Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

SCHEDULING QUEUES WITH SIMULTANEOUS AND HETEROGENEOUS REQUIREMENTS FROM MULTIPLE TYPES OF SERVERS

Noa Zychlinski

Industrial Engineering and Management
Technion Institute of Technology
Haifa 32000, ISRAEL

Carri W. Chan, Jing Dong

Graduate School of Business
Columbia University
New York, NY 10027, USA

ABSTRACT

We study the scheduling of a new class of multi-class multi-pool queueing systems where different classes of customers have heterogeneous – in terms of the type and amount – resource requirements. In particular, a customer may require different numbers of servers from different server pools to be allocated simultaneously in order to be served. We apply stochastic simulation to study properties of the model and identify two types of server idleness: avoidable and unavoidable idleness, which play important, but different, roles in dictating system performance, and need to be carefully managed in scheduling. To minimize the long-run average holding cost, we propose a generalization of the $c\mu$-rule, called Generalized Idle-Aware (GIA) $c\mu$-rule. We provide insights into how to set the hyper parameters of the GIA $c\mu$-rule. We also demonstrate that, with properly chosen hyper parameters, the GIA $c\mu$-rule achieves superior and robust performance compared to reasonable benchmarks.

1 INTRODUCTION

In this paper, we develop a special parallel processing network with multiple types of resources and multiple classes of customers. Each class of customers can require multiple types and/or multiple units of resources simultaneously to get served. This model is relevant for many operations management applications. For example, in healthcare systems, patients are classified based on the level of medical attention/supervision they require. Each class of patients requires multiple types of resources (physicians/nurses/beds/medical equipment), as well as, a different amount of each type of resources. In an Intensive Care Unit (ICU), high acuity patients typically require one dedicated nurse per patient; however, the same nurse can take care of two to three ICU patients at lower acuity levels (Masterson and Baudouin 2015). We need both a bed and the required staff to admit a patient. Other examples include emergency services such as firefighting and police patrol (Altay 2012); manufacturing, and inventory systems (Ramakrishnan and Gannon 2008).

We study the scheduling of the proposed model, with the objective of minimizing the long-run average holding cost. Due to the complex architecture and dynamics of the proposed network model, very few analytical results can be derived. In this context, discrete-event simulation is the main tool for performance evaluation and optimization of these systems (Glynn and Asmussen 2007).

The heterogenous resource requirements pose challenges on managing priority-induced idleness. In particular, policies that myopically maximize the cost-reduction rate may lead to system instability, even if the system can be stabilized under some properly designed policies. In this work, we identify two types of idleness: *avoidable* idleness and *unavoidable* idleness. We demonstrate that these two types of idleness need to be carefully managed to achieve system stability. We then propose a class of scheduling rules that carefully balance the cost-reduction rate and the two types of idleness. We refer to this class of policies as the Generalized Idle-Aware (GIA) $c\mu$-rule. Under this policy, the scheduling decision at each event time can be formulated as an integer max-min problem. By properly choosing the hyper parameters in the max-min problem, we can decide how much weight we put on maximizing the instantaneous cost-reduction rate, and how much weight we put on minimizing the two types of idleness.

Using extensive simulation experiments, we provide insights into how to choose the hyper parameters, which we refer to as the idle-aware parameters. The numerical results demonstrate the superior and robust performance of the GIA $c\mu$-rule over naive benchmarks. In addition, when dealing with highly non-stationary demand, our proposed policy also performs well when considering the cumulative cost incurred over a finite time-horizon, i.e., transient cost-minimization problems.

1.1 Brief Literature Review

Scheduling parallel processing networks has important implications for various engineering and business applications, and is a very challenging problem due to the large state-space and policy-space involved (Papadimitriou and Tsitsiklis 1999). Two classes of methods are commonly used to tackle these problems. One is asymptotic approximations; see, for example, Mandelbaum and Stolyar (2004). The other is discrete even simulation; see, for example, Mandelbaum and Feldman (), Ma and Whitt ().

Our work is a direct extension of Zychlinski et al. (2020), which studies a similar parallel processing network but with only a single type of resource. The extension from a single type of resource to multiple types of resources is highly non-trivial, as there is no notion over which to differentiate the avoidable and unavoidable idleness when there is a single type of resource. In Section 3, we demonstrate that a naive extension of the policy developed in Zychlinski et al. (2020) to our setting can lead to very poor performance. Multi-class queues where different classes of customers have different resource requirements are also studied in Green (1981) and Reiman (1991). Green (1981) propose a heuristic scheduling policy that prioritizes jobs with more resource requirements. Reiman (1991) studies the system with blocking and develop asymptotic approximations for the blocking probability.

Previous research has shown the importance of managing idleness when scheduling parallel processing networks (Harrison 1998). Policies that are throughput optimal have been developed in the literature (Armony and Bambos 2003). In this paper, we show that when there are simultaneous resource requirements for several types of servers, managing the idleness has to be done in two levels: first, manage the avoidable idleness and then the unavoidable one. Gurvich and Van Mieghem (2017) study scheduling of parallel processing networks with collaboration and multi-tasking. In such networks, they show that the network capacity can be smaller than the capacity of the bottleneck resource. They then propose scheduling policies that are throughput optimal. In our setting, we add the extra feature that different customers can also require different units of resources. We also explicitly take holding cost into account.

Scheduling jobs with different resource requirements were first studied in communication/computer systems. In those systems, different jobs may require a different amount of memory and CPU capacity (Grandl et al. 2014). The question of how to fairly share the available bandwidth between competing streams has been extensively studied; see, for example Kelly et al. (1998), Massoulié and Roberts (). The difference between these models and ours is the integrality constraints, which do not allow us to partially admit jobs. This difference poses the challenge of properly managing priority-induced idleness.

2 THE MODEL

We consider a parallel processing network with I classes of customer and J types of servers. There can be multiple servers of each type. Let $N = (N_1, \dots, N_J) \in \mathbb{N}^J$ denote the number of servers in each pool, where $\mathbb{N}$ denotes the set of natural numbers including 0. We further introduce a matrix $M = \{M_{ji}\}_{1 \le j \le J, 1 \le i \le I} \in \mathbb{N}^{J \times I}$ to denote the resource requirements of different classes of customers. In particular, M_{ji} denotes the number of Type j resource required by a Class i customer. We focus on Markovian systems with independent Poisson arrival processes and exponential service times. Let $\{\lambda_i(t) : t \ge 0\}$ denote the arrival rate function of Class i. Then, the cumulative number of Class i arrivals up until time t follows a Poisson distribution with rate $\int_0^t \lambda_i(u) \mathrm{d}u$. We also write μ_i as the service rate of Class i and the service times of Class i customers are independent and identically distributed exponential random variables with rate μ_i. A scheduling policy determines how to allocate available servers to different classes of customers. Customers within each class are served on a first-come-first-served basis.

Figure 1 provides an illustration of our model with two classes of customers and two types of resources (i.e., $I = 2$, $J = 2$). There are $N_1 = 3$ servers of Type 1 (each represented by a triangle), and $N_2 = 4$ servers of Type 2 (each represented by a circle). Additionally, we have $M = [1 \;\; 1 \;;\; 2 \;\; 1]$; that is, each Class 1 customer requires one Type 1 server and two Type 2 servers; each Class 2 customer requires one server of each type. There are three Class 1 customers and two Class 2 customers in both scenarios illustrated in the figure. In Scenario A, two Class 1 customers are admitted to service, while one Class 1 customer and two Class 2 customers wait in queue. This leaves one Type 1 server (triangle) idle. In Scenario B, one Class 1 customer and two Class 2 customers are admitted to service, while two Class 1 customers wait in queue. In this scenario, all servers are utilized.

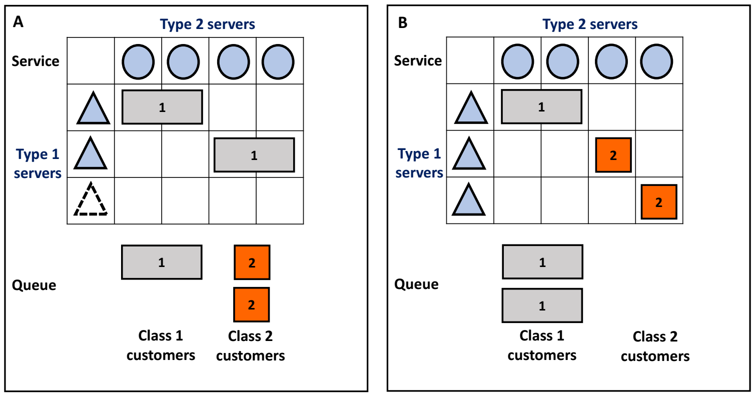

Figure 1: Model illustration – Two classes of customers and two types of resources.

Let $X_i(t)$ denote the number of Class i customers in the system at time t. Let $\pi(t) = (\pi_1(t), \ldots, \pi_I(t))$, where $\pi_i(t)$ is the number of Class i customers in service at time t. $\pi = \{\pi(t) : t \geq 0\}$ can then be interpreted as the scheduling policy (it is determined by the scheduling policy). We restrict ourselves to non-anticipative preemptive policies.

Let c_i denote the holding cost rate of a Class i customer. Our objective is to find a scheduling policy that minimizes the long-run average holding cost, i.e.,

$$\min_{\pi \in \Omega} \; \limsup_{T \to \infty} \frac{1}{T} \int_{t=0}^{T} \sum_{i=1}^{I} E[c_i X_i(u)] \mathrm{d}u, \tag{1}$$

where Ω denote the set of admissible controls.

The scheduling problem (1) is an infinite-horizon continuous-time Markov decision process (MDP). Note that the state space is countable and the action space at each state is compact, so it is without loss of optimality to consider deterministic Markovian policies only (Puterman 2005). In particular, at each $t \geq 0$, $\pi(t)$ can be viewed as a mapping from the state of Markov chain, $X(t)$, to the admitted number of customers. The huge state-space and policy space of the MDP (even after proper truncation) makes it prohibitively expensive to solve. Moreover, even if we can solve the MDP approximately, the resulting policy may be highly dependent on system primitives and hard to implement in practice. Our goal in this paper is to find scheduling policies that achieve good and robust performance and are easy to implement.

Lastly, we comment that since our objective function is formulated as the long-run average cost, we need some notion of long-run regularity of the arrival rate, e.g. periodicity, for the long-run average to be well defined (Heyman and Whitt 1984). We also focus on parameter regimes for the system primitives (e.g., arrival rates, service rates, and number of servers) under which the system can be stabilized under some controls. To facilitate subsequent discussions, when $\lambda_i(t) = \lambda_i$, i.e., is a constant. We define the traffic intensity of the system, ρ, as

$$\begin{aligned} &\min \rho \\ &\text{s.t. } \sum_{k=1}^{K} \alpha_k \phi_k(i) \mu_i = \lambda_i \text{ for } i = 1 \ldots, I; \quad \sum_{k=1}^{K} \alpha_k \le \rho; \\ &\rho \ge 0, \quad \alpha_k \ge 0 \text{ for } k = 1, \ldots K, \end{aligned} \tag{2}$$

where $\phi_k = (\phi_k(1), \ldots, \phi_k(I))$ denote the k-th possible service configuration, i.e., $\phi_k(i)$ is the number of Class i customers admitted into service under configuration k. In particular, ρ can be considered as a measure of the 'network' load (Gurvich and Van Mieghem 2015). As will be seen in Section 3.2, with $\rho < 1$, the system is stabilizable.

3 MANAGING IDLENESS

As can be seen from Figure 1, different scheduling rules can induce different levels of idleness (e.g., Scenario A versus Scenario B). When the system is critically loaded (ρ close to 1), it is important to properly manage the policy-induced idleness.

A natural way to avoid policy-induced idleness is to add a penalty term to the amount of incurred idleness when evaluating a scheduling rule. For example, Zychlinski et al. (2020) propose a scheduling policy that balances the $c\mu$-index and the idleness incurred through an integer program (IP). We can easily adapt their idea to our setting. In particular, define

$$\begin{aligned} \max_z \; & \sum_{i=1}^{I} c_i \mu_i z_i + \Gamma^{(0)} \sum_{j=1}^{J} \sum_{i=1}^{I} M_{ji} z_i \\ \text{s.t.} \; & Mz \le N \\ & 0 \le z \le x, \quad z_i \in \mathbb{N}, \qquad i = 1, \ldots, I, \end{aligned} \tag{3}$$

where $\Gamma^{(0)} \ge 0$ is a hyper parameter for idle-awareness, i.e., by maximizing the objective function in (3), we try to utilize as many servers as possible. We denote by G^0 the mapping from x to the optimal z defined by the IP (3). Then, the corresponding scheduling policy sets $\pi(t) = G^0(X(t))$. We refer to this policy as the naive idle-aware $c\mu$-rule.

We next use simulation to evaluate the performance of this policy and other benchmark policies. In the next and subsequent simulation studies, we plot how the average number of customers in the system evolves over time for systems starting from some pre-specified initial state. This provides a good amount of details on the system dynamics, including stability. In these experiments, the average number of customers in the system is estimated based on 20 independent replications. Different systems with different primitives are used in different examples. We provide more details about the system parameters in the caption of the figures. When the system is stable, we also compare the long-run average costs under different policies in some experiments. These long-run average are estimated using long-time average for $T = 6 \times 10^3$.

We next demonstrate the performance of G^0 through a simple numerical example. Consider a system with $N = (3,3)$ and $M = [1 \;\; 1 \; ; \; 1 \;\; 3]$. Figure 2 illustrates the four possible service configurations in this system (in addition to the trivial configuration under which no customer is admitted to service). Figure 3 shows the average number of customers in the system as a function of time. In addition to the naive idle-aware $c\mu$-rule, we also consider the classic $c\mu$-rule where we prioritize the class with a larger $c_i\mu_i$ index, and SNOS (smallest number of servers first). SNOS was proposed in Green (1981) for a single-type of servers. In our case, we adjust it to prioritize customers that require the smallest number of servers in total (i.e., Class 2 in this example). We observe that the classic $c\mu$-rule, the naive idle-aware $c\mu$-rule, and SNOS all fail to stabilize the system, while this particular system can be stabilized. In particular, under the system primitives in Figure 3, the traffic intensity defined in (2) is strictly less than 1.

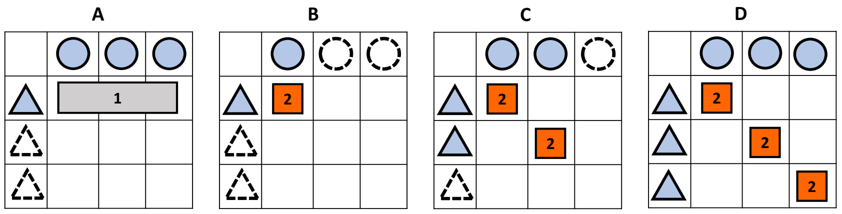

Figure 2: Possible service configurations when $N = (3,3)$ and $M = [1 \;\; 1 \;;\; 1 \;\; 3]$. The white dashed line resources represent the idle servers.

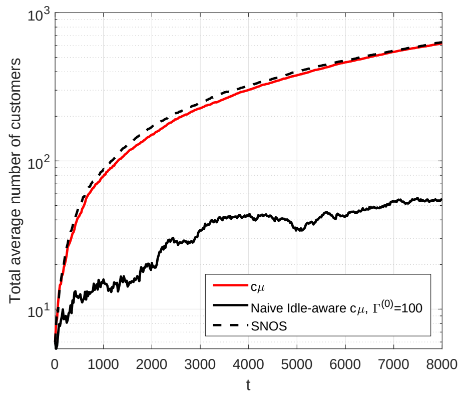

Figure 3: The average number of customers (two classes combined) in the system as a function of time over a finite horizon $T = 8 \times 10^3$, under different scheduling policies. ($N = (3,3)$, $M = [1 \;\; 1 \;;\; 1 \;\; 3]$, $\mu = (0.34, 0.8)$, $\lambda = (0.25, 0.45)$, $c = (1, 0.5)$, $x(0) = (5,2)$.)

3.1 Different Types of Idleness

The observation from Figure 3, which we also see in many other examples, motivates us to look closely into the different types of idleness. Specifically, we distinguish between two types of idleness: avoidable idleness and unavoidable idleness.

Definition 1 Unavoidable Idleness: A service configuration induces unavoidable idleness if it is infeasible to admit more customers to service, even though there are idle servers.

Definition 2 Avoidable Idleness: A service configuration induces avoidable idleness if at least one additional customer could be admitted to service if there were such customers waiting in the queue.

Configuration A in Figure 2 includes unavoidable idleness: even though two servers of Type 1 are idling, no other customers can be admitted into service. Configurations B and C induce avoidable idleness: if there were more Class 2 customers in the system, they could have been admitted. In Configuration D there is no idleness of any type. Note that in this example, even though serving Class 1 customers incurs unavoidable idleness of two servers, Configuration A has to be used in order to serve Class 1 customers. On the other hand, it is possible to serve Class 2 customers according to Configuration D, without inducing any amount of idleness. Thus, under a reasonable policy, even though Configuration A and Configuration C have the same amount of idle servers, Configuration A should be preferred to Configuration C. This suggests that the two types of idleness need to be managed differently.

3.2 The Generalized Idle-aware (GIA) $c\mu$ Rule

Based on the definition of the two types of idleness, we next introduce a modification to the naive idle-aware $c\mu$-rule where we allow different penalties for the two different types of idleness.

We first define the following IP max-min problem:

$$\begin{aligned} \max_{z}\ \min_{v} R(z,v) &:= \sum_{i=1}^{I} c_i\mu_i z_i - \Gamma_1 \sum_{j=1}^{J}\sum_{i=1}^{I} M_{ji} v_i - \Gamma_2 \sum_{j=1}^{J} u_j \\ \text{s.t.}\quad & M(z+v)+u = N \\ & 0 \le z \le x, \quad z_i, v_i, u_j \in \mathbb{N}, \qquad i=1,\dots,I,\ \ j=1,\dots,J, \end{aligned} \tag{4}$$

where $\Gamma_1, \Gamma_2 > 0$, are hyper parameters for idle-awareness. To understand the intuition behind (4), we note that $v = (v_1, \dots, v_I)$ can be interpreted as virtual customers. The inner minimization problem tends to push v to be as large as possible. Thus, the resulting Mv quantifies the amount of avoidable idleness. Meanwhile, from the first constraint, after sending v to its maximum possible value, u quantifies the amount of unavoidable idleness. By introducing two different tuning parameters, Γ_1 and Γ_2, we can put different weight on the two types of idleness. Let G denote the mapping from state x to the optimal z defined in (4). Our proposed policy sets $\pi(t) = G(X(t))$. We refer to this new policy as the Generalized Idle-Aware (GIA) $c\mu$-rule.

To implement the policy, (4) only needs to be solved at event times (arrival or departure epochs). Solving (4) has limited computational burdens in most settings. We would first try to prioritize the classes according to their $c\mu$-index. In the 'boundary' states where we incur some idleness after admitting jobs according to the $c\mu$-index, we can try swapping some of the admitted jobs with waiting jobs of other classes to reduce idleness. In general, if we have enough servers to admit all the jobs or if we have enough high index jobs to fill up all the servers, the solution would be straightforward. We only need to solve (4) in the 'boundary' cases. In addition, the IP for these states can be solved off-line and stored. Then, we only need to look up the solution whenever encountering these states.

In the next section, we study the performance GIA $c\mu$ rule with different values of Γ_1 and Γ_2 using simulation. As a quick preview, the left plot in Figure 5 consider the same system as in Figure 3. We observe that opposed to the $c\mu$ rule and the naive idle-aware $c\mu$ rule, the GIA $c\mu$ rule with $\Gamma_1 = 10$ and $\Gamma_2 = 1$ stabilizes the system with an estimated long-run average cost of 10.

4 IDLE-AWARE PARAMETERS

The GIA $c\mu$-rule is a very flexible class of policies. Note that if $\Gamma_1 = \Gamma_2 = 0$, we retrieve the $c\mu$-rule, and when $\Gamma_1 = \Gamma_2$, we retrieve the naive idle-aware $c\mu$-rule. As was shown in Figure 3, both policies can lead to poor performances due to instability. Thus, it is important to set some basic rules as how to choose the idle-aware parameters.

Our first rule is that both parameters need to be positive, i.e., both types idleness need to be managed. To see this, we consider two different models. The first one is the one shown in Figure 2. The second one is illustrated in Figure 4. Note that both Configurations A and B in Figure 4 do not incur any avoidable idleness. However, Configuration A incurs less unavoidable idleness than Configuration B. Figure 5 shows the average number of customers in the system as a function of time in the two models (left versus right) under the GIA $c\mu$-rule with different values of (Γ_1, Γ_2). We observe that in the left plot, having $\Gamma_1 = 0$ leads to instability. That is because when $\Gamma_1 = 0$, under the system parameters, Configurations B, C, and D are preferred to Configuration A in Figure 2. In other words, Class 1 customers are served only when there are no Class 2 customers in the system. The Class 1 queue blows up in this case as we incur too much avoidable idleness when implementing Configurations B and C. In the right plot we see that $\Gamma_2 = 0$ can lead to instability. This is because when $\Gamma_2 = 0$, under the system parameters, Configuration B is preferred to Configuration A in Figure 4. In this case, we end up serving the Class 2 queue too fast that when we switch to serve the Class 1 queue, there is no Class 2 customer left, i.e., we can serve only one Class 1 customer while incurring two units of avoidable idleness. Thus, even though the Class 2 queue is maintained close to zero, the Class 1 queue blows up.

Figure 5 also shows that Γ_1 needs to be larger than Γ_2; i.e., we should put more penalty (weight) on avoidable idleness than unavoidable idleness. Thus, *our second rule is that* Γ_1 *should be larger than* Γ_2*, i.e., it is more important to reduce avoidable idleness.*

Our third rule is that both Γ_1 *and* Γ_2 *need to be sufficiently large.* To see this, in Figure 6, we compare the average number of customers as a function of time for the two models (left versus right) under the GIA $c\mu$-rule with different values of (Γ_1,Γ_2) satisfying $\Gamma_1 \gg \Gamma_2$. We use a periodic time-varying arrival rate in this figure. We observe that having $\Gamma_1 \gg \Gamma_2 > 0$ alone may not be enough. Both Γ_1 and Γ_2 need be sufficiently large, i.e., we need to put enough weight on both types of idleness, to ensure system stability.

To sum up, we note that the "appropriate" values of (Γ_1,Γ_2) can be highly dependent on system parameters, i.e., c, μ, N and M. For example, setting $\Gamma_1 = 1$ and $\Gamma_2 = 0.1$ stabilizes the system in the left plot of Figure 6, but fails to stabilize the system in the right plot of that figure. We next take a closer look at the two examples in Figure 6. In the left plot (for the system depicted in Figure 2), it is important to make sure that Γ_1 is large enough so that Configuration A is preferred to Configurations B and C (i.e. eliminate the avoidable idleness). In the right plot (for the system depicted in Figure 4), it is important to make sure that Γ_2 is large enough so that Configuration A is preferred to Configuration B (i.e. eliminate the unavoidable idleness). More generally, from the perspective of throughput optimality, we need to make sure that Γ_1 and Γ_2 are chosen such that we put a higher weight on minimizing the idleness than maximizing the $c\mu$ index. In particular, Γ_1 should be larger than $\max_{1\le i\le I} c_i\mu_i s_i$, where $s_i = \min_{1\le j\le J} N(j)/M(j,i)$ is the maximal number of Class i customers allowed in service. We can then fine-tune the value of Γ_2 to put enough weight on unavoidable idleness.

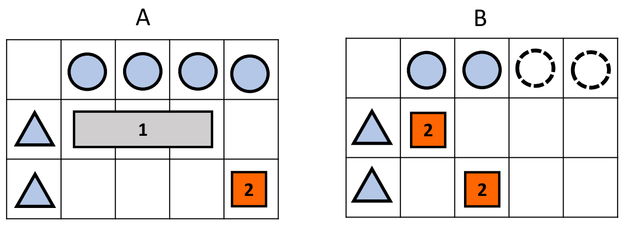

Figure 4: Service configuration for $N = (2,4)$ and $M = [1 \;\; 1 \; ; \; 1 \;\; 3]$.

When taking cost minimization into account, the optimal choice of (Γ_1,Γ_2) may depend on the load of the system. The general rule of thumb is that when the system is lightly loaded, we should put more weight on the instantaneous cost-reduction rate, i.e., the $c_i\mu_i$'s, as managing idleness is of less concern. When the system is heavily loaded, we should put more weight on the two types of idleness. For example, in Figure 7 we compare the long-run average costs of the two models (left versus right) with different traffic intensities (defined in (2)) under the GIA $c\mu$ rule with different values of (Γ_1,Γ_2). We observe that when traffic intensity is low, policies with different hyper-parameters perform similarly. Indeed, when ρ is sufficiently small, the GIA $c\mu$ rule with smaller values (Γ_1,Γ_2) performs slightly better than that with that with large values of (Γ_1,Γ_2), e.g., when $\rho < 0.7$ in the right plot, $(\Gamma_1,\Gamma_2) = (0.5,0.25)$ or $(1,0.25)$ are performing better than $(\Gamma_1,\Gamma_2) = (4,2)$. However, as traffic intensity grows, the larger values of (Γ_1,Γ_2) lead to better performance. Note that when ρ is large (gray area in the plots), the GIA $c\mu$ rule with small values of (Γ_1,Γ_2) can not stabilize the system. To set a general rule of thumb, we note that when ρ is small, the performances of GIA $c\mu$ rule is similar for different values of (Γ_1,Γ_2). When ρ is large, however, large values of (Γ_1,Γ_2) perform substantially better. Thus, we suggest to first set Γ_1 to be larger than $\max_{1\le i\le I} c_i\mu_i s_i$, where $s_i = \min_{1\le j\le J} N(j)/M(j,i)$; then to set Γ_2 to be a positive number that is smaller than Γ_1.

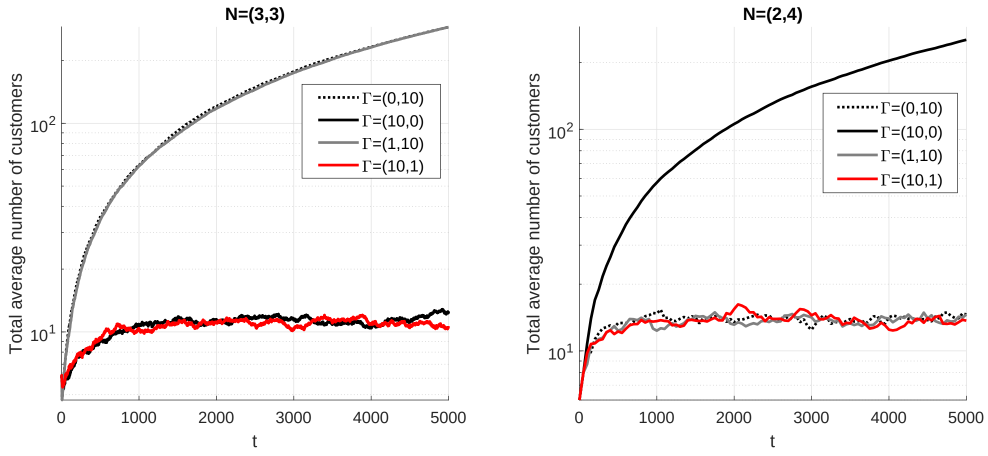

Figure 5: Comparing the average number of customers in the system as a function of time over a finite horizon $T = 5\times10^3$, for different values of Γ_1 and Γ_2. ($M = [1\ \ 1\ ;\ 1\ \ 3]$, and $x(0) = (3,3)$. In the left plot, $N = (3,3)$, $\mu = (0.34,0.8)$, $\lambda = (0.25,0.45)$, and $c = (1,0.5)$; In the right plot, $N = (2,4)$, $\mu = (0.3,0.8)$, and $\lambda = (0.26,0.75)$. Average is estimated using 20 replications.)

Lastly, as the system size grows, the policy-induced idleness is of less concern. The intuition is that idling one server in a two server system is half of the capacity, while idling one server in an n server system is only $1/n$ of the capacity. Figure 8 compares performance of the GIA $c\mu$-rule for different sizes of systems. In particular, we scale up the size of the systems, so that for the η-th system, $\eta = (5,\dots,35)$, there are $N^\eta = N\eta$ servers, and the average arrival rate of Class i is $\eta\lambda_i(t)$, $t\ge0$. We observe that while there could be large differences in the performances among the policies in small systems, as the size of the system grows large, these differences diminish. We do not include systems with η smaller than 5 in the plots, as $(\Gamma_1,\Gamma_2) = (0,10)$ or $(10,0)$ can lead to instability in those small systems. This observation highlights the importance of choosing the "right" hyper parameters when scheduling small systems.

5 COMPARISON WITH THE MAX-WEIGHT POLICY

A well-studied class of policies that is throughput optimal is the max-weight policy (Armony and Bambos 2003; Dai and Lin 2005). The max-weight policy also has certain cost-minimization features (Stolyar 2004). In this section, we compare the performances of the *GIA* $c\mu$-rule to the max-weight policy.

For the max-weight policy, at each event time t, given $X(t) = x$, the scheduling policy allocates servers according to the following IP.

$$\begin{aligned}
&\max_z \sum_{i=1}^{I} c_i\mu_i x_i z_i \\
&\text{s.t.}\quad Mz \le N \\
&\qquad\ \ 0\le z_i\le x_i, z_i\in\mathbb{N}, \qquad i=1,\dots,I,
\end{aligned} \tag{5}$$

Table 1 compares the average cost of each policy for the model illustrated in Figure 2, i.e., $N = (3,3)$ and $M = [1\ \ 1\ ;\ 1\ \ 3]$, with different parameters (scenarios). We define the workload of each class on the bottleneck resource (Type 2, in this case) as

$$W_i = \frac{M_{i2}\lambda_i}{N_2\mu_i},\quad i = 1,2.$$

In Scenarios 1a–1d, the workload of Class 1 is smaller than the workload of Class 2, and vice versa in Scenarios 2a–2d. We also define $c_i\mu_i/M_{2i}$, $i = 1,2$, as the instantaneous cost reduction rate of Class i jobs

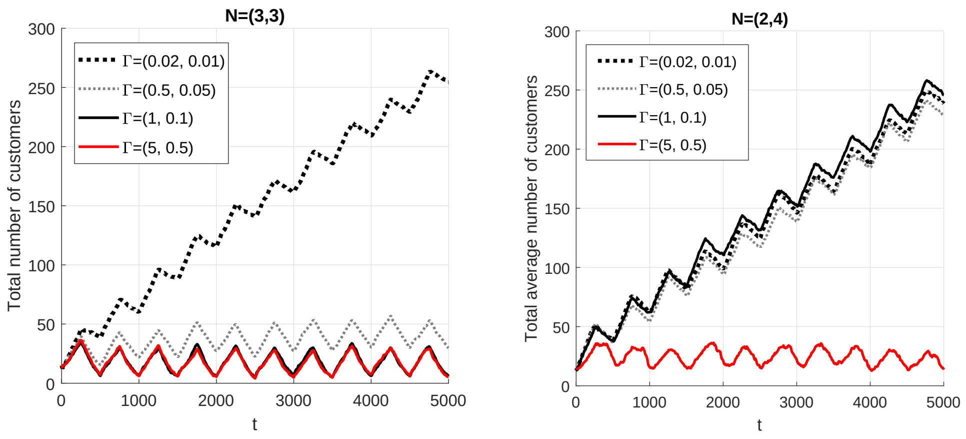

Figure 6: Comparing the average number of customers in the system as a function of time over a finite horizon $T = 5\times 10^3$, for different values of Γ_1 and Γ_2. ($M = [1\ \ 1\ ;\ 1\ \ 3]$, $c = (1, 0.5)$, $x(0) = (10, 5)$. In the left plot, $N = (3, 3)$, $\mu = (0.34, 0.8)$, $\lambda(t) = (0.4, 0.2)$ when $t = k+1, \dots, k+250$, $k = 0, 500, \dots, T$, and $(0.1, 0.7)$, otherwise. In the right plot, $N = (2, 4)$, $\mu = (0.3, 0.8)$, $\lambda(t) = (0.4, 0.1)$ when $t = k+1, \dots, k+250$, $k = 0, 500, \dots, T$, and $(0.5, 0.1)$, otherwise. Average is estimated using 20 replications.)

on the bottleneck resource (Type 2). For each scenario in the table, we allow different values of Γ_1 and Γ_2 for the GIA $c\mu$-rule. We observe that when properly tuned, the GIA $c\mu$-rule achieves significantly better performance than the max-weight policy. This is well-expected as max-weight is not designed to minimize linear holding costs. Lastly, we comment that the max-weight is throughput optimal. Based on our simulation experiments for systems under different traffic intensities, the GIA $c\mu$ rule with properly chosen (Γ_1, Γ_2) also seems to be throughput optimal.

Table 1: Comparing the long-run average cost under the max-weight policy and the GIA $c\mu$-rule ($N = (3, 3)$, $M = [1\ \ 1\ ;\ 1\ \ 3]$. In Scenarios 1a–1d, we have $\mu = (0.45, 0.55)$, $\lambda = (0.15, 0.94)$ thus, $W_1 < W_2$; in Scenarios 2a–2d, we have $\mu = (0.34, 0.8)$, $\lambda = (0.25, 0.45)$ thus, $W_1 > W_2$. The long-run average is estimated based on long-time average with $T = 6\times 10^3$.)

Scenario	Workload	Cost reduction rate	$(c_1,\ c_2)$	Max-weight	GIA $c\mu$	$(\Gamma_1,\ \Gamma_2)$	Improvement
1a	$W_1 < W_2$	$c_1\mu_1/M_{21} > c_2\mu_2/M_{22}$	(10, 0.5)	24.3	13.3	(100,0.01)	45%
1b			(6, 1)	29.9	22.3	(100,0.1)	25%
1c		$c_1\mu_1/M_{21} < c_2\mu_2/M_{22}$	(1, 5)	31.8	18.5	(100,0.1)	42%
1d			(0.5, 10)	51.3	32.2	(100,0.01)	37%
2a	$W_1 > W_2$	$c_1\mu_1/M_{21} > c_2\mu_2/M_{22}$	(10, 0.5)	85.3	48.3	(100,0.01)	43%
2b			(8, 1)	74.4	63	(100,0.01)	15%
2c		$c_1\mu_1/M_{21} < c_2\mu_2/M_{22}$	(1, 5)	35.2	19.7	(100,1)	44%
2d			(0.5,10)	44.6	23.3	(100,0.1)	48%

6 EXTENSION TO TRANSIENT COST MINIMIZATION

In various applications, there can be random shocks that take the system far from its usual mode of operation. In those scenarios, it is more important to improve the transient performance of the system, e.g., finding a good scheduling policy to manage the surge demand. A very relevant example, is the 2020 Coronavirus

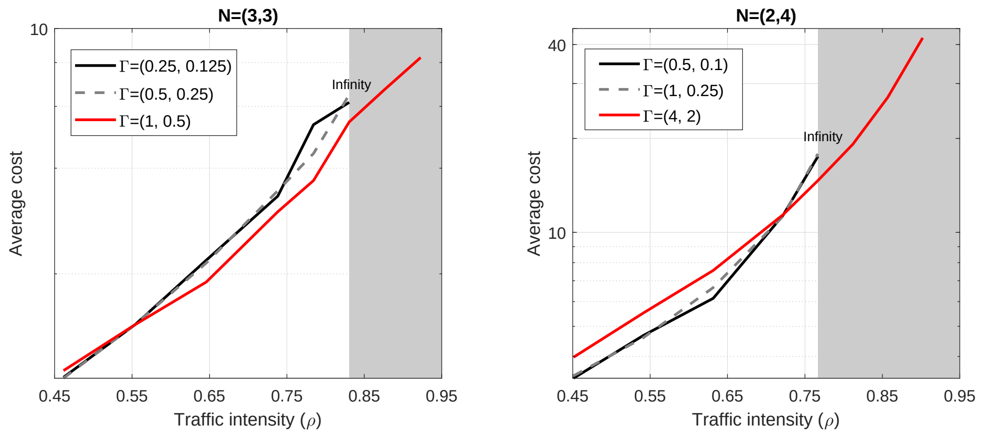

Figure 7: Comparing the long-run average cost for different traffic intensities, ρ, and different values of Γ_1 and Γ_2. ($M = [1\ \ 1\ ;\ 1\ \ 3]$. In the left plot, $N = (3,3)$, $c = (1,0.5)$, $\mu = (0.34,0.8)$, $\lambda(t) = \eta(0.125,0.225)$, $\eta = (1,1.2,\ldots,2)$. In the right plot, $N = (2,4)$, $c = (1,10)$, $\mu = (0.3,0.8)$, $\lambda(t) = \eta(0.13,0.375)$. The long-run average cost is estimated based on the long-time average with $T = 6\times 10^3$.)

pandemic (Zhe et al. 2020) , which caused a sudden surge in demand on hospital capacity, especially ICU capacity.

In this section, we consider the objective of minimizing the cumulative expected holding cost over a finite time horizon T, i.e.

$$\min_{\pi\in\Omega} \int_{t=0}^{T} \sum_{i=1}^{I} E[c_i X_i(u)]\mathrm{d}u,$$

when the arrival rate is highly non-stationary over this time interval. We observe through extensive numerical experiments that that the GIA $c\mu$-rule still performs well in minimizing the transient holding costs.

Figure 9 presents two scenarios in which at time $t = 2000$, both classes experience a surge in demands that lasts for 200 units of time. The figure presents the average number of customers in the system for each class as a function of time over the time horizon $[0,10^4]$. In the left plot, the cumulative cost over this time horizon is 59×10^4 for the Max-weight policy, and 41×10^4 for the GIA $c\mu$-rule. In the right plot, the cumulative cost is 33×10^4 for max-weight, and 22×10^4 for the GIA $c\mu$-rule. In both cases, the GIA $c\mu$-rule substantially outperform the max-weight policy, demonstrating the robustness of our policy, even in minimizing the transient costs.

7 CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we study a new class of multi-class multi-pool queueing systems, where different classes of customers have heterogeneous resource requirements. We propose the GIA $c\mu$ rule that balances two types of idleness: avoidable and unavoidable, and the instantaneous cost reduction rate. By using extensive simulation experiments, we show that when tuned properly, the proposed policy performs much better than other benchmark policies.

We identify two directions for future research. First, our simulation experiments suggest that with properly tuned parameters, the GIA $c\mu$ rule is throughput optimal. Rigorously establishing this would be an interesting future research direction. Second, it would be interesting to include more real-life complications into the model and develop appropriate scheduling policies accordingly. These complications include non-preemption, predictable pattern of time-variability in demand, etc. Note that our current policy is oblivious to the arrival rate. Taking the patterns of arrival rates into account may lead to further improvement of the policy in time-varying settings.

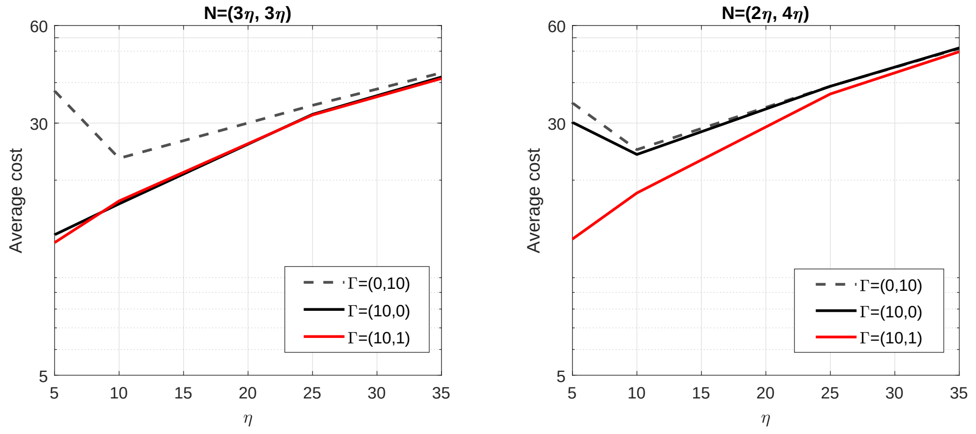

Figure 8: Comparing the long-run average cost for different system sizes and different values of Γ_1 and Γ_2. ($M = [1 \ \ 1 \ ; \ 1 \ \ 3]$, $c = (1, 0.5)$. In the left plot, $N = (3\eta, 3\eta)$, $\lambda = \eta(0.25, 0.45)$, and $\mu = (0.34, 0.8)$; In the right plot, $N = (2\eta, 4\eta)$, $\mu = (0.3, 0.8)$, and $\lambda = \eta(0.26, 0.75)$. The long-run average cost is estimated based on the long-time average with $T = 6 \times 10^3$).

REFERENCES

Altay, N. 2012. "Capability-Based Resource Allocation for Effective Disaster Response". *IMA Journal of Management Mathematics* 24(2):253–266.

Armony, M., and N. Bambos. 2003. "Queueing Dynamics and Maximal Throughput Scheduling in Switched Processing Systems". *Queueing systems* 44(3):209–252.

Dai, J. G., and W. Lin. 2005. "Maximum Pressure Policies in Stochastic Processing Networks". *Operations Research* 53(2):197–218.

Glynn, P. W., and S. Asmussen. 2007. *Stochastic Simulation: Algorithms and Analysis*. New York, NY: Springer.

Grandl, R., G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella. 2014. "Multi-Resource Packing for Cluster Schedulers". *ACM SIGCOMM Computer Communication Review* 44(4):455–466.

Green, L. 1981. "Comparing Operating Characteristics of Queues in which Customers Require a Random Number of Servers". *Management Science* 27(1):65–74.

Gurvich, I., and J. A. Van Mieghem. 2015. "Collaboration and Multitasking in Networks: Architectures, Bottlenecks, and Capacity". *Manufacturing & Service Operations Management* 17(1):16–33.

Gurvich, I., and J. A. Van Mieghem. 2017. "Collaboration and Multitasking in Networks: Prioritization and Achievable Capacity". *Management Science* 64(5):2390–2406.

Harrison, J. M. 1998. "Heavy-Traffic Analysis of a System with Parallel Servers: Asymptotic Optimality of Discrete-Review Policies". *Annals of applied probability*:822–848.

Heyman, D. P., and W. Whitt. 1984. "The Asymptotic Behavior of Queues with Time-Varying Arrival Rates". *Journal of Applied Probability* 21(1):143–156.

Kelly, F. P., A. K. Maulloo, and D. K. H. Tan. 1998. "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability". *Journal of the Operational Research society* 49(3):237–252.

Ma, N., and W. Whitt. "Using Simulation to Study Service-Rate Controls to Stabilize Performance in a Single-Server Queue with Time-Varying Arrival Rate". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti. Piscataway, NJ, USA.

Mandelbaum, A., and Z. Feldman. "Using Simulation-Based Stochastic Approximation to Optimize Staffing of Systems with Skills-Based Routing". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yucesan, 3307–3317. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Inc.

Mandelbaum, A., and A. L. Stolyar. 2004. "Scheduling Flexible Servers with Convex Delay Costs: Heavy-Traffic Optimality of the Generalized cμ-Rule". *Operations Research* 52(6):836–855.

Massoulié, L., and J. Roberts. In *Proceedings of the 1999 Institute of Electrical and Electronics Engineers INFOCOM Conference.* New York, NY, USA: Institute of Electrical and Electronics Engineers, Inc.

Masterson, G., and S. Baudouin. 2015. "Guidelines for the Provision of Intensive Care Services". Technical report, London: Faculty of Intensive Care Medicine.

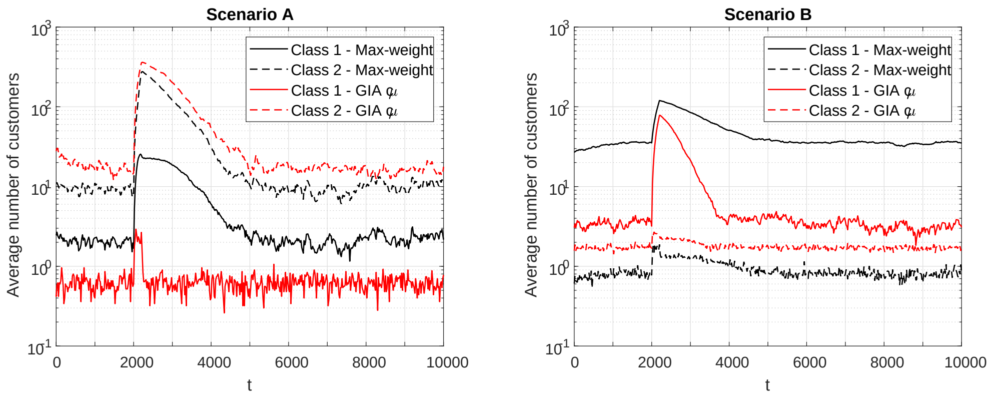

Figure 9: Average number of customers in the system as a function of time: the GIA $c\mu$-rule with $\Gamma = (100,1)$ and the Max-weight policy. ($N = (3,3)$ and $M = [1 \;\; 1 \;;\; 1 \;\; 3]$. In the left plot, $\mu = (0.4,0.6)$, $c = (10,0.5)$, $\lambda(t) = (0.3,2.25)$, when $2000 \le t \le 2200$, and $(0.15,0.9)$ otherwise. In the right plot, $\mu = (0.4,0.8)$, $c = (0.5,10)$, $\lambda(t) = (0.625,0.9)$, when $2000 \le t \le 2200$, and $(0.25,0.45)$, otherwise.)

Papadimitriou, C., and J. Tsitsiklis. 1999. "The Complexity of Optimal Queuing Network Control". *Mathematics of Operations Research* 24(2):293–305.

Puterman, M. 2005. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, New Jersey: Wiley-Interscience.

Ramakrishnan, L., and D. Gannon. 2008. "A Survey of Distributed Workflow Characteristics and Resource Requirements". Technical report, Department of Computer Science, Indiana University.

Reiman, M. I. 1991. "A Critically Loaded Multiclass Erlang Loss System". *Queueing Systems* 9(1-2):65–81.

Stolyar, A. L. 2004. "Maxweight Scheduling in a Generalized Switch: State Space Collapse and Workload Minimization in Heavy Traffic". *The Annals of Applied Probability* 14(1):1–53.

Zhe, X., S. Lei, W. Yijin, Z. Jiyuan, L. H., Z. Chao, L. Shuhong, Z. Peng, L. Hongxia, and Z. Li. 2020. "Pathological Findings of COVID-19 Associated with Acute Respiratory Distress Syndrome". *The Lancet respiratory medicine* 8(4):420–422.

Zychlinski, N. and Chan, C.W. and Dong, J. 2020. "Managing Queues with Different Resource Requirements". "https://noazy.net.technion.ac.il/files/2020/09/Zychlinski_Managing_Queues.pdf".

AUTHOR BIOGRAPHIES

Noa Zychlinski is an Assistant Professor in the Faculty of Industrial Engineering and Management at the Technion – Israel Institute of Technology. Noa has completed her postdoctoral fellowship in the Division of Decision, Risk, and Operations at Columbia Business School. Her research interests focus on service and healthcare operations management, the analysis of queueing networks and their applications, the theory of stochastic process approximation, and data analysis of large service systems. Her eamil is noazy@technion.ac.il.

Carri W. Chan is an Associate Professor in the Division of Decision, Risk, and Operations at Columbia Business School. Her primary research interests are in data-driven modeling of complex stochastic systems, dynamic optimization, and queueing with applications in health-care operations management. Her current focus is on combining empirical approaches with mathematical modeling to develop evidence-based approaches to improving patient flow through hospitals, and particularly intensive care units. Her email address is cwchan@columbia.edu.

Jing Dong Jing Dong is an Associate Professor in the Decision, Risk, and Operations division at the Graduate School of Business, Columbia University. Her primary research interests are in applied probability and stochastic simulation, with an emphasis on applications in service operations management. Her current research focuses on developing data-driven stochastic modeling to improve patient flow in hospitals. Her e-mail is jd2736@columbia.edu.