Manufacturing & Service Operations Management

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

Optimally Scheduling Heterogeneous Impatient Customers

Achal Bassamboo, Ramandeep Randhawa, Chenguang (Allen) Wu

To cite this article:
Achal Bassamboo, Ramandeep Randhawa, Chenguang (Allen) Wu (2023) Optimally Scheduling Heterogeneous Impatient Customers. Manufacturing & Service Operations Management 25(3):1066-1080. https://doi.org/10.1287/msom.2023.1190

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

MANUFACTURING & SERVICE OPERATIONS MANAGEMENT
Vol. 25, No. 3, May–June 2023, pp. 1066–1080
ISSN 1523-4614 (print), ISSN 1526-5498 (online)

Optimally Scheduling Heterogeneous Impatient Customers

Achal Bassamboo,[a] Ramandeep Randhawa,[b] Chenguang (Allen) Wu[c,*]

[a] Kellogg School of Management, Northwestern University, Evanston, Illinois 60208; [b] Marshall School of Business, University of Southern California, Los Angeles, California 90089; [c] Department of Industrial Engineering and Decision Analytics, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong

*Corresponding author

Contact: a-bassamboo@northwestern.edu, https://orcid.org/0000-0001-7758-4751 (AB); ramandeep.randhawa@marshall.usc.edu, https://orcid.org/0000-0003-4795-1252 (RR); allenwu@ust.hk, https://orcid.org/0000-0002-2528-0286 (C(A)W)

Received: October 6, 2020
Revised: June 1, 2022
Accepted: January 9, 2023
Published Online in Articles in Advance: February 7, 2023

https://doi.org/10.1287/msom.2023.1190

Abstract. ***Problem definition*:** We study scheduling multi-class impatient customers in parallel server queueing systems. At the time of arrival, customers are identified as being one of many classes, and the class represents the service and patience time distributions as well as cost characteristics. From the system's perspective, customers of the same class at time of arrival get differentiated on their residual patience time as they wait in queue. We leverage this property and propose two novel and easy-to-implement multi-class scheduling policies. ***Academic/practical relevance*:** Scheduling multi-class impatient customers is an important and challenging topic, especially when customers' patience times are nonexponential. In these contexts, even for customers of the same class, processing them under the first-come, first-served (FCFS) policy is suboptimal. This is because, at time of arrival, the system only knows the overall patience distribution from which a customer's patience value is drawn, and as time elapses, the estimate of the customer's residual patience time can be further updated. For nonexponential patience distributions, such an update indeed reveals additional information, and using this information to implement within-class prioritization can lead to additional benefits relative to the FCFS policy. ***Methodology*:** We use fluid approximations to analyze the multi-class scheduling problem with ideas borrowed from convex optimization. These approximations are known to perform well for large systems, and we use simulations to validate our proposed policies for small systems. ***Results*:** We propose a multi-class time-in-queue policy that prioritizes both across customer classes and within each class using a simple rule and further show that most of the gains of such a policy can be achieved by deviating from within-class FCFS for at most one customer class. In addition, for systems with exponential patience times, our policy reduces to a simple priority-based policy, which we prove is asymptotically optimal for Markovian systems with an optimality gap that does not grow with system scale. ***Managerial implications*:** Our work provides managers ways of improving quality of service to manage parallel server queueing systems. We propose easy-to-implement policies that perform well relative to reasonable benchmarks. Our work also adds to the academic literature on multi-class queueing systems by demonstrating the joint benefits of cross- and within-class prioritization.

Funding: A. Bassamboo received financial support from the National Science Foundation [Grant CMMI 2006350]. C. (A.) Wu received financial support from the Hong Kong General Research Fund [Early Career Scheme, Project 26206419].

Supplemental Material: The online appendix is available at https://doi.org/10.1287/msom.2023.1190.

Keywords: call center management • queueing theory • service operations • stochastic methods

1. Introduction

In this paper, we study scheduling multi-class impatient customers in parallel-server queueing systems—call centers being a canonical example. Customers in such systems are modeled as being one of many classes, and each class is associated with class-specific service and patience time distributions as well as cost characteristics. At the time of arrival, each customer's class is identified, and the customer is placed at the end of a queue corresponding to that class. The system manager's objective is to route idle servers to customers in order to minimize the abandonment and waiting costs aggregated over all arriving customers.

To achieve this objective, existing research proposes a number of scheduling policies and proves the optimality of these policies in asymptotic regimes. Examples include the well-known $h\mu/\gamma$ policy (Atar et al. 2010) for exponential patience times and the generalized $h\mu/\gamma$ policy (Long et al. 2020) for nonexponential patience times. An implicit yet fundamental assumption in these policies is that customers belonging to the same class are processed in a first-come, first-served (FCFS)

manner. However, as Bassamboo and Randhawa (2015) suggest, processing customers of the same class under FCFS itself is suboptimal; deviating from FCFS to allow within-class prioritization can lead to significant cost benefits relative to FCFS.

These benefits are primarily driven by another dimension of customer heterogeneity that is less understood in the literature. From the system's perspective, customers of the same class at the time of arrival get further differentiated on their residual patience time as they wait in the system. This is because, at the time of arrival, the system manager only knows the overall patience distribution from which an arriving customer's patience value is drawn, and as time elapses, the estimate of the customer's residual patience time can be updated. For nonexponential patience distributions, such an update indeed reveals additional information. Bassamboo and Randhawa (2015) use this information to propose a time-in-queue (TIQ) policy that differentiates customers of the same class based on their time in queue and shows that this policy can significantly outperform FCFS in certain circumstances.

These findings bring a natural question of how to incorporate this new dimension of customer heterogeneity into a multi-class analysis when customers at time of arrival are already heterogeneous in their classes. Correspondingly, a scheduling policy for a multi-class queueing system can be viewed as comprising two decisions: when a server becomes available, to which customer class should the server be allocated and, then, to which customer within that class should the server be allocated. Noting the analytical difficulty in obtaining an exact solution, we undertake a fluid-based approach and study the optimal policy in a fluid model of an overloaded queueing system. We then use the fluid solution to propose scheduling policies for the original stochastic queueing system.

Our fluid optimization yields some key insights. We show that the fluid solution for multi-class systems splits at most one customer class into two subclasses. This strengthens the result in Bassamboo and Randhawa (2015) developed for single-class systems in which the class is split into at most two subclasses. In other words, although, in principle, we allow differentiation and prioritization within each class, the fluid optimal policy only differentiates customers on their wait times for at most one class, and all other classes are processed under FCFS.

We use this observation to propose a scheduling policy for the stochastic system in which all but one class are processed under FCFS. We refer to this policy as *mostly-FCFS*. For this, we introduce an easy-to-compute index that allows us to divide all classes into three sets labeled as high priority $\mathcal{F}$ (these classes are fully served, and asymptotically, no customers from these classes abandon), medium priority $\mathcal{P}$ (these classes are partially served), and low priority $\mathcal{E}$ (these classes are not served at all, and asymptotically, all customers from these classes abandon). The set $\mathcal{F}$ consists of classes that are given full priority over classes in other sets, followed by the classes in $\mathcal{P}$ and, finally, those in $\mathcal{E}$. The set $\mathcal{P}$ has at most one class that is processed under non-FCFS.

The mostly-FCFS policy simplifies considerably if the customer-based costs are solely due to abandonments. In this case, this policy reduces to a $p\mu$ priority rule that prioritizes classes based on the ranking of the penalty per abandonment, p, times the service rate of each class, μ, and processes customers within each class under FCFS. This policy can hold under general patience distributions because, in overloaded systems, the abandonment metric depends primarily on the rate imbalance between the arrival and service processes irrespective of the patience distributions. Noting that, for exponential patience times, a queue length–based objective can be expressed in terms of an abandonment-based objective, we obtain that the $h\mu/\gamma$ (holding cost × service rate × mean patience time) priority policy is optimal for the queue-length metric. In fact, Atar et al. (2010) shows that this policy is asymptotically optimal at the fluid scale; that is, its optimality gap divided by the system size tends to zero as the system size grows indefinitely. We strengthen this result by showing that this policy is in fact $\mathcal{O}(1)$-optimal for Markovian systems; that is, its (unscaled) optimality gap to the optimal policy remains bounded as the system size grows without bound.

Turning to the queue-length metric under general patience distributions, if the patience time distributions have decreasing hazard rates, our proposed policy entails processing each class under FCFS and prioritizing different classes using a ranking that depends on the entire patience time distribution of each class beyond its mean. If the patience time distributions have increasing hazard rates, our proposed policy becomes the $h\mu/\gamma$ priority policy with one important caveat, that is, processing one class under the last-come, first-served (LCFS) policy. We show that this distinction is important and the performance can drop significantly if we process all classes under FCFS.

As an alternative to the mostly-FCFS policy, we propose a multi-class time-in-queue policy, and we refer to it as *mTIQ*. This policy has a more robust implementation but requires differentiating in real time customers of all classes based on their time in queue. Unlike the mostly-FCFS policy, which requires the knowledge of the total capacity to compute the threshold wait times associated with classes in the set $\mathcal{P}$, this information is not needed to implement the mTIQ policy. The mTIQ policy assigns an available server to the class that currently has the highest cost gradient. That is, whenever a server becomes idle, it is allocated to the class that currently has the highest marginal benefit of utilizing that server and, then, within that class, allocated to a customer

using the TIQ policy. We demonstrate that the mTIQ policy is robust to changes in capacity levels and this robustness is also exhibited for the queue-length metric under patience time distributions with increasing hazard rates.

Finally, we extend our policies to systems with dependent service and patience times within each class. We find that, in some cases, our proposed policies are identical to those under independent service and patience times within each class, whereas in other cases, these policies can be very different, and further, in these cases, there can be significant benefits of utilizing our proposed policies that take into account the underlying within-class dependence.

1.1. Literature Review

Our work is related to a growing literature on scheduling heterogeneous impatient customers in service queueing systems. This literature assumes that customers are a priori differentiated by certain characteristics, such as service or patience time distributions or cost parameters. Exact analysis of the optimal scheduling policy is often intractable (a notable exception is Down et al. 2011, who establish the optimal policy for a two-class, single-server system); thus, a common mode of analysis in this literature is to build on the fluid or diffusion approximations to original queueing systems to develop optimal policies in asymptotic regimes. For example, Dai and Tezcan (2008) employ diffusion approximations to develop policies for parallel-server systems with pool-dependent service times. Gurvich and Whitt (2010) propose control policies for parallel-server systems that maintain fixed queue ratios to meet service-level targets. Kim et al. (2018) study multi-class queueing systems with heterogeneous patience time distributions and constructs near-optimal policies by solving a diffusion control problem. Apart from diffusion approximations, fluid approximations are also commonly used to develop scheduling policies for large systems. Examples include Atar et al. (2010), who prove the asymptotic optimality of the $h\mu/\gamma$ priority rule under exponential patience times and linear cost functions. Long et al. (2020) extend the $h\mu/\gamma$ priority policy to allow nonexponential patience times and general cost functions. Long and Zhang (2019) propose a virtual allocation policy that assigns a fixed portion of servers to each class and proves the asymptotic optimality of this policy when patience time distributions have decreasing hazard rates. As mentioned, an implicit yet fundamental assumption in this literature is that customers within each class are processed under FCFS although Bassamboo and Randhawa (2015) suggest that deviating from FCFS to allow within-class differentiation can lead to significant benefits in some circumstances. Bassamboo and Randhawa (2015) further propose a TIQ policy for a single-class queueing system that allocates servers to customers based on their time in queue. However, it is not clear from Bassamboo and Randhawa (2015) how the TIQ policy defined in that paper extends to our multi-class systems for which scheduling policies must specify the joint allocation of capacity both across classes and within each class between customers with different wait times. To answer this question, we generalize their single-class analysis to multi-class systems. We introduce an index that measures the marginal benefit of allocating capacity to each class with the TIQ policy applied and then use this index to propose two novel and easy-to-implement scheduling policies for multi-class queueing systems: mostly-FCFS and mTIQ. It is worth noting that, although, in principle, we allow differentiation and prioritization within each customer class, the mostly-FCFS policy can be implemented by deviating from within-class FCFS for at most one class. This endows the mostly-FCFS policy with fairness benefits and practical relevance.

Our fluid optimization builds on Whitt's (2006) fluid approximation to single-class $G/GI/n+GI$ queueing systems under FCFS. Kang and Ramanan (2010) and Zhang (2013) formally prove that Whitt's fluid model is a bona fide fluid limit in the many-server heavy-traffic regime. Because we can reduce our multi-class scheduling problem to a subclass-based optimization problem that processes each subclass under FCFS, the single-class fluid analysis in Whitt (2006) readily applies. Such a fluid model can often yield accurate approximations to large stochastic queueing systems in the overloaded regime (Bassamboo and Randhawa 2010, Bassamboo et al. 2010). Motivated by this, we directly work with a fluid model in this paper. A similar fluid approach is employed in Bassamboo and Randhawa (2015) and Wu et al. (2018) to study queueing systems with dependent service and patience times. (Reich 2012 presents empirical evidence of such dependence.) We extend our policies to incorporate such dependence in Section 6.

2. Model

2.1. Queueing System

We consider a multi-class parallel queueing system, in which m classes of customers are processed by a single pool of N agents, each processing work deterministically at a unit rate. Customers of each class $i=1,\ldots,m$ arrive to the system according to a stationary renewal process with arrival rate Λ_i. Customers of each class have independent and identically distributed (i.i.d.) service times with the cumulative distribution function denoted by $G_i(\cdot)$ and mean service time denoted by $1/\mu_i$. Each customer is associated with a patience time and abandons the system if service does not start within this time upon the customer's arrival. We denote the cumulative distribution function of class i customers' patience times by $F_i(\cdot)$, the probability density function by $f_i(\cdot)$, and the reciprocal of the mean patience time by γ_i. Each customer's patience time is independent of the

customer's service time (we discuss dependent service and patience times within each class in Section 6).

The system manager's goal is to find a scheduling policy that routes idle agents to customers in order to minimize the long-run average costs. Specifically, let $\xi_i(x, y, w)$ denote the cost experienced by a class i customer who has service time x, patience time y, and offered waited w (which is the amount of time the customer would wait before entering service if the customer were infinitely patient). Denote the set of all nonanticipating (non-forward-looking) policies by Π. For any policy $\pi \in \Pi$, the total customer-based cost can be written as

$$\inf_{\pi\in\Pi} \sum_{i=1}^{m} \Lambda_i \mathbb{E}[\xi_i(X_i, Y_i, W_i^{\pi})], \tag{1}$$

where W_i^{π} is the random variable representing the steady-state offered wait for a class I customer with patience time Y_i and service time X_i under the policy π. The expectation is taken over the random variables X_i, Y_i, and W_i^{π}.

With an appropriate choice of cost function ξ_i in (1), we can capture different system costs. Specifically, we can capture abandonment costs by setting $\xi_i(x, y, w) = p_i \mathbb{I}(y < w)$ with p_i denoting the penalty per abandoned class i customer. To capture queue-length costs, we can set the cost rate as $\xi_i(x, y, w) = h_i \min\{y, w\}$ with h_i denoting the cost per unit time waiting in queue per class I customer. We can further capture the holding cost for all customers in the system by setting the cost rate as $\xi_i(x, y, w) = h_i(\min\{y, w\} + x\mathbb{I}(y > w))$, which incorporates a customer's total time spent in the system (time in queue and in service). We can also consider combining abandonment and queue-length costs by adding up their corresponding cost functions.

While this formulation allows us to work with general cost functions, we focus on analyzing the abandonment and queue-length cost functions to draw clean insights. We then discuss how our analysis extends to other (more general) cost functions.

2.2. Fluid Model

A scheduling policy for the queueing system in our multi-class setting comprises two decisions: when a server becomes available, to which class should the server be allocated and, then, to which customer within that class should the server be allocated. Noting the analytical intractability of an exact solution, we undertake a fluid approach to solve the policy optimization problem. This is a typical approach in dealing with intractability of stochastic systems and affording an intimate relation between the policies for the fluid model and the original queueing system.

In the fluid model, customers arrive to the system in the form of a fluid both deterministically and continuously at the corresponding arrival rate. Further, the capacity is considered in a fluid manner too and can be allocated fractionally to different classes. Thus, when considering the fluid version of a scheduling policy in steady state, the first decision of to which class allocate an idle server becomes equivalent to computing the fraction of capacity allocated to each class in steady state.

Regarding the second decision of to which customer within a class to allocate the server, we note that most of the existing literature focuses on FCFS by allocating the server to the longest waiting customer in each class. In this paper, we build on Bassamboo and Randhawa (2015), who suggest that deviating from FCFS to allow within-class prioritization can lead to cost benefits under nonexponential patience time distributions. Bassamboo and Randhawa (2015) show that the decision of how to allocate servers within a class in the fluid model can be solved by splitting the class into at most two subclasses and processing each subclass under FCFS. This policy can be connected back to the fluid model using a TIQ policy.

In our multi-class setting, the two decisions of how to allocate servers across classes and then to which customer within a class to allocate a server are inherently linked. We next analyze the fluid-optimization problem to obtain a fluid solution, and we discuss in Section 4 how to implement the fluid solution in the original stochastic queueing system.

We formulate our fluid optimization problem in two stages: the first stage focuses on allocating capacity across classes and, then, the second stage optimizes the capacity allocated to each subclass within a class following the first stage allocation. The first stage optimization problem can be stated as follows:

$$\min_{\{n:\sum_i n_i \le N\}} \sum_i C_i(n_i), \tag{2}$$

where $C_i(n_i)$ is the total cost incurred from class I by allocating capacity n_i to that class.

To characterize $C_i(n_i)$, we utilize the solution approach in Bassamboo and Randhawa (2015). Specifically, we divide each class into multiple subclasses, each operating under FCFS, and optimize the amount of capacity allocated to each subclass. We use $J(i)$ to denote the number of subclasses created from class i. (If $J(i) = 1$, then there is only one subclass for class i, and it comprises the entire class.) We denote the arrival rate to subclass $j = 1, \ldots, J(i)$ by $\lambda_{i,j}$ and the corresponding capacity allocated to this subclass by $n_{i,j}$. Because the mean service time of a class i customer is $1/\mu_i$, it follows that, in the fluid model, customers of subclass j can be processed deterministically at rate $n_{i,j}\mu_i$. Moreover, the capacity constraint states that $\sum_{j=1}^{J(i)} n_{i,j} \le n_i$, where the inequality allows us to withhold capacity when necessary.

The offered wait for subclass j of class i solves a "rate balance" equation so that the rate of customers entering service (accounting for customer abandonment) equals

the service rate. In this way, if $\lambda_{i,j} \geq n_{i,j}\mu_i$, then the offered wait $w_{i,j}$ solves

$$\lambda_{i,j}\bar{F}_i(w_{i,j}) = n_{i,j}\mu_i, \tag{3}$$

and otherwise, if $\lambda_{i,j} < n_{i,j}\mu_i$, then $w_{i,j} = 0$ because this subclass has excess capacity to process all arrivals without delays. Thus, the offered wait $w_{i,j}$ solves

$$\lambda_{i,j}\bar{F}_i(w_{i,j}) = \min\{n_{i,j}\mu_i, \lambda_{i,j}\}. \tag{4}$$

Using the offered wait $w_{i,j}$ for subclass j of class i, we characterize the average customer-based cost for a representative customer in this subclass as

$$\bar{c}_i(w_{i,j}) = \int_y \int_x \xi(x, y, w_{i,j}) dG_i(x) dF_i(y). \tag{5}$$

The total cost rate for subclass j is $\lambda_{i,j}\bar{c}_i(w_{i,j})$. Then, the cost function $C_i(n_i)$ for class i is obtained as the optimal objective value of the following (second stage) within-class optimization problem:

$$C_i(n_i) = \inf_{\{J(i), n_{i,j}, w_{i,j}, \lambda_{i,j}, j=1,\ldots,J(i)\}} \sum_{j=1}^{J(i)} \lambda_{i,j}\bar{c}_i(w_{i,j}) \tag{6}$$

$$\text{s.t.} \quad \sum_{j=1}^{J(i)} \lambda_{i,j} = \Lambda_i,$$

$$\lambda_{i,j}\bar{F}_i(w_{i,j}) \leq n_{i,j}\mu_i,$$

$$\sum_{j=1}^{J(i)} n_{i,j} \leq n_i.$$

Naturally, the optimal capacity allocation across classes obtained by solving (2) depends on the marginal benefit of allocating capacity to each class, which further depends on the optimal within-class allocation across subclasses obtained by solving (6). Hence, it is important to first characterize the optimal solution to (6). We next review the main results in Bassamboo and Randhawa (2015) on the single-class analysis that help with this.

2.3. Single-Class Analysis

The optimization problem (6) determines the optimal capacity allocation across subclasses within class i. Bassamboo and Randhawa (2015) provide a structural result for this allocation, which we restate subsequently. That is, the optimal solution splits a single class into at most two subclasses.

Lemma 1. *The optimal number of subclasses in the optimization problem* (6) *satisfies* $J^*(i) \leq 2$.

It is useful to introduce $\bar{w}_i(\cdot)$, which denotes the offered wait as a function of capacity when the entire class i is processed under FCFS. If $\Lambda_i < n_i\mu_i$, then the offered wait under FCFS equals zero, and the cost for this class $C_i(n_i)$ trivially equals zero too. Otherwise, if $\Lambda_i \geq n_i\mu_i$, then the offered wait $\bar{w}_i(n_i)$ under FCFS solves

$$\lambda_i\bar{F}_i(\bar{w}_i(n_i)) = n_i\mu_i. \tag{7}$$

In this case, it is easy to see that the optimal allocation utilizes the entire capacity. Then, using Lemma 1, we can write (6) as follows:

$$C_i(n_i) = \min_{\{w_{i,1}, w_{i,2}, \lambda_{i,1}, \lambda_{i,2}\}} \lambda_{i,1}\bar{c}_i(w_{i,1}) + \lambda_{i,2}\bar{c}_i(w_{i,2}) \tag{8}$$

$$\text{s.t} \quad \lambda_{i,1} + \lambda_{i,2} = \Lambda_i, \tag{9}$$

$$\lambda_{i,1}\bar{F}_i(w_{i,1}) + \lambda_{i,2}\bar{F}_i(w_{i,2}) = \mu_i n_i, \tag{10}$$

$$w_{i,1} \leq \bar{w}_i(n_i) < w_{i,2}, \tag{11}$$

$$\lambda_{i,1}, \lambda_{i,2} \geq 0. \tag{12}$$

Constraint (11) is equivalent to $w_{i,1} < w_{i,2}$ and $w_{i,2} > \bar{w}_i(n_i)$, collectively ensuring that FCFS is represented by a unique offered wait vector. (In the absence of (11), one could potentially also represent FCFS by setting both $w_{i,1}$ and $w_{i,2}$ equal to $\bar{w}_i(n_i)$ and setting an arbitrary $\lambda_{i,1} \in (0, \Lambda_i)$.) Note that, for any $(w_{i,1}, w_{i,2})$ satisfying (11), Relations (9) and (10) allow us to uniquely determine $\lambda_{i,1}$ and $\lambda_{i,2}$ as follows:

$$\lambda_{i,1} = \frac{\mu_i n_i - \Lambda_i \bar{F}_i(w_{i,2})}{\bar{F}_i(w_{i,1}) - \bar{F}_i(w_{i,2})},$$
$$\lambda_{i,2} = \Lambda_i - \lambda_{i,1}.$$

Thus, we can consider (8) as an optimization problem over the variables $(w_{i,1}, w_{i,2})$ only. We denote an optimizer to this problem by $(w^*_{i,1}, w^*_{i,2})$.

If the optimal solution to (8) gives $w^*_{i,1} = \bar{w}_i(n_i)$, then it is optimal to have only one subclass for class i. In this case, our formulation generates multiple optimal solutions because setting $w^*_{i,2}$ to any amount greater than $\bar{w}_i$ does not affect this one-subclass solution. Specifically, we have $\lambda^*_{i,1} = \Lambda$ and $\lambda^*_{i,2} = 0$ for all such $w^*_{i,2}$. For convenience, we set $w^*_{i,2} = \infty$ in this case. Also note that, in our formulation, the LCFS policy can be represented by an offered wait vector $(0, \infty)$; that is, one subclass has zero offered wait and is served immediately, whereas the other subclass is never served and abandons after waiting out its patience time.

3. Optimizing the Fluid Model

The analysis in Section 2.3 answers the question of how to allocate capacity within a class. Yet it does not answer how to allocate capacity across classes, a decision that must be made before one can further allocate capacity within a class. To this end, we introduce an index that measures the marginal value of allocating capacity to a class, taking into account the optimal within-class allocation (6). We then apply this index to rank and prioritize across classes.

3.1. Characterizing the Optimal Solution

We first establish a structural property of the class-dependent cost function $C_i(\cdot)$ as the optimal objective value of (6).

Lemma 2. *The cost function C_i is nonincreasing and convex. Further, $C_i(n)$ is differentiable for all $n < \Lambda_i/\mu_i$.*

Lemma 2 proves that, even though there may be multiple solutions to (8), the class-dependent cost function is differentiable for all $n < \Lambda_i/\mu_i$. We denote the negative of the derivative of the cost function by

$$\beta_i(n) := -C_i'(n) \quad \text{for} \quad n < \Lambda_i/\mu_i.$$

In other words, $\beta_i(n)$ is the marginal decrease in cost when capacity allocated to this class is increased, in other words, the marginal benefit of capacity. Notice that the cost function C_i may not be differentiable at $n = \Lambda_i/\mu_i$. This is so because, for $n = \Lambda_i/\mu_i$, the processing capacity equals the incoming work, and thus, the offered wait is zero and the cost $C_i(\Lambda_i/\mu_i) = 0$. At this level of capacity, the cost function may have different left and right derivatives. In particular, increasing capacity has no impact on the already zero cost, and so the right derivative is zero. However, reducing capacity increases the cost, and this may happen in a manner such that the left derivative is strictly negative. To circumvent this issue, we define

$$\beta_i(n_i) := \lim_{n \uparrow n_i} -C_i'(n) \text{ for } n_i \geq \Lambda_i/\mu_i$$

as the left-side limit.

We discuss how to compute this index $\beta(\cdot)$ for specific cost functions, namely, abandonment costs, in Section 3.2 and queue-length costs in Section 3.3. However, using a general formulation of this index, we are able to solve the multi-class fluid optimization problems (2) and (6). To facilitate our characterization of the optimal solution, we first provide a structural property of the total number of subclasses in the optimal solution.

Proposition 1. *The fluid-optimal solution to (2) with C_i defined in (6) splits the m customer classes into at most $m + 1$ subclasses. In particular, $J(i) > 1$ for at most one class i, and further, for this class, we have $J(i) = 2$.*

Bassamboo and Randhawa (2015) show that, when focusing on a single class, the fluid-optimal solution splits the class into at most two subclasses (cf. Lemma 1). So, for our multi-class optimization problem, it seems a priori a good idea to split each class into two subclasses. However, Proposition 1 shows that, somewhat surprisingly, the fluid-optimal solution splits at most one class. In other words, although we allow differentiation within each class, the optimal policy differentiates among customers of at most one class.

Now, using Proposition 1 and the index β, we characterize the optimal solution to the fluid optimization problems (2) and (6) under general cost functions.

Proposition 2. *Defining sets of classes $\mathcal{F} := \{l : n_l^* = \Lambda_l/\mu_l\}$, $\mathcal{P} := \{l : 0 < n_l^* < \Lambda_l/\mu_l\}$, and $\mathcal{E} := \{l : n_l^* = 0\}$. For any $f \in \mathcal{F}$, $p \in \mathcal{P}$, and $e \in \mathcal{E}$, we have*

$$\beta_e(0) \leq \beta_p(n_p^*) \leq \beta_f(n_f^*). \tag{13}$$

Further, for any $p, p' \in \mathcal{P}$, we have

$$\beta_p(n_p^*) = \beta_{p'}(n_{p'}^*). \tag{14}$$

In addition, we have $J^*(i) = 1$ for all $i \in \mathcal{F} \cup \mathcal{E}$, and there is at most one class i_s in $\mathcal{P}$ with $J^*(i_s) = 2$.

Proposition 2 characterizes the necessary conditions that the optimal capacity allocation across classes must satisfy. The relation $\beta_e(0) \leq \beta_p(n_p^*)$ in (13) must hold because it states that no class in $\mathcal{E}$ should be allocated any capacity, and thus, for this to be optimal, the marginal value of allocating capacity to these classes must be less than the marginal cost of decreasing capacity from any class p in $\mathcal{P}$. A similar reasoning suggests that all classes within the set $\mathcal{P}$ should have equal marginal values of increasing capacity. Among the set $\mathcal{P}$, at most one class can have two subclasses as a result of Proposition 1. Finally, $\beta_p(n_p^*) \leq \beta_f(n_f^*)$ must hold because it states that the marginal value of allocating capacity to any class p in $\mathcal{P}$ should be less than the marginal cost of decreasing capacity from any class f in $\mathcal{F}$.

Notice that, by Lemma 2, our optimization problem is convex, and thus, the conditions listed in Proposition 2 are also sufficient. However, the objective function is not strictly convex in general, and there can be multiple optimal solutions. As we show in the following sections, with additional regularity conditions on patience time distributions and cost functions, the objective function is strictly convex so that there exists a unique optimal solution.

We next focus on specific cost functions to draw clean insights. We show how Proposition 2 applies to these special cost functions and can be further strengthened.

3.2. Abandonment Metric

For the abandonment metric, we obtain the cost function $\xi_i(x, y, w) = p_i \mathbb{I}(y < w)$ for all classes $i = 1, \ldots, m$. Thus, for a subclass with offered wait w, we have $\bar{c}_i(w) = p_i F_i(w)$.

In (6), the objective function now can be written as

$$\sum_j \lambda_{i,j} p_i F_i(w_{i,j}) = \sum_j p_i (\lambda_{i,j} - n_{i,j}\mu_i)^+,$$

where the equality follows from the definition of $w_{i,j}$ in (4). Hence, the optimization problem reduces to

$$C_i(n_i) = \min_{\{J(i),\, n_{i,j},\, \lambda_{i,j},\, j=1,\ldots,J(i)\}} \sum_j p_i (\lambda_{i,j} - n_{i,j}\mu_i)^+ \tag{15}$$

$$\text{s.t.} \quad \sum_{j=1}^{J(i)} \lambda_{i,j} = \Lambda_i,$$

$$\sum_j n_{i,j} \leq n_i.$$

It then follows from (15) that $C_i(n_i) = p_i(\Lambda_i - \mu_i n_i)^+$. Thus, the marginal gain in cost for class i when increasing the capacity allocated to this class is equal to

$$\beta_i(n_i) = \begin{cases} p_i\mu_i, & \text{if } \Lambda_i \geq n_i\mu_i, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Proposition 2 implies that the solution to (2) and (6) has a "bang-bang" structure based on the penalty cost rate $p_i\mu_i$, in which classes are processed up to the available capacity in descending order of their penalty cost rates, and once the capacity is exhausted, the remaining classes are not processed at all. Customers within each class are processed under FCFS, and so each class has only one subclass. The following result formalizes this solution.

Proposition 3. *For the abandonment metric, if the classes are ranked such that $p_i\mu_i > p_{i+1}\mu_{i+1}$ for $i = 1, \ldots, m-1$, then the solution to (2) and (6) is $J(i) = 1$ for all $i = 1, 2, \ldots, m$. Defining*

$$i' = \inf_{\ell}\left\{\sum_{i=1}^{\ell}\frac{\Lambda_i}{\mu_i} \geq N\right\},$$

we have $n_i^* = \Lambda_i/\mu_i$ for $i < i'$, $n_{i'}^* = N - \sum_{i=1}^{i'-1}\frac{\Lambda_i}{\mu_i}$ and $n_i^* = 0$ for $i > i'$.

3.3. Queue-Length Metric

We next consider the queue-length metric aggregated over customer-level holding costs. Specifically, we consider $\xi_i(x, y, w) = h_i \min\{y, w\}$ for $i = 1, \ldots, m$, which gives the following class-dependent cost function

$$\bar{c}_i(w) = h_i \int_0^w \bar{F}_i(y)dy. \tag{17}$$

This leads to a more involved fluid-optimization problem than that under the abandonment metric. We plug this cost function (17) into (8) to obtain the fluid queue-length optimization problem. We next compute the marginal value of allocating capacity to class i. To this end, we use $H_i(\cdot)$ to denote the hazard rate of the patience time distribution of class i. The marginal value is computed by noting that, within class i, the capacity is utilized and allocated to subclasses optimally.

Lemma 3. *For class i with arrival rate Λ_i and capacity $n_i < \Lambda_i/\mu_i$, the marginal value of increasing capacity to this class for the queue-length metric is given by*

$$\beta_i(n_i) = \begin{cases} \dfrac{h_i\mu_i}{H_i(w_{i,1}^*)}, & \text{if } w_{i,1}^* > 0, \\ \dfrac{h_i\mu_i}{H_i(w_{i,2}^*)}, & \text{if } w_{i,1}^* = 0,\ w_{i,2}^* < \infty, \\ \dfrac{h_i\mu_i}{\gamma_i}, & \text{if } w_{i,1}^* = 0,\ w_{i,2}^* = \infty, \end{cases} \tag{18}$$

where $(w_{i,1}^*, w_{i,2}^*)$ solves the fluid minimization problem (8) for the queue-length metric and $H_i(w) = f_i(w)/\bar{F}_i(w)$ is the hazard rate of the patience time distribution of class i.

The marginal value of increasing capacity to class i stated in Lemma 3 can be understood as follows. First, consider the case $(w_{i1}^*, w_{i2}^*) = (0, \infty)$, that is, when it is optimal to process class i under LCFS. In this case, a marginal increase in capacity leads to an increase in customers who can be processed immediately upon arrival instead of never being processed. Because the customers who are never processed wait on average $1/\gamma_i$ time units before abandoning, the corresponding decrease in cost because of the increased capacity is $\beta_i(n_i) = h_i\mu_i/\gamma_i$. Next, consider the case in which FCFS is optimal for this class so that $w_{i,1}^* = \bar{w}_i(n)$ and $w_{i,2}^* > \bar{w}_i(n_i)$. In this case, the queue-length cost $h_i\Lambda_i\int_0^{\bar{w}_i(n_i)}\bar{F}_i(y)dy$ is captured by the area under the curve $h_i\Lambda_i\bar{F}_i(y)$ over $[0, \bar{w}_i(n_i)]$ so that the marginal value of additional capacity is calculated as the product of the height of the curve, which is $h_i\Lambda_i\bar{F}_i(\bar{w}_i(n_i))$, and the marginal change in the interval length, which equals $\mu_i/(\Lambda_i f_i(\bar{w}_i(n_i)))$. Multiplying them together, we obtain $\beta_i(n_i) = h_i\mu_i/H_i(\bar{w}_i(n_i))$. Finally, when the solution is of the form $(0, w_{i,2}^*)$, (w_{i1}^*, w_{i2}^*), or (w_{i1}^*, ∞), the same logic applies, and we obtain the marginal value $\beta_i(n_i) = h_i\mu_i/H_i(w)$, where $w = w_{i,1}^*$ if $w_{i,1}^* > 0$ and $w = w_{i,2}^*$ otherwise. Note that, if $w_{i,1}^* > 0$ and $w_{i,2}^* < \infty$, then the optimality of $(w_{i,1}^*, w_{i,2}^*)$ within class i in fact implies $H_i(w_{i,1}^*) = H_i(w_{i,2}^*)$ (Bassamboo and Randhawa 2015, proposition 2).

Using the characterization of β in Lemma 3, we can draw on Proposition 2 to provide more structures of the optimal solution to the fluid queue-length optimization problem when patience time distributions have constant or monotone hazard rates. We discuss this next.

3.3.1. Exponential Patience Time: Constant Hazard Rates. Under exponential patience times, we can simplify the cost for each subclass j of class i to $\bar{c}_i(w_{i,j}) = \frac{h_i}{\gamma_i}F_i(w_{i,j})$. Notice that this is identical to the cost under the abandonment metric by setting $p_i = h_i/\gamma_i$. Thus, Proposition 3 also characterizes the optimal solution to the fluid queue-length optimization problem.

3.3.2. Increasing Patience Time Hazard Rates. Now, consider patience time distributions with increasing hazard rates. In this case, the optimal solution for each class is captured by offered waits $(0, \infty)$, that is, to process each class under LCFS (Bassamboo and Randhawa 2015, corollary 1). An intuitive explanation for this policy is that, with increasing hazard rates, when splitting a class into two subclasses, the marginal value of increasing capacity to the subclass with a lower offered wait is always greater than the marginal cost of decreasing capacity from the subclass with a higher offered wait, and thus, the optimal subclass configuration should be such that one subclass has a zero offered wait and the other has an infinite offered wait. Then, using Lemma 3, the marginal value of allocating capacity to class i is given by $\beta_i(n_i) = h_i\mu_i/\gamma_i$ for $n_i < \Lambda_i/\mu_i$. It follows that the optimal solution under the queue-length metric is similar to that under the abandonment metric; that is, it is prescribed by Proposition 3 with the abandonment

penalty p_i replaced by the queue-length related cost h_i/γ_i. Proposition 1 further implies that at most one class is processed under LCFS and all other classes, if processed, are processed under FCFS. We formalize these results as follows.

Proposition 4. *If the patience time distributions of all classes have increasing hazard rates and, further, the classes are ranked such that $h_i\mu_i/\gamma_i > h_{i+1}\mu_{i+1}/\gamma_{i+1}$ for $i = 1,\ldots, m-1$, then for the queue-length metric, the solution to (2) and (6) is $J(i) = 1$ for all $i \neq i'$ and $J(i') = 2$, where*

$$i' = \inf_{\ell}\left\{\sum_{i=1}^{\ell}\frac{\Lambda_i}{\mu_i} \geq N\right\}.$$

Moreover, we have $n_i^* = \Lambda_i/\mu_i$ for $i < i'$, $n_{i'}^* = N - \sum_{i=1}^{i'-1}\frac{\Lambda_i}{\mu_i}$ and $n_i^* = 0$ for $i > i'$. In the optimal solution, if $\sum_{i=1}^{i'}\Lambda_i/\mu_i > N$, then class i' has $w_{i'1}^* = 0$ and $w_{i'2}^* = \infty$.

3.3.3. Decreasing Patience Time Hazard Rates. Next, consider patience time distributions with decreasing hazard rates. In this case, the optimal policy for each class is to have one single subclass for that class, that is, to process the entire class under FCFS (Bassamboo and Randhawa 2015, corollary 1). Thus, the marginal value of allocating capacity to class i is given by $\beta_i(n_i) = h_i\mu_i/H_i(\bar{w}_i(n_i))$ for all $n_i < \Lambda_i/\mu_i$, and this marginal value is decreasing in capacity n_i. This implies that the objective function in (2) is strictly convex. Thus, we have the following characterization of the optimal solution.

Proposition 5. *If the patience time distributions of all classes have decreasing hazard rates, then for the queue-length metric, the cost function C_i is strictly convex for all $i = 1,2,\ldots,m$, and there is a unique characterization of the optimal solution in the form (13) and (14) that utilizes the entire capacity. Further, no class is split into two subclasses, that is, $J(i) = 1$ for all i.*

3.4. Combining Abandonment and Queue-Length Metrics

We next consider a cost function that combines abandonment and queue-length metrics. Specifically, this new cost function would be a sum of abandonment and queue-length costs, that is,

$$\bar{c}_i(w) = p_iF_i(w) + h_i\int_0^w \bar{F}_i(y)dy. \tag{19}$$

Plugging this cost function into (8), we obtain the corresponding fluid-optimization problem. As before, Proposition 2 provides a structural property of the optimal solution to this optimization problem with the marginal value of allocating capacity to class i given as follows.

Lemma 4. *For class i with arrival rate Λ_i and capacity $n_i < \Lambda_i/\mu_i$, the marginal value of increasing capacity to this class for the cost function $\bar{c}_i$ in (19) is*

$$\beta_i(n_i) = \begin{cases} p_i\mu_i + \dfrac{h_i\mu_i}{H_i(w_{i,1}^*)}, & \text{if } w_{i,1}^* > 0, \\ p_i\mu_i + \dfrac{h_i\mu_i}{H_i(w_{i,2}^*)}, & \text{if } w_{i,1}^* = 0,\ w_{i,2}^* < \infty, \\ p_i\mu_i + \dfrac{h_i\mu_i}{\gamma_i}, & \text{if } w_{i,1}^* = 0,\ w_{i,2}^* = \infty, \end{cases} \tag{20}$$

where $(w_{i,1}^*, w_{i,2}^*)$ solves the optimization problem (8).

Another plausible way of combining abandonment and queue-length metrics would be to omit the holding cost for an abandoned customer when the penalty of losing that customer is already included in the cost function. For such a system, the class-dependent cost would be

$$\bar{c}_i(w) = p_iF_i(w) + h_iw\bar{F}_i(w). \tag{21}$$

To explain this cost function (that ignores holding cost for abandoned customers), note that the holding cost is only incurred for a $\bar{F}_i(w)$ fraction of class i customers who eventually get served, and each of them waits w time units in the fluid system before getting served.

For this cost function, the following result characterizes the marginal value of allocating capacity to each class, which we use to characterize the optimal solution in the spirit of Proposition 2.

Lemma 5. *For class i with arrival rate Λ_i and capacity $n_i < \Lambda_i/\mu_i$, the marginal value of increasing capacity to this class for the cost function $\bar{c}_i$ in (21) is*

$$\beta_i(n_i) = \begin{cases} p_i\mu_i + h_i\mu_i\left(\dfrac{1}{H_i(w_{i,1}^*)} - w_{i,1}^*\right), & \text{if } w_{i,1}^* > 0, \\ p_i\mu_i + h_i\mu_i\left(\dfrac{1}{H_i(w_{i,2}^*)} - w_{i,2}^*\right), & \text{if } w_{i,1}^* = 0,\ w_{i,2}^* < \infty, \\ p_i\mu_i, & \text{if } w_{i,1}^* = 0,\ w_{i,2}^* = \infty, \end{cases} \tag{22}$$

where $(w_{i,1}^*, w_{i,2}^*)$ solves the optimization problem (8).

4. Implementing the Fluid Solution in Queueing Systems

Recall that a scheduling policy for the multi-class queueing system consists of two decisions: (1) when a server becomes idle, to which class should the server be allocated and (2) to which customer within that class should the server be allocated. The analysis of the fluid optimization in the previous section provides guidance for developing scheduling policies for the stochastic queueing system, and based on the fluid solution derived in the previous section, we propose two scheduling policies for the stochastic queueing system: a mostly-FCFS

policy that maintains within-class FCFS for all but at most one class and an mTIQ policy that optimally differentiates customers of all classes based on their time in queue. These two policies are equivalent at the fluid level, but the mTIQ policy has additional robustness to changes in system parameters.

4.1. Mostly-FCFS Policy

In the previous section, we establish that the optimal solution to the fluid-optimization problem had an important characteristic: there was at most one class, denoted by i_s, such that $J^*(i_s) = 2$; that is, in the fluid limit, customers of this class are split into two subclasses with different wait times. Given this fact, we propose a mostly-FCFS policy in which all classes except class i_s are processed under FCFS. In fact, if there does not exist i_s such that $J^*(i_s) = 2$, then all classes would be processed under FCFS. We later discuss this special case. More generally, to decide to which class an idle server should be allocated, we differentiate classes based on their marginal values of increasing capacity using the index calculated from the fluid solution.

Formally, we propose the following priority policy to implement the fluid solution in the stochastic queueing system:

1. First, process customer classes in the set $\mathcal{F}$, and prioritize these classes in descending order of $\beta_i(n_i^*)$. Within each class, process customers under FCFS.

2. Then, process customer classes in the set $\mathcal{P}$ using the following rule.

   a. First, consider customers of class $i \in \mathcal{P}$ such that $i \neq i_s$ (i.e., those with $J(i) = 1$). Process customers with wait times greater than $\bar{w}_i(n_i^*)$ and give priority to the customer who has waited longest among them.

   b. Then, if there exists a class with $J^*(i_s) = 2$ with $(w^*_{i_s,1}, w^*_{i_s,2})$ being the solution to the fluid optimization problem (8), then process this class using the TIQ policy as follows (Bassamboo and Randhawa 2015):

      i. First, process customers who have waited more than $w^*_{i_s,2}$ time units as well as customers who have waited less than $w^*_{i_s,1}$ time units. Give priority to the customer who has waited longest among them.

      ii. Then, process all remaining customers of class i_s. Give priority to the customer who has waited least among them.

Notice that, in step 1, for classes in the set $\mathcal{F}$, any prioritization policy of these classes would lead to a zero fluid offered wait of these classes because there is ample capacity to process all of them. However, because the marginal value of allocating capacity can vary across classes, we propose prioritizing these classes based on their marginal values of increasing capacity. Similarly, in step 2, the priority between (a) and (b) can be swapped because all classes in the set $\mathcal{P}$ have identical marginal values of increasing capacity.

In an overloaded system, this policy leaves no idle capacity in the steady state. However, to ensure nonidling in the stochastic system, we augment this policy by allocating any idle server remaining after the primary allocation to, first, any of the classes in $\mathcal{P}$ and, then, any of the classes in $\mathcal{E}$.

4.1.1. Settings in Which Mostly-FCFS Reduces to All-FCFS.

Note that, under the mostly-FCFS policy, for all classes except class i_s, customers within each class are served under FCFS. That is, this policy serves all classes under FCFS with the exception of class i_s. Further, if we have $J^*(i) = 1$ for all classes, then the mostly-FCFS policy reduces to an all-FCFS policy.

Specifically, if we focus on the abandonment metric, then as per Proposition 3, the mostly-FCFS policy reduces to a $p\mu$ priority policy that processes all classes under FCFS and prioritizes classes in descending order of $p_i\mu_i$. For the queue-length metric, if all patience time distributions have decreasing hazard rates, then as per Proposition 4, we again have $J^*(i) = 1$ for all classes so that the mostly-FCFS policy reduces to an all-FCFS policy. In this case, classes are prioritized in descending order of $h_i\mu_i/H_i(\bar{w}_i(n_i^*))$, where n_i^* is the solution to the fluid program (2) and $\bar{w}_i$ is the solution to (7).

4.2. Multi-class TIQ Policy

Note that implementing the mostly-FCFS policy requires computing the optimal fluid solution. We next extend the mostly-FCFS policy to a more robust mTIQ policy. This policy differentiates between customers of all classes based on their time in queue. It allocates an idle server to a target class so as to minimize the instantaneous fluid cost rate (defined in (16) for the abandonment metric and in (18) for the queue-length metric using the current server allocation). In particular,

The mTIQ policy allocates an idle server to

$$\text{a class in the set } \arg\max_i \beta_i(\min\{n_i, (\Lambda_i/\mu_i)\}), \tag{23}$$

where n_i represents the current number of servers allocated to class i. If the set $\arg\max_i \beta_i(\cdot)$ is not a singleton, then ties are broken randomly. Once the class to which to allocate the server is identified using (23), the server is assigned to a customer within that class using the TIQ policy in Bassamboo and Randhawa (2015) (this TIQ policy is analogous to the one presented in 2(b) of the mostly-FCFS policy).

The mTIQ policy is motivated by the fact that the fluid-optimization problem is convex in the capacity of each class (cf. Lemma 2). The mTIQ policy utilizes a greedy allocation rule to obtain the fluid optimal solution as it starts with zero capacity allocated to each class and then allocates in a repeated manner available capacity in a

fixed and small increment to the class with the highest marginal value of capacity until the entire capacity is utilized or the entire customer arrival rates are satisfied. Unlike the mostly-FCFS policy, which requires knowing the total capacity to compute the fluid solution (in particular, the offered waits associated with classes in the set $\mathcal{P}$), this information is not needed to implement the mTIQ policy. In this sense, the mTIQ policy is more robust to changes in capacity levels.

4.2.1. The mTIQ Policy Can Have a Robust Implementation. Consider minimizing the queue-length metric for a system with increasing patience time hazard rates. The discussion preceding Proposition 4 suggests that the optimal fluid solution for each class is $(0,\infty)$; that is, each class should be processed under LCFS. Thus, within each class, the idle server is allocated to the most recently arriving customer. Further, we have $\beta_i(n_i) = h_i\mu_i/\gamma_i$ for all $n_i < \Lambda_i/\mu_i$ (which does not depend on the current allocation n_i), and thus, the mTIQ policy prioritizes classes in descending order of $h_i\mu_i/\gamma_i$ and within each class processes customers under LCFS. This suggests that the mTIQ policy has a more robust implementation as it operates (the optimal) within-class LCFS for all classes as opposed to the mostly-FCFS policy that operates LCFS for at most one class.

If we consider patience time distributions with decreasing hazard rates, then the mTIQ policy is almost the same as the mostly-FCFS policy with all classes being processed under FCFS but with one important caveat that idle servers are allocated to classes in descending order of $h_i\mu_i/H_i(\bar{w}_i(n_i))$ in the mTIQ policy, where n_i represents the current number of servers allocated to class i rather than the fluid optimal solution n_i^* (which we use in the mostly-FCFS policy).

5. Performance of the Proposed Policy

In this section, we study the performance of our proposed policies: mostly-FCFS and mTIQ. In Section 5.1, we provide a theoretical result for our policies in Markovian systems in which the interarrival, service, and patience times are all exponentially distributed. For such systems, we show that our proposed policies reduce to an $h\mu/\gamma$ priority policy and exhibit $\mathcal{O}(1)$-optimality as the system size grows; that is, its optimality gap relative to the optimal policy remains bounded as the system size grows without bound. Then, in Section 5.2, we perform numerical studies to demonstrate the performance of our policies under nonexponential patience time distributions.

5.1 $O(1)$-Optimality in Markovian Systems

For Markovian systems, the patience time distributions are exponential, and following the discussion in Section 3.3, we obtain that the mostly-FCFS and mTIQ policies both reduce to an $h\mu/\gamma$ priority policy that prioritizes classes in descending order of $h_i\mu_i/\gamma_i$ and processes customers within each class under FCFS.

To formally state the result on the performance of this policy, we introduce the following notations. We denote the total arrival rate to all classes by Λ, and we write the individual arrival rates to each class and total number of servers in terms of Λ; that is, the arrival rate to class i is $\Lambda_i = a_i\Lambda$, where $a_i > 0$ and $\sum_{i=1}^m a_i = 1$. We fix the offered load at $\rho > 1$ by setting the number of servers as $n_\Lambda = \frac{1}{\rho}(\sum_{i=1}^m a_i/\mu_i)\Lambda$. We denote the average queue-length cost under the optimal policy by K_Λ^*, the average queue-length cost under the $h\mu/\gamma$ priority policy by $K_\Lambda^{h\mu/\gamma}$, and the optimal objective value of the fluid optimization problem (2) by $C^*(n_\Lambda,\Lambda)$.

Proposition 6. *If the interarrival, service and patience times are all exponentially distributed, the classes are ranked such that $h_i\mu_i/\gamma_i > h_{i+1}\mu_{i+1}/\gamma_{i+1}$ for $i = 1,\dots,m-1$, and for any fixed $\rho > 1$, $\sum_{i=1}^j \frac{\Lambda_i/\mu_i}{n_\Lambda} \neq 1$ for all $j = 1,\dots,m$, then the $h\mu/\gamma$ priority policy is $\mathcal{O}(1)$-optimal. That is, there exists a finite constant $A \geq 0$ such that*

$$K_\Lambda^{h\mu/\gamma} - K_\Lambda^* \leq A, \quad \text{for all } \Lambda > 0. \tag{24}$$

Further, the optimal cost is lower bounded by the solution to the fluid-optimization program. That is, $K_\Lambda^* \geq C^*(n_\Lambda,\Lambda)$.

This result shows that the optimality gap between our policy, namely, the $h\mu/\gamma$ priority policy, and the optimal policy is bounded and does not grow with the system size. This result strengthens the finding in Atar et al. (2010) that the $h\mu/\gamma$ priority policy is fluid-scale optimal for Markovian systems; that is, the optimality gap divided by the system size tends to zero as the system size grows. In terms of the technical condition we impose to obtain the $\mathcal{O}(1)$-optimality, we require the system to be effectively overloaded; that is, if we remove any class that is not processed at all in the fluid solution, then the remaining system is still overloaded. The condition $\sum_{i=1}^j \frac{\Lambda_i/\mu_i}{n_\Lambda} \neq 1$ $\left(\text{or, equivalently, } \frac{\sum_{i=1}^m \Lambda_i/\mu_i}{\sum_{i=1}^j \Lambda_i/\mu_i} \neq \rho\right)$ for all $j = 1,\dots,m$ ensures that this is the case.

5.2. Numerical Study

We use simulations to illustrate the performance of our proposed mostly-FCFS and mTIQ policies. We simulate a two-class system in which customers arrive to the system according to a Poisson process with a total arrival rate Λ. The service times of each class are exponentially distributed with unit mean, that is, $\mu_1 = \mu_2 = 1$. The queue-length holding costs are such that $h_1 = 1.5$ and $h_2 = 1$. For arrival rates $\Lambda = 25, 50, 100, 200$ corresponding to different system scales, we consider offered loads $\rho = 1.05, 1.1, 1.5$ by selecting the number of servers as $n_\Lambda = \frac{1}{\rho}(\sum_i a_i/\mu_i)\Lambda = \Lambda/\rho$.

Table 1. Performance of mTIQ Policy Relative to Fluid Lower Bound for Exponential Patience Times

	$\rho = 1.05$			$\rho = 1.1$			$\rho = 1.5$		
Arrival rate	mTIQ	Fluid	Difference	mTIQ	Fluid	Difference	mTIQ	Fluid	Difference
25	6.0	4	2.0	7.5	6	1.5	19.1	18	1.1
50	8.5	6	2.5	11.6	10	1.6	35.2	34	1.2
100	12.9	10	2.9	21.3	20	1.3	69.3	68	1.3
200	22.7	20	2.7	38.8	38	0.8	135.2	134	1.2

5.2.1. $\mathcal{O}(1)$-Performance for Markovian Systems. We start by demonstrating the $\mathcal{O}(1)$-optimality of our proposed policies for Markovian systems. In our simulations, we let the patience times of each class be exponentially distributed with mean two, that is, $\gamma_1 = \gamma_2 = 0.5$, and let the two classes share the same arrival rate, $a_1 = a_2 = 0.5$. Under exponential patience times, all customers of the same class in queue have an identical residual patience time because of the memoryless property. This reduces our proposed mostly-FCFS and mTIQ policies to the $h\mu/\gamma$ priority policy.

We compare the cost obtained from the $h\mu/\gamma$ priority policy (i.e., prioritizing class 1 over class 2 and processing each class under FCFS) with the fluid lower bound. For this numerical study, we report the average queue-length cost (per time unit) by simulating each queueing system for 10,000 time units and taking an average of 20 independent runs; the 95% confidence interval half width is less than 1% of the reported values in all cases we consider. Table 1 presents the results and provides a numerical validation of Proposition 6: for Markovian systems, the cost difference between the $h\mu/\gamma$ priority policy and the fluid lower bound remains bounded as the system size grows without bound. We also find that the cost difference is lower for systems with higher offered load ρ.

5.2.2. Performance of mTIQ Policy Under Lognormal Patience Times. We next illustrate the performance of our proposed policies under general patience time distributions. We simulate a two-class system in which both classes have lognormal patience such that the natural logarithm of the patience times is normally distributed with mean one and variance four (this is obtained by setting the mean and variance of the lognormal patience distribution to be e^3 and $(e^{10} - e^6)$, respectively). Again, we let the two classes have the same arrival rate, $a_1 = a_2 = 0.5$. We compare the performance of the mTIQ policy, which allocates available servers to minimize the instantaneous cost rate, to other reasonable benchmarks, such as the $h\mu/\gamma$ priority policy, the reverse $h\mu/\gamma$ priority (that prioritizes classes in ascending order of $h\mu/\gamma$) policy, and the virtual allocation policy in Long and Zhang (2019) (which assigns a fixed portion of servers to each class) with all these benchmark policies processing customers within each class under FCFS. Table 2 presents the results. We find that the mTIQ policy outperforms the reverse $h\mu/\gamma$ priority and virtual allocation policies in all cases. (The mTIQ and virtual allocation policies are equivalent at the fluid scale under $\rho = 1.5$, so there are only tiny cost differences between these two policies for large systems.) We also find that the mTIQ policy performs slightly worse than the $h\mu/\gamma$ priority policy when the system size ($\Lambda = 25$ and 50) and offered load ($\rho = 1.05$ and 1.1) are not too large. Indeed, one can verify that the $h\mu/\gamma$ priority policy is fluid-optimal in these cases. The mTIQ policy fails to strictly prioritize class 1 in smaller systems because of stochastic fluctuation and, thus, performs slightly worse. However, as the system size grows, the effect of stochastic fluctuation diminishes, making the mTIQ policy effectively equivalent to the $h\mu/\gamma$ priority policy. We also observe that the mTIQ policy significantly outperforms the $h\mu/\gamma$ priority policy in heavily loaded systems ($\rho = 1.5$) and the cost difference between these two policies grows with the system size. This is because these two policies have different fluid-level characteristics in heavily loaded systems: the mTIQ policy partially serves each class, whereas the $h\mu/\gamma$ priority policy completely serves class 1 by always prioritizing that class.

Table 2. A Comparison of Steady-State Queue-Length Holding Cost of mTIQ, $h\mu/\gamma$ Priority, Reverse $h\mu/\gamma$ Priority, and Virtual Allocation Policies of Long and Zhang (2019) Under Lognormal Patience Times

	$\rho = 1.05$				$\rho = 1.1$				$\rho = 1.5$			
Arrival rate	mTIQ	$\frac{h\mu}{\gamma}$	Reverse	Virtual	mTIQ	$\frac{h\mu}{\gamma}$	Reverse	Virtual	mTIQ	$\frac{h\mu}{\gamma}$	Reverse	Virtual
25	7.5	−1.9%	+33%	+12.4%	9.9	−1.4%	+33%	+2.8%	33.4	+26%	+82%	+3.6%
50	10.1	−1.8%	+38%	+13.4%	14.4	−0.5%	+41%	+3.8%	59.5	+33%	+94%	+0.9%
100	14.6	+0.0%	+42%	+16.7%	25.6	+0.6%	+45%	+2.6%	116.9	+41%	+109%	+0.4%
200	24.6	+0.1%	+46%	+14.8%	45.2	+0.4%	+47%	+2.2%	225.3	+46%	+117%	+0.3%

5.2.3. Comparing mTIQ with Mostly-FCFS. We propose two scheduling policies for multi-class queueing systems: mostly-FCFS and mTIQ. Each policy has its advantages: mostly-FCFS has the least deviation from within-class FCFS and, thus, has advantages in fairness considerations, and mTIQ does not need the information of total capacity and has robustness properties best exhibited for the queue-length metric under patience distributions with increasing hazard rates. In terms of their cost performance, both policies have the same fluid-level characteristics. Turning to smaller systems, we numerically find that there is no clear dominance between the two policies, and there are cases for which either one can dominate the other.

Recall that, under exponential patience times, both policies are trivially identical because the TIQ policy within each class is the same as FCFS. Table 3 compares the performance of the two policies under lognormal and Erlang patience distributions. We find that, between these two policies, mostly-FCFS performs slightly better under lognormal patience distributions, and mTIQ performs slightly better under Erlang patience distributions (which have increasing hazard rates).

Given the similarity in the cost performance achieved by these two policies, we recommend considering both policies for the general prescription, selecting mostly-FCFS if there is a preference to adhere as close to within-class FCFS as possible, and selecting mTIQ if there is a preference to implement a more robust policy.

6. Dependent Service and Patience Times

In this section, we consider the scheduling problem for multi-class systems with dependent service and patience times within each class. Following Bassamboo and Randhawa (2015) and Wu et al. (2018), we assume that each class i customer arrives with a finite service and patience time, which are i.i.d. draws from a class-specific bivariate distribution; its probability density function is denoted by f_i^D. We use S_i and T_i to denote the bivariate random variables representing a class i customer's service and patience times. Define

$$\phi_i(w) := \int_{x=0}^{\infty}\int_{y=w}^{\infty} x f_i^D(x,y)dydx,$$

which represents the average amount of work required by a nonabandoning class i customer who has waited w time units in queue. Further define $\mu_i := 1/\phi_i(0)$, which represents a server's average rate of processing class i customers that are not delayed in queue. For each subclass j of class i processed under FCFS, analogously to (4), the offered wait $w_{i,j}$ solves

$$\lambda_{i,j}\phi_i(w_{i,j}) = \min\left\{n_{i,j}, \frac{\lambda_{i,j}}{\mu_i}\right\}. \tag{25}$$

Using the offered wait in (25), the abandonment cost for this subclass is $\lambda_{i,j}\bar{c}_i(w_{i,j}) = p_i\lambda_{i,j}F_i(w_{i,j})$. Notice that this abandonment cost is in general not equal to $p_i(n_i\mu_i - \lambda_i)^+$ because the average processing rate of nonabandoning class i customers (Wu et al. 2018 terms this rate the effective service rate) may be different than μ_i. The queue-length cost for this subclass is $h_i\lambda_{i,j}\int_0^{w_{i,j}}\bar{F}_i(y)dy$ by applying Little's law.

Similar to Section 3.2, when focusing on the fluid model, we can formulate the policy optimization problem as a two-stage optimization problem: the first stage optimizes the capacity allocation across classes as stated in (2), and the second stage optimizes the capacity allocation across subclasses within each class. The counterpart of (6) for systems with dependent service and patience times within each class is as follows:

$$\begin{aligned}\inf_{\{J(i),\, w_{i,j},\, \lambda_{i,j},\, j=1,\ldots,J(i)\}} \quad & \sum_{j=1}^{J(i)}\lambda_{i,j}\bar{c}_i(w_{i,j}) \\ \text{s.t.} \quad & \sum_{j=1}^{J(i)}\lambda_{i,j} = \Lambda_i, \\ & \sum_{j=1}^{J(i)}\lambda_{i,j}\phi_i(w_{i,j}) \le n_i.\end{aligned} \tag{26}$$

Proposition 1 extends to the new fluid-optimization problems (2) and (26). In other words, the dependence between service and patience times within each class does not alter the structure of the fluid-optimal policy that implements within-class differentiation for at most one class.

To provide more insights into the fluid optimal policy, we consider a special class of joint service and patience

Table 3. A Comparison of mTIQ and Mostly-FCFS Policies

	Erlang patience				Lognormal patience			
	$\rho = 1.05$		$\rho = 1.5$		$\rho = 1.05$		$\rho = 1.5$	
Arrival rate	mTIQ	mostly-FCFS	mTIQ	mostly-FCFS	mTIQ	Mostly-FCFS	mTIQ	Mostly-FCFS
25	11.0	+0.5%	31.9	+4.1%*	7.5	−1.9%*	33.5	−1.5%*
50	15.8	+0.2%	55.7	+1.3%*	10.1	−1.7%*	59.6	−0.8%*
100	24.1	+0.1%	107.1	+0.4%*	14.6	−1.0%*	116.6	−0.2%
200	41.4	+0.0%	206.3	+0.0%	24.7	−0.4%	225.8	−0.1%

*Statistically significant at the 5% significance level.

distributions generated by the Gaussian copula (see Cario and Nelson (1997) for a background of copulas and Wu et al. (2018) for an application of the Gaussian copula in queueing systems), and we denote this class of joint distributions by $\mathcal{G}_i := \mathcal{G}_i(f_{S_i}, f_{T_i})$ with f_{S_i} and f_{T_i} being the marginal distributions of the service and patience times of class i. The Gaussian copula is a useful and flexible instrument to construct bivariate distributions with any arbitrary marginals and attainable correlation coefficients r_i (Cario and Nelson 1997). To state our result, further define the conditional service time function $g_i(t) := \mathbb{E}[S_i \mid T_i = t]$, which represents the average amount of work required by a class i customer whose patience equals t time units.

We discuss how the structure of the fluid-optimal solution is affected by the dependence between the service and patience times within each class. We focus on the abandonment metric because optimizing this metric yields a simple $p\mu$ priority policy in the absence of dependence and provides a good benchmark with which to compare. The analysis of the queue-length metric follows analogously but creates additional complexity without offering new insights, so we omit it for brevity.

Recall that, when each class has independent service and patience times, the optimal policy is a $p\mu$ priority policy that prioritizes classes up to available capacity in descending order of their penalty cost of abandonment, and the marginal value of increasing capacity to each class is given by (16). Now, for systems with dependent service and patience times within each class, if g_i is increasing for all classes, then customers within each class should be processed under LCFS (Bassamboo and Randhawa 2015, Wu et al. 2018). The marginal value of increasing capacity to a class remains unchanged, namely, $\beta_i(n_i) = p_i\mu_i$ for $n_i < \Lambda_i/\mu_i$. As a result, the $p\mu$ priority policy that prioritizes classes in descending order of $p\mu$ remains optimal. However, if g_i is decreasing for all classes, then customers within each class should be processed under FCFS (Bassamboo and Randhawa 2015, Wu et al. 2018). Correspondingly, the marginal value of increasing capacity to class i should be revised to

$$\beta_i(n_i) = p_i / g_i(\bar{w}_i(n_i)) \text{ for } n_i < \Lambda_i/\mu_i,$$

where $\bar{w}_i(n_i)$ is the offered wait of class i processed under FCFS with capacity n_i.

Based on these observations, we next provide a structural property of the fluid-optimal solution for systems with service and patience times generated by the Gaussian copula within each class.

Proposition 7. *Consider minimizing the abandonment cost.*

i. *Suppose* $(S_i, T_i) \in \mathcal{G}_i$ *with* $r_i > 0$ *for each class i. Then, the* $p\mu$ *priority policy is optimal, and the optimal capacity allocation follows the structure specified in Proposition 3.*

ii. *Suppose* $(S_i, T_i) \in \mathcal{G}_i$ *with* $r_i < 0$ *for each class i. Define sets of classes* $\mathcal{F} := \{l : n_l^* = \Lambda_l/\mu_l\}$, $\mathcal{P} := \{l : 0 < n_l^* < \Lambda_l/\mu_l\}$ *and* $\mathcal{E} := \{l : n_l^* = 0\}$. *If* $\sum_{i=1}^m \Lambda_i/\mu_i > N$, *then* $\mathcal{E} = \emptyset$, $\mathcal{F} = \emptyset$, *and* $\beta_p(n_p^*) = \beta_{p'}(n_{p'}^*)$ *for any* $p, p' \in \mathcal{P}$. *If* $\sum_{i=1}^m \Lambda_i/\ \mu_i \le N$, *then all classes are in the set* $\mathcal{F}$.

Proposition 7 shows that the optimal policy under a negative dependence between the service and patience times within each class can be significantly different than the $p\mu$ priority policy, which is otherwise optimal under a positive dependence or no dependence within each class. Under a negative dependence, each class should be allocated some capacity because the marginal value of increasing capacity to a class that is not served at all can be extremely large. This is in contrast to the $p\mu$ priority policy, which, under a limited total capacity, chooses not to allocate any capacity to classes with low abandonment costs. Further, under a negative dependence, if the system is overloaded with insufficient capacity to process all arrivals, then no class should be served completely because the marginal cost of decreasing capacity from a fully served class can be extremely small. In this case, each class should be allocated some capacity but not to the full level. This again is in contrast to the $p\mu$ priority policy, which prioritizes classes with high abandonment costs and serves those classes completely.

The mTIQ policy proposed in Section 4.2 extends in a straightforward manner to systems with dependent service and patience times within each class. Whenever a server becomes available, the mTIQ policy allocates the server to the class with the highest instantaneous cost gradient by substituting the new expressions of $\beta_i(\cdot)$ into (23) and allocates this server to a customer within that class using the TIQ policy. We conduct a numerical study to demonstrate the performance of this policy. We follow the setting in Section 5.2 and consider a two-class system in which customers arrive to the system according to a Poisson process with total arrival rates $\Lambda = 25, 50, 100$, and 200. Class i, $i = 1, 2$, has an arrival rate of $a_i\Lambda$, where $a_1 = a_2 = 0.5$. The abandonment penalties are $p_1 = 1$ and $p_2 = 1.5$. For each arrival rate Λ, we consider different offered loads $\rho = 1.05, 1.1$, and 1.5 by selecting the number of servers as $n_\Lambda = \frac{1}{\rho}(\sum_i a_i/\mu_i)\Lambda$. The service and patience times of both classes are exponentially distributed with means one and two, respectively. They are independent for class 1 and are generated by the Gaussian copula with correlation coefficient $r_2 = -0.6$ for class 2.

We compare the performance of our mTIQ policy with other simple policies, namely, the $p\mu$ priority policy and the reverse $p\mu$ priority policy in Table 4. We find that the mTIQ policy significantly outperforms the $p\mu$ priority policy (which is the optimal policy that minimizes the abandonment cost in the absence of within-class dependence) in all cases, especially for large systems. This suggests a substantial loss in cost performance if one devises scheduling policies by considering the system

Table 4. A Comparison of Abandonment Cost Under mTIQ, $p\mu$ Priority, and Reverse $p\mu$ Priority Policies for Negatively Dependent Service and Patience Times

	$\rho = 1.05$			$\rho = 1.1$			$\rho = 1.5$		
Arrival rate	mTIQ	$p\mu$	Reverse	mTIQ	$p\mu$	Reverse	mTIQ	$p\mu$	Reverse
25	2.2	+3.7%	+2.0%	2.7	+6.9%	+3.3%	6.0	+8.0%	+14%
50	3.0	+14%	+1.2%	4.2	+17%	+0.9%	12.8	+17%	+10%
100	4.3	+26%	−0.2%	6.5	+38%	+0.7%	24.0	+25%	+8.4%
200	6.7	+46%	−0.1%	19.6	+61%	−0.1%	48.6	+30%	+6.9%

primitives within each class to be independent when they are indeed dependent. We also observe that, in some cases with moderate offered loads ($\rho = 1.05$ and 1.1), the mTIQ policy performs slightly worse than the reverse $p\mu$ priority policy. The cost differences between these two policies, however, are statistically insignificant and are covered by the 95% confidence interval. Indeed, in these cases, the mTIQ policy and the reverse $p\mu$ priority policy have the same fluid-level characteristics. For heavily loaded systems ($\rho = 1.5$), the mTIQ policy deviates from the reverse $p\mu$ priority policy at the fluid scale and, thus, outperforms the latter.

7. Conclusion

In this paper, we propose cost-minimizing scheduling policies for queueing systems with multiple customer classes characterized by their service and patience time distributions and cost parameters. Our policies differentiate between customers across classes and further within each class based on the time they have spent in queue. We propose two policy types: mostly-FCFS and mTIQ, each performing reasonably well with its own useful properties. Mostly-FCFS is, as its name suggests, very close to processing all customer classes under FCFS and, in fact, processes at most one class under non-FCFS. This endows mostly-FCFS with fairness benefits. The mTIQ policy differentiates between customers of all classes and is robust to changes in system parameters. Under exponential patience times, both policies are equivalent to a priority policy that we formally prove to be $\mathcal{O}(1)$-optimal for Markovian systems; that is, the optimality gap remains bounded even if the optimal cost grows without bound.

Our work can be extended in multiple directions. For the most part, we focus on analyzing the fluid model and translating the fluid solution to implementable scheduling policies for stochastic queueing systems. Our theoretical results for exponential patience suggest that similar results may hold in nonexponential patience settings. A formal analysis of the asymptotic performance of our proposed policies in these more general settings constitutes an interesting avenue for future study.

Our formulation of customer-based cost objectives can be extended to incorporate a reference effect that captures the notion of fairness. Considering fairness is especially important to classes that are processed under non-FCFS. We expect that a similar result to Proposition 1 (i.e., at most one class is processed under non-FCFS) continues to hold in this setting, but the index β that measures the marginal cost benefit of increasing capacity to a class must be revised to reflect fairness considerations. A formal analysis of these fairness-based settings is an interesting direction for future study.

We also believe that the topic of dependent service and patience times should be explored further. In our work, we obtain tractability by focusing on copulas, which provide us with a specific framework for an insightful analysis. Extending this framework to a broader class of dependencies tied to empirically observed characteristics would be another valuable endeavor.

Acknowledgments

The authors thank the department editor Morris Cohen, the associate editor, and two anonymous reviewers for their careful reading of the paper and for providing constructive feedback.

References

Atar R, Giat C, Shimkin N (2010) The $c\mu/\theta$ rule for many-server queues with abandonment. *Oper. Res.* 58(5):1427–1439.

Bassamboo A, Randhawa RS (2010) On the accuracy of fluid models for capacity sizing in queueing systems with impatient customers. *Oper. Res.* 58(5):1398–1413.

Bassamboo A, Randhawa RS (2015) Scheduling homogeneous impatient customers. *Management Sci.* 62(7):2129–2147.

Bassamboo A, Randhawa RS, Zeevi A (2010) Capacity sizing under parameter uncertainty: Safety staffing principles revisited. *Management Sci.* 56(10):1668–1686.

Cario MC, Nelson BL (1997) Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. Technical report.

Dai J, Tezcan T (2008) Optimal control of parallel server systems with many servers in heavy traffic. *Queueing Systems* 59(2):95–134.

Down DG, Koole G, Lewis ME (2011) Dynamic control of a single-server system with abandonments. *Queueing Systems* 67(1):63–90.

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

Kang W, Ramanan K (2010) Fluid limits of many-server queues with reneging. *Ann. Appl. Probab.* 20(6):2204–2260.

Kim J, Randhawa RS, Ward AR (2018) Dynamic scheduling in a many-server, multiclass system: The role of customer impatience in large systems. *Manufacturing Service Oper. Management* 20(2): 285–301.

Long Z, Zhang J (2019) Virtual allocation policies for many-server queues with abandonment. *Math. Methods Oper. Res.* 90(3):399–451.

Long Z, Shimkin N, Zhang H, Zhang J (2020) Dynamic scheduling of multiclass many-server queues with abandonment: The generalized cμ/h rule. *Oper. Res.* 68(4):1218–1230.

Reich M (2012) The offered-load process: Modeling, inference and applications. Unpublished PhD thesis, Technion-Israel Institute of Technology, Haifa.

Whitt W (2006) Fluid models for multiserver queues with abandonments. *Oper. Res.* 54(1):37–54.

Wu C, Bassamboo A, Perry O (2018) Service system with dependent service and patience times. *Management Sci.* 65(3):1151–1172.

Zhang J (2013) Fluid models of many-server queues with abandonment. *Queueing Systems* 73(2):147–193.