbol{x}_{2k})$ and $\hat{f}(\boldsymbol{x}_{2k-1})$, which in turn yield the FD approximation for the gradient $\nabla f(\bar{\boldsymbol{x}}_k)$:

$$\boldsymbol{H}_k \equiv \frac{\hat{f}(\boldsymbol{x}_{2k}) - \hat{f}(\boldsymbol{x}_{2k-1})}{\delta_k}.$$

Then, LiQUAR updates the control parameter according to a SGD recursion as $\bar{\boldsymbol{x}}_{k+1} = \Pi_{\mathcal{B}}(\bar{\boldsymbol{x}}_k - \eta_k \boldsymbol{H}_k)$, where $\Pi_{\mathcal{B}}$ is the operator that projects $\bar{\boldsymbol{x}}_k - \eta_k \boldsymbol{H}_k$ to $\mathcal{B}$. We give the outline of LiQUAR below.

Outline of LiQUAR:

0. Input: hyper-parameters $\{T_k, \eta_k, \delta_k\}$ for $k = 1, 2, ...$, initial policy $\bar{\boldsymbol{x}}_1 = (\bar{\mu}_1, \bar{p}_1)$.
   For $k = 1, 2, ..., L$,
1. Obtain $\boldsymbol{x}_l$ according to (5) for $l = 2k-1$ and $2k$. In cycle l, operate the system with policy $\boldsymbol{x}_l$ for T_k units of time.
2. Compute $\hat{f}(\boldsymbol{x}_{2k-1})$ and $\hat{f}(\boldsymbol{x}_{2k})$ from the queueing data to build an estimator $\boldsymbol{H}_k$ for $\nabla f(\mu_k, p_k)$.
3. Update $\bar{\boldsymbol{x}}_{k+1} = \Pi_{\mathcal{B}}(\bar{\boldsymbol{x}}_k - \eta_k \boldsymbol{H}_k)$.

Next, we explain in details how the gradient estimator $\boldsymbol{H}_k$, along with $\hat{f}(\boldsymbol{x}_{2k-1})$ and $\hat{f}(\boldsymbol{x}_{2k})$, are computed from the queueing data in Step 2.

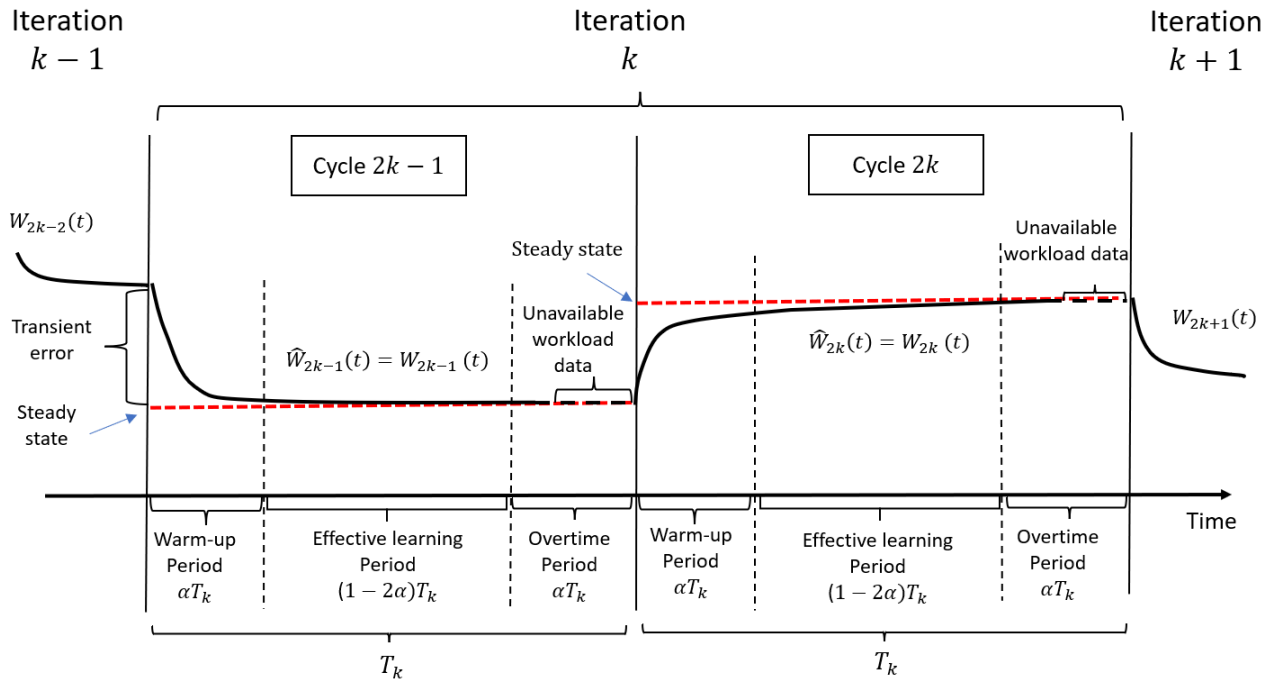


Figure 2 The system dynamics under LiQUAR.

4.2. Computing Gradient Estimator from Queueing Data

We first introduce some notation to describe the system dynamics under LiQUAR and the queueing data generated by LiQUAR. For $l \in \{2k-1, 2k\}$, let $W_l(t)$ be the present workload at time $t \in [0, T_k]$ in cycle l. By definition, we have $W_{l+1}(0) = W_l(T_k)$ for all $l \geq 1$. We assume that the system starts empty, i.e., $W_1(0) = 0$. At the beginning of each cycle l, the control parameter is updated to (μ_l, p_l). The customers arrive in cycle l according to a Poisson process $N_l(t)$ with rate $\lambda(p_l)$, $0 \leq t \leq T_k$. Let $\{V_i^l : i = 1, 2, ..., N_l\}$ be a sequence of I.I.D. random variables denoting customers' individual workloads, where $N_l = N_l(T_k)$ is the total number of customer arrival in cycle l. Then, the dynamics of the workload process $W_l(t)$ is described by the SDE:

$$W_l(t) = W_l(0) + \sum_{i=1}^{N_l(t)} V_i^l - \mu_l \int_0^t \mathbf{1}(W_l(s) > 0) ds. \tag{6}$$

If the system dynamics is available continuously in time (i.e. $W_l(t)$ was known for all $t \in [0, T_k]$ and $l = 2k-1, 2k$), then a natural estimator for $f(\mu_l, p_l)$ would be

$$\hat{f}(\mu_l, p_l) = \frac{-pN_l}{T_k} + \frac{h_0}{T_k} \int_0^{T_k} W_l(t) dt + c(\mu_l).$$

4.2.1. Retrieving workload data from service and arrival times. We assume that LiQUAR can observe each customer's arrivals in real time, but can only recover the individual workload at the service completion time. This assumption is consistent with real practice in many service systems. For example,

in call center, hospital, etc., customer's individual workload is realized only after the service is completed. Hence, the workload process $W_l(t)$ is not immediately observable at t.

In LiQUAR, we approximate $W_l(t)$ by $\hat{W}_l(t)$ which we elaborate below. For given $l \geq 1$ and $t \in [0, T_k]$, if all customers arriving by time t can finish service by the end of cycle l, then all of their service times are realized, so we can recover $W_l(t)$ from the arrival times and service times of these customers using (6). Since customers are served under FIFO, it is straightforward to see that this happens if and only if $W_l(t) \leq \mu_l(T_k - t)$, i.e., the workload at time t is completely processed by T_k. Hence, we define the approximate workload as

$$\hat{W}_l(t) = \begin{cases} W_l(t), & \text{if } W_l(t) \leq \mu_l(T_k - t) \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

As illustrated in Figure 2, to reduce approximation error incurred by delayed observations of service times, we discard the $\hat{W}_l(t)$ data for $t \in ((1-\alpha)T_k, T_k]$; we call the subinterval $((1-\alpha)T_k, T_k]$ the overtime period in cycle l. The following Proposition 1 ensures that the approximation error $|\hat{W}_l(t) - W_l(t)|$ incurred by delayed observation vanishes exponentially fast as length of the overtime period increases. This result will be used in Section 5 to bound the estimation errors of the FD gradient estimator H_k.

PROPOSITION 1 **(Bound on Error of Delayed Observation)**. *Under Assumptions 1 and 2, there exist some constants M and $\theta_0 > 0$ such that, for all $l \geq 1$ and $0 \leq t \leq T_k$,*

$$\mathbb{E}[|\hat{W}_l(t) - W_l(t)|] \leq \exp(-\theta_0 \underline{\mu}/2 \cdot (T_k - t))M.$$

Roughly speaking, M is the moment bound under the busiest traffic intensity, and θ_0 is a small number depending on θ in Assumption 2. The existence of them are shown in the Lemma EC.9 in Section EC.2.2.

4.2.2. Computing the gradient estimator. As illustrated in Figure 2, we also discard the data at the beginning of each cycle (i.e., $\hat{W}_l(t)$ for $t \in [0, \alpha T_k]$ in cycle l) in order to reduce the bias due to transient queueing dynamics incurred by the changes of the control parameters. We call $[0, \alpha T_k]$ the warm-up period of cycle l. Thus, we give the following system performance estimator under control x_l, $l \in \{2k-1, 2k\}$:

$$\hat{f}^G(\mu_l, p_l) = \frac{-pN_l}{T_k} + \frac{h_0}{(1-2\alpha)T_k} \int_{\alpha T_k}^{(1-\alpha)T_k} \hat{W}_l(t)dt + c(\mu_l), \tag{8}$$

and the corresponding FD gradient estimator

$$\boldsymbol{H}_k = \frac{\boldsymbol{Z}_k \cdot (\hat{f}^G(\mu_{2k}, p_{2k}) - \hat{f}^G(\mu_{2k-1}, p_{2k-1}))}{\delta_k}. \tag{9}$$

Unlike standard zero-order methods used in offline optimization problems, our data is generated through online interactions with the real system. Consequently, we must carefully tune our algorithm parameters to control the variance of H_k, as techniques like common random numbers can not be used to achieve variance reduction for $\hat{f}^G(\mu_{2k}, p_{2k}) - \hat{f}^G(\mu_{2k-1}, p_{2k-1})$.

Algorithm 1: LiQUAR

Input: number of iterations L;

parameters $0 < \alpha < 1$, and T_k, η_k, δ_k for $k = 1, 2, .., L$;

initial value $\bar{x}_1 = (\bar{\mu}_1, \bar{p}_1)$, $W_1(0) = 0$;

for $k = 1, 2, ..., L$ **do**

Randomly draw $\boldsymbol{Z}_k \in \{(0,2),(2,0)\}$;

Run Cycle $2k-1$**:** Run the system for T_k units of time under control parameter

$$\boldsymbol{x}_{2k-1} = \bar{\boldsymbol{x}}_k - \delta_k \boldsymbol{Z}_k/2 = (\bar{\mu}_k, \bar{p}_k) - \delta_k \boldsymbol{Z}_k/2,$$

Run Cycle $2k$**:** Run the system for T_k units of time under control parameter

$$\boldsymbol{x}_{2k} = \bar{\boldsymbol{x}}_k + \delta_k \boldsymbol{Z}_k/2 = (\bar{\mu}_k, \bar{p}_k) + \delta_k \boldsymbol{Z}_k/2,$$

Compute FD gradient estimator:

$$\boldsymbol{H}_k = \frac{\boldsymbol{Z}_k}{\delta_k}\left[\frac{h_0}{(1-2\alpha)T_k}\int_{\alpha T_k}^{(1-\alpha)T_k}\left(\hat{W}_{2k}(t) - \hat{W}_{2k-1}(t)\right)dt - \frac{p_{2k}N_{2k} - p_{2k-1}N_{2k-1}}{T_k} + c(\mu_{2k}) - c(\mu_{2k-1})\right]$$

where $\hat{W}_l(\cdot)$ is an approximate of $W_l(\cdot)$ as specified in (7).

Update $\bar{\boldsymbol{x}}_{k+1} = \Pi_{\mathcal{B}}(\bar{\boldsymbol{x}}_k - \eta_k \boldsymbol{H}_k)$.

end

The psuedo code of LiQUAR is given in Algorithm 1. To complete the design of LiQUAR algorithm, we still need to specify the hyperparameters T_k, η_k, δ_k for $k \geq 1$. We seek to optimize these hyperparameters in Section 5 to achieve minimized regret bound.

REMARK 2 (INTEGRATING QUEUEING FEATURES INTO LIQUAR). For LiQUAR to effectively address our queueing control problem, its design should be well informed by the features and structures inherent to the underlying queueing system. First, we utilize workload data to construct the gradient estimator, moving beyond traditional reliance on arrival and service time data. This shift is particularly advantageous in heavy-traffic scenarios where queueing formulas are highly sensitive to input estimates such as demand and service distributions. To calculate the gradient estimator which involves integrating the workload process with delayed observations, we leverage the fact that the workload process is almost surely piecewise linear. This helps simplify the computation of the gradient estimator. Second, to mitigate transient biases and errors caused by delayed observations in the queueing data, we exclude data from a designated warm-up interval at the start of each cycle and an over-time interval at its end. The lengths of these intervals

are determined by the exponential ergodic rate of the $M/GI/1$ queue. Third, the efficiency of learning algorithms, particularly gradient-based methods, critically depends on the choice of hyperparameters. Leveraging the regret analysis in Section 5, which extensively utilizes queueing properties such as transient bias and autocorrelation in the data, we carefully determine optimal hyperparameters to minimize regret.

5. Convergence Rate and Regret Analysis

In Section 5.1, we establish the rate of convergence for our decision variables (μ_k, p_k) under LiQUAR (Theorem 1). Besides, our analysis illustrate how the estimation errors in the queueing data will propagate to the iteration of (μ_k, p_k) and thus affect the quality of decision making. We follow three steps: First, we quantify the bias and mean square error of the estimated system performance $\hat{f}^G(\mu_l, p_l)$ computed from the queueing data via (8) (Proposition 2). To bound the estimation errors, we need to deal with the transient bias and stochastic variability in the queueing data. Next, using these estimation error bounds, we can determine the accuracy of the FD gradient estimator H_k in terms of the algorithm hyperparameters (Proposition 3). Finally, following the convergence analysis framework of SGD algorithms, we obtain the convergence rate of LiQUAR in terms of the algorithm hyperparameters (Theorem 1). The above three steps together form a quantitative explanation of how the errors are passed on from the queueing data to the learned decisions (whereas there is no such steps in PTO so its performance is much more sensitive to the errors in the data). In addition, the convergence result enables us to obtain the optimal choice of hyperparameters if the goal is to approximate $\boldsymbol{x}^* = (\mu^*, p^*)$ accurately with minimum number of iterations, which is often preferred in simulation-based offline learning settings.

In Section 5.2, we investigate the cost performance of dynamic pricing and capacity sizing decisions made by LiQUAR, via analyzing the total regret, which is the gap between the amount of cost produced by LiQUAR and that by the optimal control $\boldsymbol{x}^*$. Utilizing the convergence rate established by Theorem 1 and a separate analysis on the transient behavior of the system dynamics under LiQUAR (Proposition 4), we obtain a theoretic bound for the total regret of LiQUAR in terms of the algorithm hyperparameters. By simple optimization, we obtain an optimal choice of hyperparameters which leads to a total regret bound of order $O(\sqrt{T}\log(T))$ (Theorem 2), where T is the total amount of time in which the system is operated by LiQUAR .

5.1. Convergence Rate of Decision Variables

As $\bar{\boldsymbol{x}}_k$ evolves according to an SGD iteration in Algorithm 1, its convergence depends largely on how accurate the gradient is approximated by the FD estimator H_k. In the theoretical analysis, the accuracy of $\boldsymbol{H}_k$ is measured by the following two quantities:

$$B_k \equiv \mathbb{E}\left[\|\mathbb{E}[\boldsymbol{H}_k - \nabla f(\bar{\boldsymbol{x}}_k)|\mathcal{F}_k]\|^2\right]^{1/2} \quad \text{and} \quad \mathcal{V}_k \equiv \mathbb{E}[\|\boldsymbol{H}_k\|^2],$$

where $\mathcal{F}_k$ is the σ-algebra including all events in the first $2(k-2)$ cycles and $\|\cdot\|$ is Euclidean norm in $\mathbb{R}^2$. Intuitively, B_k measures the bias of the gradient estimator $\boldsymbol{H}_k$ and $\mathcal{V}_k$ measures its variability.

According to (9), the gradient estimator $\boldsymbol{H}_k$ is computed using the estimated system performance $\hat{f}^G(\mu_{2k}, p_{2k})$ and $\hat{f}^G(\mu_{2k-1}, p_{2k-1})$. So, the accuracy of $\boldsymbol{H}_k$ essentially depends on the estimation errors of the system performance, i.e., how close is $\hat{f}^G(\mu_l, p_l)$ to $f(\mu_l, p_l)$. Note that the control parameters (μ_l, p_l) for $l \in \{2k-1, 2k\}$ are random and dependent on the events in the first $2(k-2)$ cycles. Accordingly, we need to analyze the estimation error of $\hat{f}^G(\mu_l, p_l)$ conditional on the past events, which is also consistent with our definition of B_k. For this purpose, we denote by $\mathcal{G}_l$ the σ-algebra including all events in the first $l-1$ cycles and write $\mathbb{E}_l[\cdot] \equiv \mathbb{E}[\cdot|\mathcal{G}_l]$. The following Proposition 2 establishes bounds on the conditional bias and mean square error of $\hat{f}^G(\mu_l, p_l)$, in terms of the initial workload $W_l(0)$ and the hyperparameter T_k.

PROPOSITION 2 **(Estimation Errors of System Performance)**. *Under Assumptions 1 and 2, for any $T_k > 0$, the bias and mean square error of $\hat{f}^G(\mu_l, p_l)$, conditional on $\mathcal{G}_l$, have the following bounds:*

1. *Bias*

$$\left|\mathbb{E}_l\left[\hat{f}^G(\mu_l, p_l) - f(\mu_l, p_l)\right]\right| \leq \frac{2\exp(-\theta_1 \alpha T_k)}{(1-2\alpha)\theta_1 T_k} \cdot M(M + W_l(0))(\exp(\theta_0 W_l(0)) + M).$$

2. *Mean square error*

$$\mathbb{E}_l[(\hat{f}^G(\mu_l, p_l) - f(\mu_l, p_l))^2] \leq K_M T_k^{-1}(W_l^2(0) + 1)\exp(\theta_0 W_l(0)),$$

where $\theta_1 \equiv \min(\gamma, \theta_0 \underline{\mu}/2)$, and γ and K_M are two positive constants that are independent of $l, T_k, W_l(0), \mu_l$ and p_l.

The proof of Proposition 2 is given in Section EC.1, where the specification of the two constants γ and K_M are given in (EC.9) and (EC.5) respectively. The key step in the proof is to bound the transient bias (from the steady-state distribution) and auto-correlation of the workload process $\{W_l(t) : 0 \leq t \leq T_k\}$, utilizing an ergodicity analysis. This approach can be applied to other queueing models which share similar ergodicity properties, e.g., GI/GI/1 queue and stochastic networks (Blanchet and Chen 2020).

Based on Proposition 2, we establish the following bounds on B_k and $\mathcal{V}_k$ in terms of the algorithm hyperparameters T_k and δ_k.

PROPOSITION 3 **(Bounds for B_k and $\mathcal{V}_k$)**. *Under Assumptions 1 and 2, the bias and variance of the gradient estimator satisfy*

$$B_k = O\left(\delta_k^2 + \delta_k^{-1}\exp(-\theta_1 \alpha T_k)\right), \quad \mathcal{V}_k = O\left(\delta_k^{-2} T_k^{-1} \vee 1\right). \tag{10}$$

Assumption 1 guarantees that the objective function $f(\mu, p)$ in (3) has desired convex structure (see Lemma EC.5 in Section EC.1 for details). Hence, the SGD iteration is guaranteed to converge to its optimal solution x^* as long as the gradient bias B_k and variance $\mathcal{V}_k$ are properly bounded. Utilizing the bounds on B_k and $\mathcal{V}_k$ as given in Proposition 3, we are able to prove the convergence of LiQUAR and obtain an explicit expression of the convergence rate in terms of algorithm hyperparameters.

Theorem 1 (Convergence rate of decision variables) *Suppose Assumption 1 holds. If there exists a constant $\beta \in (0,1]$ such that the following inequalities hold for all k large enough:*

$$\left(1+\frac{1}{k}\right)^{\beta} \le 1+\frac{K_0}{2}\eta_k, \quad B_k \le \frac{K_0}{8}k^{-\beta}, \quad \eta_k \mathcal{V}_k = O(k^{-\beta}). \tag{11}$$

Then, we have

$$\mathbb{E}\left[\|\bar{\boldsymbol{x}}_k - \boldsymbol{x}^*\|^2\right] = O(k^{-\beta}). \tag{12}$$

If, in further, Assumption 2 holds and the algorithm hyperparameters are set as $\eta_k = O(k^{-a})$, $T_k = O(k^b)$, and $\delta_k = O(k^{-c})$ for some constants $a, b, c \in (0,1]$. We have

$$\mathbb{E}\left[\|\bar{\boldsymbol{x}}_k - \boldsymbol{x}^*\|^2\right] = O\left(k^{\max(-a,-a-b+2c,-2c)}\right). \tag{13}$$

REMARK 3 (OPTIMAL CONVERGENCE RATE). According to the bound (13), by minimizing the term $\max(-a, a-b+2c, -2c)$, one can obtain an optimal choice of hyperparameters $\eta_k = O(k^{-1}), T_k = O(k)$ and $\delta_k = O(k^{-1/2})$ under which the decision parameter $\boldsymbol{x}_k$ converges to $\boldsymbol{x}^*$ at a fastest rate of $O(L^{-1})$, in terms of the total number of iterations L. Of course, the above convergence rate analysis does not focus on reducing the total system cost generated through the learning process, which is what we will do in Section 5.2.

5.2. Regret Analysis

Having established the convergence of control parameters under Assumption 1, we next investigate the efficacy of LiQUAR as measured by the cumulative regret which measures the gap between the cost under LiQUAR and that under the optimal control. According to the system dynamics described in Section 4.2, under LiQUAR, the expected cost incurred in cycle l is

$$\rho_l \equiv \mathbb{E}\left[h_0 \int_0^{T_k} W_l(t)dt + c(\mu_l)T_k - p_l N_l\right], \tag{14}$$

where $k = \lceil l/2 \rceil$. The total regret in the first L iterations (each iteration contains two cycles) is

$$R(L) = \sum_{k=1}^{L}\sum_{l=2k-1}^{2k} R_l = \sum_{l=1}^{2L} R_l, \quad \text{with } R_l \equiv \rho_l - T_k f(\mu^*, \rho^*).$$

Our main idea is to separate the total regret $R(L)$ into three parts as

$$\begin{aligned} R(L) = &\sum_{k=1}^{L} \underbrace{\mathbb{E}\left[2T_k(f(\bar{\boldsymbol{x}}_k) - f(\boldsymbol{x}^*))\right]}_{\equiv R_{1k}\text{: regret of suboptimality}} \\ &+ \sum_{k=1}^{L} \underbrace{\mathbb{E}\left[(\rho_{2k-1} - T_k f(\boldsymbol{x}_{2k-1})) + (\rho_{2k} - T_k f(\boldsymbol{x}_{2k}))\right]}_{\equiv R_{2k}\text{: regret of nonstationarity}} \\ &+ \sum_{k=1}^{L} \underbrace{\mathbb{E}\left[T_k(f(\boldsymbol{x}_{2k-1}) + f(\boldsymbol{x}_{2k}) - 2f(\bar{\boldsymbol{x}}_k))\right]}_{\equiv R_{3k}\text{: regret of finite difference}}, \end{aligned} \tag{15}$$

which arise from the errors due to the suboptimal decisions (R_{1k}), the transient system dynamics (R_{2k}), and the estimation of gradient (R_{3k}), respectively. Then we aim to minimize the orders of all three regret terms by selecting the "optimal" algorithm hyperparameters T_k, η_k and δ_k for $k \geq 1$.

Treating R_{1k}, R_{2k}, R_{3k} ***separately.*** Suppose the hyperparameters of LiQUAR are set in the form of $\eta_k = O(k^{-a})$, $T_k = O(k^b)$, and $\delta_k = O(k^{-c})$ for some constants $a, b, c \in (0, 1]$. The first regret term R_{1k} is determined by the convergence rate of control parameter $\bar{x}_k$. By Taylor's expansion, $f(\bar{\boldsymbol{x}}_k) - f(\boldsymbol{x}^*) = O(\|\bar{\boldsymbol{x}}_k - \boldsymbol{x}^*\|_2^2)$, and hence, $R_{1k} = O(T_k \|\bar{\boldsymbol{x}}_k - \boldsymbol{x}^*\|_2^2)$. Following Theorem 1, we have $R_{1k} = O(k^{\max(b-a, b-2c, -a+2c)})$. By the smoothness condition in Assumption 1, we can check that $R_{3k} = O(T_k \delta_k^2) = O(k^{b-2c})$ (Lemma EC.8 in Section EC.1).

The remaining regret analysis will focus on the regret of nonstationarity R_{2k}. Intuitively, it depends on the rate at which the (transient) queueing dynamics converges to its steady state. Applying the same ergodicity analysis as used in the analysis of estimation errors of system performance, we can find a proper bound on the transient bias after the warm-up period, i.e., for $W_l(t)$ with $t \geq \alpha T_k$. Derivation of a desirable bound on the transient bias in the warm-up period, i.e., for $W_l(t)$ with $t \in [0, \alpha T_k]$, is less straightforward. The main idea is based on the two facts that (1) $W_l(t)$, when t is small, is close to the steady-state workload corresponding to (μ_{l-1}, p_{l-1}) and that (2) the steady-state workload corresponding to (μ_{l-1}, p_{l-1}) is close to that of (μ_l, p_l). We formalize the bound on R_{2k} in Proposition 4 below. The complete proof is given in Section EC.1.6.

PROPOSITION 4 **(Regret of Nonstationarity)**. *Suppose Assumptions 1 and 2 hold. If* $T_k > \log(k)/\gamma$ *and there exists some constant* $\xi \in (0, 1]$ *such that* $\max(\eta_k \sqrt{\mathcal{V}_k}, \delta_k) = O(k^{-\xi})$. *Then,*

$$R_{2k} = O\left(k^{-\xi} \log(k)\right). \tag{16}$$

If, in further, the algorithm hyperparameters are set as $\eta_k = O(k^{-a})$, $T_k = O(k^b)$, *and* $\delta_k = O(k^{-c})$ *for some constants* $a, b, c \in (0, 1]$, *we have*

$$R_{2k} = O\left(k^{\max(-a-b/2+c, -a, -c)} \log(k)\right).$$

By summing up the three regret terms, we can conclude that

$$R(L) \leq \sum_{k=1}^{L} C\left(k^{\max(-a-b/2+c, -a, -c)} \log(k) + k^{\max(b-a, b-2c, -a+2c)} + k^{b-2c}\right),$$

for some positive constant C that is large enough. The order of the upper bound on the right hand side reaches its minimum at $(a, b, c) = (1, 1/3, 1/3)$. The corresponding total regret and time elapsed in the first L iterations are, respectively,

$$R(L) = O(L^{2/3} \log(L)) \quad \text{and} \quad T(L) = O(L^{4/3}).$$

As a consequence, we have $R(T) = O(\sqrt{T} \log(T))$.

Theorem 2 (Regret Upper Bound) *Suppose Assumptions 1 and 2 hold. If we choose $\eta_k = c_\eta k^{-1}$ for some $c_\eta > 2/K_0$, $T_k = c_T k^{1/3}$ for some $c_T > 0$ and $\delta_k = c_\delta k^{1/3}$ for some $0 < c_\delta < \sqrt{K_0/32c}$, where c is a smoothness constant given in Lemma EC.4, then the total regret accumulated in the first L rounds by LiQUAR*

$$R(L) = O(L^{2/3}\log(L)) = O(\sqrt{T(L)}\log(T(L))).$$

Here $T(L)$ is the total units of time elapsed in L cycles.

REMARK 4 (ON THE $O(\sqrt{T}\log(T))$ REGRET BOUND). Consider a hypothetical setting in which we are no longer concerned with the transient behavior of the queueing system, i.e., somehow we can directly observe an unbiased and independent sample of the objective function with uniform bounded variance in each iteration. In this case, we know that the Kiefer-Wolfowitz algorithm and its variate provide an effective approach for model-free stochastic optimization (Broadie et al. 2011). According to Broadie et al. (2011), the convergence rate of Kiefer-Wolfowitz algorithm is $\|\bar{\boldsymbol{x}}_k - \boldsymbol{x}^*\|^2 = O(\eta_k/\delta_k^2)$. In addition, the regret of finite difference is $f(\boldsymbol{x}_{2k-1}) + f(\boldsymbol{x}_{2k}) - 2f(\bar{\boldsymbol{x}}_k) = O(\delta_k^2)$. Since $\eta_k/\delta_k^2 + \delta_k^2 \geq 2\sqrt{\eta_k} \geq k^{-1/2}$, we can conclude that the optimal convergence rate in such a hypothetical setting is $O(k^{-1/2})$. This accounts for the $\sqrt{T}$ part of our regret in Theorem 2. Unfortunately, unlike the hypothetical setting, our queueing samples are biased and correlated. Such a complication is due to the nonstationary error at the beginning of cycles which gives rise to the extra $\log(T)$ term in the regret bound; see Proposition 4 for additional discussion of the $\log(T)$ term in our regret. In Section EC.3, we present a theoretic result showing that the lower bound for the regret of suboptimality is $\Omega(\sqrt{T})$.

5.3. LIQUAR in Heavy Traffic

We now evaluate LiQUAR's performance under heavy traffic conditions. To do this, we construct a series of queueing models with traffic intensities that approach the critical threshold of 1, and define the associated profit optimization problems. Our goal is to derive an explicit regret expression as a function of traffic intensity. To achieve this, we will need to consider a simplified model in order for reduced technicalities in our regret analysis. We consider two simplifications: first, we focus on the case of exponential service times; second, we focus on a pricing problem, treating the service rate μ as a constant. We stress that the simplified model still preserve the challenge of dealing with unknown demand (the core aspect of the learning problem).

Consider a sequence of $M/M/1$ systems indexed by the parameter $h > 0$, which is the queueing congestion cost per time unit. They share a common demand curve $\lambda(p)$ and service rate $\mu = 1$. For the h^{th} model, we aim to minimize the objective function below

$$\min_{p \in \mathcal{B}_h} \quad f_h(p) \equiv -p\lambda(p) + \frac{h\lambda(p)}{1-\lambda(p)}, \tag{17}$$

where $\mathcal{B}_h$ will be specified later. When the holding cost h in (17) decreases, the service provider is incentivized to increase service utilization to maximize profit which places the system under the heavy-traffic regime. Below we will formally show that the traffic intensity under the optimal price ρ_h^* converges to 1 as $h \to 0$, specifically, below we will show $1 - \rho_h^* = O(\sqrt{h})$.

We denote by p_h^* as the optimal solution to (17). To explicitly show the relationship between ρ_h^* and h, we impose the following assumptions on the demand curve.

ASSUMPTION 3 **(Demand Curve)**. *We assume that the arrival rate function $\lambda(p)$ satisfies the following conditions:*

1. *The demand function is non-increasing and twice-differentiable.*
2. *The function $r(p) = p\lambda(p)$ is strictly concave.*
3. *The demand function is elastic in the feasible regions: $-\frac{\lambda'(p)}{\lambda(p)} \cdot p > 1$ for $p \in \mathcal{B}_h$.*

The third technical condition is commonly used in the literature of revenue management in queues; see for example, Assumption 1 in Maglaras and Zeevi (2003). Essentially, this condition assumes that customers are price sensitive in the feasible region.

Denote p_0 as the price that makes the system critically loaded, i.e., $\lambda(p_0) = 1$. Before giving our regret bound, we first characterize the optimal pricing decision p_h^* as a function of h and relate the traffic intensity ρ_h^* to h.

PROPOSITION 5. *Under Assumption 3, we have the optimal price*

$$p_h^* \equiv \arg\min f_h(p) = p_0 + \sqrt{h} \cdot \sqrt{\frac{1}{(1 + p_0\lambda'(p_0))\lambda'(p_0)}} + o(\sqrt{h}),$$

and the corresponding optimal service excess

$$1 - \rho_h^* = \sqrt{h} \cdot \sqrt{\frac{-\lambda'(p_0)}{1 + p_0\lambda'(p_0)}} + o(\sqrt{h}) = O(\sqrt{h}).$$

To investigate LiQUAR's performance in heavy traffic, we consider $\mathcal{B}_h$ which asymptotically operates the system in heavy traffic as $h \to 0$. Following Proposition 5, we let

$$p_h^* \in \left[\, p_0 + c_1\sqrt{h}, p_0 + c_2\sqrt{h} \,\right] \equiv \mathcal{B}_h,$$

where the constants c_1 and c_2 satisfy $0 < c_1 < c_0 \equiv 1/\sqrt{\lambda'(p_0)(1 + p_0\lambda'(p_0))} < c_2$. Note that as $\rho_h^* \to 1$ (or equivalently, $h \to 0$), the queueing system takes a longer time to converge to its steady-state. Therefore, as $h \to 0$, we need to increase the length of the learning cycle. We operate the h^{th} model under LiQUAR with a total time duration of $T^h \equiv T_0/h$ units where T_0 is a positive constant independent of h. We evaluate our performance using the regret

$$R^h(T_0) \equiv R(T^h).$$

In what follows, we report the theoretical regret bound as a function of the traffic intensity.

Theorem 3 (Regret Bound in Heavy Traffic) *For the h^{th} system operated under LiQUAR to minimize* (17)*, when the hyper-parameters are chosen as*

$$T_k^h = c_T h^{-1} k^{1/3}, \quad \delta_k^h = c_\delta \sqrt{h} k^{-1/3}, \quad \eta_k^h = c_\eta \sqrt{h} k^{-1}, \tag{18}$$

where the constants c_T, c_δ and c_η are independent of h, then the regret is given by

$$R^h(T_0) \le C\sqrt{h^{-1} T_0 \log T_0} = O(\sqrt{T_0 \log T_0}/(1-\rho_h^*)), \tag{19}$$

where C is a positive constant independent of h.

The result in Theorem 3 refines that in Theorem 2 by emphasizing how the regret depends on the system's traffic intensity. In Section 7, we also conduct heavy-traffic analysis for a nonparametric PTO framework; and we compare the performance under both methods in our heavy-traffic regime both theoretically and numerically. In addition, in Section EC.3, we show that under our heavy-traffic scheme, for **any** algorithm, the lower bound for the regret of suboptimality is $\Omega(\sqrt{T_0/h}) = \Omega\left(\sqrt{T_0}/(1-\rho_h^*)\right)$.

6. Numerical Experiments

We provide engineering confirmations of the effectiveness of LiQUAR by conducting a series of numerical experiments. We will use simulated data to visualize the convergence of LiQUAR, estimate the regret curves and benchmark them with our theoretical bounds. In Section 6.1, we evaluate the performance of LiQUAR using an $M/M/1$ base example with logit demand functions. In Section 6.2, we discuss how to fine-tune the algorithm's hyperparameters including T_k and η_k. In Section 6.3, we generalize LiQUAR to $GI/GI/1$ queues with non-Poisson arrivals and evaluate its performance.

6.1. An $M/M/1$ Base Example

Our base model is an $M/M/1$ queues having Poisson arrivals with rate $\lambda(p)$ and exponential service times with rate μ. We consider a logistic demand function (Besbes and Zeevi 2015)

$$\lambda(p) = M_0 \cdot \frac{\exp(a-bp)}{1+\exp(a-bp)}, \tag{20}$$

with $M_0 = 10, a = 4.1, b = 1$ and a linear staffing cost function

$$c(\mu) = c_0 \mu. \tag{21}$$

The demand function is shown in the top left panel in Figure 4. Then, the service provider's profit optimization problem (2) reduces to

$$\max_{\mu,p} \left\{ p\lambda(p) - h_0 \frac{\lambda(p)/\mu}{1-\lambda(p)/\mu} - c_0\mu \right\}. \tag{22}$$

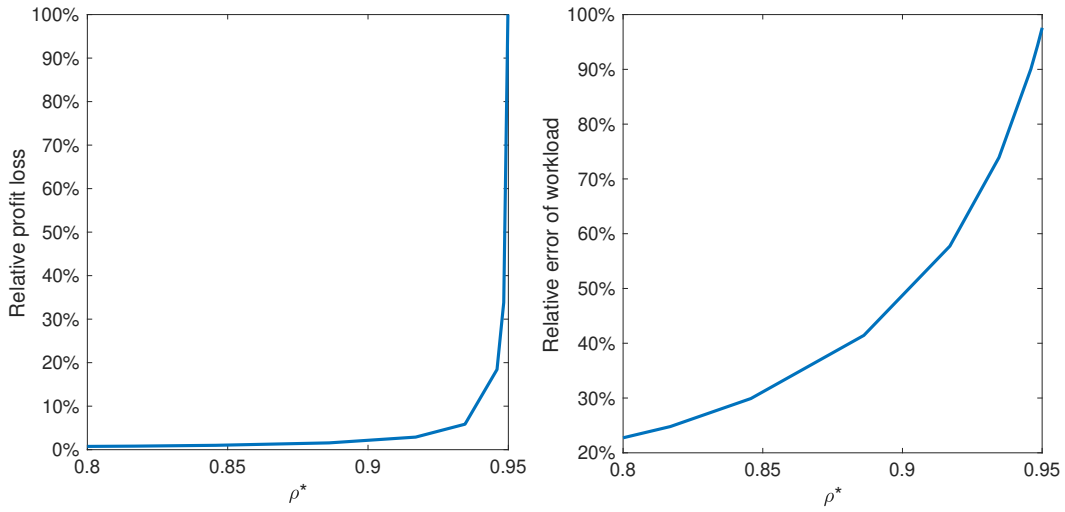


Figure 3 Relative profit loss (left) and workload error (right) for the $M/M/1$ example with $M_0 = 10, a = 4.1$, and $b = 1$ and linear staffing cost $c(\mu) = \mu$.

6.1.1. Performance sensitivity to parameter errors without learning We first illustrate how the parameter estimation error impacts the performance. Here we assume the service provider does not know the true value of $\lambda(p)$ but rather make decisions based on an estimated arrival rate $\hat{\lambda}_\epsilon(p) \equiv (1-\epsilon\%)\lambda(p)$, where ϵ is the percentage estimator error. Let $(\hat{\mu}_\epsilon, \hat{p}_\epsilon)$ and (μ^*, p^*) be the solutions under the estimated $\hat{\lambda}_\epsilon$ and the true value of λ. We next compute the relative profit loss due to the misspecification of the demand function $\left(\mathcal{P}(\mu^*, p^*) - \mathcal{P}(\hat{\mu}_\epsilon, \hat{p}_\epsilon)\right)/\mathcal{P}(\mu^*, p^*)$, which is the relative difference between profit under the miscalculated solutions using the believed $\hat{\lambda}_\epsilon$ and the true optimal profit under λ.

Let $\rho^* \equiv \lambda(p^*)/\mu^*$ be the traffic intensity under the true optimal solution. We are able to impact the value of ρ^* by varying the queueing penalty coefficient h_0. We provide an illustration in Figure 3 with $\epsilon = 5$. From the left panel of Figure 3, we can see that as ρ^* increases, the model fidelity becomes more sensitive to the misspecification error in the demand rate and the relative loss of profit grows dramatically as ρ^* goes closer to 1. This effect arises from the fact that the error predicted workload is extremely sensitive to that in the arrival rate and is disproportionally amplified by the PK formula when the system is in heavy traffic (see panel (b) for the relative error of the workload). Later in Section 7, we will conduct a careful comparison to the PTO method where we will compute the PTO regret including profit losses in both the prediction and optimization steps.

6.1.2. Performance of LiQUAR Using the explicit forms of (22), we first numerically obtain the exact optimal solution (μ^*, p^*) and the maximum profit $\mathcal{P}(\mu^*, p^*)$ which will serve as benchmarks for LiQUAR. Taking $h_0 = 1$ and $c_0 = 1$ yields $(\mu^*, p^*) = (8.18, 3.79)$, and the corresponding profit plot is shown in the top right panel of Figure 4. To test the criticality of condition (b) in Assumption 1, we implement LiQUAR

when condition (b) does not hold. For this purpose, we set $\mathcal{B} = [6.5, 10] \times [3.5, 7]$, in which the objective (3) is not always convex, let alone the condition (b) of Assumption 1 (top right and middle right panel of Figure 4).

Then we implement LiQUAR without exploiting the specific knowledge of the exponential service distribution or the form of $\lambda(p)$. In light of Theorem 2, we set the hyperparameters $\eta_k = 4k^{-1}, \delta_k = \min(0.1, 0.5k^{-1/3})$, $T_k = 200k^{1/3}$ and $\alpha = 0.1$. From Figure 4, we observe that the pair (μ_k, p_k), despite some stochastic fluctuations, converges to the optimal decision rapidly. The regret is estimated by averaging 100 sample paths and showed in the bottom left panel of Figure 4. To better relate the regret curve to its theoretical bounds as established in Theorem 2, we also draw the logarithm of regret as a function of the logarithm of the total time; we fit the log-log curve to a straight line (bottom right panel of Figure 4) so that the slope of the line may be used to quantify the theoretic order of regret: the fitted slope (0.38) is less than its theoretical upper bound (0.5). Such "overperformance" is not too surprising because the theoretic regret bound is established based on a worst-case analysis. In summary, our numerical experiment shows that the technical condition (b) in Assumption 1 does not seem to be too restrictive.

6.2. Tuning the hyperparameters for LiQUAR

Next, we test the performance of LiQUAR on the base $M/M/1$ example under different hyperparameters. We also provide some general guidelines on the choices of hyperparameters when applying LiQUAR in practice.

6.2.1. Step lengths η_k and δ_k. In the first experiment, we tune the step length η_k and δ_k jointly within the following form:

$$\eta_k = c \cdot 4k^{-1}, \quad \text{and} \quad \delta_k = \min(0.1, c \cdot 0.5k^{-1/3}). \tag{23}$$

To understand the rationale of this form, note that these parameters give critical control to the variance of the gradient estimator. We aim to keep the variance of the term $\eta_k H_k$ at the same level in the gradient descent update

$$\boldsymbol{x}_{k+1} = \Pi_{\mathcal{B}}(\boldsymbol{x}_k - \eta_k \boldsymbol{H}_k), \qquad \text{with} \quad \eta_k \boldsymbol{H}_k = \eta_k \frac{\hat{f}(\boldsymbol{x}_k + \delta_k/2 \cdot \boldsymbol{Z}_k) - \hat{f}(\boldsymbol{x}_k - \delta_k/2 \cdot \boldsymbol{Z}_k)}{\delta_k}.$$

In this experiment, we let $c \in \{0.6, 1.0, 1.2\}$ and fix $T_k = 200k^{1/3}$ and $\alpha = 0.1$. For each case, the regret curve is estimated by 100 independent runs for $L = 1000$ iterations. The regret and its linear fit are reported in Figure 5. As shown in the right panel of Figure 5, the regret of LiQUAR has slopes of the linear regret fit close to 0.5 in all three cases. Comparing the two curves with $c = 0.6$ and $c = 1.2$ (left panel of Figure 5), we find that the larger value of c immediately accumulates a large regret in the early stages but performs better in the later iterations. This observation may be explained by the trade-off between the level of exploration

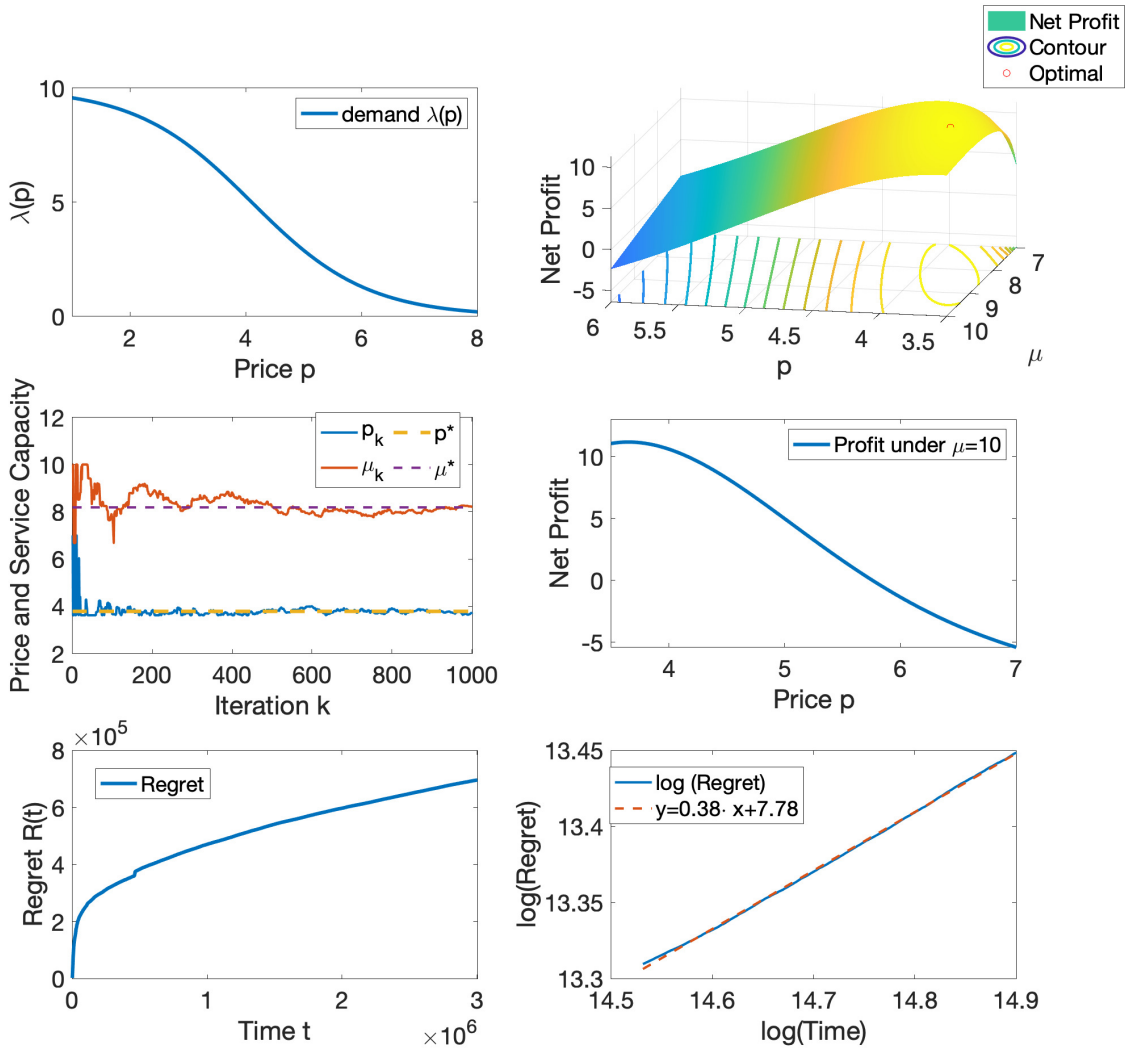


Figure 4 Joint pricing and staffing in the $M/M/1$ logistic demand base example with $\eta_k = 4k^{-1}, \delta_k = \min(0.1, 0.5k^{-1/3})$, $T_k = 200k^{1/3}$, $p_0 = 5$, $\mu_0 = 10$ and $\alpha = 0.1$: (i) Demand function $\lambda(p)$ (top left panel); (ii) net profit function (top right panel); (iii) sample trajectories of decision parameters (middle left); (iv) One dimensional net profit function when $\mu = 10$; (v) average regret curve estimated by 100 independent runs (bottom left); (vi) a linear fit to the regret curve in logarithm scale.

and learning rate of LiQUAR. In particular, a larger value of c leads to larger values of η_k and δ_k, which allows more aggressive exploration and higher learning rate.

Although the tuning of c will not affect the convergence of asymptotic regret of the algorithm, it may be critical to decision making in a finite-time period. For example, a myopic decision maker who prefers good system performance in a short term should consider small values of c, while a far-sighted decision maker who values more the long-term performance should adopt a larger c.

6.2.2. Cycle length T_k. In this experiment, we test the impact of T_k on the performance of LiQUAR. We again use the $M/M/1$ base example. The step-length hyperparameters are set to $\eta_k = 4k^{-1}$ and $\delta_k =$

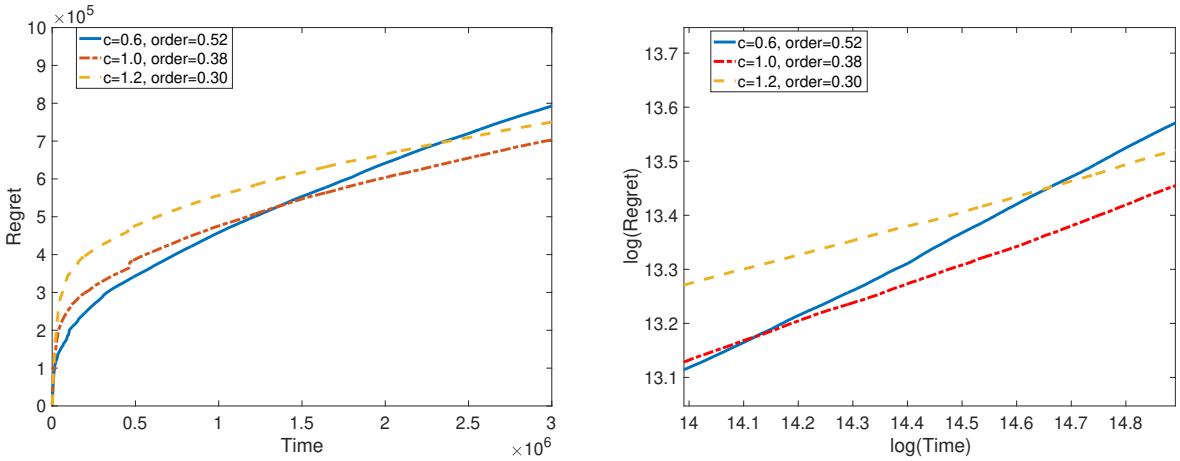


Figure 5 Regret under different $c \in \{0.6, 1.0, 1.2\}$: (i) average regret from 100 independent runs (left panel); (ii) regret curve in logarithm scale, with $T_k = 200k^{1/3}$, $\eta_k = c \cdot 4k^{-1}$, $\delta_k = \min(0.1, c \cdot 0.5k^{-1/3})$ and $\alpha = 0.1$.

$\min(0.1, 0.5k^{-1/3})$. We choose different values of T_k in the form of

$$T_k = T \cdot k^{1/3},\ T \in \{40, 200, 360\}.$$

For different values of T, iteration numbers L_T are chosen to maintain equal total running times for LiQUAR. In particular, we choose $L_T = \left\lceil 1000 \cdot (200/T)^{3/4} \right\rceil$. Results of all above-mentioned cases are reported in Figure 6.

The right panel of Figure 6 shows that the slope of the linear fits all below 0.5. According to the three regret curves in the left panel, we can see how different values of T impact the exploration-exploitation trade-off: a larger value of T, e.g., $T = 360$, yields a higher regret in the early iterations but ensures a flatter curve in the later iterations. This is essentially due to the trade-off between the learning cost and the quality of the gradient estimator. A larger cycle length T_k guarantees a high-quality gradient estimators as more data are generated and used in each iteration which help reduce the gradient estimator's transient bias and variance. On the other hand, it demands that the system be operated for a longer time under suboptimal control policies, especially in the early iterations. The above analysis provides the following guidance for choosing T in practice: A smaller T is preferred if the service provider's goal is to make the most efficient use of the data in order to make timely adjustment on the control policy. This guarantees good performance in short term (the philosophy here is similar to that of the temporal-difference method with a small updating cycle). But if the decision maker is more patient and aims for good long-term performance, he/she should select a larger T which ensures that the decision update is indeed meaningful with sufficient data (this idea is similar to the Monte-Carlo method with batch updates).

6.3. Queues with non-Poisson arrivals

In this section, we consider the more general $GI/GI/1$ model having arrivals according to a renewal process. Similar to the service times, we model the interarrival times using scaled random variables

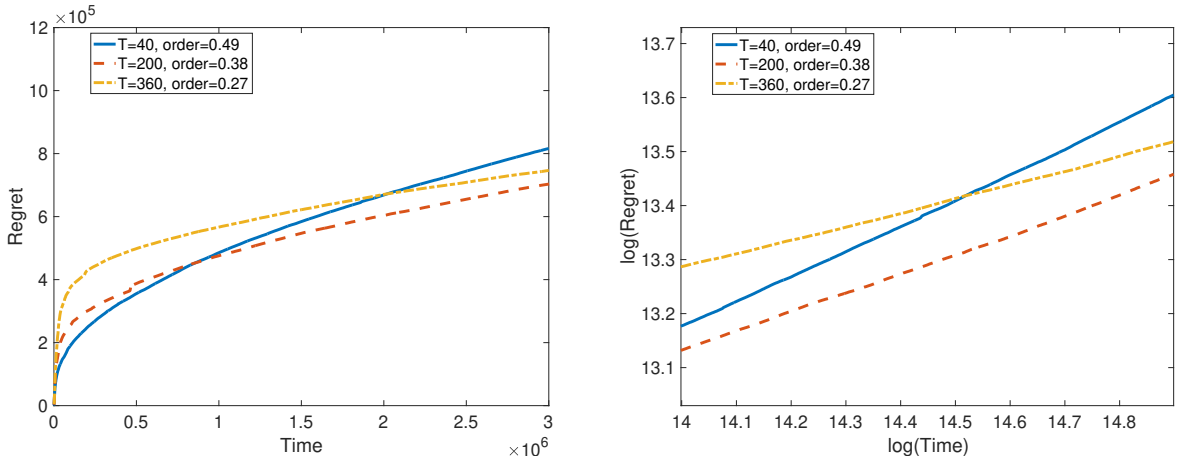


Figure 6 Regret under different $T \in \{40, 200, 360\}$: (i) average regret from 100 independent runs (left panel); (ii) regret curve in logarithm scale, with $T_k = T \cdot k^{1/3}$, $\eta_k = 4k^{-1}, \delta_k = \min(0.1, 0.5k^{-1/3})$ and $\alpha = 0.1$.

$U_1/\lambda(p), U_2/\lambda(p), \ldots$ for a given p, with $U_1, U_2, \ldots$ being a sequence of I.I.D. random variables with $\mathbb{E}[U_n] = 1$.

The PTO framework is not applicable here because $\mathbb{E}[W_\infty]$ does not have a closed-form solution in the $GI/GI/1$ setting. This provides additional motivations for our online learning approach. On the other hand, generalizing the theoretical regret analysis rigorously from $M/GI/1$ to $GI/GI/1$ is by no means a straightforward extension. A key step in our analysis is to give a proper bound for the bias of the gradient estimator. When the arrival process is Poisson, the memoryless property ensures that N_l/T_k in (8) is an unbiased estimator for the arrival rate. For renewal arrivals, the arrival rate bias has an order $O(1/T_k) = O(k^{-1/3})$ (see for example Lorden's inequality (Asmussen 2003, Section V, Proposition 6.2)), which contributes to the bias of the FD with an order of $O(1/T_k\delta_k) = O(1)$. This contradicts Theorem 1 which requires $B_k = O(k^{-1})$. This part of the analysis requires additional investigations (in order to establish a more delicate bias bound). We leave the careful regret analysis of $GI/GI/1$ to future research.

Nevertheless, from the engineering perspective, the increased bias due to the GI arrival process may not be too significant (note that the theoretical bias bound is obtained from a worst-case analysis). We next conduct some preliminary numerical experiments to test the performance of LiQUAR under GI arrivals. We consider an $E_2/M/1$ queue example having Erlang-2 interarrival times with mean $1/\lambda(p)$ and exponential service times with rate μ to illustrate the performance of LiQUAR in $GI/GI/1$'s case. We continue to consider the logit demand function (20) with $M = 10, a = 4.1, b = 1$ and linear staffing cost function (21). Unlike the $M/GI/1$ case where the PK formula provides a closed-form formula for the steady-state waiting time, here we numerically compute the optimal solution (μ^*, p^*) by using matrix geometric method (note that the state process of $E_2/M/1$ is quasi-birth-and-death process). Letting $h_0 = c_0 = 1$ yields the optimal decision $(\mu^*, p^*) = (7.78, 3.75)$. We implement LiQUAR with hyperparameters $\eta_k = 4k^{-1}, \delta_k =$

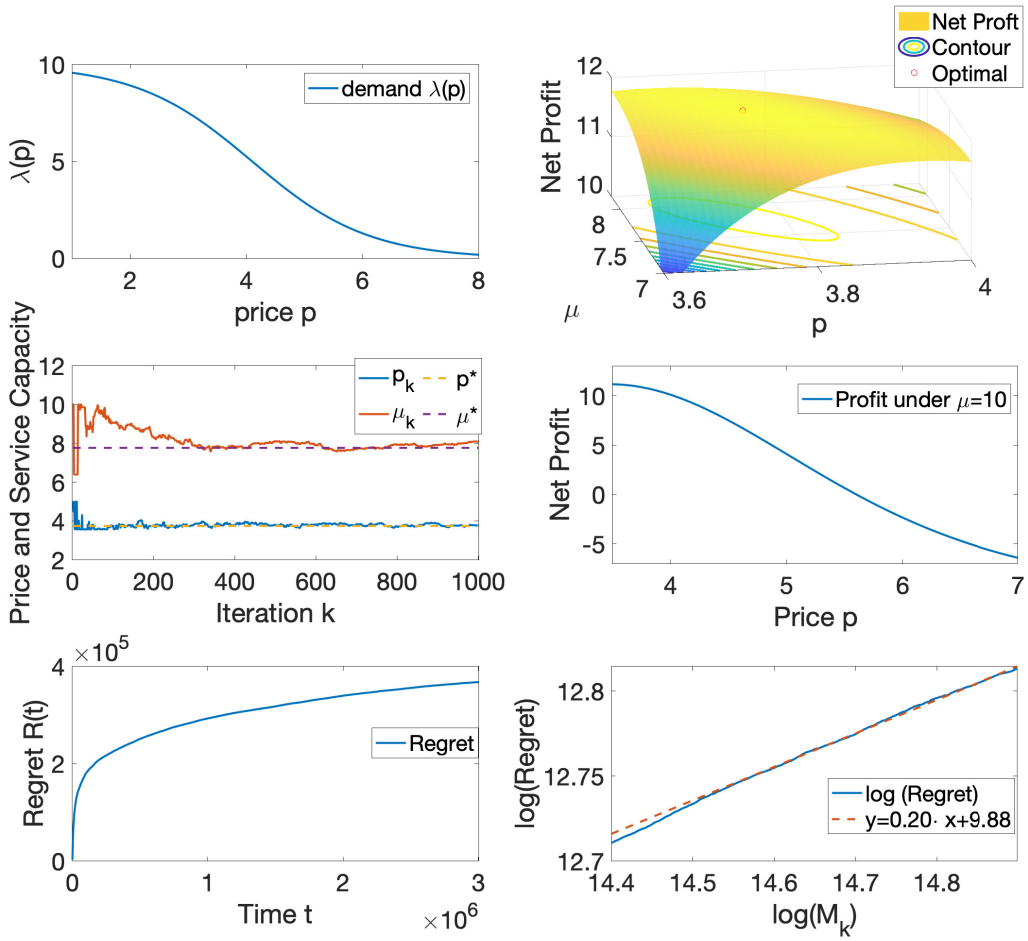


Figure 7 Joint pricing and staffing in the $E_2/M/1$ queue with $\eta_k = 4k^{-1}, \delta_k = \min(0.1, 0.5k^{-1/3})$, $T_k = 200k^{1/3}$, $\alpha = 0.1$, $p_0 = 5$ and $\mu_0 = 10$.

$\min(0.1, 0.5k^{-1/3})$, $T_k = 200k^{1/3}$, and $\alpha = 0.1$. Figure 7, as an analog to Figure 4, shows that the refined LiQUAR continues to be effective, exhibiting a rapid converge to the optimal decision and a slowly growing regret curve (bottom left panel of Figure 7). Despite of the good performance of the above $E_2/M/1$ example, we acknowledge that this is only a preliminary step, and the full investigation of the $GI/GI/1$ case requires careful theoretical analysis and comprehensive numerical studies.

7. LiQUAR vs. PTO

In this section, we contrast the performance of LiQUAR to that of the conventional PTO method. In principle, a PTO algorithm undergoes two phases: (i) "prediction" of the model (e.g., estimation of the demand function and service distribution) and (ii) "optimization" of the decision variables (e.g., setting the optimal service price and capacity). Taking the demand function $\lambda(\cdot)$ as an example, PTO relies on the "prediction" phase to provide a good estimate $\widehat{\lambda}(p)$, which will next be fed to the "optimization" phase for generating desired control policies. In case no historical data is available so that the "prediction" completely relies

on the newly generated data, one needs to learn the unknown demand curve $\lambda(p)$ by significantly experimenting the decision parameters in real time in order to generate sufficient demand data that can be used to obtain an accurate $\widehat{\lambda}(p)$.

We begin by establishing theoretical results to compare the performance of LiQUAR and PTO within a heavy-traffic framework. These findings are then supplemented by numerical experiments to give engineering confirmations.

7.1. LiQUAR vs. PTO in heavy traffic

In this section, we compare the regret bounds of LiQUAR and PTO when the system is in heavy traffic. Consider the sequence of h-indexed systems described in Section 5.3, we now intend to use PTO to find the optimal value of (17) within $\mathcal{B}_h$ and measure its performance by computing the regret $R^h(T_0)$ at time $T^h = T_0/h$. We consider the following PTO algorithm (Besbes and Zeevi 2009):

- **Input:** Total running time T^h, number of testing points κ^h, time of prediction t_0^h
- **Step 1. Prediction:**
  a. Organize $\mathcal{B}_h$ into κ^h evenly spaced grids and distribute the testing points in all grids.
  b. For each $i = 1, \cdots, \kappa^h$, operate the system at i^{th} under the testing point p_i for t_0^h/κ^h units of time, and approximate the demand curve $\hat{\lambda}(p_i)$ by the time-averaged arrival rate.
- **Step 2. Optimization:**
  a. Calculate $\hat{f}(p_i)$ using $\hat{\lambda}(p_i)$ in the PK formula.
  b. Operate the system under $\hat{p}^* = \arg\max_i \hat{f}(p_i)$ for the rest of time horizon.

We next give a regret bound for the above PTO method under the heavy-traffic learning scheme.

PROPOSITION 6 **(PTO in heavy traffic)**. *Under Assumption 3, in the h^{th} system, PTO with hyperparameters κ^h, t_0^h yields the regret bound:*

$$R^h(T_0) \le C\sqrt{h}t_0^h + CT_0 \cdot \frac{\sqrt{\kappa^h \log T_0/h}}{h\sqrt{t_0^h}} + C\frac{\sqrt{h}T_0}{(\kappa^h)^2}. \tag{24}$$

In addition, if we select $t_0^h = O\left(\frac{T_0^{5/7}(\log T^h)^{2/7}}{h}\right)$ and $\kappa = O\left(\frac{T_0^{1/7}}{\log T^h}\right)$, then the PTO regret bound can be minimized as below:

$$R^h(T_0) \le C \cdot \frac{T_0^{5/7}\log(T_0/h)^{2/7}}{\sqrt{h}} = \tilde{O}\left(\frac{T_0^{5/7}}{1-\rho_h^*}\right),$$

with C being some constant independent with h and $1-\rho_h^*$.

REMARK 5 (LIQUAR VS. PTO IN HEAVY TRAFFIC). Compared to the original regret bounds presented in Besbes and Zeevi (2009), we have re-optimized the hyperparameters and derived reduced regret bounds under Assumption 1, enabling a fair comparison between LiQUAR and PTO. According to Theorem 3 and Proposition 6, the regret bounds for both LiQUAR and PTO share a dependence on the traffic intensity in

the order of $1/(1-\rho_h^*)$. However, LiQUAR exhibits a slower growth rate with respect to the time horizon, scaling as $\sqrt{T_0}$. This implies that over the long run, LiQUAR achieves a smaller regret bound than PTO. Furthermore, as the two terms involving T_0 and ρ^* interact multiplicatively in the regret bound, the factor $1/(1-\rho^*)$ amplifies LiQUAR's advantage over PTO in heavy-traffic conditions, i.e., as $\rho^* \to 1$. This trend is further validated by the numerical results illustrated in Figure 8.

Next, we numerically investigate the performance of LiQUAR and PTO for a one-dimensional pricing problem under heavy traffic. We consider an $M/M/1$ system having exponential demand function

$$\lambda(p) = \exp(a - bp),$$

with $a = 1 + \log 2$, $b = 1$ and exponential service time distributions. Following the settings in Theorem 3, we consider a sequence of objective functions in (22) indexed by h. We keep $\mu = 1$ held fixed and allow $h \in \{0.1, 0.01, 0.005, 0.001\}$ to account for different values of the traffic intensity. As $h \to 0$, the feasible region $\mathcal{B}_h$ takes the form

$$\mathcal{B}_h = \left[\, p_0 + c_1\sqrt{h}, p_0 + c_2\sqrt{h} \,\right],$$

with $c_1 = 0.6 \cdot c_0, c_2 = 5c_0$ and $c_0 = 1/\sqrt{\lambda'(p_0)(1 + p^*\lambda'(p_0))} = 1.20$.

Then, we apply LiQUAR and PTO in all the instances with different h. For LiQUAR, following Theorem 3, we choose $\eta_k^h = 4\sqrt{h}k^{-1}, \delta_k^h = 2\sqrt{h}k^{-1/3}$ with $T_k^h = h^{-1}k^{1/3}$ for 500 iterations. To make a fair comparison, we pick an equal runtime for LiQUAR and PTO with $T_0^h = \sum_{k=1}^{500} T_k^h$ and $T_0 = h \cdot T_0^h$ for all h. PTO's hyperparameters are chosen as $t_0^h = t \cdot \frac{T_0^{5/7}\log(T_0/h)^{2/7}}{\sqrt{h}}$ and $\kappa^h = h \cdot (t_0/\log T^h)^{1/5}$ with $t \in \{0.1, 0.2, 0.5\}$.

In Figure 8, we report the scaled regret curves of both methods under different holding costs h where each regret curve is estimated by the average of 100 independent replications. To understand how the regret is influenced by the heavy-traffic scaling factor h, we scale time by $h \cdot T^h = T_0$ and scale the regret by $\sqrt{h} \cdot R^h(T_0)$ (as in Theorem 3 and Proposition 6). From Figure 8, we confirm that LiQUAR significantly outperforms PTO in all heavy-traffic scenarios.

7.2. LiQUAR vs. objective-informed PTO

In this section, we compare LiQUAR to an advanced parametric PTO framework, where the prediction step incorporates information from the downstream objective function. We refer to this approach as *objective-informed PTO* (oiPTO). In oiPTO, it is assumed that the decision-maker knows the parametric form of the demand function, $\lambda(\cdot;\boldsymbol{\beta})$, with parameters $\boldsymbol{\beta}$ that are initially unknown. During the prediction phase, oiPTO estimates $\boldsymbol{\beta}$ by leveraging the structure of the downstream objective function (see (25)). In the optimization phase, oiPTO uses the demand function with the estimated parameters, denoted as $\boldsymbol{\beta}_{oi}$, to compute the optimal decisions.

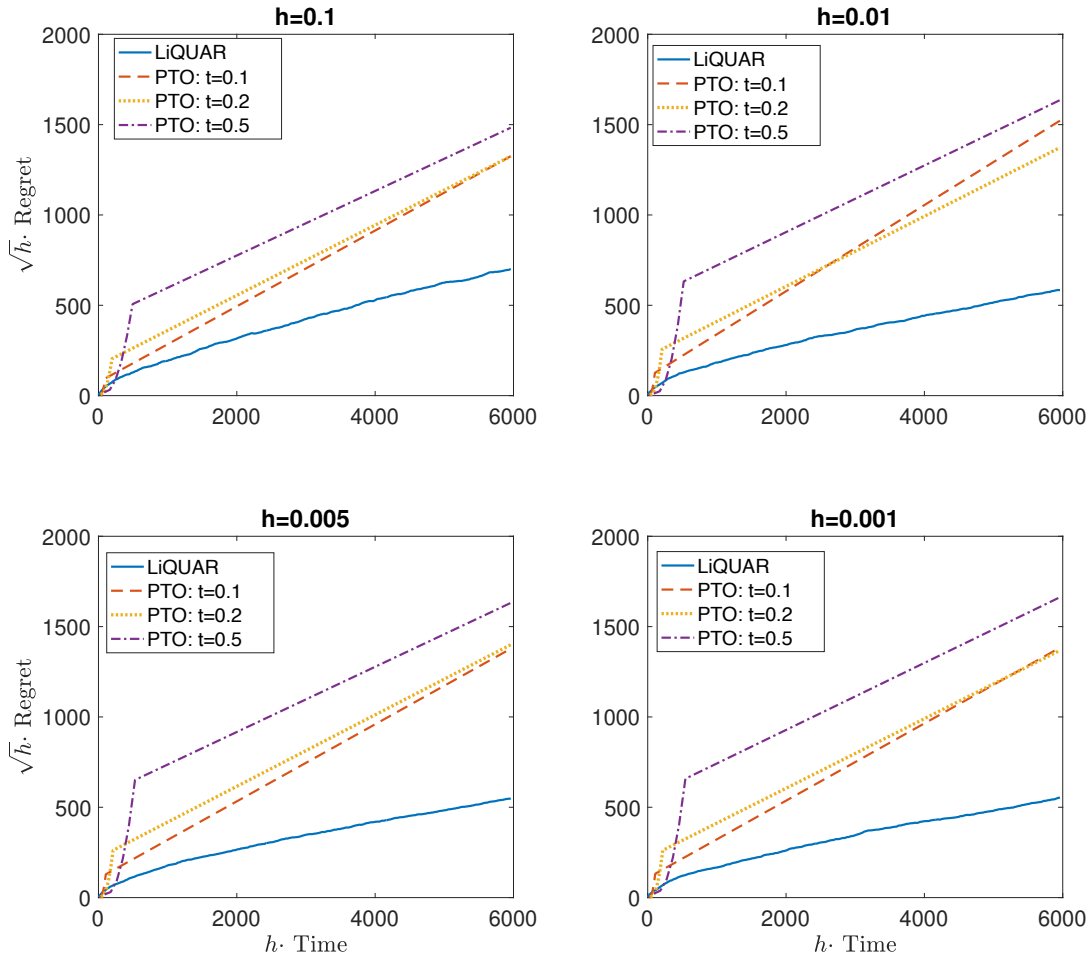


Figure 8 PTO vs. LiQUAR; $\eta_k^h = 4\sqrt{h}k^{-1}, \delta_k^h = 2\sqrt{h}k^{-1/3}$ with $T_k^h = h^{-1}k^{1/3}$; $t_0^h = t \cdot \frac{T_0^{5/7}\log(T_0/h)^{2/7}}{\sqrt{h}}$ and $\kappa = t_0^{1/5} \cdot h^{1/5}$ with $t \in \{0.1, 0.2, 0.5\}$

Specifically, let $\theta \in (0,1)$ represent the exploration ratio and T denote the total time (or learning budget). The oiPTO approach divides the total time horizon T into two phases. In the first phase, corresponding to the interval $[0, \theta T]$, the focus is on learning the parameters of the demand function. In the second phase, covering the interval $[\theta T, T]$, the system operates using decisions optimized based on the estimated parameters. The details of these two steps are provided below:

- **Prediction:** Suppose m parameters of the demand function are to estimated, and we uniformly select $(p_1, \mu_1), \cdots, (p_m, \mu_m) \in \mathcal{B}$ as experimentation decisions. We sequentially operate the system under each of the experimentation decision for $\theta T/m$ units of time. Then, based on the arrival and workload data, the estimated parameter is given by

$$\boldsymbol{\beta}_{oi} = \arg\min_{\boldsymbol{\beta}} \sum_{k=1}^{m} (f(p_k, \mu_k; \boldsymbol{\beta}) - \hat{f}(p_k, \mu_k))^2, \tag{25}$$

where $f(p,\mu;\boldsymbol{\beta}) = -p\lambda(p;\boldsymbol{\beta}) + h\cdot\frac{\lambda(p;\boldsymbol{\beta})}{\mu-\lambda(p;\boldsymbol{\beta})} + c(\mu)$ and $\hat{f}(p,\mu)$ is the time average cost estimation of $f(p,\mu)$.

- **Optimization:** Next, we obtain the oiPTO-optimal policy $\hat{x}^*$ by maximizing our objective function with $\lambda(\cdot;\boldsymbol{\beta})$ replaced by $\lambda(\cdot;\boldsymbol{\beta}_{oi})$. Then we implement this policy for the rest of time horizon.

Experiment settings and results. We consider our base logit example in Section 6.1 having demand function (20) with $M_0 = 10, a = 4.1, b = 1$ and exponential service times. Throughout this experiment, we fix the staffing cost $c(\mu) = \mu$. To understand the impact of the system's congestion level on performance of oiPTO and LiQUAR, we consider two scenarios specified by the optimal traffic intensity ρ^*: (i) A light-traffic case with $\rho^* = 0.709$ ($h_0 = 1$) and (ii) A heavy-traffic case with $\rho^* = 0.987$ ($h = 0.001$).

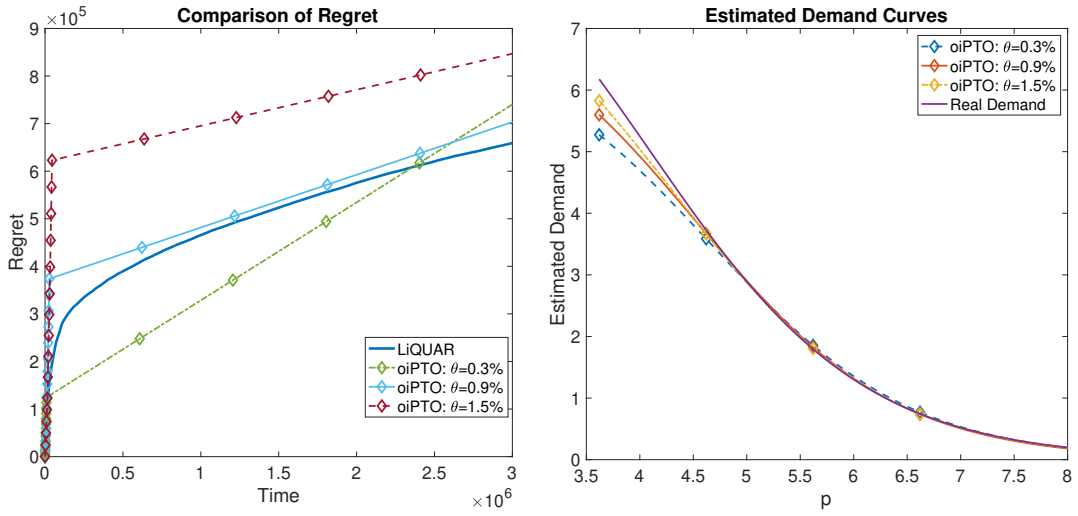
For LiQUAR, we consistently select the hyperparameters $\eta_k = 4k^{-1}, \delta_k = \min(0.1, 0.5k^{-1/3})$, initial values $(\mu_0, p_0) = (10, 7)$ and $T_k = 200k^{1/3}$ for $L = 1000$ iterations with a total running time $T = 2\sum_{k=1}^{L} 200k^{1/3}$. For oiPTO, we use the same total time T and consider several values of the exploration ratio $\theta \in \{0.3\%, 0.9\%, 1.5\%, 6\%, 15\%\}$ to account for different levels of exploration efforts.

In Figure 9, we present the regret results for LiQUAR and oiPTO, showcasing the three oiPTO curves with the lowest regrets. The left-hand panels illustrate that the exploration ratio θ has a significant impact on oiPTO's performance. The regret for oiPTO exhibits a piecewise linear pattern: during the prediction phase, regret grows rapidly due to periodic exploration across all experimentation variables; in the optimization phase, regret continues to increase linearly, albeit at a slower rate, as the system operates based on the oiPTO-optimized solution, which remains suboptimal. A larger (smaller) θ leads to higher (lower) regret during the prediction phase but results in a more (less) accurate model. This improved accuracy generates decisions that are closer to optimal, resulting in a slower (faster) regret growth during the optimization phase.

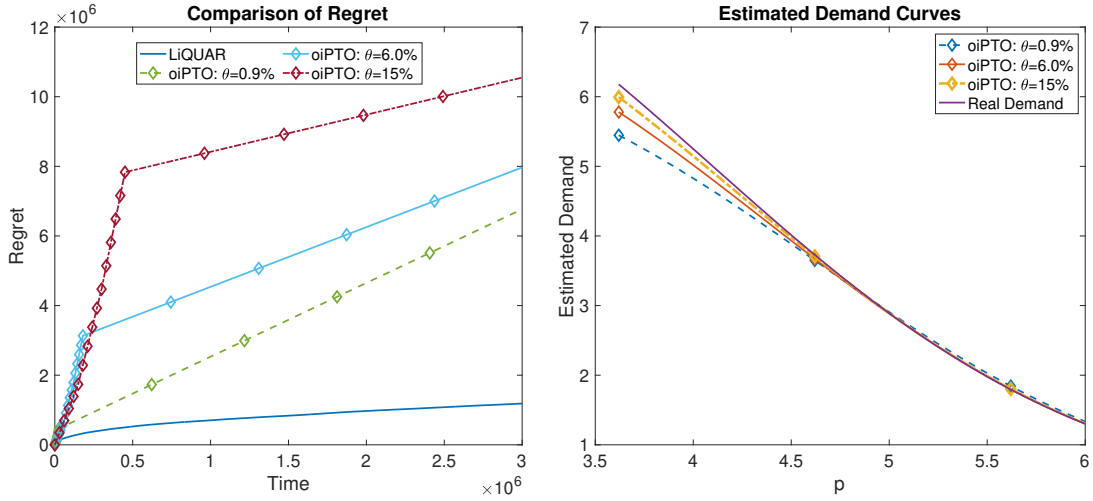
Furthermore, comparing case (a) to case (b), we observe that while oiPTO incorporates downstream objective information, LiQUAR consistently outperforms oiPTO by achieving a lower regret. This advantage is especially pronounced in heavy-traffic scenarios. The primary reason is that LiQUAR employs an integrated learning approach, continuously refining its decision-making through direct interaction with the environment. In contrast, oiPTO follows a static learning strategy, investing fixed efforts into parameter prediction and relying entirely on these predictions in the optimization phase. The limitations of oiPTO become more apparent in heavy traffic, where the nonlinear structure of workload amplifies the cost of suboptimal decisions.

8. LiQUAR vs. Reinforcement Learning

In this section, we compare LiQUAR with reinforcement learning (RL) methods. While the machine learning literature offers a wide range of RL approaches, we focus on the policy gradient type methods (PG



(a) Light traffic case with $\rho^* = 0.706$



(b) Heavy traffic case with $\rho^* = 0.987$

Figure 9 oiPTO vs. LiQUAR: (i) low traffic scenario $\rho^* = 0.705$; (ii) high traffic scenario $\rho^* = 0.987$. Hyperparameters for LiQUAR are $\eta_k = 4k^{-1}, \delta_k = \min(0.1, 0.5k^{-1/3}), T_k = 200k^{1/3}$ in both scenarios. All regret curves are estimated by averaging 1,000 independent simulation runs.

methods) for comparison due to the following reasons: (i) both LiQUAR and PG methods rely on gradient-based optimization, making them conceptually aligned; and (ii) our problem involves an infinite state space (e.g., queue length or workload) and a continuous action space, where the PG methods demonstrate particular advantages over other RL techniques.

Problem Settings and Algorithms. Because an RL method is underpinned by its corresponding Markov decision process (MDP), and setting up an MDP requires the model to be Markovian, we now restrict our attention to the $M/M/1$ queue. Specifically, we consider the following discrete-time MDP with the objective of maximizing its long-run average reward. Our MDP has

- *Time steps:* $t = 1, 2, \ldots$.
- *States:* Queue length at beginning of period t, denoted by S_t.
- *Actions:* Choices of price and service rate at each time step $A_t = (p, \mu)$.
- *Rewards:* The net profit gained in time slot t, denoted by R_t.

Under the above setting, we write the Bellman equation as below:

$$q_\pi(s,a) + h_\pi = \mathbb{E}_{s' \sim P(s,a), a' \sim \pi}[R(s,a) + q(s',a')],$$

with $q_\pi(s,a)$ is the q-function of policy π and h_π is the long-run average revenue under π.

Following (Sutton and Barto 2018, Section 13.6), we apply the Gaussian parameterization for our actions. Specifically, we draw $p \sim N(\bar{p}, \sigma_p^2)$ and $\mu \sim N(\bar{\mu}, \sigma_\mu^2)$ independently. Let $\theta \equiv (\bar{p}, \bar{\mu}, \sigma_p^2, \sigma_\mu^2)$ and we denote π_θ as the Gaussian density with parameter θ. According to the policy gradient theorem (Sutton and Barto 2018, p.339), the gradient on the policy function can be represented as $\nabla_\theta h_{\pi_\theta} = \mathbb{E}[\nabla \log \pi_\theta(A_t|S_t) \cdot q(S_t, A_t)]$. We consider two types of PG algorithms, i.e., a base policy gradient algorithm (PG-base) and a policy gradient algorithm with score-aware gradient (PG-SAGE) introduced in Comte et al. (2023). See EC.8 for more details about the two algorithms.

Experiment settings and results. We now compare PG algorithms with LiQUAR using our base example as described in Section 6.1. Specifically, we consider an $M/M/1$ queue having logit demand function (20) with $M_0 = 10$, $a = 4.1$, $b = 1$ and exponential service times with holding cost $c(\mu) = \mu$. For LiQUAR, the hyperparameter are $\eta_k = 4k^{-1}$, $T_k = 200k^{1/3}$, and $\delta_k = \min(0.1, 0.5k^{-1/3})$. For PG type algorithms, we consider picking both the constant step sizes $\eta \in \{1, 0.1, 0.01, 0.001, 0.0001\}$ and the decreasing step sizes $\eta_k \in \{4k^{-1}, 2k^{-1}, k^{-1}, 0.1k^{-1}, 0.01k^{-1}\}$. The cycle lengths are $T \in \{10, 100, 300\}$. In addition, we keep the total running time of LiQUAR and PG equal in order for a fair comparison.

We report the regret curves in Figure 10. For the clarity of the figure, we report the curves with the lowest regret for each PG algorithm. From Figure 10, we find that LiQUAR is more effective than PG in a wide range of hyper-parameter choices.

REMARK 6 (LIQUAR VS. PG). In the PG algorithms, the gradient estimators rely on accurately learning the $q_\pi(s,a)$ function (as outlined in the policy gradient theorem), which is a two-dimensional function. Inaccuracies in this estimation can result in significant variance in the gradient calculations. In contrast, LiQUAR only requires learning the values of individual actions, substantially reducing the complexity and effort required for learning. Furthermore, by the design of LiQUAR, the tuning of its hyperparameters can leverage domain-specific knowledge of the queueing system, e.g., the transient bias and auto-correlation between queueing data; also see Remark 2. In comparison, tuning hyperparameters in the PG algorithms is considerably more challenging, as RL methods are generally considered black-box approaches.

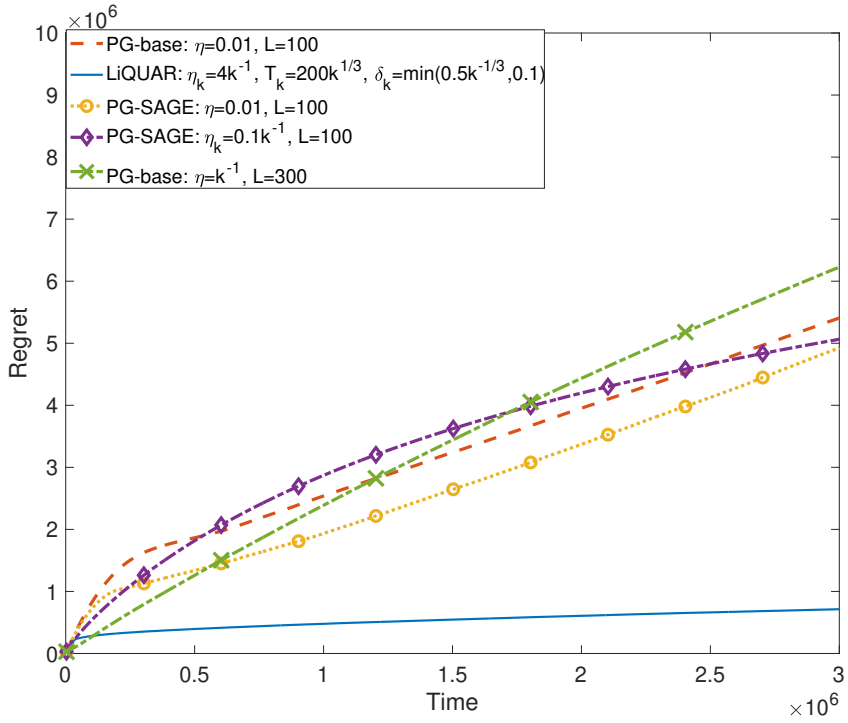


Figure 10 Comparison of the regret of LiQUAR with PG type algorithms in our base example. The hyperparameter choices are: (i) LiQUAR : $\eta_k = 4k^{-1}, T_k = 200k^{1/3}, \delta_k = \min(0.5k^{1/3}, 0.1)$; (ii) PG: $\eta_k \in \{1, 0.1, 0.01, 0.001, 4k^{-1}, 2k^{-1}, k^{-1}, 0.1k^{-1}, 0.01k^{-1}\}, L \in \{1, 10, 100, 300\}$, Episode length $T = 3000/L$. For clarity of the figure, we plot the lowest regret curve for PG algorithms. All regret curves are estimated from 100 independent simulation replications.

9. Conclusions

In this paper we develop an online learning framework, dubbed LiQUAR, designed for dynamic pricing and staffing in an $M/GI/1$ queue with unknown arrival rate function and service distribution. LiQUAR's main appeal is its "model-free" attribute. Unlike the conventional "predict-then-optimize" approach where precise estimations of the demand function and service distribution must be conducted (as a separate step) before the decisions may be optimized, LiQUAR is an integrated method that recursively evolves the control policy to optimality by effectively using the newly generated queueing data (e.g., arrival times and service times). LiQUAR's main advantage is its solution robustness; its algorithm design is able to automatically relate the parameter estimation errors to the fidelity of the optimized solutions. Comparing to the conventional method, this advantage becomes more significant when the system is in heavy traffic.

Effectiveness of LiQUAR is substantiated by (i) theoretical results including the algorithm convergence and regret analysis, and (ii) engineering confirmation via simulation experiments of a variety of representative queueing models. Theoretical analysis of the regret bound in the present paper may shed lights on the design of efficient online learning algorithms (e.g., bounding gradient estimation error and controlling

proper learning rate) for more general queueing systems. In addition, the analysis on the statistical properties for our gradient estimator has independent interests and may contribute to the general literature of stochastic gradient decent. We also extend LiQUAR to the more general $GI/GI/1$ model and confirm its good performance by conducting numerical studies.

There are several venues for future research. One dimension is to extend the method to queueing models under more general settings such as non-Poisson arrivals, customer abandonment and multiple servers, which will make the framework more practical for service systems such as call centers and healthcare. Another interesting direction is to theoretically relax the assumption of uniform stability by developing a "smarter" algorithm that automatically explores and then settles on control policies that guarantee stable system performance. Another practical direction is to investigate how to learn the optimal dynamic policy, where for a finite horizon T the optimal service rate μ and price p are inherently state- and time-dependent as given by a continuous-time optimal control problem.

Acknowledgments

The authors thank the editors and anonymous referees for their helpful comments. The first author acknowledges grant supports from NSFC 72571237, NSFC 72394361.

References

Abate J, Choudhury GL, Whitt W (1993) Calculation of the GI/G/1 waiting-time distribution and its cumulants from Pollaczek's formulas. *Archiv für Elektronik und Ubertragungstechnik* 47(5/6):311–321.

Abate J, Whitt W (1988a) The correlation functions of RBM and M/M/1. *Communications in Statistics. Stochastic Models* 4(2):315–359.

Abate J, Whitt W (1988b) Transient behavior of the M/M/1 queue via Laplace transforms. *Advances in Applied Probability* 20(1):145–178.

Asmussen S (2003) *Applied Probability and Queues*, volume 2 (Springer).

Baron O, Krass D, Senderovich A, Sherzer E (2023) Supervised ML for solving the GI/GI/1 queue. *INFORMS Journal on Computing* 2015(586-594).

Bergquist J, Elmachtoub AN (2023) Static pricing guarantees for queueing systems. *arXiv preprint arXiv:2305.09168* .

Besbes O, Zeevi A (2009) Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Operations research* 57(6):1407–1420.

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

Blanchet J, Chen X (2020) Rates of convergence to stationarity for reflected Brownian motion. *Mathematics of Operations Research* 45(2):660–681.

Broadie M, Cicek D, Zeevi A (2011) General bounds and finite-time improvement for the Kiefer-Wolfowitz stochastic approximation algorithm. *Operations Research* 59(5):1211–1224.

Broder J, Rusmevichientong P (2012) Dynamic pricing under a general parametric choice model. *Operations Research* 60(4):965–980.

Chen X, Liu Y, Hong G (2024) An online learning approach to dynamic pricing and capacity sizing in service systems. *Operations Research* 72(6):2677–2697, URL `http://dx.doi.org/10.1287/opre.2020.0612`.

Cheung WC, Simchi-Levi D, Wang H (2017) Dynamic pricing and demand learning with limited price experimentation. *Operations Research* 65(6):1722–1731.

Chong EKP, Ramadge PJ (1993) Optimization of queues using an infnitesimal perturbation analysis-based stochastic algorithm with general update times. *SIAM Journal on Control and Optimization* 31:698–732.

Comte C, Jonckheere M, Sanders J, Senen-Cerda A (2023) Score-aware policy-gradient methods and performance guarantees using local lyapunov conditions: Applications to product-form stochastic networks and queueing systems. *arXiv preprint arXiv:2312.02804* .

Cosmetatos GP (1976) Some approximate equilibrium results for the multi-server queue (M/G/r). *Journal of the Operational Research Society* 27(3):615–620.

Dai JG, Gluzman M (2021) Queueing network controls via deep reinforcement learning. *Stochastic Systems* 12(1):30–67.

Elmachtoub AN, Shi J (2023) The power of static pricing for reusable resources. *arXiv preprint arXiv:2302.11723* .

Fu MC (1990) Convergence of a stochastic approximation algorithm for the GI/G/1 queue using infinitesimal perturbation analysis. *Journal of Optimization Theory and Applications* 65:149–160.

Garyfallos S, Liu Y, Barlet-Ros P, Cabellos-Aparicio A (2024) Service level prediction in non-Markovian nonstationary queues: A simulation-based deep learning approach. *Winter Simulation Conference (WSC)* 2015(586-594).

Garyfallos S, Liu Y, Barlet-Ros P, Cabellos-Aparicio A (2025a) NeuraliNQ: A neural network method for the transient performance analysis in non-Markovian queues. *Queueing Systems* 109(24).

Garyfallos S, Liu Y, Wang X, Barlet-Ros P, Cabellos-Aparicio A (2025b) Solving nonstationary non-Markovian queueing systems: A transfer learning neural network approach. *Working Paper*.

Glasserman P (1992) Stationary waiting time derivatives. *Queueing Systems* 12:369–390.

Huh WT, Janakiraman G, Muckstadt JA, Rusmevichientong P (2009) An adaptive algorithm for finding the optimal base-stock policy in lost sales inventory systems with censored demand. *Mathematics of Operations Research* 34(2):397–416.

Jia H, Shi C, Shen S (2022) Online learning and pricing with reusable resources: Linear bandits with sub-exponential rewards. *International Conference on Machine Learning*, 10135–10160.

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

Keskin NB, Zeevi A (2014) Dynamic pricing with an unknown demand model: Asymptotically optimal semi-myopic policies. *Operations research* 62(5):1142–1167.

Kim J, Randhawa RS (2018) The value of dynamic pricing in large queueing systems. *Operations Research* 66(2):409–425.

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

Kumar S, Randhawa RS (2010) Exploiting market size in service systems. *Manufacturing Service Oper. Management* 12(3):511–526.

L'Ecuyer P, Giroux N, Glynn PW (1994) Stochastic optimization by simulation: Numerical experiments with the M/M/1 queue in steady-state. *Management Science* 40(10):1245–1261.

L'Ecuyer P, Glynn PW (1994) Stochastic optimization by simulation: Convergence proofs for the GI/GI/1 queue in steady state. *Management Science* 40(11):1562–1578.

Lee C, Ward AR (2014) Optimal pricing and capacity sizing for the GI/GI/1 queue. *Operations Research Letters* 42:527–531.

Lee C, Ward AR (2019) Pricing and capacity sizing of a service facility: Customer abandonment effects. *Production and Operations Management* 28(8):2031–2043.

Liu B, Xie Q, Modiano E (2019) Reinforcement learning for optimal control of queueing systems. *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 663–670.

Maglaras C, Zeevi A (2003) Pricing and capacity sizing for systems with shared resources: Approximate solutions and scaling relations. *Management Science* 49(8):1018–1038.

Murthy Y, Grosof I, Maguluri ST, Srikant R (2024) Performance of npg in countable state-space average-cost rl. *arXiv preprint arXiv:2405.20467* .

Nair J, Wierman A, Zwart B (2016) Provisioning of large-scale systems: The interplay between network effects and strategic behavior in the user base. *Management Science* 62(6):1830–1841.

Nakayama MK, Shahabuddin P, Sigman K (2004) On finite exponential moments for branching processes and busy periods for queues. *Journal of Applied Probability* 41(A):273–280.

Olivares J, Martin P, Valero E (2018) A simple approximation for the modified bessel function of zero order i0 (x). *Journal of Physics: Conference Series*, volume 1043, 012003 (IOP Publishing).

Pollaczek F (1930) Über eine aufgabe der wahrscheinlichkeitstheorie. I. *Mathematische Zeitschrift* 32(1):64–100.

Raeis M, Tizghadam A, Leon-Garcia A (2021) Queue-learning: A reinforcement learning approach for providing quality of service. *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 461–468.

Shah D, Xie Q, Xu Z (2020) Stable reinforcement learning with unbounded state space. Bayen AM, Jadbabaie A, Pappas G, Parrilo PA, Recht B, Tomlin C, Zeilinger M, eds., *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, 581–581.

Sutton RS, Barto AG (2018) *Reinforcement Learning: An Introduction* (The MIT Press), 2nd edition.

Walton N, Xu K (2021) Learning and information in stochastic networks and queues. *Tutorials in Operations Research: Emerging Optimization Methods and Modeling Techniques with Applications*, 161–198.

Yuan H, Luo Q, Shi C (2021) Marrying stochastic gradient descent with bandits: Learning algorithms for inventory systems with fixed costs. *Management Science* 67(10):6089–6115, URL `http://dx.doi.org/10.1287/mnsc.2020.3799`.

Zhang H, Chao X, Shi C (2020) Closing the gap: A learning algorithm for lost-sales inventory systems with lead times. *Management Science* 66(5):1962–1980.

Zhong Y, Birge JR, Ward AR (2024) Learning to schedule in multiclass many-server queues with abandonment. *Operations Research* .

Author Biographies

Xinyun Chen is an associate professor in the School of Data Science and School of Management and Economics at the Chinese University of Hong Kong, Shenzhen. Her research interests include stochastic simulation, queueing theory, and reinforcement learning with applications in healthcare and telecommunication networks.

Guiyu Hong is an assistant professor in the College of Business at Shanghai University of Finance and Economics. His research interests include queueing theory, online learning, stochastic modeling, and applications in telecommunication networks.

Yunan Liu is a Principal Research Scientist in the Supply Chain Optimization Technology (SCOT) department at Amazon, and an adjunct professor in the Industrial and Systems Engineering department at North Carolina State University. His research interests include stochastic modeling, simulations, optimal control, reinforcement learning, and ML-based stochastic optimization, and their applications to queueing systems and supply chain systems. His work was awarded first place in the INFORMS Junior Faculty Interest Group Paper Competition in 2016. His website is https://yliu48.github.io/.