Queueing Systems
https://doi.org/10.1007/s11134-020-09669-5

A survey on skill-based routing with applications to service operations management

Jinsheng Chen[1] · **Jing Dong**[2] · **Pengyi Shi**[3]

Received: 14 September 2020 / Revised: 17 September 2020

Abstract
Service systems often feature multiple classes of customers with different service needs and multiple pools of servers with different skillsets. How to efficiently match customers of different classes with servers of different skillsets is of great importance to the management of these systems. In this survey, we review works on skill-based routing in queues. We first summarize key insights on routing/scheduling policies developed in the literature. We then discuss complications brought by modern service operations management problems, particularly healthcare systems. These complications stimulate a growing body of literature on new modeling and analysis tools. Lastly, we provide additional numerical experiments to highlight the complex nature of a routing problem motivated from hospital patient-flow management, and provide some useful intuition to develop good skill-based routing policies in practice. Our goal is to provide a brief overview of the skill-based routing research landscape and to help generate interesting research ideas.

Keywords Queueing theory · Skill-based routing · Asymptotic analysis

Mathematics Subject Classification 90B22

✉ Jing Dong
jing.dong@gsb.columbia.edu

Jinsheng Chen
jc4823@columbia.edu

Pengyi Shi
shi178@purdue.edu

[1] Department of IEOR, Columbia University, 500 West 120th Street, New York, NY 10027, USA

[2] Graduate School of Business, Columbia University, 3022 Broadway, New York, NY 10027, USA

[3] Krannert School of Management, Purdue University, 403 W State St, West Lafayette, IN 47907, USA

Published online: 04 October 2020

1 Introduction

In service systems, customers are typically grouped into different classes based on their service needs. For example, in a call center, different customer types can represent different service requirements, for example, individual banking, business banking, and loans in a bank call center, or different priority levels. In a hospital inpatient department, different customer types can correspond to different medical specialties of patients, for example, general medicine, cardiology, and neurology, or different severity levels, for example, general ward, intensive care unit, and step-down unit. Correspondingly, servers (resources) are often grouped into different server pools based on their skillset/specialization to serve different types of customers. For example, call center staff are grouped into different pools based on their training or language skills, for example, English, Spanish, and bilingual. Inpatient ward beds are grouped into different specialty wards to better facilitate the coordination and standardization of care.

There are several interesting design questions that arise in these service systems: (i) how to define the customer/server classes and how to design the network structure; (ii) how to size each server pool; and (iii) how to route customers to match with servers. Note that these three design questions are often made at different timescales: for example, in call centers or hospitals, (i) is usually planned on a yearly basis, (ii) is on a monthly/weekly basis, while (iii) needs to be implemented in the real-time fashion. These questions are also highly interdependent. However, tackling them all at once is often too ambitious a goal. Most of the literature looks at one of them while holding the others fixed. In this survey paper, we focus on studying the third design question listed above. We refer to this problem as skill-based routing (SBR) following [40,43].

Queueing models are commonly used modeling tools to study the dynamics of these service systems. In this paper, we consider a multi-class multi-pool queue with I classes of customers and J types of parallel server pools. Customer class i, $i = 1, 2, \ldots, I$, is characterized by an arrival process with rate λ_i and a patience time distribution with mean $1/\theta_i$. (The patience time of a customer is the amount of time he/she is willing to wait for service before abandoning the queue.) We allow each server pool to have multiple homogeneous servers. In particular, server pool j, $j = 1, 2, \ldots, J$, has s_j homogeneous servers. Focusing on service operations applications, we assume each server can only serve one customer and each customer can only be served by one server at a time. Note that in some applications, it is possible that a server can serve multiple customers at a time. For example, in an online chatting system, an agent can chat with multiple customers at a time. It can also be the case that multiple (types of) servers are required to serve a customer. For example, in a healthcare facility, to treat a patient with complex medical conditions, multiple resources from multiple pools are needed simultaneously. We do not consider such models in this paper, but refer interested readers to Gurvich and Van Meghem [47] and Luo and Zhang [64].

A customer from class i can potentially be matched with a server in more than one server pool. If a customer from class i is served by a server from pool j, we denote the mean of its service time distribution as $1/\mu_{ij}$. Note that if there is no 'compatibility' between customer class i and server type j, we can define $\mu_{ij} = 0$. Figure 1 provides

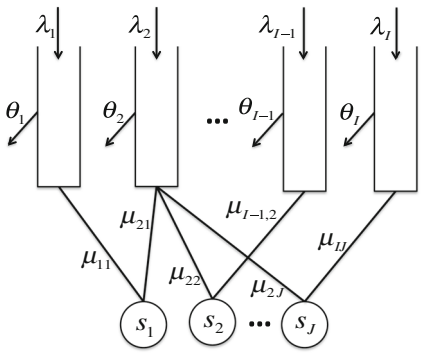


Fig. 1 A multi-class multi-pool queue

an illustration of such a system. Following Harrison and López [54], we call it a parallel-server system (PSS).

In a PSS, there are two types routing decisions to be made: (1) when a server becomes available (finishes service), which customer, if any, should the server attend to next; (2) when a customer arrives and there are multiple idle servers, to which server, if any, should the customer be routed. An SBR policy specifies how these decisions are made. The goal is to design a good policy according to some performance criteria, for example, waiting cost, abandonment cost, matching (overflow) cost, or combinations of these.

To facilitate subsequent discussions, we introduce some notation and terminology. First, to describe the system state, at a particular time t, let $X_i(t)$ denote the total number of class i customers in the system, $Q_i(t)$ denote the number of class i customers waiting in queue, and $Z_{ij}(t)$ denote the number of class i customers in service in pool j. Note that

$$X_i(t) = Q_i(t) + \sum_{j=1}^{J} Z_{ij}(t).$$

For ease of exposition and analytical tractability, we next introduce the system dynamics for a Markovian system where interarrival times, service times, and patience times all have exponential distributions. Let A_i, S_{ij}, and R_i, $i = 1, \ldots, I$, $j = 1, \ldots, J$, be unit rate Poisson processes modeling the arrival processes, (non-interrupted) service completion processes, and abandonment processes, respectively. Then, the system dynamics can be described as

$$X_i(t) = X_i(0) + A_i(\lambda_i t) - \sum_{j=1}^{J} S_{ij}\left(\mu_{ij}\int_0^t Z_{ij}(s)\,\mathrm{d}s\right) - R_i\left(\theta_i \int_0^t Q_i(s)\,\mathrm{d}s\right),$$

where the evolution of Z_{ij} are determined by the routing policy.

We also consider a discrete-time version of the problem, $\tilde{X}$, for which, at each time slot, arrivals follow Poisson distributions and departures follow Binomial distributions.

In this case, the system dynamics can be described as

$$\tilde{X}_i(t+1) = \tilde{X}_i(t) + \tilde{A}_i(t) - \sum_{j=1}^{J} \tilde{S}_{ij}(t) - \tilde{R}_i(t).$$

where, given a scheduling policy $\tilde{Z}_{ij}(t)$, $\tilde{A}_i(t)$ is a Poisson random variable with rate λ_i, $\tilde{S}_{ij}(t)$ conditional on $\tilde{Z}_{ij}(t)$ is a Binomial random variable with parameters $\tilde{Z}_{ij}(t)$ and μ_{ij}, and $\tilde{R}_i(t)$ conditional on $\tilde{Q}_i(t)$ is a Binomial random variable with parameters $\tilde{Q}_i(t)$ and θ_i.

The routing decisions are often to be made upon a customer's arrival or upon a service completion. We next introduce two major service disciplines: preemptive versus non-preemptive. A policy is said to be preemptive if a customer in service can be interrupted. Here, interruption can mean being transferred to a server in a different pool or being put back in the queue. Non-preemption, on the other hand, means that once a customer starts service, it stays with the assigned server until service completion.

Preemptive policies are typically easier to analyze. In particular, under Markovian system primitives, one can view a non-anticipatory preemptive scheduling policy as a mapping from $X(t) := (X_i(t) : i = 1, \ldots, I)$ to an allocation of the servers, $Z(t) := (Z_{ij}(t) : i = 1, \ldots, I, j = 1, \ldots, J)$. When preemption is allowed, in general we can focus on work-conserving policies. A policy is said to be work-conserving, or non-idling, if a server will not idle whenever there is at least one compatible customer waiting in the queue. On the other hand, work-conservation is in general sub-optimal when preemption is not allowed. In particular, we can hold a server idle in anticipation of the arrival of a more preferred customer, or we can hold a customer waiting in anticipation of a more preferred server becoming available.

Finally, to optimize the routing policies, one needs to define a performance metric or an objective function. The objective function is often defined as some form of cumulative cost, and a good policy aims to minimize the cost objective. Specifically, let $C(t)$ denote the cost rate incurred at time t in the continuous-time setting, or the cost incurred at epoch t in the discrete-time setting. $C(t)$ can include holding costs, for example, $h_i Q_i(t)$, or overflow costs (for using non-preferred servers), for example, $\sum_j \phi_{ij} Z_{ij}(t)$, etc. The cumulative cost over a finite time horizon is then defined as

$$\mathbb{E}\left[\int_0^T C(t)\,\mathrm{d}t\right] \text{ and } \mathbb{E}\left[\sum_{t=1}^{T} C(t)\right]$$

for the continuous-time and discrete-time problem settings, respectively. When dealing with an infinite time horizon, we can define the objective function either as a long-run average cost, i.e.,

$$\limsup_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\int_0^T C(t)\,\mathrm{d}t\right] \text{ and } \limsup_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} C(t)\right],$$

or a cumulative discounted cost with a discount rate $\gamma > 0$, i.e.,

$$\mathbb{E}\left[\int_0^\infty e^{-\gamma t} C(t)\,\mathrm{d}t\right] \text{ and } \mathbb{E}\left[\sum_{t=1}^{\infty} e^{-\gamma t} C(t)\right].$$

The rest of the paper is organized as follows: Taking system stability as the first-order goal, in Sect. 2, we review an important class of policies that achieves the maximum stability region. In Sect. 3, we review a powerful analytic tool—heavy-traffic asymptotic analysis—and discuss the structural properties of optimal/asymptotically optimal SBR policies in some special PSSs. We note that some of the results in Sects. 2 and 3 do not rely on Markovian system primitives, i.e., they hold for more general arrival processes and service time distributions. Next, in Sect. 4, we discuss the challenges and opportunities when modeling healthcare applications. Lastly, we provide a numerical study to demonstrate the effect of overflow cost in Sect. 5.

2 Stability

Before we discuss cost minimization, the first-order criterion that a good routing policy should achieve is to stabilize the system. There are several notions of stability. Following [5], we say that the system is stable if

$$\lim_{t\to\infty} Q_i(t)/t = 0, \text{ for } i = 1, \ldots, I \text{ , almost surely.}$$

This notion of stability is also known as rate stability as it preserves job inflow–outflow balance. In particular, in a PSS, rate stability implies that the arrival rate of class i customers is equal to the departure rate of class i customers in the long run.

If customers abandon after waiting for too long, the system is guaranteed to be stable. However, when abandonment is absent, stability is not guaranteed. For a system with given service rates, server pool sizes, and a fixed SBR policy, the set of arrival rates for which the system is stable is referred to as the stability region. The maximum stability region $\mathcal{S}$ is the union of the stability regions over all SBR policies and can be characterized as

$$\mathcal{S} = \left\{ \lambda : \ \lambda_i = \sum_{j=1}^{J} \mu_{ij} s_j \pi_{ij}, \text{ for some } \pi_{ij} \geq 0 \text{ with } \sum_{i=1}^{I} \pi_{ij} \leq 1, \right.$$
$$\left. i = 1, 2, \ldots, I, \, j = 1, \ldots J \right\}.$$

In the definition of $\mathcal{S}$, π_{ij} can be interpreted as the proportion of time resources from pool j are used to serve class i customers. For any $\lambda \in \mathcal{S}$, the system can be stabilized

under some SBR policy regardless of whether preemption is allowed [31]. However, different scheduling policies may have very different stability regions. An SBR policy that achieves the maximum stability region $\mathcal{S}$ is called throughput optimal. We next review an important class of throughput optimal policies—the maximum pressure policies [83].

Maximum pressure policy

For a PSS, we define the network pressure at time t as

$$p(t) = \sum_{i=1}^{I} \alpha_i X_i(t) \left(\sum_{j=1}^{J} \mu_{ij} Z_{ij}(t) - \lambda_i \right) = \sum_{i,j} \alpha_i X_i(t) \mu_{ij} Z_{ij}(t) + K,$$

for some $\alpha = (\alpha_1, \ldots, \alpha_I) > 0$, where $K = -\sum_{i=1}^{I} \alpha_i X_i(t) \lambda_i$ does not depend on the server allocation $Z_{ij}(t)$. The weights α_i are positive and are often determined by the holding cost of each class, as we shall explain below. When there are no cost concerns, we can set $\alpha_i = 1/I$, i.e., assigning equal weights to each class. The maximum pressure policy tries to maximize $p(t)$ among all feasible server allocations. In the preemptive version, at each time t, we set

$$Z(t) = \arg\max_{z} \Big\{ p(t) : \ z_{ij} \in \mathbb{N}_0, \sum_{i=1}^{I} z_{ij} \le s_j, \sum_{j=1}^{J} z_{ij} \le X_i(t), \\ i = 1, \ldots, I, j = 1, \ldots, J \Big\},$$

where $\mathbb{N}_0$ denotes the set of nonnegative integers. In the non-preemptive version, when a pool j server becomes available, the server will next serve a waiting class i^* customer if

$$i^* \in \arg\max_{i: Q_i(t) > 0} \alpha_i X_i(t) \mu_{ij}.$$

Dai and Lin [29] show that maximum pressure policies are throughput optimal for a fairly general class of queueing networks. These networks allow, for example, customers to be sent to another queue upon service completion and include as a special case PSSs. To see the intuition behind the throughput optimality of the maximum pressure policy, we consider a 'counterpart' deterministic fluid model

$$\frac{\mathrm{d}\bar{x}_i(t)}{\mathrm{d}t} = \lambda_i - \sum_{j=1}^{J} \mu_{ij} \bar{z}_{ij}(t), \quad i = 1, 2, \ldots, I,$$

where $\sum_{i=1}^{I} \bar{z}_{ij}(t) \le s_j$. Under the maximum pressure policy with preemption,

$$\bar{z}(t) = \arg\max_{z} \left\{ \sum_{i=1}^{I} \alpha_i x_i(t) \sum_{j=1}^{J} \mu_{ij} z_{ij} : \sum_{i=1}^{I} z_{ij} \le s_j, \sum_{j=1}^{J} z_{ij} \le x_i(t), \right.$$
$$\left. i = 1, \ldots, I, j = 1, \ldots, J \right\}.$$

Now, consider a quadratic Lyapunov function $V(t) = \sum_{i=1}^{I} \alpha_i \bar{x}_i(t)^2$. Under the maximum pressure policy, for any $\lambda \in \mathcal{S}$, when $x_i(t) \ge N_j$ for any compatible (i, j) pair,

$$\begin{aligned} \frac{\mathrm{d}V(t)}{\mathrm{d}t} &= 2 \sum_i \alpha_i \bar{x}_i(t) \frac{\mathrm{d}\bar{x}_i(t)}{\mathrm{d}t} \\ &= 2 \sum_i \alpha_i \bar{x}_i(t) \left(\lambda_i - \sum_j \mu_{ij} \bar{z}_{ij}(t) \right) \\ &\le 2 \sum_i \alpha_i \bar{x}_i(t) \left(\lambda_i - \sum_j \mu_{ij} s_j \pi_{ij} \right), \end{aligned}$$

for any $\pi_{ij} \ge 0$ satisfying $\sum_{i=1}^{I} \pi_{ij} \le 1$. If we choose π_{ij} such that $\lambda_i \le \sum_{j=1}^{J} \mu_{ij} s_j \pi_{ij}$, which is feasible due to the definition of $\mathcal{S}$, then

$$2 \sum_i \alpha_i \bar{x}_i(t) \left(\lambda_i - \sum_j \mu_{ij} s_j \pi_{ij} \right) \le 0,$$

which implies that $\frac{\mathrm{d}V(t)}{\mathrm{d}t} \le 0$. The negative drift of the Lyapunov function implies the stability of the fluid model.

One remarkable feature of the maximum pressure policy is that it is oblivious to the arrival rates. On the other hand, we emphasize that the policy does require knowledge of the service rates. When the service rates are not known, we may consider algorithms to learn them while doing the routing/scheduling [62].

In addition to being throughput optimal, the maximum pressure policy is also shown to be asymptotically cost optimal for certain cost structures. Dai and Lin [30] prove that for quadratic holding cost, $\sum_i \alpha_i X_i(t)^2$, the maximum pressure policy is asymptotically optimal under the conventional heavy-traffic scaling (see Sect. 3.1 for details of the asymptotic regime). Under the same scaling, Stolyar [81] studies discrete-time generalized switch model and establishes the asymptotic optimality of a general class of maximum pressure policies, which is also known as the Max-Weight policy. In

particular, at each time slot t, the policy is trying to maximize

$$\sum_{i,j} \alpha_i X_i(t)^\beta \mu_{ij} Z_{ij}(t)$$

for some $\beta > 0$. Here, the holding cost takes the form $\sum_i \alpha_i X_i(t)^{1+\beta}$.

We comment that developing throughput optimal policies can be highly nontrivial. For example, a policy that maximizes the instantaneous processing rate $\sum_{i,j} \mu_{ij} Z_{ij}(t)$ is in general not throughput optimal [5]. In addition to the maximum pressure policy, Armony and Bambos [5] study other scheduling policies that are throughput optimal. Examples include FastEmpty and BatchAdapt. Under FastEmpty, $Z_{ij}(t)$ are chosen to minimize the time until the system is empty, assuming that there are no further arrivals. Under BatchAdapt, customers are grouped and served in batches, where each batch comprises the customers who arrived during the processing time of the previous batch. Within each batch, we try to maximize the processing rate. Similar batching-and-matching ideas are also used in Gurvich and Ward [48] and Harrison [53].

There is a vast amount of literature studying stability and scaling properties (as the traffic intensity approaches 1) of various routing/scheduling policies. An important class of these policies are load-balancing policies [39], such as join-the-shortest-queue [38], power-of-d [68], and join-idle-queue [63]. Even though load-balancing is important in scheduling service systems, the above-mentioned policies and analysis are more relevant for computer systems (for example, cloud computing facilities), where there is often a huge number of parallel servers, making it difficult to maintain a centralized queue [80].

3 Asymptotic analysis: insights into special cases

The exact analysis of SBR is usually analytically intractable due to the large state space and policy space. In addition, even if we can solve for the optimal SBR policy numerically, the instance-by-instance solutions may not provide much insight into the structure of a good policy. One important method to overcome the challenge is through heavy-traffic asymptotic analysis. In essence, a sequence of queueing systems with the same architecture is considered. The corresponding limiting processes (deterministic dynamical systems or diffusion processes), which are justified by limit theorems such as functional laws of large numbers or functional central limit theorems, are in some cases much more tractable. The tractability often comes from the fact that, under the appropriate scaling, one can leverage the 'state-space collapse' (SSC) property, where a multidimensional process can be represented by a much lower-dimensional process in the limit [75].

In what follows, we first introduce the heavy-traffic asymptotic regimes in Sect. 3.1. We then review works that apply the heavy-traffic asymptotic framework to study SBR in Sect. 3.2. Many of these works focus on special cases of the network presented in Fig. 1, such as the V-model and the N-model, because analyzing the full network can be very complicated. We summarize the main insights gleaned from studying these

special cases. In Sect. 3.3, we discuss a few other related works/policies that enjoy certain analytical tractability.

3.1 Heavy-traffic asymptotic regime

There are several heavy-traffic asymptotic regimes considered in the literature. Policies that are appropriate for one regime may not be appropriate for the others. In fact, a policy that is asymptotically optimal for one regime can cause instability in another regime. In practice, we usually deal with only a single system rather than a sequence of systems, and it is essential to identify the appropriate regime for analysis in order to determine a good scheduling policy. We review two important asymptotic regimes here: the conventional heavy traffic regime and the many-server heavy traffic regime.

Consider a sequence of systems indexed by n. We denote $\lambda^n := (\lambda_i^n : i = 1, \ldots, I)$ as the arrival rates of the nth system, $\mu^n := (\mu_{ij}^n : i = 1, \ldots, I, j = 1, \ldots, J)$ as its service rates, and $s^n := (s_j^n : j = 1, \ldots, J)$ as the sizes of the server pools. As we are dealing with multiple customer classes and server pools, the traffic intensity ρ^n is defined through a linear program [54]:

$$\begin{aligned} \text{minimize} \quad & \rho^n \\ \text{subject to} \quad & \sum_{j=1}^{J} s_j^n \mu_{ij}^n \pi_{ij} = \lambda_i^n \text{ for } i = 1, \ldots, I, \\ & \sum_{i=1}^{I} \pi_{ij} \le \rho^n \text{ for } j = 1, \ldots, J, \\ & \rho^n \ge 0, \pi_{ij} \ge 0 \text{ for } i = 1, \ldots, I, j = 1, \ldots, J. \end{aligned} \tag{1}$$

When there is no abandonment, $\rho^n \le 1$ is necessary for stability.

Conventional Heavy Traffic Under the conventional heavy traffic scaling, the number of servers is held fixed, while time is scaled up by n. The arrival rates and service rates can vary with n as long as $\rho^n \to \rho$ as $n \to \infty$ [58]. To keep subsequent discussions concise, we keep the service rates fixed, i.e., $\mu_{ij}^n = \mu_{ij}$, and assume that each server pool only has a single server. We then impose the following complete resource pooling (CRP) condition: for some $b_i \in \mathbb{R}, i = 1, \ldots, I$, $\sqrt{n}(\lambda_i^n - \lambda_i) \to b_i$, where $\lambda = (\lambda_1, \ldots, \lambda_I)$ satisfies

$$\lambda_i = \sum_{j=1}^{J} \mu_{ij} \pi_{ij} \text{ for some } \pi_{ij} \ge 0 \text{ and } \sum_{i=1}^{I} \pi_{ij} = 1.$$

Note that under CRP, $\rho = 1$. When there is customer abandonment, we scale down the abandonment rates when scaling up time (see [74,86] for more details). In the special case of a sequence of single class $GI/GI/1$ queues, the limiting diffusion scaled queue length process is a reflected Brownian motion [58]. In this case, almost all arriving customers have to wait for service. Thus, under the conventional heavy-

traffic scaling, we can focus on the question: which class should be prioritized when a server becomes available? In other words, which server an arriving customer should be routed to is not relevant (see, for example, Mandelbaum and Stolyar [66]).

Many-server Heavy Traffic Under the many-server heavy traffic scaling, we send the arrival rates and the number of servers to infinity while keeping the service rates and abandonment rates fixed. We again denote $\rho = \lim_{n\to\infty} \rho^n$. There are three commonly considered (sub)regimes in this setting: the Quality-Driven (QD) regime where $\rho < 1$, the Efficiency-Driven (ED) regime where $\rho > 1$, and the Quality-and-Efficiency-Driven (QED) regime where $\rho = 1$ and ρ^n approaches 1 at rate $1/\sqrt{n}$. For what follows, we focus on the QED regime. As its name suggests, this regime is often able to strike a balance between the quality of service and the server utilization rate (efficiency) [19]. We also impose the following complete resource pooling condition in this case: For some $\xi_i \in \mathbb{R}$, $i = 1, \ldots, I$, and $\gamma_j \in \mathbb{R}$, $j = 1, \ldots, J$, $(\lambda_i^n - u_i n)/\sqrt{n} \to \xi_i$ and $(s_j^n - v_j n)/\sqrt{n} \to \gamma_j$, where $u_i, v_j > 0$ satisfy

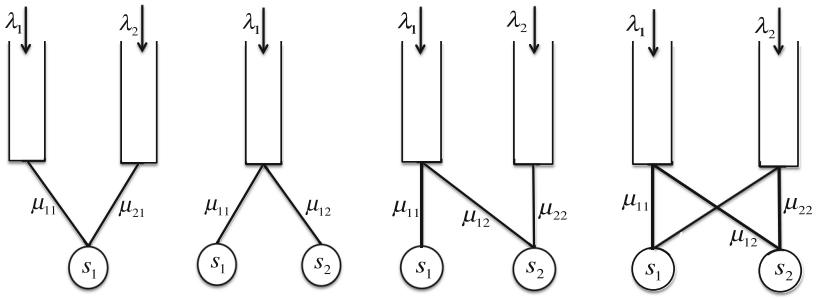
$$\sum_{j=1}^{n} \mu_{ij} v_j \pi_{ij} = u_i \text{ for some } \pi_{ij} \geq 0 \text{ and } \sum_{i=1}^{I} \pi_{ij} = 1.$$

In the special case of a sequence of single class $GI/M/s^n$ queues, the limiting diffusion-scaled queue length process is a piecewise-linear diffusion process [52]. (The result is extended in Garnett et al. [44] to accommodate abandonment.) In the limit, there is a positive yet less than one probability that an arriving customer has to wait. In this case, we need to specify both how to route customers to available servers upon arrivals and how to prioritize different classes of customers upon service completions (see, for example, Gurvich and Whitt [50]). It is also worth noticing that in the heavy-traffic limit, the difference between preemption and non-preemption sometimes diminishes [9].

Other asymptotic regimes In addition to the two asymptotic regimes discussed above, there are other asymptotic regimes studied in the literature. For example, the non-degenerate slowdown (NDS) regime is a 'midpoint' between the conventional heavy-traffic regime and the many-server QED regime [87]. Under NDS, we scale up the arrival rate and the number of servers while scaling down the service rate. In the case of a single-class $M/M/n$ model, for a given $\alpha \in [0, 1]$, $\lambda > 0$, and $v > 0$, we set $\lambda^n = n - \lambda\sqrt{n}$, $s^n = vn^\alpha$ and $\mu^n = v^{-1}n^{1-\alpha}$. The NDS regime enjoys the property that delay and service times are of the same order under scaling [10]. Another asymptotic regime is one where the arrival rates scale super-linearly in n while the service rates and abandonment rates scale linearly in n. This asymptotic regime has been applied to study systems with demand uncertainty [13,15].

3.2 Special cases

Much of the literature focuses on certain special cases of PSSs, generating interesting insights into the management of such systems. Four important special cases are: (a) the V-model, where there is a single pool of servers but two or more classes of customers



(a) V-model (b) Inverted V-model (c) N-model (d) X-model

Fig. 2 Special cases of multi-class multi-pool queueing systems

that typically differ in their service times and/or holding cost; (b) the inverted V-model, where there is a single class of customers but two or more pools of servers, which typically differ in their rates of service; (c) the N-model; and (d) the X-model. In the latter two models, there are two classes of customers, classes 1 and 2, and two pools of servers, pools 1 and 2, which are primarily dedicated to the two classes of customers, respectively. The N-model allows only one way of 'helping,' i.e., pool 2 can help serve class 1 customers but not the other way around. The X-model, on the other hand, allows 'helping' in both directions, i.e., pool 1 can help serve class 2 customers and pool 2 can also help serve class 1 customers. However, there is often a cost for helping. For example, customers served by servers from the non-primary pool may experience a slower service rate (service slowdown), or there might be an overflow penalty cost for each non-primary assignment. Figure 2 illustrates these four special cases.

We next summarize high-level insights from the special cases developed in the literature. One aspect that we will not cover in its full complication is customer abandonment, especially when the patience time has a non-exponential distribution. This is nevertheless a very important aspect in service operations. Indeed, there is a growing body of literature analyzing customer abandonment behavior from both empirical and modeling perspectives. We refer to Puha and Ward [72] for more details on the modeling and analysis of customer abandonment.

In what follows, our default assumption is that the system parameters (arrival rates, service rates, abandonment rates, and the number of servers) are time-homogeneous. We also assume there is no overflow cost for a non-primary assignment. We delay the discussion of overflow cost to Sects. 4 and 5. We write $C_i(q_i)$ as the holding cost rate for class i when there are q_i class i customers in the queue.

V-model In the V-model, the relevant scheduling decision is: when a server becomes available, which customer class it should serve next?

We start with a single-server system with no abandonment and set the objective function as the total cumulative cost over a finite time horizon. If static priority is imposed (i.e., the relative priority of different classes is not state-dependent), the conventional heavy-traffic asymptotic framework leads to a state-space collapse where the scaled queue in the limit only contains customers in the lowest priority class [75]. We also note that under any work-conserving scheduling policies, the workload-

process is the same. Thus, if work-conservation is imposed, the holding cost rate is approximately $C_{\tilde{i}}(W(t)\mu_{\tilde{i}1})$, where $\tilde{i}$ is the class with the lowest priority and $W(t)$ is the workload process. (Note that in this model, there is one server pool ($J = 1$) but each customer class can have a different service rate. Hence, we use μ_{i1} to denote the class-dependent service rate.) This holding cost rate suggests that we should give lower priority to the class with a smaller holding cost rate and a smaller service rate (longer average service time). More generally, as our goal is to minimize the holding cost, when there is a single server and no abandonment is allowed, a good heuristic is to maximize the cost reduction rate in a myopic way. In particular, when a server becomes available, it should next serve class i^* if

$$i^* = \arg\max_i \{C_i'(Q_i(t))\mu_{i1}\}. \tag{2}$$

This policy is known as the generalized $c\mu$-rule. In a multi-class $M/G/1$ queue with linear holding cost, i.e., $C_i(q) = c_i q$ for some $c_i > 0$, Cox and Smith [27] establish the optimality of the $c\mu$-rule where one prioritizes the class with a larger $c_i\mu_{i1}$ index (see [21] for an elegant proof of the discrete-time model). Van Mieghem [67] studies the multi-class $G/G/1$ queue with convex holding cost and establishes the asymptotic optimality of the generalized $c\mu$ rule in (conventional) heavy traffic.

For multi-class $M/M/n + M$ queues in the QED regime, the above state-space collapse phenomenon is absent. This can change the structure of the optimal scheduling policy. Consider a linear holding cost with the objective of minimizing the infinite-horizon discounted cost. When allowing preemption, Harrison and Zeevi [55] derive a diffusion limit (approximation) for the scaled queue length process and study the associated diffusion control problem. They find that the optimal control is extremal in the sense that it is optimal to have strict priority. Notably, the priority is state-dependent, i.e., not static. Thus, the optimal policy is no longer as simple as the $c\mu$-rule. Atar [11] takes a similar approach as [55], but considers more general cost functions and allows non-preemption. Moreover, Atar [11] establishes the asymptotic optimality of the scheduling policy constructed by translating the optimal diffusion control to the pre-limit queueing systems.

Inverted V-model In this model, different server pools in general differ in the service rates only. The relevant decision to be made is: upon a customer's arrival, to which available server should this customer be routed? From the perspective of maximizing the system's efficiency, a good heuristic is to prioritize the pool with faster servers. This is known as the fastest-server-first (FSF) policy. Indeed, when we allow preemption, FSF has shown to be optimal [4]. However, when service is non-preemptive, implementing FSF in a work-conserving fashion is not optimal in general. In particular, we might want to hold the customer waiting for a faster server while keeping the slower server idling [90]. In [4], Armony studies the Markovian inverted V-model without abandonment and shows that FSF is asymptotically optimal with respect to minimizing the steady-state queue length in the QED regime. She proves this by showing that the non-preemptive FSF is 'close' to the preemptive FSF in the limit. She also establishes a state-space collapse result, which shows that, under FSF, all the faster servers are constantly busy, and the only possible idleness is within the slowest servers.

While FSF is optimal to minimize costs, it may be considered as 'unfair' to the servers in terms of how the workload is distributed. It is of interest to determine good policies subject to 'fairness' constraints (for example, idleness should be distributed across different server pools). SBR subject to such fairness considerations is studied in Armony and Ward [7] and Ward and Armony [85]. There, good routing policies also take server idle times into account. One such policy is called the longest-idle-server-first. An attractive feature of the longest-idle-server-first policy is that the stationary distribution of systems under this policy often has a product form [1], which makes it analytically tractable for performance analysis.

N-model In the N-model, one usually assumes that class 1 customers have a higher holding cost, in which case pool 2 servers should be used to help class 1 customers. We start by considering the N-model without abandonment. In this case, letting pool 2 give strict priority to class 1 can lead to system instability under non-preemption even for $\rho < 1$ [43,53]. This is because pool 2 may provide too much help, leaving pool 1 idling while there are still class 2 customers waiting. One way to overcome the unwanted idleness is to impose a threshold. In particular, pool 2 gives priority to class 1 only when the class 1 queue is larger than the threshold. This idea can be viewed as having a safety stock for class 1. Bell and Williams [16] show that if the threshold is properly chosen and preemptive priority is allowed for class 1 customers when their queue length is larger than the threshold, the threshold-based priority rule is asymptotically optimal in the conventional heavy-traffic regime. The cost structure in Bell and Williams [16] contains linear holding cost and the objective is to minimize the infinite horizon discounted cost.

In the many-server QED regime, the idleness induced by strict priority is of less concern. This is because each class 1 customer only takes a $1/s_2^n$ fraction of the service capacity from pool 2, which is almost negligible when n is large. This reflects a fundamental difference between the conventional heavy traffic scaling and the many-server heavy-traffic scaling. Tezcan and Dai [84] study the N-model with pool-dependent service rates and linear holding costs in the QED regime. In particular, pool-dependent service rates require that $\mu_{12} = \mu_{22} = \mu_2$. They show that a $c\mu$-type greedy policy is asymptotically optimal. The policy keeps servers non-idling; when a server becomes available, it next serves the waiting customer from the more expensive queue; and when an arriving customer (in class 1) can choose from multiple servers, it picks a faster server. The intuition is that, in general, a good policy should aim to keep all of the servers busy except the slowest one and all of the queues empty except the cheapest one. Tezcan and Dai [84] also allow abandonment, assuming the abandonment rates are smaller than the service rates.

We note that abandonment can sometimes change the structure of the optimal policy. For example, if the more expensive class (class 1) also happens to have a higher abandonment rate, then when there are many customers in the system, it can be more cost-effective to give priority to class 2 jobs. In particular, by keeping a longer class 1 queue, we can take advantage of the higher abandonment rate of class 1 to help reduce the queue more quickly. Ghamami and Ward [45] study the N-model with abandonment and establish the asymptotic optimality of a two-threshold policy under the conventional heavy-traffic regime. Under this policy, when the abandonment rate of class 1 is large enough, the pool 2 server gives priority to class 1 customers when the

class 1 queue is larger than a threshold but the total workload is smaller than another threshold.

X-model and beyond When the holding cost is increasing and (strongly) convex, Mandelbaum and Stolyar [66] consider a general PSS with multiple customer classes and multiple pools of fully flexible (multi-skilled) servers. They show that the generalized $c\mu$-rule minimizes the instantaneous and cumulative queueing costs, asymptotically, over essentially all scheduling disciplines, preemptive or non-preemptive, under the conventional heavy-traffic scaling. (Each pool has only one server in Mandelbaum and Stolyar [66] and there is no abandonment.) The generalized $c\mu$-rule says that when a pool j server becomes available (at time t), it serves a class i^* customer next, where

$$i^* \in \arg\max_{i:Q_i(t)>0} C_i'(Q_i(t))\mu_{ij}.$$

The policy essentially aims at myopically maximizing the instantaneous cost decreasing rate, which agrees with the intuition developed in the V-model.

Gurvich and Whitt [50] consider a similar multi-class multi-pool system, but allow each server pool to have multiple servers. ([50] also allows abandonment.) They propose a family of routing policies called queue-idleness-ratio (QIR) rules. Under QIR, a newly available server next serves the customer class (among the eligible ones) whose queue length most exceeds a state-dependent proportion of the total queue length, and an arriving customer is routed to the server pool whose idleness exceeds a state-dependent proportion of the total idleness the most. In the QED regime, this policy achieves state-space collapse under certain regularity conditions (see Theorem 3.1 in [50]). See also [32] for a similar result. In particular, the state-space collapse result indicates that the scaled queue length of each class is a pre-specified state-dependent ratio of the total queue length. Similarly, the scaled idleness at each pool is a pre-specified state-dependent ratio of the aggregate idleness. Thus, in the limit, we only need to keep track of the total queue length and the aggregated idleness processes. Utilizing the state-space collapse result, Gurvich and Whitt [49] establish that when costs C_i are convex and the service rates are only pool-dependent, a properly specified QIR control is asymptotically optimal in the QED regime. The objective there is to minimize the cumulative holding cost over a finite horizon. Intuitively, the optimal QIR control works as follows: the routing component places all the idleness to the slowest server pool, and the scheduling component distributes the total queue length among the individual customer classes so that the instantaneous holding cost is minimized. In particular, the scheduling policy is tried to mimic/stay close to the solution of the following:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{I} C_i(q_i) \\ \text{subject to} \quad & \sum_{i=1}^{I} q_i = \bar{q}, \\ & q_i \geq 0 \text{ for } i = 1, \ldots, I. \end{aligned} \tag{3}$$

This indicates that when there are $\bar{q}$ customers in the queue, the queue ratio is $q_i^*(\bar{q})/\bar{q}$, where $q_i^*(\bar{q})$ is the solution to (3).

The most general asymptotic optimality result for PSSs in the QED regime is established in Atar [9]. The objective is to minimize the infinite horizon discounted cost. It derives asymptotically optimal policies by properly 'translating' the optimal policy for the associated limiting diffusion control problem to the stochastic systems. However, it is worth pointing out that, due to the generality of the result, the Hamilton–Jacobi–Bellman (HJB) equation associated with the diffusion control problem cannot be solved analytically in general. Thus, it is hard to derive structural insights from this class of HJB-derived policies. Atar [9] studies both the preemptive version and non-preemptive version of the problem and shows that the two versions are not, in general, asymptotically equivalent in this regime. However, he also shows that under appropriate assumptions on the structure of the system, the two versions of the problem are asymptotically equivalent. In particular, the structural assumption requires the graph describing the compatibility of the network to form a tree and all the compatible class-pool pairs to have non-negligible flow in the corresponding fluid model.

Perry and Whitt [70,71] study how to use overflow (routing class i customers to pool $j, i \neq j$) to handle unexpected demand shocks (overload) in the X-model. They assume overflow incurs an efficiency loss, i.e., $\mu_{11}\mu_{22} > \mu_{12}\mu_{21}$, so that each pool in general prefers to serve its own customer class. To handle demand shocks, they propose a fixed-queue-ratio policy with threshold (FQR-T). The fixed-queue-ratio policy is studied in Gurvich and Whitt [51] for PSSs with pool-dependent service rates. The extra thresholds in FQR-T are introduced to avoid unwanted (excessive) sharing in normal load. The thresholds are also used to automatically detect when one class becomes overloaded.

3.3 Other related works

Our focus in this section is on asymptotically optimal scheduling policies. Most work reviewed above relies on some heavy-traffic asymptotic mode of analysis. Beyond these works, many other analytically tractable scheduling policies have been studied in the literature. An important class of policies are those under which the PSS has an explicit product-form stationary distribution. One such policy is First-Come-First-Served-and-Assign-Longest-Idle-Server [2]. Under this policy, a server always chooses to serve the longest waiting compatible customer, and a customer is assigned to the longest idle compatible server. Other policies that lead to closed-form stationary distributions include queues with redundant requests in computer scheduling [42], and store-and-forward allocation policy in processor-sharing queues [18]. See also Gardner and Righter [41] for a survey on product forms for PSSs under FCFS scheduling.

For the cost objective, we only considered different forms of holding cost. We will discuss the effect of overflow cost in Sect. 5. In addition to these costs, other performance metrics have been considered in the literature. These include chance constraints [46], deadline constraints [57], fairness constraints [88], etc. The optimal routing policies with respect to these performance metrics remain largely open.

4 Complications brought by healthcare applications

In this section, we review some recent developments of SBR in the context of healthcare applications, especially in patient-flow management.

Various specialized queueing models have been developed to study patient flow in hospitals [6]. For example, to capture the fact that the patients may return to service (consulting the physician) several times during their stay in the Emergency Department, Yom-Tov and Mandelbaum propose and study a queueing model that explicitly accommodates reentrant customers [92]. The paper [57] then studies how to prioritize new versus reentrant patients with deadline constraints. To capture the batch departure phenomenon in inpatient wards, models with special discharge mechanisms have been proposed and studied in Dong and Perry [36] and Shi et al. [77].

In what follows, we focus on how routing decisions may affect patient outcomes, and how that in turn may affect the routing decisions. The challenge of balancing the delicate trade-off between system efficiency and quality of care in face of this interplay stimulates a lot of interesting research opportunities.

The effect of admission delay on patient outcome. Delay arises often in various healthcare settings due to the high utilization of resources and the high variability in demand. Delay in receiving care can lead to adverse outcomes such as a longer length of stay [24] or even a higher mortality rate [22]. The paper [35] shows that when delay leads to longer service requirement, even a simple single-class multi-server queue with abandonment can exhibit bi-stability. In particular, the system can alternate randomly between two performance regimes. The underlying mechanism can be intuitively understood as a snowball effect where delayed patients have increased service requirements, causing further delays to other patients, who in turn might require even longer service times [76].

When there are multiple classes of patients and a single pool of servers, the more severe patients are often prioritized. However, delayed treatment of less critical patients may cause them to deteriorate into the more severe classes. This deterioration phenomenon can be captured by allowing patients to 'transition' from a less severe class to a more severe one while waiting. In this case, how to prioritize different patient classes can be highly nontrivial. On the one hand, providing care for patients when they are less critical could mean that fewer resources are needed. This can be beneficial from the system throughput perspective. On the other hand, utilizing limited capacity for less severe patients takes the resource away from other more critical patients. Hu et al. [56] study this problem using fluid approximations and optimal control theory. More generally, analyzing the benefit of proactive care or proactive services (providing service to less critical/urgent patients) in a limited resource environment has attracted much attention recently; see, for example, Delana et al. [34] and Örmeci and Güneş [69].

The effect of off-service placement on patient outcome. Inpatient ward beds are usually grouped into different specialized units, with each unit designated to serve patients in certain primary specialties. This focused care model is known to facilitate better coordination of care and nurse training [17]. However, to avoid excessive admission delay, many hospitals choose to assign patients whose designated unit (server pool) is full to an available bed in a unit of a different specialty. This is referred to as off-

service placement. Empirical evidence has shown that off-service placement can lead to adverse outcomes such as a longer length of stay or a higher readmission rate [79]. This poses an interesting trade-off. On the one hand, off-service placement can help create more resource pooling and thus reduce admission delay. On the other hand, patients who are placed off-service may require a longer service time, which can create more congestion in the system. This can further lead to a snowball effect as well: Delayed patients are placed off-service, leading to increased service times that can cause more delays for future patients. Recently, Dong et al. [37] take a data-driven approach to study this trade-off.

From an analysis point of view, when the service rate is both class-dependent and pool-dependent, the state space and policy space soon grow prohibitively large to conduct any exact or even numerical analysis. Adding to the complication is the non-preemption assumption in most healthcare settings. To optimize the routing decisions, good approximation techniques are needed. Recent developments in solving these constrained Markov decision processes include approximate dynamic programming [20,28] and Lagrangian relaxation [12,33].

When patients are placed off-service, in addition to a longer length of stay, there might be other costs incurred from the hospital's perspective. For example, nurses in the off-service unit may incur a heavier workload as they now have to take care of multiple types of patients. Physicians may also incur greater workload as they now have to travel from the primary unit to the off-service units to see the patients. These can be modeled via an overflow cost (penalty). Adding the overflow cost to the cost objective can change the structure of the optimal policy. We conduct detailed numerical experiments of this setting in Sect. 5.

Readmission. A prominent feature in patient flow management and a key measure for quality of care is patient readmission. Several operational measures are associated with the readmission rate. We have mentioned that off-service placement can lead to a higher readmission rate. It has also been shown that shortening service time (service speed-up) is likely to cause a higher readmission rate [61]. This poses an interesting tension. Hospitals tend to use early discharge (speed-up service rate) to alleviate congestion at present [59,60], which will allow newly arrived patients to get timely treatment. However, this may substantially increase the future workload due to increased readmissions. If this trade-off is not managed properly, it can lead to bi-stability due to, again, a snowball effect [25]. More generally, analyzing the trade-off between service speed and service quality is an active area of research; see, for example [23,93].

Routing with predictive information. In recent years, the growing availability of data and the development of statistical learning techniques have provided us with various accurate predictive models. For example, predictive models have been developed to evaluate hospital-acquired infection risk, the risk of intensive care unit (ICU) admission, the risk of cardiovascular events, etc. There is a growing number of works studying how to effectively incorporate the predictive information into operational decision making. Xu and Chan [91] study admission control in the Emergency Department using predicted patient demand information. Bassamboo and Randhawa [14] and Mandelbaum and Momcilovic [65] study scheduling using customers' patience time. Chen and Dong [26] and Wierman and Nuyens [89] study scheduling using predicted

service times. Argon and Ziya [3] and Sun et al. [82] study patient prioritization using predicted class information. Shi et al. [78] study patient discharge using predicted readmission risk.

5 Patient routing with overflow cost: a numerical study

In this section, we focus on one specific example motivated by the patient routing problem in hospital inpatient flow management. In particular, on top of the typical SBR setting reviewed earlier, we incorporate a one-time overflow cost when a patient is routed to a bed (server) in a non-primary ward (non-dedicated server pool). Overflow cost has not been extensively studied in the literature, and adding it to the cost objective can change the structure of the optimal policy from the ones discussed in Sect. 3. To understand the complexity in this case, we first introduce the Markov decision process (MDP) formulation for modeling the routing problem. We then numerically solve the MDP for some small problem instances to generate insights into the structure of the optimal routing policy. Finally, we adapt several existing heuristics, such as the $c\mu$-rule and the maximum pressure policy, to incorporate the overflow cost, and compare their performance numerically.

5.1 MDP formulation

We consider a discrete-time model where each unit of time can be thought of as a day. For the simplicity of exposition, we assume the service rate is pool-dependent and there is no abandonment. Motivated by the patient-flow application, we do not allow preemption. The order of events is as follows: At the start of each day t, we observe the queue length of each class, $Q_i(t)$, and the number of busy servers in each pool, $Z_j(t)$. (Note that since the service rate is pool-dependent, we do not need to track each $Z_{ij}(t)$; we only need to know the total number of busy servers in pool j.) After observing the state, we make the admission decision $a_{ij}(t)$, which is how many class i patients to admit to pool j. This incurs a cost of $\sum_{i,j} \phi_{ij} a_{ij}(t)$. In addition, a holding cost of $\sum_i h_i(Q_i(t) - \sum_j a_{ij}(t))$ is incurred. After the admission decision is made, random departures occur, followed by random arrivals, bringing the system into new states $Q_i(t+1)$ and $Z_j(t+1)$ at the start of the next day.

We consider a long-run average cost-minimization problem and focus on the class of Markovian policies (non-anticipatory). Let

$$\mathcal{S}(t) = (Q_1(t), \ldots, Q_I(t), Z_1(t), \ldots, Z_J(t))$$

be the state of the system. For a given state $\mathcal{S}(t) = \zeta$, we denote by $\mathcal{A}(\zeta)$ the set of feasible actions:

$$\mathcal{A}(\zeta) = \left\{ (a_{ij})_{i=1,\ldots,I, j=1,\ldots,J} : a_{ij} \geq 0, \sum_i a_{ij} + z_j \leq s_j, \sum_j a_{ij} \leq q_i \right\}.$$

Recall that s_j is the capacity of pool j. Under suitable regularity conditions on the ergodicity of the Markov process, let g be the gain and $V(\zeta)$ be the bias of the MDP; see, for example, [73]. Then,

$$g + V(\zeta) = \min_{a \in \mathcal{A}(\zeta)} \left\{ \sum_{i,j} \phi_{ij} a_{ij} + \sum_i h_i \left(Q_i(t) - \sum_j a_{ij} \right) + \sum_{\zeta'} V(\zeta') P(\zeta'; \zeta, a) \right\},$$

where $P(\zeta'; \zeta, a)$ denotes the transition probability from state ζ to state ζ' under the policy $a = \{a_{ij}\}$. Specifically, for $q_i' = q_i - \sum_j a_{ij} + k_i$ and $0 \le z_j' \le z_j + \sum_i a_{ij}$,

$$P(\zeta'; \zeta, a) = \prod_i P(\tilde{A}_i = k_i) \prod_j P\left(\tilde{S}_j = z_j + \sum_i a_{ij} - z_j' \,\middle|\, z_j + \sum_i a_{ij} \right),$$

where $\tilde{A}_i$ is a Poisson random variable with rate λ_i and $\tilde{S}_j$ given $z_j + \sum_i a_{ij}$ is a binomial random variable with parameters $z_j + \sum_i a_{ij}$ and μ_j.

5.2 Optimal solution

In this subsection, we consider two small instances of the problem and solve the corresponding MDPs numerically. Motivated by the patient flow application, we assume $I = J$ and each class i has a primary server pool $j = i$. When sending a customer from class i to the dedicated server pool i, we incur no overflow cost, i.e., $\phi_{ii} = 0$. However, when sending a customer from class i to a non-primary server pool j, $j \neq i$, we incur some overflow cost, i.e., $\phi_{ij} > 0$.

To enable efficient computation, we implement two simplifications. First, to handle the infinite state space, we truncate the state space by imposing a common pre-specified threshold m for each class i. In particular, if so many class i customers arrive that the queue length of class i will exceed m, the excess customers (beyond m) are rejected from the system at no cost. Second, we enforce maximum primary assignments. In particular, as many primary assignments a_{ii} as possible are made at each epoch and only then are overflow decisions a_{ij} ($j \neq i$) made.

5.2.1 Two-class example

We set $m = 45$ and numerically solve the MDP in a two-class two-pool setting where each pool has a capacity of 5. The arrival rates are (0.65, 0.85) and the service rates are 0.25 for each pool. The holding costs are 1 for each class, and we try different values of the overflow cost, i.e., (i) $\phi_{ij} = 0.2$, (ii) $\phi_{ij} = 2$, and (iii) $\phi_{ij} = 10$ for $i \neq j$. Note that the classes are symmetric in everything except the arrival rates, and class 2 has a higher arrival rate than class 1.

Figures 3, 4 and 5 illustrate the optimal policy under different overflow costs. We make a few observations from the figures. First, as the overflow cost increases, we do less overflow. In particular, when $\phi_{ij} = 0.2$ (Fig. 3), we do as much overflow as possible after maximum primary assignment. When $\phi_{ij} = 2$ (Fig. 4), we start to reserve some pool 2 capacity even when there are class 1 customers waiting. For

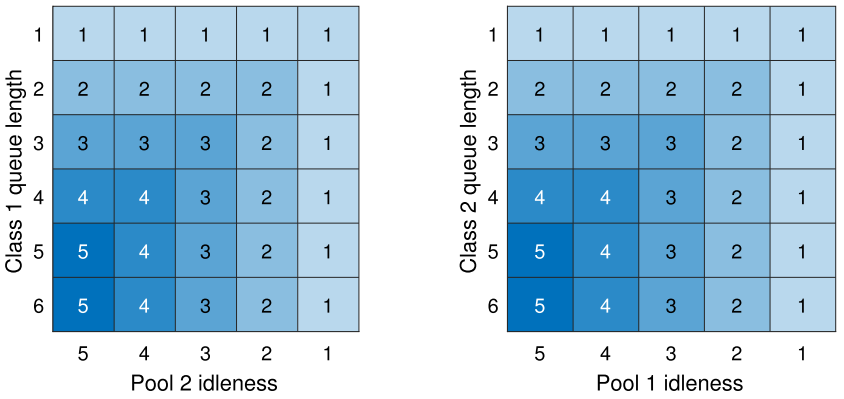


Fig. 3 Number overflowed when $\phi = 0.2$, class i has a queue and pool j has idleness

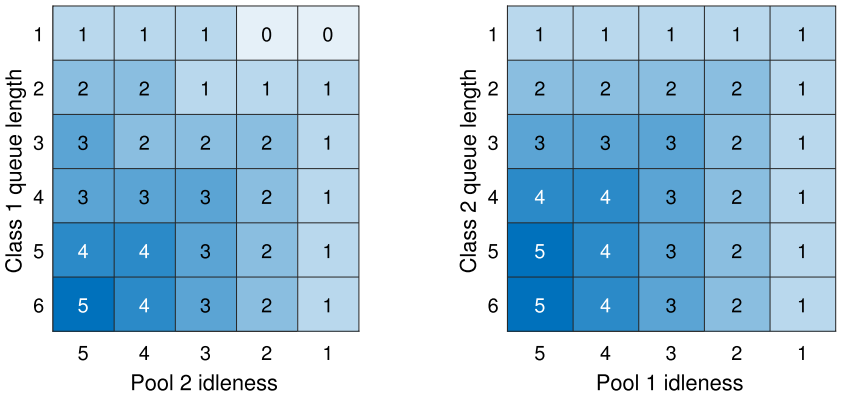


Fig. 4 Number overflowed when $\phi = 2$, class i has a queue and pool j has idleness

example, when pool 2 has two available servers and class 1 has one customer waiting after its primary assignment, the optimal policy keeps the two servers in pool 2 idling while keeping the class 1 customer waiting. When $\phi_{ij} = 10$ (Fig. 5), we only overflow when the queue is large enough. For example, when pool 2 has one idle server, the optimal policy only starts overflowing class 1 customers to pool 2 when there are more than seven class 1 customers waiting after its primary assignment. Likewise, when pool 1 has an idle server, the optimal policy only starts overflowing class 2 customer to pool 1 when there are more than four class 2 customer waiting after its primary assignment. Secondly, we observe from Fig. 5 that the number of customers we overflow is increasing in the queue length and increasing in the number of idle servers. For instance, when $\phi_{ij} = 10$ and pool 2 has three idle servers, we overflow one class 1 customer when there are seven class 1 customers waiting, and we overflow three class 1 customer when there are more than nine waiting. Lastly, because class 2 is more heavily loaded than class 1, i.e., $\lambda_2 > \lambda_1$, we tend to do more overflow for class 2. In particular, when $\phi_{ij} = 2$, we start reserving pool 2 capacity by keeping

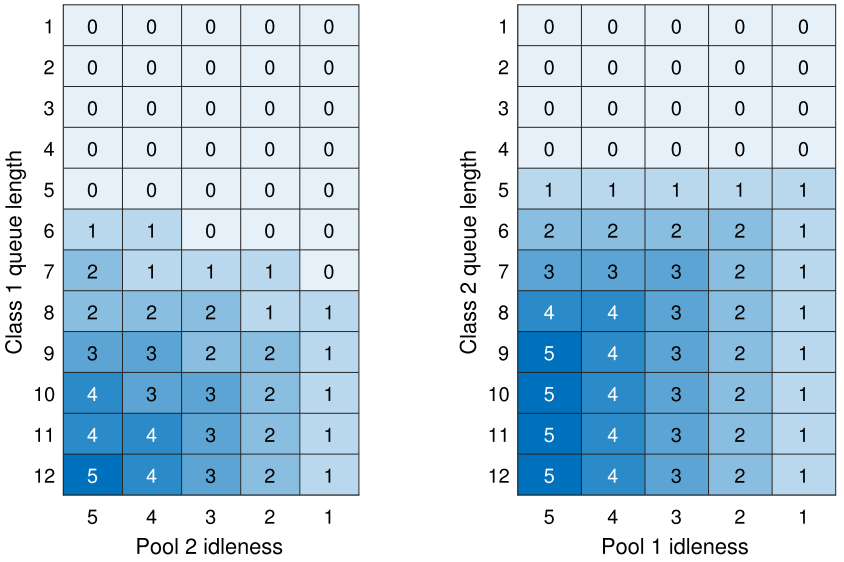


Fig. 5 Number overflowed when $\phi = 10$, class i has a queue and pool j has idleness

them idling when there are still class 1 customers waiting. In contrast, we still do the maximum possible overflow for class 2 customers.

5.2.2 Three-class example

We next consider a three-class three-pool setting where each pool has capacity 5. The arrival rates are $(0.65, 0.75, 0.85)$ and the service rates are 0.25 for each pool. The holding costs are 1 for each class, and we test two different overflow costs: (i) $\phi_{ij} = 0.2$ and (ii) $\phi_{ij} = 2$ for $i \neq j$. We set $m = 15$.

Figures 6, 7, 8 and 9 illustrate the optimal policy under different overflow costs. We make the following observations from the figures.

When deciding which pool (if any) to overflow a customer (Figs. 6, 7), we tend to overflow the customer to the pool with more idleness. When two pools have a similar numbers of idle servers, we tend to prioritize the pool that has a smaller primary offered load. For example, in the first plot in Fig. 6, when there are two idle servers in pool 3, we overflow the class 1 customer to pool 3 when pool 2 has fewer than two idle servers, but overflow to pool 2 when it has more than two idle servers. When both pool 2 and pool 3 have two idle servers, we overflow the class 1 customer to pool 2 as class 2 has a smaller offered load than class 3. Comparing the three cases in Fig. 6, we also note that the larger the difference between the primary offered loads, the more preferred the pool with a lighter load is to overflow the customer. Moreover, similar to the two-class case, as ϕ_{ij} increases, we do less overflow, especially when the number of idle servers is small. For example, when $\phi_{ij} = 2$, we do not overflow the waiting class 1 customer when pool 2 and pool 3 both have fewer than two idle servers.

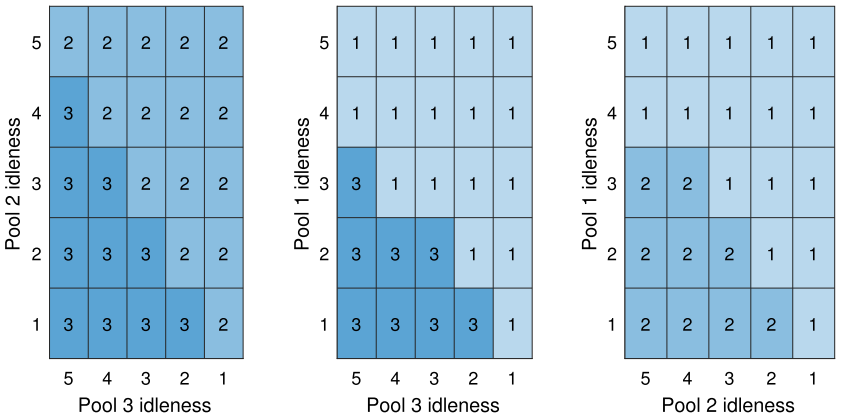


Fig. 6 Overflow pool when $\phi_{ij} = 0.2$, there is one class i in queue and the other pools have different numbers of idle servers

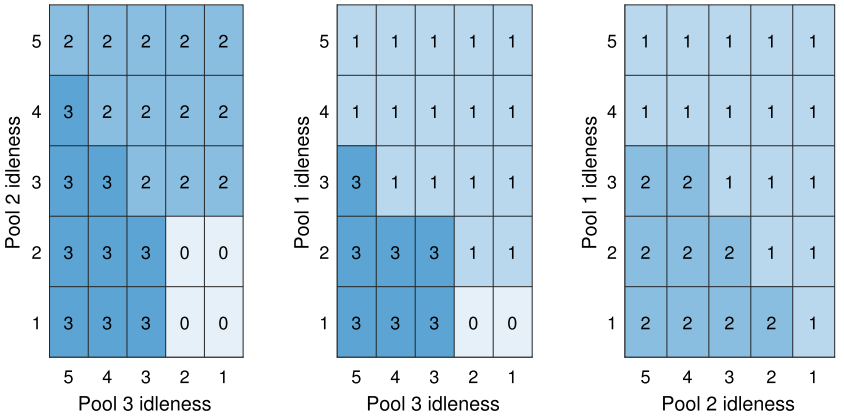


Fig. 7 Overflow pool when $\phi_{ij} = 2$, there is one class i in queue and the other pools have different numbers of idle servers

When deciding which non-primary class a newly available server should help next (Figs. 8, 9), we tend to help the class with a larger queue. For example in the first plot in Fig. 8, when class 3 has three customers waiting after its primary assignment, the idling pool 1 server helps class 3 when there are less than or equal to three class 2 customers waiting, but switches to help class 2 when there are more than three class 2 customers waiting. When two queues have similar lengths, we prioritize the queue with a larger arrival rate. For example, in the second plot in Fig. 8, when both queues have six customers waiting, the idling pool 2 server helps serve class 3, which is the more overloaded class. Moreover, similar to the two-class case, as ϕ_{ij} increases, we do less overflow, especially when the queue lengths are small.

The above numerical results suggest that when deciding which pool to route customers to, the system state should be taken into account. In general we would prioritize the pool with more idleness and a lighter primary offered load. Similarly, we should

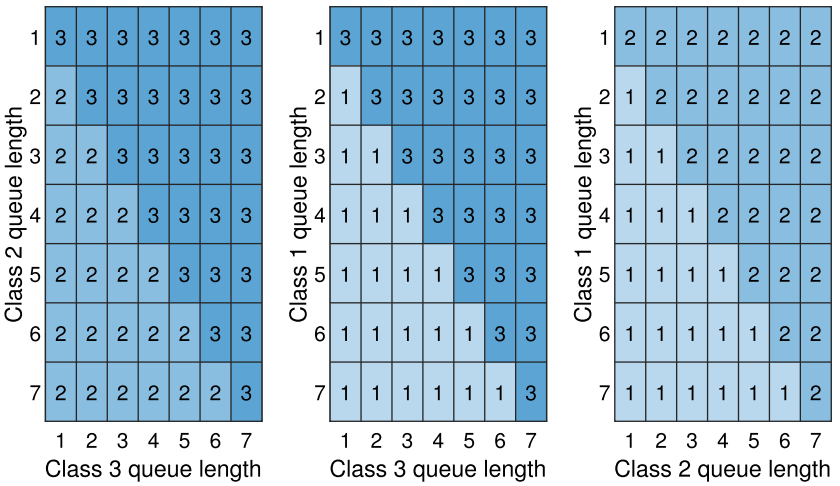


Fig. 8 Overflow class when $\phi = 0.2$, there is 1 idle server in pool j and the other two queues have different numbers of customers waiting

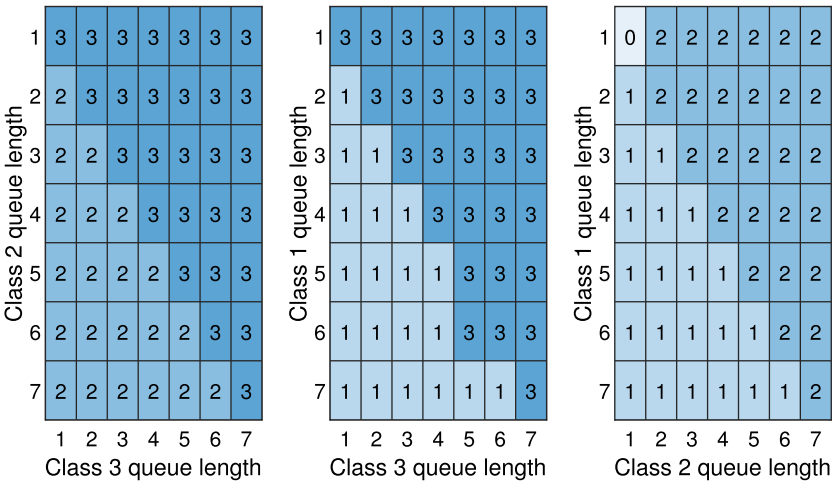


Fig. 9 Overflow class when $\phi = 2$, there is 1 idle server in pool j and the other two queues have different numbers of customers waiting

consider the system state when deciding which class to prioritize when scheduling servers. In general we would prioritize the class with a longer queue and a larger arrival rate.

5.3 Heuristic policies

To manage large-scale systems, numerically solving the MDP is computationally prohibitive due to the large state space and action space. In such cases, it is natural to come

up with some heuristic policies. In this section, we compare five heuristic policies: (i) the classic $c\mu$-rule, (ii) the classic maximum pressure policy, (iii) the modified $c\mu$-rule, which takes the overflow cost into account, (iv) the modified maximum pressure policy, which takes the overflow cost into account, and (v) a policy based on estimated waiting time. We provide more details about policies (iii)–(v) next.

For the **modified $c\mu$ rule**, we derive the assignment action $a_{ij}(t)$ according to the following integer program:

$$\begin{aligned} \max \ & \sum_{i=1}^{I}\sum_{j=1}^{J}(h_i\mu_{ij} - \phi_{ij})a_{ij}(t) \\ \text{s.t. } & a_{ij}(t) \geq 0, \ \sum_{i=1}^{I} a_{ij}(t) + Z_{ij}(t) \leq s_j \text{ and } \sum_{j=1}^{J} a_{ij}(t) \leq Q_i(t). \end{aligned} \tag{4}$$

The optimization problem (4) suggests that we rank the patient-bed assignment pair using the $(h_i\mu_{ij} - \phi_{ij})$ index. This index is adjusting the original $c\mu$-index, $h_i\mu_{ij}$, by subtracting an overflow cost ϕ_{ij}. Our goal here is to maximize the instantaneous cost reduction rate.

For the **modified maximum pressure policy**, we derive the assignment action $\{a_{ij}\}$ from the following integer program:

$$\begin{aligned} \max \ & \sum_{i=1}^{I}\sum_{j=1}^{J}(h_i Q_i(t)\mu_{ij} - \phi_{ij})a_{ij}(t) \\ \text{s.t. } & a_{ij}(t) \geq 0, \ \sum_{i=1}^{I} a_{ij}(t) + Z_{ij}(t) \leq s_j \text{ and } \sum_{j=1}^{J} a_{ij}(t) \leq Q_i(t). \end{aligned} \tag{5}$$

Compared to the original max-pressure policy, we now adjust the index $h_i Q_i(t)\mu_{ij}$ with the overflow cost ϕ_{ij}.

The last heuristic we consider takes estimated waiting time into account. In particular, we balance the overflow cost and the waiting cost by solving the following integer program:

$$\begin{aligned} \max \ & \sum_{i=1}^{I}\sum_{j=1}^{J}(h_i W_i(t) - \phi_{ij})a_{ij}(t) \\ \text{s.t. } & a_{ij}(t) \geq 0, \ \sum_{i=1}^{I} a_{ij}(t) + Z_{ij}(t) \leq s_j \text{ and } \sum_{j=1}^{J} a_{ij}(t) \leq Q_i(t). \end{aligned} \tag{6}$$

Here, $W_i(t)$ represents an estimate of the waiting time for the class i customers. We refer to this policy as the **look-ahead policy**.

Table 1 Performance comparison for a small system [$s_j = 5$, $\mu_j = 0.25$, $\lambda = (0.65, 0.75, 0.85)$]

	optimal	$c\mu$	max-p.	mod. $c\mu$	mod. max-p.	look-ahead
$\phi_{ij} = 0.2$						
Total cost	0.11	0.33	0.22	0.11	0.12	0.12
Holding	0.07	0.06	0.06	0.07	0.07	0.07
Overflow	0.04	0.26	0.15	0.05	0.05	0.05
$\phi_{ij} = 0.2$						
Total cost	0.50	2.71	1.57	1.12	0.91	0.57
Holding	0.12	0.06	0.07	1.12	0.88	0.12
Overflow	0.38	2.65	1.50	0.00	0.03	0.45

The standard error for each entry is less than 0.001

Different methods for estimating the waiting time have been developed in the literature. Here we take a conservative approach, as in [8], and set

$$W_i(t) = Q_i(t)/(s_i\mu_{ii} - \lambda_i). \tag{7}$$

$W_i(t)$ as defined in (7) can be interpreted as an approximation of the time to clear the current queue using dedicated capacity only.

We consider the same three-class three-pool model as studied in Sect. 5.2.2. We test both a small-scale setting where exact analysis is still feasible, and a larger-scale setting with higher arrival rates and more servers in each pool.

We start with the small system that is exactly the same as the three-pool system studied in Sect. 5.2.2. In this setting, we are able to compare the performance of different heuristics with the performance under the optimal policy we have numerically solved. Table 1 reports the corresponding long-run average cost evaluated through simulation. We simulate 10 replications for each policy to evaluate the mean and standard error of the average cost. Each replication contains 5×10^5 days, with the first 10^4 days excluded when calculating the average to eliminate the initial transiency. A common sequence of random numbers is used when comparing different policies. We observe that when ϕ_{ij} is small, i.e., $\phi_{ij} = 0.2$, the modified $c\mu$ rule (mod. $c\mu$), the modified maximum pressure policy (mod. max-p.), and the look-ahead policy all perform very well. In this case, not incorporating the overflow cost, as in the original $c\mu$ rule ($c\mu$) or the original maximum pressure policy (max-p.), leads to too much overflow. However, when ϕ_{ij} is large, i.e., $\phi_{ij} = 2$, naively incorporating the overflow cost into the $c\mu$-index or the max-pressure index, as in the modified $c\mu$ rule or the modified maximum pressure policy, leads to too little overflow. In this case, the look-ahead policy works very well, while the $c\mu$-rule and the max-pressure policy still induce too much overflow.

We next compare the five heuristic policies for a larger-scale system. In particular, we increase the arrival rates to (6.5, 7.5, 8.5) and the number of servers to (33, 36, 39). This system size is more realistic for the hospital inpatient setting. In our collaborating hospital, each inpatient ward (server pool) contains 20 to 40 beds. Solving the MDP is

Table 2 Performance comparison for a large system [$s = (33, 36, 39)$, $\mu_j = 0.25$, $\lambda = (6.5, 7.5, 8.5)$]

	$c\mu$	max-p.	mod. $c\mu$	mod. max-p.	look-ahead
$\phi_{ij} = 0.2$					
Total cost	1.70	2.02	0.34	0.40	0.40
Holding	0.19	0.18	0.19	0.19	0.19
Overflow	1.51	1.83	0.15	0.21	0.21
$\phi_{ij} = 2$					
Total cost	15.26	18.53	3.21	1.76	1.85
Holding	0.19	0.19	3.21	0.24	0.21
Overflow	15.07	18.34	0.00	1.52	1.64

The standard error for each entry is less than 0.005

computationally prohibitive in this case. Thus, Table 2 only reports the performance (estimated long-run average cost) for the heuristic policies. We again run the simulation experiments for 10 replications, with 5×10^5 days in each replication and the first 10^4 days being excluded. A common sequence of random numbers is used when comparing different policies. We make similar observations as for the small systems. When ϕ_{ij} is small, the modified $c\mu$ rule, the modified maximum pressure policy, and the look-ahead policy all perform quite well. When ϕ_{ij} is large, the modified maximum pressure and the look-ahead policy perform much better than the other policies.

Above all, the observations suggest that when we design heuristic algorithms in this setting, the overflow cost plays an important role and a more dynamic priority rule in general performs better than static priority rules. In this section, we have not considered other complications in patient flow management, such as class-and-pool-dependent service rates and time-varying arrival rates. How to derive good routing policies when incorporating these realistic features is a topic for fruitful future research.

6 Conclusion

In this survey, we study skill-based routing problems in multi-class multi-pool parallel-server systems. These problems arise in various service operations management settings. We start by discussing stability and reviewing the maximum pressure policy, which is throughput optimal in a variety of settings. We then review good policies that can minimize different cost objectives in various special cases of PSSs. Heavy-traffic asymptotic analysis is a powerful tool for generating structural insights into the optimal policy for these systems. Lastly, we discuss complications brought by modern healthcare applications. The goal is to help generate interesting future research directions. We also provide a numerical study on one complication raised in the hospital inpatient flow: whether and where to overflow a waiting patient. We show that adding an overflow penalty cost to the objective function significantly changes the optimal policy. We compare a few heuristic policies adapted from the literature and show that it is important to adjust for the overflow cost in these heuristics.

Acknowledgements The authors would like to thank Kristen Gardner, Yoni Nazarathy, and the referee for their insightful suggestions. Support from NSF Grant CMMI-1762544 is gratefully acknowledged by J. Dong.

References

1. Adan, I., Weiss, G.: A loss system with skill-based servers under assign to longest idle server policy. Probab. Eng. Inf. Sci. **26**(3), 307–321 (2012)
2. Adan, I., Weiss, G.: A skill based parallel service system under FCFS-ALIS: steady state, overloads, and abandonments. Stoch. Syst. **4**(1), 250–299 (2014)
3. Argon, N., Ziya, S.: Priority assignment under imperfect information on customer type identities. Manuf. Serv. Oper. Manag. **11**(4), 674–693 (2009)
4. Armony, M.: Dynamic routing in large-scale service systems with heterogeneous servers. Queueing Syst. **51**(3–4), 287–329 (2005)
5. Armony, M., Bambos, N.: Queueing dynamics and maximal throughput scheduling in switched processing systems. Queueing Syst. **44**(3), 209–252 (2003)
6. Armony, M., Israelit, S., Mandelbaum, A., Marmor, Y.N., Tseytlin, Y., Yom-Tov, G.B.: On patient flow in hospitals: a data-based queueing-science perspective. Stochas. Syst. **5**(1), 146–194 (2015)
7. Armony, M., Ward, A.R.: Fair dynamic routing in large-scale heterogeneous-server systems. Oper. Res. **58**(3), 624–637 (2010)
8. Ata, B., Peng, X.: An optimal callback policy for general arrival processes: a pathwise analysis. Oper. Res. **68**(2), 327–347 (2020)
9. Atar, R.: Scheduling control for queueing systems with many servers: asymptotic optimality in heavy traffic. Ann. Appl. Probab. **15**(4), 2606–2650 (2005)
10. Atar, R.: A diffusion regime with nondegenerate slowdown. Oper. Res. **60**(2), 490–500 (2012)
11. Atar, R., Mandelbaum, A., Reiman, M.I.: Scheduling a multi class queue with many exponential servers: asymptotic optimality in heavy traffic. Ann. Appl. Probab. **14**(3), 1084–1134 (2004)
12. Balseiro, S., Brown, D.B., Chen, C.: Dynamic pricing of relocating resources in large networks. Manage. Sci. (forthcoming) (2020)
13. Bassamboo, A., Harrison, J.M., Zeevi, A.: Design and control of a large call center: asymptotic analysis of an lp-based method. Oper. Res. **54**(3), 419–435 (2006)
14. Bassamboo, A., Randhawa, R.: Scheduling homogeneous impatient customers. Manag. Sci. **62**(7), 2129–2147 (2016)
15. Bassamboo, A., Zeevi, A.: On a data-driven method for staffing large call centers. Oper. Res. **57**(3), 714–726 (2009)
16. Bell, S.L., Williams, R.J.: Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: asymptotic optimality of a threshold policy. Ann. Appl. Probab. **11**(3), 608–649 (2001)
17. Best, T.J., Sandçı, B., Eisenstein, D.D., Meltzer, D.O.: Managing hospital inpatient bed capacity through partitioning care into focused wings. Manuf. Serv. Oper. Manag. **17**(2), 157–176 (2015)
18. Bonald, T., Proutiere, A.: Insensitive bandwidth sharing in data networks. Queueing Syst. **44**(1), 69–100 (2003)
19. Borst, S., Mandelbaum, A., Reiman, M.I.: Dimensioning large call centers. Oper. Res. **52**(1), 17–34 (2004)
20. Braverman, A., Gurvich, I., Huang, J.: On the Taylor expansion of value functions. Oper. Res. **68**(2), 631–654 (2020)
21. Buyukkoc, C., Varaiya, P., Walrand, J.: The cμ rule revisited. Adv. Appl. Probab. **17**(1), 237–238 (1985)
22. Chalfin, D., Trzeciak, S., Likourezos, A., Baumann, B., Dellinger, R.: Impact of delayed transfer of critically ill patients from the emergency department to the intensive care unit. Crit. Care Med. **35**, 1477–1483 (2007)
23. Chan, C.W., Farias, V.F., Bambos, N., Escobar, G.J.: Optimizing intensive care unit discharge decisions with patient readmissions. Oper. Res. **60**(6), 1323–1341 (2012)
24. Chan, C.W., Farias, V.F., Escobar, G.J.: The impact of delays on service times in the intensive care unit. Manag. Sci. **63**(7), 2049–2072 (2017)

25. Chan, C.W., Yom-Tov, G., Escobar, G.J.: When to use speedup: an examination of service systems with returns. Oper. Res. **62**(2), 462–482 (2014)
26. Chen, Y., Dong, J.: Scheduling with service-time information: the power of two priority classes. Working paper (2019)
27. Cox, D.R., Smith, W.L.: Queues. Methuen, London (1961)
28. Dai, J., Shi, P.: Inpatient bed overflow: an approximate dynamic programming approach. Manuf. Serv. Oper. Manag. **21**(4), 894–911 (2019)
29. Dai, J.G., Lin, W.: Maximum pressure policies in stochastic processing networks. Oper. Res. **53**(2), 197–218 (2005)
30. Dai, J.G., Lin, W.: Asymptotic optimality of maximum pressure policies in stochastic processing networks. Ann. Appl. Probab. **18**(6), 2239–2299 (2008)
31. Dai, J.G., Tezcan, T.: Optimal control of parallel server systems with many servers in heavy traffic. Queueing Syst. **59**(2), 95 (2008)
32. Dai, J.G., Tezcan, T.: State space collapse in many-server diffusion limits of parallel server systems. Math. Oper. Res. **36**(2), 271–320 (2011)
33. Daniel Adelman, A.J.M.: Relaxations of weakly coupled stochastic dynamic programs. Oper. Res. **56**(3), 712–727 (2008)
34. Delana, K., Savva, N., Tezcan, T.: Proactive customer service: operational benefits and economic frictions. Manuf. Serv. Oper. Manag., Articles in Advance (2020)
35. Dong, J., Feldman, P., Yom-Tov, G.B.: Service systems with slowdowns: potential failures and proposed solutions. Oper. Res. **63**(2), 305–324 (2015)
36. Dong, J., Perry, O.: Queueing models for patient-flow dynamics in inpatient wards. Oper. Res. **68**(1), 250–275 (2020)
37. Dong, J., Shi, P., Zheng, F., Jin, X.: Off-service placement in inpatient ward network: resource pooling versus service slowdown. Working paper (2019)
38. Foley, R.D., McDonald, D.R.: Join the shortest queue: stability and exact asymptotics. Ann. Appl. Probab. **11**(3), 569–607 (2001)
39. Foss, S., Chernova, N.: On the stability of a partially accessible multi-station queue with state-dependent routing. Queueing Syst. **29**, 55–73 (1998)
40. Gans, N., Koole, G., Mandelbaum, A.: Telephone call centers: tutorial, review, and research prospects. Manuf. Serv. Oper. Manag. **5**(2), 79–141 (2003)
41. Gardner, K., Righter, R.: Product forms for FCFS queueing models with arbitrary server-job compatibilities: an overview. In: Queueing Systems (2020)
42. Gardner, K., Zbarsky, S., Doroudi, S., Harchol-Balter, M., Hyytiä, E., Scheller-Wolf, A.: Queueing with redundant requests: exact analysis. Queueing Syst. **83**(3), 227–259 (2016)
43. Garnett, O., Mandelbaum, A.: An introduction to skills-based routing and its operational complexities. Teaching Note, Technion, Haifa, Israel (2000)
44. Garnett, O., Mandelbaum, A., Reiman, M.: Designing a call center with impatient customers. Manuf. Serv. Oper. Manag. **4**(3), 208–227 (2002)
45. Ghamami, S., Ward, A.R.: Dynamic scheduling of a two-server parallel server system with complete resource pooling and reneging in heavy traffic: Asymptotic optimality of a two-threshold policy. Math. Oper. Res. **38**(4), 761–824 (2013)
46. Gurvich, I., Luedtke, J., Tezcan, T.: Staffing call centers with uncertain demand forecasts: a chance-constrained optimization approach. Manag. Sci. **56**(7), 1093–1115 (2010)
47. Gurvich, I., Van Meghem, J.: Collaboration and multitasking in networks: prioritization and achievable capacity. Manag. Sci. **64**(5), 2390–2406 (2018)
48. Gurvich, I., Ward, A.R.: On the dynamic control of matching queues. Stochas. Syst. **4**(2), 479–523 (2014)
49. Gurvich, I., Whitt, W.: Scheduling flexible servers with convex delay costs in many-server service systems. Manuf. Serv. Oper. Manag. **11**(2), 237–253 (2008)
50. Gurvich, I., Whitt, W.: Queue-and-idleness-ratio controls in many-server service systems. Math. Oper. Res. **34**(2), 363–396 (2009)
51. Gurvich, I., Whitt, W.: Service-level differentiation in many-server service systems via queue-ratio routing. Oper. Res. **58**(2), 316–328 (2010)
52. Halfin, S., Whitt, W.: Heavy-traffic limits for queues with many exponential servers. Oper. Res. **29**(3), 567–588 (1981)

53. Harrison, J.M.: Heavy traffic analysis of a system with parallel servers: asymptotic optimality of discrete-review policies. Ann. Appl. Probab. **8**(3), 822–848 (1998)
54. Harrison, J.M., López, M.J.: Heavy traffic resource pooling in parallel-server systems. Queueing Syst. **33**(4), 339–368 (1999)
55. Harrison, J.M., Zeevi, A.: Dynamic scheduling of a multiclass queue in the Halfin–Whitt heavy traffic regime. Oper. Res. **52**(2), 243–257 (2004)
56. Hu, Y., Chan, C.W., Dong, J.: Optimal scheduling of proactive service with customer deterioration and improvement. Manage. Sci. (forthcoming) (2020)
57. Huang, J., Carmeli, B., Mandelbaum, A.: Control of patient flow in emergency departments, or multi-class queues with deadlines and feedback. Oper. Res. **63**(4), 892–908 (2015)
58. Iglehart, D.L., Whitt, W.: Multiple channel queues in heavy traffic. II: sequences, networks, and batches. Adv. Appl. Probab. **2**(2), 355–369 (1970)
59. Jaeker, J.A.B., Tucker, A.L.: Past the point of speeding up: the negative effects of workload saturation on efficiency and patient severity. Manag. Sci. **63**(4), 1042–1062 (2017)
60. Kc, D.S., Terwiesch, C.: An econometric analysis of patient flows in the cardiac intensive care unit. Manuf. Serv. Oper. Manag. **14**(1), 50–65 (2012)
61. Kc, D.S., Terwiesch, C.: Impact of workload on service time and patient safety: an econometric analysis of hospital operations. Manag. Sci. **55**(9), 1486–1498 (2009)
62. Krishnasamy, S., Sen, R., Johari, R., Shakkottai, S.: Learning unknown service rates in queues: a multi-armed bandit approach. Oper. Res. Articles in Advance (2020)
63. Lu, Y., Xie, Q., Kliot, G., Geller, A., Larus, J.R., Greenberg, A.: Join-idle-queue: a novel load balancing algorithm for dynamically scalable web services. Perform. Eval. **68**(11), 1056–1071 (2011)
64. Luo, J., Zhang, J.: Staffing and control of instant messaging contact centers. Oper. Res. **61**(2), 328–343 (2013)
65. Mandelbaum, A., Momcilovic, P.: Personalized queues: the customer view, via a fluid model of serving least-patient first. Queueing Syst. **87**, 1–31 (2017)
66. Mandelbaum, A., Stolyar, A.L.: Scheduling flexible servers with convex delay costs: heavy-traffic optimality of the generalized cμ-rule. Oper. Res. **52**(6), 836–855 (2004)
67. van Mieghem, J.A.: Dynamic scheduling with convex delay costs: the generalized $c\mu$ rule. Ann. Appl. Probab. **5**(3), 809–833 (1995)
68. Mitzenmacher, M.: The power of two choices in randomized load balancing. IEEE Trans. Parallel Distrib. Syst. **12**(10), 1094–1104 (2001)
69. Örmeci, E.L., Güneş, E.D., Kunduzcu, D.: A modeling framework for control of preventive services. Manuf. Serv. Oper. Manag. **18**(2), 227–244 (2015)
70. Perry, O., Whitt, W.: Responding to unexpected overloads in large-scale service systems. Manage Sci. **55**(8), 1353–1367 (2009)
71. Perry, O., Whitt, W.: A fluid approximation for service systems responding to unexpected overloads. Oper. Res. **59**(5), 1159–1170 (2011)
72. Puha, A.L., Ward, A.R.: Tutorial paper: scheduling an overloaded multiclass many-server queue with impatient customers. In: Tutorials in Operations Research (2019)
73. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley-Interscience, Hoboken (2005)
74. Reed, J.E., Ward, A.R.: Approximating the GI/GI/1+GI queue with a nonlinear drift diffusion: hazard rate scaling in heavy traffic. Math. Oper. Res. **33**(3), 606–644 (2008)
75. Reiman, M.I.: Some diffusion approximations with state space collapse. In: Baccelli, F., Fayolle, G. (eds.) Modelling and Performance Evaluation Methodology. Lecture Notes in Control and Information Sciences, pp. 207–240. Springer, Berlin (1984)
76. Selen, J., Adan, I.J., Kulkarni, V.G., van Leeuwaarden, J.S.: The snowball effect of customer slowdown in critical many-server systems. Stochas. Models **32**(3), 366–391 (2016)
77. Shi, P., Chou, M.C., Dai, J., Ding, D., Sim, J.: Models and insights for hospital inpatient operations: time-dependent ED boarding time. Manag. Sci. **62**(1), 1–28 (2016)
78. Shi, P., Helm, J., Deglise-Hawkinson, J., Pan, J.: Timing it right: balancing inpatient congestion versus readmission risk at discharge. Oper. Res. (forthcoming) (2020)
79. Song, H., Tucker, A., Graue, R., Moravick, S., Yang, J.: Capacity pooling in hospitals: the hidden consequences of off-service placement. Manag. Sci. **66**(9), 3825–3842 (2020)
80. Srikant, R., Ying, L.: Communication Networks: An Optimization, Control, and Stochastic Networks Perspective. Cambridge University Press, Cambridge (2013)

81. Stolyar, A.L.: MaxWeight scheduling in a generalized switch: state space collapse and workload minimization in heavy traffic. Ann. Appl. Probab. **14**(1), 1–53 (2004)
82. Sun, Z., Argon, N., Ziya, S.: Patient triage and prioritization under austere conditions. Manag. Sci. **64**(10), 4471–4489 (2018)
83. Tassiulas, L., Ephremides, A.: Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. In: 29th IEEE Conference on Decision and Control, IEEE, pp. 2130–2132 (1990)
84. Tezcan, T., Dai, J.G.: Dynamic control of N-systems with many servers: asymptotic optimality of a static priority policy in heavy traffic. Oper. Res. **58**(1), 94–110 (2010)
85. Ward, A.R., Armony, M.: Blind fair routing in large-scale service systems with heterogeneous customers and servers. Oper. Res. **61**(1), 228–243 (2013)
86. Ward, A.R., Glynn, P.W.: A diffusion approximation for a Markovian queue with reneging. Queueing Syst. **43**, 103–128 (2003)
87. Whitt, W.: How multiserver queues scale with growing congestion-dependent demand. Oper. Res. **51**(4), 531–542 (2003)
88. Wierman, A., Harchol-Balter, M.: Classifying scheduling policies with respect to unfairness in an M/GI/1. In: Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp. 238–249 (2003)
89. Wierman, A., Nuyens, M.: Scheduling despite inexact job-size informaion. ACM SIGMETRICS Perform. Eval. Rev. **36**, 25–36 (2008)
90. Woei, L., Kumar, P.: Optimal control of a queueing system with two heterogeneous servers. IEEE Trans. Autom. Control **29**(8), 696–703 (1984)
91. Xu, K., Chan, C.: Using future information to reduce waiting times in the emergency department via diversion. Manuf. Serv. Oper. Manag. **18**(3), 314–331 (2016)
92. Yom-Tov, G.B., Mandelbaum, A.: Erlang-r: a time-varying queue with reentrant customers, in support of healthcare staffing. Manuf. Serv. Oper. Manag. **16**(2), 283–299 (2014)
93. Zhan, D., Ward, A.R.: Staffing, routing, and payment to trade off speed and quality in large service systems. Oper. Res. **67**(6), 1738–1751 (2019)