# Value of Sparse Structures in Dynamic Reusable Resource Allocation with Waiting

Jing Dong
Graduate School of Business, Columbia University
jing.dong@gsb.columbia.edu

Yue Hu
Graduate School of Business, Stanford University
yuehu@stanford.edu

Shixin Wang
CUHK Business School, Chinese University of Hong Kong
shixinwang@cuhk.edu.hk

We study the dynamic resource allocation problem in online service platforms featuring reusable resources, waiting space, and heterogeneous demands and resources. The service provider aims to balance the trade-off between maximizing revenue and minimizing waiting times across various demand types. For this problem, we propose a comprehensive framework to minimize the long-run average revenue loss and waiting cost by simultaneously designing the 1) flexibility structure, 2) admission control, and 3) scheduling policy. Our proposed flexibility structures are tailored to systems with varying levels of workload intensity, and the number of arcs in the network is linear in the number of resource and demand types. In these sparse networks, we show that simple static priority rules and threshold-based admission controls are asymptotically optimal in the many-server regimes. Furthermore, our proposed algorithm for designing system flexibility, along with the scheduling and admission control policies, is both straightforward to interpret and easy to implement. Numerical experiments demonstrate the effectiveness of our approach, particularly for smaller systems, in non-asymptotic environments.

*Key words*: Dynamic Resource Allocation, Reusable Resource, Sparse Structure, Admission Control, Scheduling Policy, Asymptotic Analysis

## 1. Introduction

We study dynamic resource allocation systems with heterogeneous demands and supplies, where the central planner aims to balance the revenue collected by satisfying the demands and the waiting cost generated from delaying the service. This problem is essential for online service platforms, including cloud computing services (e.g., Amazon Web Services and Microsoft Azure), integrated service platforms (e.g., Handy and TaskRabbit), and equipment rental platforms (e.g., Peerby and Rent the Runway). In these service systems, different types of demands have heterogeneous

matching values and waiting costs. At the same time, there are different types of resources that may be geographically dispersed. Additionally, each resource type is often (endogenously) tailored for specific demand types. For instance, in cloud computing services, different demands have different service requirements and priorities. While a GPU-intensive server might be perfect for AI model training, it is not necessarily optimal for web hosting or other tasks. Making every resource a jack-of-all-trades is not only costly in terms of infrastructure investment but can also lead to operational inefficiencies. Given the variety in demands and resources, it is vital to have a clear strategy for designing the flexibility structure, i.e., the demands that a resource can accommodate, and scheduling protocol when facing resource constraints. Motivated by the aforementioned features of service systems, our work explores how to simultaneously design the flexibility structure between demands and resources, and the scheduling and admission policies to minimize the total revenue loss and waiting cost in a reusable resource system.

To better study the dynamic resource allocation systems that arise in service applications, we hope to bridge the dynamic resource allocation literature with the queueing literature. In particular, in the classic dynamic resource allocation literature with reusable resources, there is typically no waiting room, i.e., demands that cannot be matched upon arrival are lost. Therefore, the objective is to maximize the matching reward, i.e., revenue, only. In the classic queueing literature, there is usually an infinite waiting room, and the focus is primarily on minimizing the waiting/holding cost. In this work, we consider a setting with an optimized finite waiting room together with a resource allocation policy, which leads to a non-trivial tradeoff between revenue loss and waiting cost. The optimal waiting room size and scheduling policy depend on the demand-supply mismatch. We also aim to find a sparse flexibility design that is near-optimal compared to the fully flexible system where all the resources are compatible with all types of demands. In a nutshell, our research questions are as follows:

1. How should we design the resource allocation (scheduling) and admission control policy to balance the revenue loss and waiting cost across different demand types?

2. Can we design a sparse structure that has similar performance compared with a fully flexible system?

Our performance measure is the optimality gap, which is defined as the gap between the long-run average total cost – revenue loss plus waiting cost – under our proposed scheduling and admission control policy and flexibility design and and that under the optimal scheduling and admission control policy in a fully flexible system. To gain analytical insights, we conduct the many-server asymptotic mode of analysis where we send both the demand rates and the number of servers to increase without bounds. We consider three different asymptotic regimes: 1) underloaded regime where there are abundant resources, 2) overloaded regime where the resource can only serve a

fraction of the demand, and 3) critically loaded regime where the demands and resources are well-balanced, which is the most cost-effective regime to operate in if capacity costs are concerned.

We first study the optimal admission control in a one-demand-class one-server-pool system. For an underloaded system, we show that any nonnegative waiting space achieves an $o(1)$ optimality gap; for an overloaded system, any $\mathcal{O}(1)$ waiting space archives an $\mathcal{O}(1)$ optimality gap; for a critically-loaded system, a waiting space in the order of $\sqrt{n}$ can achieve an $o(\sqrt{n})$ optimality gap. Subsequently, we study the scheduling and admission control problem in the V-model where there are two demand classes and a single server pool. For an underloaded system, any non-idling scheduling policy with infinite waiting space achieves an $o(1)$ optimality gap; for an overloaded system, a policy that prioritizes the high-reward demands and blocks low-reward demands when all servers are busy achieves an $\mathcal{O}(1)$ optimality gap; for a critically loaded system, a policy that prioritizes the high-waiting-cost demands and blocks the low-reward demands when the number of waiting jobs exceeds a predetermined waiting space threshold, which is in the order of $\sqrt{n}$, achieves an $o(\sqrt{n})$ optimality gap. These policies achieve the same order of optimality gap no matter whether the system adopts a preemptive or non-preemptive priority rule. Lastly, the insights gleaned from these simple models guide our approach to flexibility design in general systems. Specifically, we aim to configure the system so that only the lowest-revenue demand class incurs revenue loss (if any) and only the lowest-waiting-cost demand class incurs waiting cost (if any).

For the general system with multiple demand classes and resource types, we design sparse structures together with the appropriate scheduling and admission control policies that are near optimal. For an underloaded system, our solution generates a sparse structure with at most $I + J - 1$ arcs and achieves an optimality gap of $o(1)$; for an overloaded system, our solution generates a sparse structure with at most $I + 2J - 2$ arcs and achieves an optimality gap of $\mathcal{O}(1)$; for a critically-loaded system, our solution generates a sparse structure with at most $I + 2J - 2$ arcs and achieves an optimality gap of $o(\sqrt{n})$. In all these sparse structures, we show that simple static priority scheduling combined with threshold-based admission control leads to near-optimal performance.

Above all, our contributions can be summarized as follows:

1. Our model incorporates heterogeneous rewards and waiting costs across different demand classes, and allows the decision-maker to minimize the sum of revenue loss and waiting cost, which is an essential objective in many service systems in practice.

2. We propose a simple algorithm to construct sparse flexibility structures in underloaded, overloaded, and critically loaded systems that can achieve near-optimal performance, compared to a fully flexible system with optimal admission and scheduling (which is an exact optimum of the problem). Our proposed sparse structure not only has a relatively small number of arcs but also offers additional practical advantages, as it requires each individual server to handle at most two

demand types. In particular, in underloaded systems, each server is dedicated to only one demand class. In overloaded and critically loaded systems, we identify a common "basic" demand class within the system. For each server, we only require it to serve a tailored dedicated demand class and this basic class. This flexibility design is particularly attractive in scenarios where it is challenging to develop fully flexible servers. The basic class, which receives a lower priority, is associated with either the lowest revenue or the lowest waiting cost, and can effectively serve as a low-priority backup option to enhance the overall system efficiency.

3. For each traffic intensity regime, under the sparse structures, we further show that a static priority scheduling together with a threshold-based admission control policy is asymptotically optimal. The policies we propose are easy to implement in practice as they do not require keeping track of the detailed system state to determine the scheduling priority.

The rest of the paper is organized as follows. In Section 2, we discuss related literature. We present our model and problem formulation in Section 3. Then, in Section 4, we analyze the admission control and scheduling policy in simple systems with one or two demand classes and a single server pool. Finally, in Section 5, we present our main results on the flexibility design, admission control, and scheduling policy in general systems. Supplementary discussions and proofs are relegated to the appendix.

**Notation** In our asymptotic analysis, for two functions $f_1(n)$ and $f_2(n)$, we write $f_1(n) = \mathcal{O}(f_2(n))$ if there exist positive constants $M$ and $n_0$ such that $|f_1(n)| \leq M|f_2(n)|, \forall\, n \geq n_0$; $f_1(n) = o(f_2(n))$ if for every positive constant $\epsilon$, there exists a constant $n_0$ such that $|f_1(n)| \leq \epsilon f_2(n), \forall\, n \geq n_0$; and $f_1(n) = \Theta(f_2(n))$ if there are constants $\kappa_1, \kappa_2$ and $n_0$ such that $\kappa_1 f_2(n) \leq f_1(n) \leq \kappa_2 f_2(n),\ \forall\, n \geq n_0$. In addition, we let $\Rightarrow$ denote convergence in distribution, and let $D$ denote the function space of right-continuous real-valued functions defined on the positive halfline with left limits, endowed with the Skorohod $J_1$ topology.

## 2. Literature Review

Our research is mainly related to three streams of literature: dynamic resource allocation with reusable resources, queueing scheduling, and process flexibility design.

### 2.1. Dynamic Resource Allocation

Our work is related to dynamic resource allocation problems with reusable resources. Levi and Radovanović (2010) study the admission control policy in a single-resource system, and propose an LP-based class selection rule that is near optimal. Chen et al. (2017) study the single-resource system with advance reservation, and propose the so-called "$\epsilon$-perturbation class selection rule" that is near-optimal. Lei and Jasin (2020) study the dynamic pricing problem in a general system

with non-stationary customer arrivals, advance reservation, and deterministic usage time. For more general networks with reusable resources, Gong et al. (2022), Feng et al. (2021, 2022), Goyal et al. (2020) study the performance of different algorithms and provide constant competitive ratios compared with the hindsight optimal resource allocation. In the aforementioned work, the demand that can not be matched upon arrival will be lost immediately. Jia et al. (2022) study the online learning and pricing problem to maximize the revenue in a system with finite homogeneous reusable resources and customers can wait in a queue if all resources are occupied. They consider a finite-horizon setting and do not consider admission control as a potential lever to control the waiting cost. Recently, Hua et al. (2024) study the ambulance dispatching policies in emergency medical services, where ambulances can be dispatched from dedicated or flexible units. Unlike our setting, they focus on a different objective of minimizing the long-run average dispatching cost, and use different control levers, specifically, without incorporating flexibility design or admission control.

In addition to reusable resource allocation, online decision-making has been widely studied in a broad range of applications (Jasin and Kumar 2012, Arlotto and Gurvich 2019, Bumpensanti and Wang 2020, Vera and Banerjee 2021, Vera et al. 2021, Wang and Wang 2022). In most cases, resources are assumed to be fixed and nonrenewable after allocation, and performance is quantified by the regret as time horizon and initial inventory scale. This stream of work develops low-regret algorithms by resolving a deterministic relaxation of the problem which is a linear program. Among these studies, Xie et al. (2022) investigates the benefits of delayed matching in online resource allocation, which is also a consideration in our study, but we explicitly incorporate the waiting cost in the objective. Compared to the nonrenewable resources setting, the literature on reusable resource allocation highlights that resource reusability adds complexity to the analysis since it necessitates meticulous tracking of resource states. This complexity is further compounded when waiting is permitted.

### 2.2. Scheduling in Queues

The performance of different routing/scheduling policies has been widely studied in queueing literature. Van Mieghem (1995), Mandelbaum and Stolyar (2004) introduce the generalized $c\mu$ rule for single-server systems. Gurvich and Whitt (2009) study the queue-and-idleness ratio for many-server systems. Helm and Waldmann (1984), Stidham (1985), Altman and Stidham (1995) study the structure of the optimal admission control policy in single-server queues. Kim and Van Oyen (1998) consider a finite-buffer M/M/1 queue with two classes of demands and prove that, even without rejection cost, the $c\mu$ rule is not optimal in this finite buffer setting. Instead, the optimal policy is more complicated and has a switching curve that depends on the buffer size. Wang et al. (2015) conduct exact analysis of a preemptive M/M/c queue with two priority classes.

In bipartite or multi-way matching queues with heterogeneous matching rewards, Nazari and Stolyar (2019), Özkan and Ward (2020), Gupta (2022), Hu and Zhou (2022), Kerimov et al. (2023a,b) study the reward maximization problem while keeping queues stable. Ünver (2010), Gurvich and Ward (2015), Buỳić and Meyn (2015) study the delay cost minimization problem. Ding et al. (2021) study the one-sided bipartite matching system and propose an index-based routing policy, which trades off the matching reward and waiting time. Aveklouris et al. (2021) study the matching policy to maximize the matching reward minus the waiting cost. Afeche et al. (2022) optimize the matching topologies that balance the matching reward and waiting cost in a Pareto efficiency sense. In contrast to the above works, we optimize the matching topologies, routing policy, and admission control simultaneously, and our objective is to minimize the sum of revenue loss and waiting cost.

### 2.3. Process Flexibility Design

The seminal work by Jordan and Graves (1995) shows the benefit of sparse structures, especially chaining structures. This work has motivated many recent works that theoretically justify the effectiveness of the long chain and other designs with limited flexibility. Bassamboo et al. (2010), Chou et al. (2010, 2011), Simchi-Levi and Wei (2012, 2015), Deng and Shen (2013), Chou et al. (2014), Wang and Zhang (2015), Chen et al. (2015), Bidkhori et al. (2016), Chen et al. (2019) discuss the effectiveness of the sparse structures when the demands have homogeneous reward. Mak and Shen (2009), Feng et al. (2017), DeValve et al. (2022), Wang et al. (2022, 2023) study the design and effectiveness of sparse structures when demands or matchings have different rewards. The objective in the aforementioned literature is to maximize the expected sales or minimize the lost sales, and there is no waiting cost in the objective. Shi et al. (2019) consider minimizing backlogging costs in a multiperiod production system. Their findings indicate that, under the max-weight policy, the expected waiting cost achieved by a sparse structure with a positive generalized chaining gap is comparable to that in a fully flexible system. Bassamboo et al. (2012), Tsitsiklis and Xu (2013, 2017) study the effectiveness of sparse structures in balancing delay cost and capacity cost in flexibility design. These works do not consider the service reward since no demand is blocked. Rutten and Mukherjee (2023) characterize the class of sparse graphs for which the mean-field approximation remains valid (as for the fully flexible structure), and the focus is on the stability of the network. In our study, the demands can have different revenue rates and waiting costs. We minimize the sum of revenue loss and waiting cost. We demonstrate that under a simple priority rule and a threshold-based admission control, the performance of a sparse structure asymptotically approaches that of a fully flexible system operating under the corresponding optimal admission control and scheduling policy.

## 3. Problem Formulation

We study a service provider handling $I$ demand classes utilizing $J$ distinct types of reusable resources. Each demand class $i \in \mathcal{I} = \{1, \ldots, I\}$ arrives according to a Poisson process at a rate of $\lambda_i$. Each resource pool $j \in \mathcal{J} = \{1, \ldots, J\}$ contains $s_j$ unit of resources, i.e., $s_j$ servers, and the service duration of each demand at each compatible server follows an exponential distribution with a mean of $1/\mu$. We denote the flexibility design of the system by $\mathcal{G}$, where $(i, j) \in \mathcal{G}$ if resource type $j \in \mathcal{J}$ can serve demand class $i \in \mathcal{I}$. For any demand class $i \in \mathcal{I}$, we denote $\Gamma_{\mathcal{G}}(i)$ as the set of resources compatible with that demand class, i.e., $\Gamma_{\mathcal{G}}(i) := \{j \in \mathcal{J} : (i, j) \in \mathcal{G}\}$. For each resource poll $j$, we denote $\Xi_{\mathcal{G}}(j)$ as the set of demand classes the resource type can serve, i.e., $\Xi_{\mathcal{G}}(j) := \{i \in \mathcal{I} : (i, j) \in \mathcal{G}\}$. Upon the arrival of a class $i$ demand, the system may take one of the following three actions: (1) assign it to a server from server pool $j \in \Gamma_{\mathcal{G}}(i)$ (preemption may or may not be allowed), (2) keep it in a queue awaiting service, or (3) reject the demand. Such actions resonate with the cloud computing application where jobs may either queue or face rejection during high traffic to manage congestion. A demand from class $i$ yields a revenue at a rate of $r_i$ when being served and incurs a waiting cost at a rate of $h_i$ when queued. Given the resource capacity and demand arrival rates, the service provider aims to optimize the trade-off between two objectives: minimizing the delays and maximizing the revenue. We hope to determine the flexibility structure $\mathcal{G}$, together with the scheduling and admission control policy $\pi$, to minimize the sum of the long-run average revenue loss and waiting cost, denoted as $\mathcal{L}(\mathcal{G}, \pi)$. Let $B_i^{\mathcal{G},\pi}(t)$ denote the number of class $i$ demand being blocked up to time $t$ and $Q_i^{\mathcal{G},\pi}(t)$ denote the number of class $i$ demand waiting in the queue at time $t$ under the flexibility structure $\mathcal{G}$ and scheduling and admission control policy $\pi$. Then,

$$\mathcal{L}(\mathcal{G}, \pi) = \limsup_{T \to \infty} \frac{1}{T} \sum_{i=1}^{I} \mathbb{E}\left[\frac{r_i}{\mu} B_i^{\mathcal{G},\pi}(T) + h_i \int_0^T Q_i^{\mathcal{G},\pi}(t) dt\right]. \tag{1}$$

As a benchmark, we denote $\mathcal{L}^* = \mathcal{L}(\mathcal{F}, \pi_{\mathcal{F}}^*)$ as the long-run average cost (revenue loss plus waiting cost) in a fully flexible system, $\mathcal{F}$, where all the resources are compatible with all demand classes, under its optimal scheduling and admission control policy, $\pi_{\mathcal{F}}^*$. We evaluate the performance of a flexibility structure $\mathcal{G}$ and a scheduling and admission control policy $\pi$ by how much its cost deviates from $\mathcal{L}^*$:

$$\mathsf{GAP}(\mathcal{G}, \pi) \triangleq \mathcal{L}(\mathcal{G}, \pi) - \mathcal{L}^*.$$

We take a many-server asymptotic mode of analysis where the demand and the number of servers increase without bounds. In particular, we construct a sequence of systems indexed by the total number of servers $n$. Let $\lambda_i(n)$ denote the arrival rate of class $i$ demand and $s_j(n)$ denote the number of servers in resource pool $j$ in the $n$th-system. Note that $\sum_{j \in \mathcal{J}} s_j(n) = n$. We assume that

$\liminf_{n\to\infty} \min_{i\in\mathcal{I}} \lambda_i(n)/n \in (0,\infty)$ and $\liminf_{n\to\infty} \min_{j\in\mathcal{J}} s_j(n)/n \in (0,\infty)$, i.e., there is a substantial amount of demand from each demand class and a substantial amount of servers of each resource type. The utilization of the $n$-th system is denoted by

$$\rho_n := \frac{\sum_{i\in\mathcal{I}} \lambda_i(n)}{n\cdot\mu}.$$

We also define

$$\rho := \lim_{n\to\infty} \rho_n,$$

assuming the limit exists. The service rates $\mu$ for each server, the revenue rate $r_i$ obtained by serving the class $i$ demand (per job per unit time), and the holding cost rate $h_i$ generated by keeping class $i$ demand waiting do not scale with $n$. Without loss of generality, we assume $\mu = 1$.

We consider three asymptotic regimes as $n\to\infty$: 1) *Underloaded regime*: $\rho < 1$. This is the regime where there are abundant resources. 2) *Overloaded regime*: $\rho > 1$. This is the regime where the resources are scarce, i.e., there are not enough resources to serve all the demands. 3) *Critically loaded*: $\rho = 1$ and $\lim_{n\to\infty} \sqrt{n}(1-\rho_n) = \beta$. This is the regime where the resource just matches the demand. Our goal is to design sparse flexibility structures $\mathcal{G}$ and easy-to-implement scheduling and admission control policies $\pi$ that achieve a small optimality gap in these asymptotic regimes.

## 4. Scheduling and Admission Control to Balance Revenue Loss and Waiting Cost

In this section, we study two relatively simple models to gain insights into how to balance the waiting cost and revenue loss. These insights lay the foundation to study the flexibility design in Section 5. We first study the optimal admission control for a single-class single-pool system in Section 4.1. We then study the optimal scheduling and admission control policy for a two-class single-pool system, i.e., the V-model, in Section 4.2.

### 4.1. Waiting Capacity for a Single Class System

In this subsection, we consider the system where there is a single demand type and a single server type. Since there is only one demand class, we omit the subscript $i$ in the revenue rate $r$, holding cost $h$, and arrival rate $\lambda(n)$. In this scenario, the service provider does not need to optimize the flexibility design and scheduling decision. However, it provides important insights into the design of the optimal admission control policy beyond this simple case.

For a given arrival process and server capacity, we optimize the admission control policy to minimize the sum of revenue loss and waiting cost. Since the system is Markovian, the optimal admission control is state-dependent. For a system indexed by $n$, we denote $X^n(t)$ as the total number of demands in the system at time $t$. In Appendix B.2, we show that the optimal admission

control is a threshold policy, i.e., whether to accept an arriving demand at time $t$ depends on whether the current system state $X^n(t)$ is smaller than a threshold. Hence, optimizing the admission control is equivalent to optimizing the waiting room capacity or total capacity in the system. Let $C^n$ denote the total capacity of the system, i.e., the total number of demands that the system can hold, including both the demands in service and in the queue. In particular, the service provider aims to determine $C^n$ that minimizes the sum of the revenue loss and waiting cost. With a little abuse of notation, we denote the long-run average cost in the system as

$$\mathcal{L}_n(C^n) = r\lambda^n \mathbb{P}(X^n(\infty; C^n) = C^n) + h\mathbb{E}[(X^n(\infty; C^n) - n)^+]$$

where $X^n(\infty; C^n)$ denotes the stationary number of demands in the system when the capacity is $C^n$. Define $\mathcal{L}_n^* = \min_{C^n} \mathcal{L}_n(C^n)$. Then, the optimality gap is

$$\mathsf{GAP}_n(C^n) = \mathcal{L}_n(C^n) - \mathcal{L}_n^*.$$

In Proposition 1, we characterize the optimality gap of some near-optimal admission control policies under different traffic intensity regimes: 1) In the *underloaded regime*, i.e., $\rho < 1$, we can show that no job needs to wait asymptotically. Thus, $C^n$ can be set as any number greater than or equal to the number of servers. 2) In the *overloaded regime*, i.e., $\rho > 1$, we can show that even if we keep no waiting room, there is a negligible amount of idleness for the resources, i.e., the resources are busy generating revenue at all times asymptotically. Thus, setting $C^n = n$ minimizes the waiting cost without sacrificing the revenue. 3) In the *critically loaded*, i.e., $\rho = 1$ and $\lim_{n\to\infty} \sqrt{n}(1-\rho_n) = \beta$, we need to carefully design the system capacity to strike a balance between the revenue loss and waiting cost. To characterize the optimal capacity design for the critically loaded regime, we define

$$\begin{aligned} g(a; r, h) :=& r\left(\Phi(\beta) + \frac{e^{-\beta^2/2}(1 - e^{-a\beta})}{\sqrt{2\pi}\beta}\right)^{-1} \left(\sqrt{2\pi} e^{a\beta + \beta^2/2}\right)^{-1} \\ &+ h\left(\Phi(\beta)\frac{e^{-\beta^2/2}(1 - e^{-a\beta})}{\sqrt{2\pi}\beta}\right)^{-1} \frac{1 - e^{-a\beta} - a\beta e^{-a\beta}}{\sqrt{2\pi}\beta^2 e^{\beta^2/2}}, \end{aligned} \tag{2}$$

where $\Phi$ is the cumulative distribution function of the standard normal distribution. As we will show in the proof of Proposition 1, the first part in $g(a; r, h)$ characterizes the revenue loss rate at the $\sqrt{n}$ scale, and the second part characterizes the holding cost rate at the $\sqrt{n}$ scale. We ignore the $\sqrt{n}$ scaling term in $g(a; r, h)$. We also write

$$a^*(r, h) = \arg\min_a g(a; r, h). \tag{3}$$

PROPOSITION 1. *Consider three asymptotic traffic intensity regimes:*

1. *When $\rho < 1$, for any $C^n \geq n$, including $C^n = \infty$, we have $\mathcal{L}_n(C^n) = o(1)$ and*

$$\mathsf{GAP}_n(C^n) = o(1).$$

2. *When $\rho > 1$, set $C^n = n + \mathcal{O}(1)$. Then, $\mathcal{L}_n(C^n) = r\lambda_n(1-\rho) + \mathcal{O}(1)$ and*

$$\mathsf{GAP}_n(C^n) = \mathcal{O}(1).$$

3. *When $\rho = 1$ and $\sqrt{n}(1-\rho_n) \to \beta$ for some finite $\beta$, set $C^n = n + a^*(r,h)\sqrt{n}$. Then, $\mathcal{L}_n(C^n) = \sqrt{n}g(a^*(r,h)) + o(\sqrt{n})$ and*

$$\mathsf{GAP}_n(C^n) = o(\sqrt{n}).$$

Proposition 1 provides a closed-form characterization of a near-optimal admission control /waiting room design. The main insight is that when the system is underloaded, almost everyone can be served immediately upon arrival. Thus, for any waiting capacity that is greater than or equal to 0, there is almost no waiting cost and no revenue loss. When the system is overloaded, the servers are busy almost all the time. In this case, by setting the waiting capacity to 0, we minimize the waiting cost without sacrificing any revenue. When the system is critically loaded, the optimal waiting capacity is of order $\sqrt{n}$. By setting the waiting capacity as $a^*(r,h)\sqrt{n}$, we strike a delicate balance between the waiting cost and the revenue loss.

### 4.2. Scheduling and Admission Control for the V Model

In this subsection, we study the scheduling and admission control policy for a V-system where two demand classes share a single pool of resources. This model provides important insights into the routing and admission control policy when facing heterogeneous revenue rates and waiting costs.

We assume that $\limsup_{n\to\infty} \lambda_i(n)/n < 1$ for $i = 1, 2$. Otherwise, it may be optimal to serve only one class. Without loss of generality, we assume $r_1 > r_2$. We first consider the preemptive case, where a high-priority job can interrupt the service of a low-priority job. We then show that the same near-optimal policy structure also applies to the non-preemptive case where a job that has already started service cannot be interrupted. Note that in cloud computing preemption is allowed. Our policy contains both a scheduling/routing part and an admission control part. The scheduling part determines which demand class to serve next and the admission control part determines whether an arriving demand should be blocked.

In Proposition 2, we characterize the optimal scheduling and admission control policy in different traffic intensity regimes: 1) In the *underloaded regime*, i.e., $\rho < 1$, under any non-idling policy, no job needs to wait asymptotically. 2) In the *overloaded regime*, i.e., $\rho > 1$, we prioritize the demand with a higher revenue rate and block demand from the lower revenue rate class when all servers are occupied. In this case, we can show that the servers are busy at all times (i.e., we maximize

the resource utilization), but the system incurs almost no waiting cost asymptotically. 3) In the *critically loaded regime*, i.e., $\rho = 1$ and $\lim_{n\to\infty} \sqrt{n}(1-\rho_n) = \beta$, we prioritize the class with a higher waiting cost but block demand of the class with a lower revenue rate when the total queue reaches a certain $O(\sqrt{n})$ threshold. In this case, all the waiting cost is incurred by the class with a lower waiting cost, and all the revenue loss is incurred by the class with a lower revenue rate asymptotically.

PROPOSITION 2 **(Preemptive V-System)**. *We consider three asymptotic regimes depending on the aggregated traffic intensity:*

1. *When $\rho < 1$, consider any policy $\pi$ that does not allow idling (i.e., it can apply arbitrary priority between the two classes of demands) and does not block any demands. Then,*

$$\mathsf{GAP}_n(\pi) = o(1).$$

2. *When $\rho > 1$, consider the policy $\pi$ under which the service provider gives strict priority to class 1 demand and blocks class 2 arrivals whenever all servers are occupied. Then,*

$$\mathsf{GAP}_n(\pi) = \mathcal{O}(1).$$

3. *When $\rho = 1$ and $\sqrt{n}(1-\rho_n) \to \beta$ for some finite $\beta$, we further consider two cases:*

(a) *If $h_1 \geq h_2$, consider the policy $\pi$ under which the service provider always prioritizes class 1 demands and blocks class 2 arrivals when the total number of jobs waiting in the queue reaches or exceeds $a^*(r_2, h_2)\sqrt{n}$; recall that $a^*(r_2, h_2) = \arg\min g(a; r_2, h_2)$. Then,*

$$\mathsf{GAP}_n(\pi) = o(\sqrt{n}).$$

(b) *If $h_1 < h_2$, we consider the policy $\pi$ under which the service provider always prioritizes class 2 demands but blocks class 2 arrivals when the total number of jobs waiting in the queue reaches or exceeds $a^*(r_2, h_1)\sqrt{n}$. Then,*

$$\mathsf{GAP}_n(\pi) = o(\sqrt{n}).$$

We next consider a non-preemptive system where a job in service can not be interrupted. We show that the scheduling and admission control policy defined in Proposition 2 leads to the same optimality gap in the non-preemptive system.

PROPOSITION 3 **(Nonpreemptive V-System)**. *For the non-preemptive system, consider the scheduling and admission control policy $\pi$ defined in Proposition 2.*

1. *When $\rho < 1$, $\mathsf{GAP}_n(\pi) = o(1)$.*
2. *When $\rho > 1$, $\mathsf{GAP}_n(\pi) = \mathcal{O}(1)$.*

3. *When* $\rho = 1$ *and* $\lim_{n\to\infty} \sqrt{n}(1-\rho_n) = \beta$, $\mathsf{GAP}_n(\pi) = o(\sqrt{n})$.

Propositions 2 and 3 characterize simple near-optimal scheduling and admission control policies in different traffic intensity regimes. Note that the policies do not differ for the preemptive versus non-preemption systems. When the system is underloaded, any non-idling policy works well. When the system is overloaded, we should prioritize the class with a larger revenue rate, and no waiting is allowed for the low-revenue class. When the system is critically loaded, we should prioritize the class with a higher cost of waiting and block incoming demands from the low-revenue class when the number of jobs waiting in the queue reaches $a^*(\min\{r_1, r_2\}, \min\{h_1, h_2\})\sqrt{n}$. In all cases, in the limit, only demand from the low-revenue class is lost (if any), and only demand from the low-waiting-cost class needs to wait (if any). These insights provide us with some guidelines for the flexibility design of general networks. In particular, we should design the network $\mathcal{G}$ such that the high-revenue classes are underloaded. Then, only the lowest-revenue class encounters revenue loss (if any) asymptotically. In addition, the scheduling priority should be designed such that only the lowest-waiting-cost class encounters waiting costs (if any) asymptotically.

## 5. Flexibility Design for General Systems

In this section, we study the joint network flexibility design and admission control and scheduling policy in a general system with class-dependent arrival rates $\lambda_1(n), \ldots, \lambda_I(n)$, revenue rates $r_1, \ldots, r_I$, and holding costs $h_1, \ldots, h_I$. The capacity levels are $s_1(n), \ldots, s_J(n)$ at the $J$ types of resources respectively. We propose sparse structures, with $\mathcal{O}(I+J)$ arcs, and easy-to-implement scheduling and admission control policies that achieve near-optimal performance. Similar to our analysis before, how to design the flexibility structure depends on the overall traffic intensity of the system.

### 5.1. Underloaded Systems

In this subsection, we consider the limiting regime where $\rho < 1$, i.e., the system is overall underloaded. In this regime, we can design the network to ensure that the subsystem associated with each demand classes is "underloaded," so that no revenue is lost and no waiting cost is incurred asymptotically. We apply the following algorithm to generate a flexibility structure.

**Algorithm 1** Generating Flexibility Structure in an Underloaded System

**Input:** Sort the demand classes $\{i \mid i = 1, \ldots, I\}$ in descending order of holding costs, i.e., $h_1 \geq h_2 \geq \cdots \geq h_I$.

Initialize $i = j = 1$, $\omega = s_1(n) - \lambda_1(n)$, $\delta = \frac{\sum_{j=1}^{J} s_j(n) - \sum_{i=1}^{I} \lambda_i(n)}{I}$, $\mathcal{G} = \emptyset$.

**while** $i < I$ or $j < J$ **do**

  $\mathcal{G} = \mathcal{G} \cup (i, j)$

  **if** $\omega \geq i \cdot \delta$ **then**

    $i = i + 1$, $\omega = \omega - \lambda_i(n)$

  **else**

    $j = j + 1$, $\omega = \omega + s_j(n)$.

  **end if**

**end while**

$\mathcal{G} = \mathcal{G} \cup \{(I, J)\}$

The basic idea behind Algorithm 1 is to reserve a positive slack capacity $\delta$ for each demand class. After the flexibility structure is designed based on Algorithm 1, we adopt the following admission control and scheduling policy.

Definition 1 (Policy $\pi$ for an underloaded system). No demand is blocked, i.e., all demands are admitted. For server pool $j$ with $\Xi_{\mathcal{G}}(j) = \{i_j, i_j + 1, \ldots, i_{j+1}\}$, it exclusively reserves/dedicates $i_j \cdot \delta + \sum_{i'=1}^{i_j} \lambda_{i'}(n) - \sum_{j'=1}^{j-1} s_j(n)$ servers for class $i_j$, $\sum_{j'=1}^{j} s_j(n) - \sum_{i'=1}^{i_{j+1}-1} \lambda_{i'}(n) - (i_{j+1} - 1)\delta$ servers for class $i_{j+1}$, and $\delta + \lambda_i(n)$ servers for demand class $i$, $i_j < i < i_{j+1}$.

The policy in Definition 1 implicitly divides the total service capacity into $I$ groups, each dedicated to one of the $I$ demand classes with a slack capacity of $\delta$. Since $\delta = \Theta(n)$, each demand class, together with the capacity reserved for it, constitutes an underloaded system. Then, according to part 1 of Proposition 1 (the case with $\rho < 1$), the revenue loss and waiting cost for each demand class is $o(1)$.

Theorem 1. *The structure generated by Algorithm 1 has at most $I + J - 1$ arcs, and achieves a total cost of $\mathcal{O}(ne^{-n})$ when operating under the admission control and scheduling policy in Definition 1.*

It should be noted that Definition 1 delineates just one of the many asymptotically optimal scheduling policies. For example, an alternative policy is for each server pool to prioritize the eligible demand classes, i.e., $i \in \Xi_{\mathcal{G}}(j)$ for server pool $j$, in descending order of their waiting costs. Here, we focus on the policy described in Definition 1 due to its analytical simplicity. Moreover, there are

also variants of Algorithm 1 that lead to similar asymptotic performance. For instance, instead of having a class-independent slack capacity $\delta$, we can set the slack capacity to be class-dependent, e.g., proportional to the corresponding arrival rate.

**5.1.1. Numerical Experiments for Underloaded Systems** We conduct numerical experiments to compare the finite system performance under our proposed policy and a properly constructed cost lower bound. In particular, we consider a sequence of systems with 3 demand classes and 2 server pools. The arrival rates are given by $(\lambda_1(n), \lambda_2(n), \lambda_3(n)) = (n/4, 3n/8, n/4)$ and the server-pool capacities are given by $(s_1(n), s_2(n)) = (n/4, 3n/4)$. The traffic intensity satisfies $\rho(n) = \rho = 0.875$, for all $n \geq 1$. In addition, we assume that the revenue rates and holding costs are given by $(h_1, h_2, h_3) = (3, 2, 1)$ and $(r_1, r_2, r_3) = (3, 2, 1)$ respectively.

For this sequence of stochastic systems, we first follow Algorithm 1 to obtain the flexibility design. All demands are admitted. Moreover, based on the scheduling policy in Definition 1, server pool 1 reserves all capacity for class-1 demand. server pool 2 is capable of serving all three demand classes and allocates service capacities of $(n/24, \lambda_2(n) + n/24, \lambda_3(n) + n/24)$ across these classes. Equivalently, the queueing network is divided into three independent underloaded systems with arrival rate equal to $\lambda_i(n)$ and server-pool capacity equal to $\lambda_i(n) + n/24$, $i = 1, 2, 3$.

We simulate the long-run average costs under policy $\pi$ for systems with sizes $n = 10, 50, 100, 150, 200, 1000$, as shown in Table 1. Each simulation experiment is conducted for 10,000 units of time, yielding relatively small standard errors. For example, based on the batch means method with 10 batches, the standard errors for the estimated costs range from 0.072 to 1.461 across all system sizes. Moreover, note that for underloaded systems, a straightforward lower bound for the optimal cost is zero. As shown in Table 1, the long-run average costs (as well as their gaps towards the cost lower bounds) under policy $\pi$ decrease significantly, from 18.211 when $n = 10$ to 0.002 when $n = 1000$, illustrating an $o(1)$ optimality gap which is consistent with Theorem 1.

As mentioned in the discussion after Theorem 1, there are alternative scheduling policies that achieve the same optimality gap. In Table 1, we further study the performance of one such alternative policy where for each server pool, instead of reserving a fixed capacity for each eligible demand class, we prioritize the eligible demand classes in descending order of their holding costs, either preemptively or non-preemptively. We observe from Table 1 that these alternative policies also achieve an $o(1)$ gap. Additionally, the alternative policies result in lower costs than policy $\pi$, especially for smaller systems, due to their increased flexibility in cross-serving, which leads to more resource pooling. Furthermore, the preemptive policy achieves lower costs than the non-preemptive policy, especially for smaller systems.

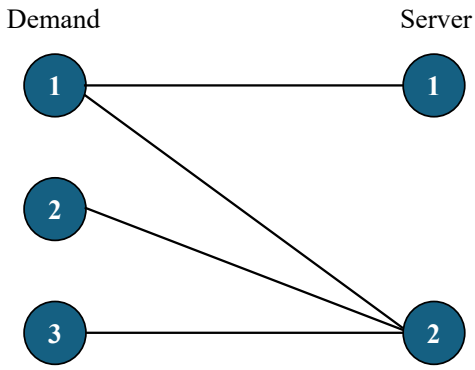


**Figure 1** **Flexibility Design for Underloaded System**

| $n$ | Lower bound | Policy $\pi$ | | Alternative scheduling (preemptive) | | Alternative scheduling (non-preemptive) | |
|---|---|---|---|---|---|---|---|
| | Cost | Cost | Gap | Cost | Gap | Cost | Gap |
| 10 | 0 | 17.863 | 17.863 | 9.531 | 9.531 | 15.706 | 15.706 |
| 50 | 0 | 17.853 | 17.853 | 2.779 | 2.779 | 5.024 | 5.024 |
| 100 | 0 | 11.699 | 11.699 | 1.237 | 1.237 | 2.290 | 2.290 |
| 150 | 0 | 10.268 | 10.268 | 0.620 | 0.620 | 1.169 | 1.169 |
| 200 | 0 | 7.374 | 7.374 | 0.359 | 0.359 | 0.685 | 0.685 |
| 1000 | 0 | 0.499 | 0.499 | 0.000 | 0.000 | 0.000 | 0.000 |

**Table 1** **Cost Comparison for Underloaded Systems**

### 5.2. Overloaded Systems

In this subsection, we consider the case where the limiting $\rho > 1$, i.e., the system is overall overloaded. We sort the demand classes $\{i \mid i = 1, \ldots, I\}$ in descending order of *revenue rate*. Define $i^* = \min\{i : \sum_{k=1}^{i} \lambda_i(n) \geq \sum_{j=1}^{J} s_j(n)\}$. Note that if $i^* < I$, there is not enough capacity to serve demand class $i$ for $i > i^*$. Without loss of generality, we assume

$$\lim_{n\to\infty} \frac{\sum_{k=1}^{i^*-1} \lambda_i(n)}{\sum_{j=1}^{J} s_j(n)} < 1 \text{ and } \lim_{n\to\infty} \frac{\sum_{k=1}^{i^*} \lambda_i(n)}{\sum_{j=1}^{J} s_j(n)} > 1.$$

Otherwise, we can consider the (sub)system consisting of classes $1, 2, \ldots, i^* - 1$ or classes $1, 2, \ldots, i^*$ as a critically loaded system and apply the construction in Section 5.3. We first apply Algorithm 2 below to generate a flexibility structure, and then adopt the admission control and scheduling policy in Definition 2.

**Algorithm 2** Generating Flexibility Structure in an Overloaded System

**Input:** Sort the demand classes $\{i \mid i = 1, \ldots, I\}$ in descending order of revenue rate, i.e, $r_1 \geq r_2 \geq \cdots \geq r_I$. Define $i^* = \min\{i : \sum_{k=1}^{i} \lambda_i(n) \geq \sum_{j=1}^{J} s_j(n)\}$.

Initialize $i = j = 1$, $\omega = s_1(n) - \lambda_1(n)$, $\delta = \frac{\sum_{j=1}^{J} s_j(n) - \sum_{i=1}^{i^*-1} \lambda_i(n)}{i^*-1}$, $\mathcal{G} = \{(i,j)\}$.

**while** $i < i^*$ or $j < J$ **do**

  **if** $\omega \geq i \cdot \delta$ **then**

    $i = i+1$, $\omega = \omega - \lambda_i(n)$

  **else**

    $j = j+1$, $\omega = \omega + s_j(n)$

  **end if**

  $\mathcal{G} = \mathcal{G} \cup \{(i,j)\}$

**end while**

$\mathcal{G} = \mathcal{G} \cup \{(i^*,1)\} \cup \{(i^*,2)\} \cup \cdots \cup \{(i^*,J)\}$

Definition 2 (Policy $\pi$ for an overloaded system). All the demands in class $i$, $i \leq i^* - 1$, are admitted. All the demands in class $i$, $i > i^*$ are rejected. For server pool $j$ with $\Xi_{\mathcal{G}}(j) = \{i_j, i_j+1, \ldots, i_{j+1}, i^*\}$, it reserves $i_j \cdot \delta + \sum_{i'=1}^{i_j} \lambda_{i'}(n) - \sum_{j'=1}^{j-1} s_j(n)$ servers for class $i_j$, $\sum_{j'=1}^{j} s_j(n) - \sum_{i'=1}^{i_{j+1}-1} \lambda_{i'}(n) - (i_{j+1}-1)\delta$ servers for class $i_{j+1}$ (Note that $i_{j+1} < i^*$), and $\delta + \lambda_i(n)$ servers for demand class $i$, $i_j < i < i_{j+1}$. In this way, we divide the system into $(i^*-1)$ $M/M/\tilde{s}_i(n)$ queues, where $\tilde{s}_i(n) = \lambda_i(n) + \delta$. For the class $i^*$ demand, upon its arrival, we route it to one of the $M/M/\tilde{s}_i(n)$ queues with equal probability, i.e., probability $1/(i^*-1)$. In addition, the class $i^*$ demand is blocked upon arrival if all servers in the sub-system that it is routed to are occupied. In subsystem $M/M/\tilde{s}_i(n)$, class $i$ jobs receive a higher priority, in a preemptive or non-preemptive manner, than class $i^*$ jobs.

Under the policy in Definition 2, we divide the system into $(i^*-1)$ subsystems, each serving two classes of demands. In particular, the $i$-th system is an $M/M/\tilde{s}_i(n)$ queue, serving demands in classes $i$ and $i^*$, and the class $i$ demand is prioritized over the class $i^*$ demand. For class $i$, the prioritized class, we have $\tilde{s}_i(n) - \lambda_i(n) = \delta$, where $\delta = \Theta(n)$, i.e., class $i$ alone is underloaded. When combining the two demand classes $i$ and $i^*$ routed to the $i$-th system together, since $\lambda_{i^*}(n)/(i^*-1) > \delta$, $\tilde{s}_i(n) - (\lambda_i(n) + \lambda_{i^*}(n)/(i^*-1)) = \tilde{\delta} < 0$, where $|\tilde{\delta}| = \Theta(n)$, the $i$-th system is overall overloaded.

We next show that with the small amount of flexibility following Algorithm 2, the admission control and scheduling policy defined in Definition 2 achieves an optimality gap of $\mathcal{O}(1)$.

Theorem 2. *The structure generated by Algorithm 2 has at most $I + 2J - 2$ arcs, and achieves an optimality gap of $\mathcal{O}(1)$ when operating under the admission control and scheduling policy in Definition 2.*

Theorem 2 demonstrates that a sparse structure with at most $I + 2J - 2$ arcs together with a static priority rule and a threshold-based admission control achieves near-optimal performance.

Other flexibility designs, admission control, and scheduling policies may achieve the same performance guarantee, similar to the case for underloaded systems. In the main paper, we focus on Algorithm 2 and Definition 2 due to their analytical elegance. In Appendix A.1, we complement the main paper by providing an alternative solution for flexibility design, admission control, and scheduling policy for overloaded systems. Numerical experiments are also conducted for this alternative solution. It is worth noting that, compared to the policy characterized in Algorithm 2 and Definition 2, the alternative policy has a more complex and less elegant definition. However, it may result in slightly better performance for certain small-scale stochastic systems by reducing the amount of server idling.

**5.2.1. Numerical Experiments for Overloaded Systems** We conduct numerical experiments to compare the system performance under our proposed policy and a properly constructed cost lower bound. In particular, we consider a sequence of systems with 3 demand classes and 2 server pools. The arrival rates are given by $(\lambda_1(n), \lambda_2(n), \lambda_3(n)) = (n/2, 3n/4, n/2)$, and the service-pool capacities are given by $(s_1(n), s_2(n)) = (n/2, n/2)$. The traffic intensity satisfies $\rho(n) = \rho = 1.75$, for all $n \geq 1$. In addition, we assume that the revenue rates and holding costs are given by $(h_1, h_2, h_3) = (3, 2, 1)$ and $(r_1, r_2, r_3) = (3, 2, 1)$ respectively.

For this sequence of stochastic systems, we first follow Algorithm 2 to obtain the flexibility design, as shown in Figure 2. Note that $i^* = 2$ in this example, so that all demands from class 3 are dropped. In addition, note that the demands combined from classes 1 and 2 are strictly larger than the total service capacity in the system. In this case, policy $\pi$ in Definition 2 regulates that all service capacity is reserved for prioritizing class-1 demands, and all class-1 demands are admitted. As for class-2 demand, its admission is contingent upon the existence of idle servers beyond those occupied by class-1 demands. The priority for class-1 demands can be implemented in either a preemptive or a non-preemptive manner. In the preemptive system, a class-1 demand can displace a class-2 demand currently in service, prompting the displaced class-2 demand to be returned to the queue for later service. Conversely, in a non-preemptive system, ongoing services cannot be interrupted.

In Table 2 below, we present the simulated costs of the system under policy $\pi$, considering both preemptive and non-preemptive services, for different values of $n$. To facilitate comparison, we also construct a lower bound for the optimal cost. Specifically, the cost lower bound for the $n$th system is calculated as $n/2 \times 1 + n/4 \times 2 = n$, where the first term in the summation accounts for the revenue loss by dropping all class-3 demand. The second term represents the remaining demand

exceeding the total service capacity, and excess demand (i.e., $n/4$) is assumed to be penalized at the lower revenue rate between class-1 and class-2 demands (i.e., $\min\{r_1, r_2\} = 2$). By comparing the simulated costs under the proposed policies and the cost lower bounds, we observe that the gaps stay bounded as $n$ increases, reflecting an optimality gap of $\mathcal{O}(1)$ established in Theorem 2.

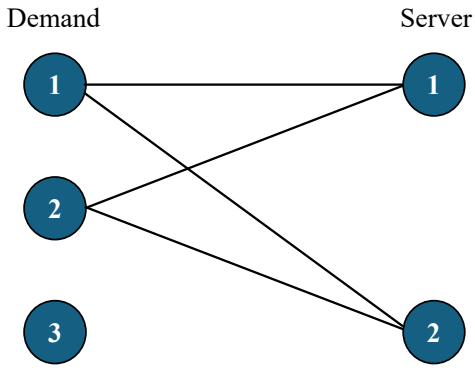


**Figure 2** **Flexibility Design for Overloaded Systems**

| $n$ | Lower bound | Policy $\pi$ (preemptive) | | Policy $\pi$ (non-preemptive) | |
|---|---|---|---|---|---|
| | Cost | Cost | Gap | Cost | Gap |
| 10 | 10 | 13.465 | 3.465 | 13.950 | 3.950 |
| 50 | 50 | 54.916 | 4.916 | 55.307 | 5.307 |
| 100 | 100 | 105.279 | 5.279 | 105.642 | 5.642 |
| 150 | 150 | 155.896 | 5.896 | 156.249 | 6.249 |
| 200 | 200 | 205.555 | 5.555 | 205.900 | 5.900 |
| 500 | 500 | 505.084 | 5.084 | 505.420 | 5.420 |
| 1000 | 1000 | 1006.274 | 6.274 | 1006.612 | 6.612 |

**Table 2** **Cost Comparison for Overloaded Systems**

### 5.3. Critically Loaded Systems

In this subsection, we consider the case where the limiting $\rho = 1$ and $\lim_{n\to\infty} \sqrt{n}(1-\rho_n) = \beta$, i.e., the system is overall critically loaded. The analysis for the critically loaded case is similar to the overloaded case. We sort the demand classes $\{i \mid i = 1, \ldots, I\}$ in descending order of *revenue rate*, and then move the demand class with the lowest cost of waiting to the last, i.e., $r_1 \geq r_2 \geq \cdots \geq r_{I-1}$ and $I = \arg\min_i h_i$. We aim to construct the flexibility structure, admission control, and scheduling policy such that the revenue loss is only incurred by the lowest-revenue class (i.e., class $I-1$ or $I$) and the waiting cost is only incurred by the lowest-waiting cost class (i.e., class $I$). Formally, we construct the flexibility structure in Algorithm 3 and the admission control and scheduling policy in Definition 3.

**Algorithm 3** Generating Flexibility Structure in a Critically Loaded System

**Input:** Sort the demand classes $\{i \mid i = 1, \ldots, I\}$ in descending order of revenue rate and then move the demand class with the lowest cost of waiting to the last, i.e., $r_1 \geq r_2 \geq \cdots \geq r_{I-1}$ and $h_I = \min_{\{1 \leq i \leq I\}} h_i$.

Initialize $i = j = 1$, $\omega = s_1(n) - \lambda_1(n)$, $\delta = \frac{\sum_{j=1}^{J} s_j(n) - \sum_{i=1}^{I-1} \lambda_i(n)}{I-1}$, $\mathcal{G} = \{(i,j)\}$.

**while** $i < I$ or $j < J$ **do**

  **if** $\omega \geq i \cdot \delta$ **then**

    $i = i + 1$, $\omega = \omega - \lambda_i(n)$

  **else**

    $j = j + 1$, $\omega = \omega + s_j(n)$.

  **end if**

  $\mathcal{G} = \mathcal{G} \cup \{(i,j)\}$

**end while**

$\mathcal{G} = \mathcal{G} \cup \{(I,1)\} \cup \{(I,2)\} \cup \cdots \cup \{(I,J)\}$

Definition 3 (Policy $\pi$ for a critically loaded system). Let $i^*$ denote the class with the lowest revenue rate. Note that $i^* = I - 1$ or $I$. For server pool $j$ with $\Xi_{\mathcal{G}}(j) = \{i_j, i_j + 1, \ldots, i_{j+1}, I\}$, it reserves $i_j \cdot \delta + \sum_{i'=1}^{i_j} \lambda_{i'}(n) - \sum_{j'=1}^{j-1} s_j(n)$ servers for class $i_j$, $\sum_{j'=1}^{j} s_j(n) - \sum_{i'=1}^{i_{j+1}-1} \lambda_{i'}(n) - (i_{j+1} - 1)\delta$ servers for class $i_{j+1}$ (Note that $i_{j+1} < I$), and $\delta + \lambda_i(n)$ servers for demand class $i$, $i_j < i < i_{j+1}$. In this way, we divide the system into $(I-1)$ $M/M/\tilde{s}_i(n)$ queues, where $\tilde{s}_i(n) = \lambda_i(n) + \delta$. For the class $I$ demand, it can be served by all servers but receives a lower priority in a preemptive manner than the class the servers are reserved for. We follow a work-conserving scheduling. In particular, the class $I$ queue is positive only when there are no idle servers. Class-$i$, $1 \leq i \leq I-1$, arrivals are blocked when the number of class-$i$ jobs in the system is larger than or equal to $\tilde{s}_i(n)$, i.e., the number of servers reserved for class $i$. In addition, when the total number of jobs in the system exceeds (larger than or equal to) $n + a^*\left(\min_{1 \leq i \leq I}\{r_i\}, \min_{1 \leq i \leq I}\{h_i\}\right) \cdot \sqrt{n}$, for $a^*(\cdot,\cdot)$ defined in (3), the system stops admitting demand from class $i^*$.

In the next theorem, we show that with the small level of flexibility defined in Algorithm 3, the simple admission control and scheduling policy defined in Definition 3 achieves an optimality gap of $o(\sqrt{n})$.

Theorem 3. *The structure generated by Algorithm 3 has at most $I + 2J - 2$ arcs, and achieves an optimality gap bounded from above by $o(\sqrt{n})$ when operating under the admission control and scheduling policy defined in Definition 3.*

Theorem 3 demonstrates that a sparse structure with at most $I+2J-2$ arcs together with a static priority rule and a threshold-based admission control achieves near-optimal performance.

Similar to underloaded and overloaded systems, there may exist other flexibility designs, admission control, and scheduling policies that achieve the same performance guarantee for critically loaded systems. In the main paper, we choose to focus on Algorithm 3 and Definition 3 due to their analytical elegance. In Appendix A.2, we describe an alternative solution for flexibility design, admission control, and scheduling policy for critically loaded systems. Numerical experiments, in Sections 5.3.1 and Appendix A.2, demonstrate that the original and alternative policies achieve very similar performance, at least for the problem instances examined.

**5.3.1. Numerical Experiments for Critically Loaded Systems** We next numerically compare the system performance under our proposed policy and a properly constructed lower bound of the optimal cost. We consider a sequence of systems with 3 demand classes and 2 server pools. The arrival rates are given by $(\lambda_1(n),\lambda_2(n),\lambda_3(n)) = ((n-\beta\sqrt{n})/3,(n-\beta\sqrt{n})/3,(n-\beta\sqrt{n})/3)$, and the server-pool capacities are given by $(s_1(n),s_2(n)) = (n/2,n/2)$. The traffic intensity satisfies $\lim_{n\to\infty}\sqrt{n}(1-\rho_n)=\beta$. We conduct two sets of numerical experiments for $\beta=0.75$ and $\beta=0.05$. In addition, we assume that the revenue rates and holding costs are given by $(h_1,h_2,h_3)=(3,2,1)$ and $(r_1,r_2,r_3)=(3,2,1)$ respectively.

For the sequence of stochastic systems with each $\beta$ specification, we first follow Algorithm 3 to obtain the flexibility design, as shown in Figure 3. Note that $i^*=3$ and $\delta = n/6+\beta\sqrt{n}/3$. According to policy $\pi$ characterized in Definition 3, server pool 1 reserves all its capacity, i.e., $n/2$ servers, to demand class 1, and server pool 2 reserves all its capacity, i.e., $n/2$ servers, to demand class 2. For class-1 (alternatively, class-2) demands, they receive priority within their reserved capacity and are blocked from entering the system when the total number of class-1 (alternatively, class-2) demands in the system is larger than or equal to $n/2$. For class-3 demands, they can be served by both server pools 1 and 2, but receive a lower priority. Moreover, class-3 demand is blocked when the total number of jobs across all demand classes in the system is larger than or equal to $n+a^*(\min r_i,\min h_i)\cdot\sqrt{n}$, where $a^*=0.989$ when $\beta=0.75$ and $a^*=0.651$ when $\beta=0.05$.

Tables 3 and 4 present the simulated costs under the proposed policy $\pi$ for both preemptive and non-preemptive scenarios. To facilitate comparison, we also construct a lower bound for the optimal cost. This lower bound is derived as follows: For each system, we first calculate the total arrival rate across all demand classes, $\sum_{i=1}^{I}\lambda_i$, and the total service capacity across all server pools, $\sum_{j=1}^{J}s_j$. We then consider a simplified system with a single demand class and a single server pool, matching the total arrival rate and service capacity of the original system. In this simplified system, we set the revenue rate to be $\min_{1\le i\le I} r_i$ and the holding cost to be $\min_{1\le i\le I} h_i$. By optimizing

the admission control (i.e., the optimal buffer size) for this single-class single-pool system through enumeration in simulation, we can determine its optimal objective value. This value serves as a lower bound for the optimal cost of the original, more complex system.

As shown in Tables 3 and 4, the performance gap between our proposed policy $\pi$ and the lower bound, after being scaled down by $\sqrt{n}$, decreases as $n$ increases. We remark that in Theorem 3, while we only formally prove the $o(\sqrt{n})$ optimality gap when preemption is allowed, we conjecture that this optimality gap also holds for non-preemptive systems as shown numerically in Tables 3 and 4. Unfortunately, due to the limited analytical tractability of critically loaded systems, we cannot provide a formal proof for the non-preemptive case.

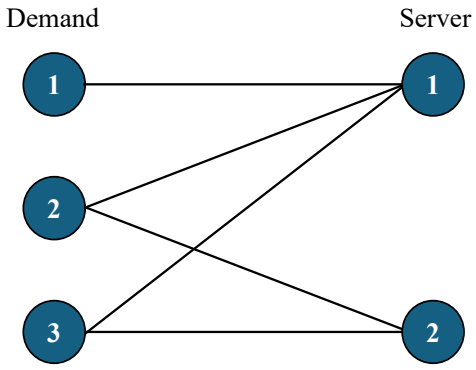


**Figure 3** **Flexibility Design for Critically Loaded Systems**

| $n$ | Lower bound | Policy $\pi$ (preemptive) | | Policy $\pi$ (non-preemptive) | |
|---|---|---|---|---|---|
| | Cost | Cost | Gap/$\sqrt{n}$ | Cost | Gap/$\sqrt{n}$ |
| 10 | 0.635 | 1.229 | 0.188 | 2.370 | 0.549 |
| 50 | 1.404 | 1.979 | 0.081 | 4.200 | 0.395 |
| 100 | 2.104 | 2.536 | 0.043 | 4.969 | 0.287 |
| 150 | 2.537 | 3.051 | 0.042 | 5.382 | 0.232 |
| 200 | 2.800 | 3.296 | 0.035 | 5.684 | 0.204 |
| 500 | 4.603 | 5.032 | 0.019 | 7.425 | 0.126 |
| 1000 | 6.682 | 7.116 | 0.014 | 9.481 | 0.088 |

**Table 3** **Cost Comparison for Critically Loaded Systems ($\beta = 0.75$)**

### 5.4. Discussion

Overall, our analysis demonstrates that as long as there is a small amount of flexibility, i.e., a flexibility network with at most $I + 2J - 2$ arcs, static (i.e., state-independent) priority rules together with threshold-type admission controls can achieve near-optimal performance, i.e., achieve an optimality gap of $o(1)$ for underloaded systems, $\mathcal{O}(1)$ for overloaded systems, and $o(\sqrt{n})$ for critically loaded systems.

| $n$ | Lower bound | Policy $\pi$ (preemptive) | | Policy $\pi$ (non-preemptive) | |
|---|---|---|---|---|---|
| | Cost | Cost | Gap/$\sqrt{n}$ | Cost | Gap/$\sqrt{n}$ |
| 10 | 1.766 | 3.531 | 0.558 | 5.558 | 1.199 |
| 50 | 4.066 | 5.330 | 0.179 | 9.781 | 0.808 |
| 100 | 5.464 | 6.690 | 0.123 | 11.666 | 0.620 |
| 150 | 6.885 | 7.828 | 0.077 | 12.405 | 0.451 |
| 200 | 8.246 | 8.830 | 0.041 | 13.398 | 0.364 |
| 500 | 12.662 | 13.373 | 0.032 | 17.661 | 0.224 |
| 1000 | 18.278 | 18.868 | 0.019 | 22.757 | 0.142 |

**Table 4 Cost Comparison for Critically Loaded Systems ($\beta = 0.05$)**

Since we focus on optimizing the long-run average cost criteria, any fixed costs associated with the flexibility design (e.g., building the infrastructure and cross-training staff) diminish with time. Therefore, we do not explicitly penalize complex flexibility structures in the objective function. Nonetheless, our findings highlight that a simple and sparse flexibility design can achieve almost the same performance as a fully flexible system.

Furthermore, the simplicity of our proposed flexibility design is evident not only from the relatively small number of arcs in the network but also from the perspective of individual servers. Importantly, in conjunction with the proposed scheduling policies, *each server needs to handle at most two demand classes*, even though the server pool to which it belongs may be connected to more than two demand types. More specifically, in underloaded systems, each server is dedicated to only one demand class. In overloaded and critically loaded systems, each server needs to be capable of serving only two demand classes: one is its reserved demand class, which receives high priority, and the other is the basic demand class with either the lowest revenue rate or the lowest cost of waiting. This second low-priority basic demand class is the same across all servers, and it effectively balances the load in the system in a low-cost way. By requiring each server to handle at most two demand types, our flexibility design further reduces the burden of staff training and system management.

## 6. Non-Markovian System

In the previous sections, we developed an asymptotically optimal flexibility design together with the admission control and scheduling policy in Markovian systems. In this section, we present some heuristics for non-Markovian systems where the inter-arrival times and service durations follow general distributions. These heuristics achieve effective performance numerically, and have theoretical performance guarantees under specific conditions.

### 6.1. Underloaded and Overloaded Systems

For non-Markovian underloaded and overloaded systems, where $\rho < 1$ and $\rho > 1$ respectively, the heuristics for flexibility design, admission control, and scheduling policy are identical to those

developed for the Markovian systems. Specifically, for non-Markovian underloaded systems, we adhere to Algorithm 1 and Definition 1, while for overloaded systems, we follow Algorithm 2 and Definition 2. Note that for these policies to be well-defined, we only need information on the average arrival and service rates, not on their distributions. Moreover, throughout Section 6.1, we assume non-preemptive system dynamics for the proposed heuristics.

We conduct numerical experiments for both underloaded and overloaded systems with general service time distributions, as presented in Tables 5 and 6, respectively. We examine two types of service time distributions characterized by coefficients of variation smaller and larger than that of the exponential distribution. Specifically, the first set of experiments assumes an Erlang distribution with a mean of 1 and a squared coefficient of variation of 0.5. In contrast, the second set of experiments assumes a Lognormal distribution with a mean of 1 and a squared coefficient of variation of 1.5. The remaining model parameters are the same as those assumed in Sections 5.1 and 5.2 for underloaded and overloaded systems respectively. Tables 5 and 6 present the simulated costs and compare them to the cost lower bounds constructed using the methods outlined in Sections 5.1 and 5.2. The numerical results indicate that the heuristic policies yield diminishing gaps compared to the lower bounds for underloaded systems, and constant gaps for overloaded systems. Although our theoretical results do not explicitly cover these cases due to the lack of analytical tractability, it is plausible that the findings hold under broader conditions, such as general service time distributions.

| $n$ | Lower bound | Erlang Service Time | | Lognormal Service Time | |
|---|---|---|---|---|---|
| | Cost | Cost | Gap | Cost | Gap |
| 10 | 0 | 13.561 | 13.561 | 23.912 | 23.912 |
| 50 | 0 | 13.767 | 13.767 | 19.608 | 19.608 |
| 100 | 0 | 9.314 | 9.314 | 12.352 | 12.352 |
| 150 | 0 | 8.595 | 8.595 | 10.406 | 10.406 |
| 200 | 0 | 6.542 | 6.542 | 8.125 | 8.125 |
| 1000 | 0 | 0.438 | 0.438 | 0.506 | 0.506 |

**Table 5 Underloaded Systems with General Service Time Distributions**

| $n$ | Lower bound | Erlang Service Time | | Lognormal Service Time | |
|---|---|---|---|---|---|
| | Cost | Cost | Gap | Cost | Gap |
| 10 | 10 | 14.143 | 4.143 | 14.188 | 4.188 |
| 50 | 50 | 55.303 | 5.303 | 54.634 | 4.634 |
| 100 | 100 | 106.577 | 6.577 | 105.237 | 5.237 |
| 150 | 150 | 156.139 | 6.139 | 154.329 | 4.329 |
| 200 | 200 | 208.183 | 8.183 | 206.440 | 6.440 |
| 1000 | 1000 | 1009.365 | 9.365 | 1006.766 | 6.766 |

**Table 6 Overloaded Systems with General Service Time Distributions**

### 6.2. Critically Loaded Systems

For non-Markovian critically loaded systems with $\rho = 1$, we first utilize the framework developed in Whitt (2004, 2005) to derive approximations for the steady-state average queue length and blocking probability in $G/GI/n/C$ models. In particular, we consider a sequence of $G/GI/n/C^n$ systems indexed by the number of servers $n$, with $C^n = n + a\sqrt{n}$ for some $a \geq 0$, and $\sqrt{n}(1-\rho_n) \to \beta$ for some $\beta \in \mathbb{R}$. Let $A := \{A(t) : t \geq 0\}$ denote the arrival process, with rate $\lambda_n$ for the $n$th system. Let $c_a^2$ be a non-negative scaling constant, and $B$ denote the standard Brownian motion (zero drift, unit diffusion coefficient). We assume that the sequence of arrival processes satisfies the following functional central limit theorem:

$$\tilde{A}_n \Rightarrow \sqrt{c_a^2} B \quad \text{in } (D, J_1),$$

where

$$\tilde{A}_n(t) := \frac{1}{\sqrt{n}} \left( A(\lambda_n t) - \lambda_n t \right), \quad t \geq 0.$$

We next elaborate on the approximations for the limiting steady-state average queue length and blocking probability. Let

$$v := \frac{(c_a^2 + w c_s^2 + 1 - w)}{2},$$

where $c_s^2$ is the squared coefficient of variation of the service time distribution. In addition, the weight $w \in [0,1]$ is a tuning parameter whose default value is set to 1 in Whitt (2004). Let $H^c$ denote the complementary cumulative distribution function of the service times. We set $z$ according to Equation (3.16) in Section 10.3 of Whitt (2002):

$$z := 1 + \left(c_a^2 - 1\right) \int_0^\infty H^c(u)^2 du.$$

Given $\nu$ and $z$, we also define

$$p := z/\nu.$$

Next, we define the function $\alpha : \mathbb{R}_+ \to \mathbb{R}$ as

$$\alpha(a) := \begin{cases} \left[1 + \beta \Phi(\beta) / \left(\phi(\beta)(1 - e^{-a\beta})\right)\right]^{-1}, & \beta \neq 0 \\ \left(1 + a^{-1}\sqrt{\pi/2}\right)^{-1}, & \beta = 0. \end{cases}$$

Then, the approximation of the asymptotic scaled steady-state average queue length is given by

$$\tilde{Q}^{approx}(a) := \begin{cases} \int_0^a \alpha(\beta/\sqrt{z}, pa/\sqrt{z}) \frac{e^{-x\beta/\nu} - e^{-a\beta/\nu}}{1 - e^{-a\beta/\nu}} dx, & \beta \neq 0 \\ \int_0^a \alpha(0, ap/\sqrt{z}) \frac{a-x}{a} dx, & \beta = 0. \end{cases} \tag{4}$$

In particular,

$$\tilde{Q}^{approx}(a) \approx \lim_{n \to \infty} \mathbb{E}[(X^n(\infty; C^n) - n)^+] / \sqrt{n}.$$

Moreover, the approximation of the asymptotic scaled steady-state blocking probability is

$$\tilde{\mathbb{P}}(BL)^{approx}(a) := \begin{cases} \nu\alpha(\beta/\sqrt{z}, pa/\sqrt{z})p\beta/\sqrt{z}\frac{e^{-a\beta/\nu}}{(1-e^{-a\beta/\nu})}, & \beta \neq 0 \\ \nu\alpha_0(pa/\sqrt{z})/a, & \beta = 0. \end{cases} \tag{5}$$

In particular,

$$\tilde{\mathbb{P}}(BL)^{approx}(a) \approx \mathbb{P}(X^n(\infty; C^n) = C^n)\sqrt{n}.$$

We remark that the approximations in (4) and (5) are the bona fide limiting performance measures for systems under specific conditions, including $G/H_2^*/n/C^n$ systems, where the service time distribution (denoted by $H_2^*$) is a mixture of an exponential distribution with some probability $p$ and a point mass at 0 with probability $1-p$ (Whitt 2004, 2005).

To derive an effective heuristic policy for critically loaded systems, we first follow Algorithm 3 to design the flexibility structure of the network, which does not require any information of the inter-arrival time and service time distributions. For the admission control and scheduling policy, we apply the policy introduced in Definition 3, but use a different blocking threshold from $a^*\left(\min_{1\le i\le I}\{r_i\}, \min_{1\le i\le I}\{h_i\}\right)$. Specifically, we write

$$g^{approx}(a; r, h) := r\,\tilde{\mathbb{P}}(BL)^{approx}(a) + h\,\tilde{Q}^{approx}(a).$$

Given $\tilde{g}^{approx}(a; r, h)$, we substitute $a^*\left(\min_{1\le i\le I}\{r_i\}, \min_{1\le i\le I}\{h_i\}\right)$ in Definition 3 by

$$a^{*,approx}\left(\min_{1\le i\le I}\{r_i\}, \min_{1\le i\le I}\{h_i\}\right) := \arg\min_{a\ge 0} g^{approx}\left(a; \min_{1\le i\le I}\{r_i\}, \min_{1\le i\le I}\{h_i\}\right).$$

We conduct numerical experiments to evaluate the performance of the proposed heuristic policy for critically loaded systems. We use the same Erlang and Lognormal service time distributions as those used in Section 6.1, with squared coefficients of variation of 0.5 and 1.5, which are lower and higher than that of the exponential distribution, respectively. The other model parameters match those in Section 5.3 for $\beta = 0.05$. Tables 7 and 8 compare, for each system, the simulated cost under the heuristic policy and the gap compared to the cost lower bound developed in Section 5.3. In addition, Tables 7 and 8 list the two key performance measures, i.e., the expected steady-state queue length and blocking probability, obtained by both the approximations in (4) and (5) and simulations. Note that the approximations in (4) and (5) apply to systems with a single class of customers and a single pool of servers. In this context, to examine the effectiveness of these approximations for the $n$th system, we consider a corresponding $M/GI/n/C^n$ queue with arrival rate of $\sum_{i\in\mathcal{I}}\lambda_i(n)$, and set $C^n = n + a^{*,approx}\left(\min_{1\le i\le I}\{r_i\}, \min_{1\le i\le I}\{h_i\}\right)\sqrt{n}$. We then compare the simulated and approximated values of the expected steady-state queue length and blocking probability for this $M/GI/n/C^n$ system in Tables 7 and 8. We observe that the approximated

values closely match the simulated ones. More importantly, the gap between the simulated costs and lower bounds, scaled by $\sqrt{n}$, decreases as the system size $n$ increases. These trends are consistent with the theoretical results for Markovian systems. Due to the lack of analytical tractability, our theoretical analysis does not easily extend to critically loaded systems with general inter-arrival or service time distributions. Nevertheless, as demonstrated by the heuristic development and numerical experiments, it is likely that the theoretical guarantees hold under broader conditions.

| $n$ | Lower bound | Heuristic policy | | Queue length | | Blocking prob. | |
|---|---|---|---|---|---|---|---|
| | Cost | Cost | Gap/$\sqrt{n}$ | Approx. | Simu. | Approx. | Simu. |
| 10 | 1.617 | 5.367 | 1.186 | 0.540 | 0.810 | 0.172 | 0.114 |
| 50 | 3.958 | 9.405 | 0.770 | 1.207 | 1.398 | 0.077 | 0.059 |
| 100 | 5.833 | 10.640 | 0.481 | 1.706 | 1.720 | 0.054 | 0.044 |
| 150 | 6.965 | 12.300 | 0.436 | 2.090 | 2.199 | 0.044 | 0.036 |
| 200 | 8.148 | 13.396 | 0.371 | 2.413 | 2.609 | 0.038 | 0.030 |
| 1000 | 17.978 | 26.396 | 0.266 | 5.396 | 5.661 | 0.017 | 0.017 |

**Table 7 Critically Loaded Systems with Erlang Service Time Distributions**

**($\beta = 0.05, c_s^2 = 0.5, a^{*,approx} = 0.785$)**

| $n$ | Lower bound | Heuristic policy | | Queue length | | Blocking prob. | |
|---|---|---|---|---|---|---|---|
| | Cost | Cost | Gap/$\sqrt{n}$ | Approx. | Simu. | Approx. | Simu. |
| 10 | 1.806 | 6.102 | 1.358 | 0.226 | 0.463 | 0.141 | 0.133 |
| 50 | 4.132 | 8.007 | 0.548 | 0.505 | 0.775 | 0.063 | 0.073 |
| 100 | 5.674 | 10.464 | 0.479 | 0.714 | 1.132 | 0.045 | 0.051 |
| 150 | 7.194 | 11.895 | 0.384 | 0.874 | 1.257 | 0.036 | 0.043 |
| 200 | 7.758 | 12.361 | 0.325 | 1.009 | 1.327 | 0.032 | 0.034 |
| 1000 | 17.769 | 24.228 | 0.204 | 2.257 | 3.235 | 0.014 | 0.019 |

**Table 8 Critically Loaded Systems with Lognormal Service Time Distributions**

**($\beta = 0.05, c_s^2 = 1.5, a^{*,approx} = 0.563$)**

## 7. Conclusion

We study dynamic resource allocation systems with heterogeneous demands and supplies, where the centralized planner aims to balance the revenue collected by satisfying the demands and the waiting cost incurred from delaying the services. To achieve analytical tractability, we consider the many-server asymptotic regime, where the demand arrival rates and numbers of servers grow without bounds. We analyze the underloaded, critically loaded, and overloaded systems, where the limiting traffic intensity is strictly smaller than, equal to, and larger than 1, respectively. Leveraging asymptotic analysis, we propose easy-to-implement solutions that jointly specify the flexibility design, admission control, and scheduling policy. We show that the proposed solutions

achieve near-optimal performance, with a respective optimality gap of order $o(1)$ for underloaded systems, $o(\sqrt{n})$ for critically loaded systems, and $\mathcal{O}(1)$ for overloaded systems.

Our findings underscore the value of maintaining a sparse network flexibility structure. In particular, our proposed flexibility design algorithm generates sparse structures with at most $I+J-1$ arcs for underloaded systems, and $I+2J-2$ arcs for critically-loaded and overloaded systems. This flexibility design is practical in reducing infrastructure investment. Furthermore, its effectiveness underscores that, with a properly designed admission control and scheduling policy, a small degree of flexibility in the network can yield similar performance as a fully flexible system.

Our theoretical framework is inherently comprehensive, as it 1) incorporates heterogeneity in demands, 2) balances both revenue loss and holding costs, and 3) simultaneously optimizes flexibility design, admission control, and scheduling policy. Despite the seemingly prohibitive nature of the problem due to its extensive action space, jointly optimizing all three aspects of system management—flexibility design, admission control, and scheduling policy—proves advantageous. This broad action space allows us to derive solutions with sparse flexibility structures, simple threshold-type admission control, and static priority rules, which are easy to interpret and implement.

We conclude by discussing several limitations of our work and identifying a few interesting future research directions. First, while our model captures customer-class-dependent arrival rates, holding costs, and revenue rates, as well as server-pool-dependent capacities, it would be valuable to generalize the framework to include class-and-pool-dependent service rates and revenue rates. Although the asymptotic analysis framework we employ can potentially be adapted for these cases, several key coupling arguments we rely on do not extend easily. Second, it would be meaningful to consider time-varying and potentially unknown arrival rates. In such scenarios, worst-case analysis based on adversarial demands, as commonly employed in resource allocation literature, or policies that are agnostic to the arrival rates may be desirable. Lastly, while our study focuses on admission control, which can block demands, we do not consider abandonment when demands are waiting in the queue. An interesting future direction is to extend the framework to incorporate abandonment behavior from the demand side. However, it is worth noting that if abandonment is anticipated, it may be more effective to block the demand initially.

## References

Afeche, P., Caldentey, R., and Gupta, V. (2022). On the optimal design of a bipartite matching queueing system. *Operations Research*, 70(1):363–401.

Altman, E. and Stidham, S. (1995). Optimality of monotonic policies for two-action markovian decision processes, with applications to control of queues with delayed information. *Queueing Systems*, 21:267–291.

Arlotto, A. and Gurvich, I. (2019). Uniformly bounded regret in the multisecretary problem. *Stochastic Systems*, 9(3):231–260.

Aveklouris, A., DeValve, L., and Ward, A. R. (2021). Matching impatient and heterogeneous demand and supply. *arXiv preprint arXiv:2102.02710*.

Bassamboo, A., Randhawa, R. S., and Mieghem, J. A. V. (2012). A little flexibility is all you need: on the asymptotic value of flexible capacity in parallel queuing systems. *Operations Research*, 60(6):1423–1435.

Bassamboo, A., Randhawa, R. S., and Van Mieghem, J. A. (2010). Optimal flexibility configurations in newsvendor networks: Going beyond chaining and pairing. *Management Science*, 56(8):1285–1303.

Bidkhori, H., Simchi-Levi, D., and Wei, Y. (2016). Analyzing process flexibility: A distribution-free approach with partial expectations. *Operations Research Letters*, 44(3):291–296.

Bumpensanti, P. and Wang, H. (2020). A re-solving heuristic with uniformly bounded loss for network revenue management. *Management Science*, 66(7):2993–3009.

Buỳić, A. and Meyn, S. (2015). Approximate optimality with bounded regret in dynamic matching models. *ACM SIGMETRICS Performance Evaluation Review*, 43(2):75–77.

Chen, X., Ma, T., Zhang, J., and Zhou, Y. (2019). Optimal design of process flexibility for general production systems. *Operations Research*, 67(2):516–531.

Chen, X., Zhang, J., and Zhou, Y. (2015). Optimal sparse designs for process flexibility via probabilistic expanders. *Operations Research*, 63(5):1159–1176.

Chen, Y., Levi, R., and Shi, C. (2017). Revenue management of reusable resources with advanced reservations. *Production and Operations Management*, 26(5):836–859.

Chou, M. C., Chua, G. A., Teo, C.-P., and Zheng, H. (2010). Design for process flexibility: Efficiency of the long chain and sparse structure. *Operations Research*, 58(1):43–58.

Chou, M. C., Chua, G. A., Teo, C.-P., and Zheng, H. (2011). Process flexibility revisited: The graph expander and its applications. *Operations Research*, 59(5):1090–1105.

Chou, M. C., Chua, G. A., and Zheng, H. (2014). On the performance of sparse process structures in partial postponement production systems. *Operations research*, 62(2):348–365.

Deng, T. and Shen, Z.-J. M. (2013). Process flexibility design in unbalanced networks. *Manufacturing & Service Operations Management*, 15(1):24–32.

DeValve, L., Pekeč, S., and Wei, Y. (2022). Approximate submodularity in network design problems. *Operations Research*.

Ding, Y., McCormick, S. T., and Nagarajan, M. (2021). A fluid model for one-sided bipartite matching queues with match-dependent rewards. *Operations Research*, 69(4):1256–1281.

Feng, W., Wang, C., and Shen, Z.-J. M. (2017). Process flexibility design in heterogeneous and unbalanced networks: A stochastic programming approach. *IISE Transactions*, 49(8):781–799.

Feng, Y., Niazadeh, R., and Saberi, A. (2021). Online assortment of reusable resources with exogenous replenishment. *Available at SSRN 3795056*.

Feng, Y., Niazadeh, R., and Saberi, A. (2022). Near-optimal bayesian online assortment of reusable resources. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 964–965.

Gong, X.-Y., Goyal, V., Iyengar, G. N., Simchi-Levi, D., Udwani, R., and Wang, S. (2022). Online assortment optimization with reusable resources. *Management Science*, 68(7):4772–4785.

Goyal, V., Iyengar, G., and Udwani, R. (2020). Asymptotically optimal competitive ratio for online allocation of reusable resources. *arXiv preprint arXiv:2002.02430*.

Gupta, V. (2022). Greedy algorithm for multiway matching with bounded regret. *Operations Research*.

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

Gurvich, I. and Whitt, W. (2009). Scheduling flexible servers with convex delay costs in many-server service systems. *Manufacturing & Service Operations Management*, 11(2):237–253.

Helm, W. E. and Waldmann, K.-H. (1984). Optimal control of arrivals to multiserver queues in a random environment. *Journal of Applied Probability*, 21(3):602–615.

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

Hua, C., Wang, T., Zhang, J., and Zhou, Z. (2024). Asymptotically optimal dispatch policies for emergency medical services. *Available at SSRN 4386239*.

Jasin, S. and Kumar, S. (2012). A re-solving heuristic with bounded revenue loss for network revenue management with customer choice. *Mathematics of Operations Research*, 37(2):313–345.

Jia, H., Shi, C., and Shen, S. (2022). Online learning and pricing for service systems with reusable resources. *Operations Research*.

Jordan, W. C. and Graves, S. C. (1995). Principles on the benefits of manufacturing process flexibility. *Management science*, 41(4):577–594.

Kerimov, S., Ashlagi, I., and Gurvich, I. (2023a). Dynamic matching: Characterizing and achieving constant regret. *Management Science*.

Kerimov, S., Ashlagi, I., and Gurvich, I. (2023b). On the optimality of greedy policies in dynamic matching. *Operations Research*.

Kim, E. and Van Oyen, M. P. (1998). Beyond the c$\mu$ rule: Dynamic scheduling of a two-class loss queue. *Mathematical Methods of Operations Research*, 48:17–36.

Lei, Y. and Jasin, S. (2020). Real-time dynamic pricing for revenue management with reusable resources, advance reservation, and deterministic service time requirements. *Operations Research*, 68(3):676–685.

Levi, R. and Radovanović, A. (2010). Provably near-optimal lp-based policies for revenue management in systems with reusable resources. *Operations Research*, 58(2):503–507.

Mak, H.-Y. and Shen, M. Z.-J. (2009). Stochastic programming approach to process flexibility design. *Flexible Services and Manufacturing Journal*, 21(3-4):75–91.

Mandelbaum, A. and Stolyar, A. L. (2004). Scheduling flexible servers with convex delay costs: Heavy-traffic optimality of the generalized c$\mu$-rule. *Operations Research*, 52(6):836–855.

Nazari, M. and Stolyar, A. L. (2019). Reward maximization in general dynamic matching systems. *Queueing Systems*, 91:143–170.

Özkan, E. and Ward, A. R. (2020). Dynamic matching for real-time ride sharing. *Stochastic Systems*, 10(1):29–70.

Puterman, M. L. (1990). Markov decision processes. *Handbooks in operations research and management science*, 2:331–434.

Rutten, D. and Mukherjee, D. (2023). Load balancing under strict compatibility constraints. *Mathematics of Operations Research*, 48(1):227–256.

Shi, C., Wei, Y., and Zhong, Y. (2019). Process flexibility for multiperiod production systems. *Operations Research*, 67(5):1300–1320.

Simchi-Levi, D. and Wei, Y. (2012). Understanding the performance of the long chain and sparse designs in process flexibility. *Operations Research*, 60(5):1125–1141.

Simchi-Levi, D. and Wei, Y. (2015). Worst-case analysis of process flexbility designs. *Operations Research*, 63(1):166–185.

Stidham, S. (1985). Optimal control of admission to a queueing system. *IEEE Transactions on Automatic Control*, 30(8):705–713.

Tsitsiklis, J. N. and Xu, K. (2013). On the power of (even a little) resource pooling. *Stochastic Systems*, 2(1):1–66.

Tsitsiklis, J. N. and Xu, K. (2017). Flexible queueing architectures. *Operations Research*, 65(5):1398–1413.

Ünver, M. U. (2010). Dynamic kidney exchange. *The Review of Economic Studies*, 77(1):372–414.

Van Mieghem, J. A. (1995). Dynamic scheduling with convex delay costs: The generalized c— mu rule. *The Annals of Applied Probability*, pages 809–833.

Vera, A. and Banerjee, S. (2021). The bayesian prophet: A low-regret framework for online decision making. *Management Science*, 67(3):1368–1391.

Vera, A., Banerjee, S., and Gurvich, I. (2021). Online allocation and pricing: Constant regret via bellman inequalities. *Operations Research*, 69(3):821–840.

Wang, J., Baron, O., and Scheller-Wolf, A. (2015). M/m/c queue with two priority classes. *Operations Research*, 63(3):733–749.

Wang, S., Wang, X., and Zhang, J. (2022). Robust optimization approach to process flexibility designs with contribution margin differentials. *Manufacturing & Service Operations Management*, 24(1):632–646.

Wang, S., Zhang, J., and Zhang, Y. (2023). The impact of profit differentials on the value of a little flexibility. *Available at SSRN 4413821*.

Wang, X. and Zhang, J. (2015). Process flexibility: A distribution-free bound on the performance of k-chain. *Operations Research*, 63(3):555–571.

Wang, Y. and Wang, H. (2022). Constant regret resolving heuristics for price-based revenue management. *Operations Research*, 70(6):3538–3557.

Whitt, W. (2002). Stochastic-process limits: an introduction to stochastic-process limits and their application to queues. *Space*, 500:391–426.

Whitt, W. (2004). A diffusion approximation for the g/gi/n/m queue. *Operations Research*, 52(6):922–941.

Whitt, W. (2005). Heavy-traffic limits for the g/h 2*/n/m queue. *Mathematics of Operations Research*, 30(1):1–27.

Xie, Y., Ma, W., and Xin, L. (2022). The benefits of delay to online decision-making. *Available at SSRN*.

## Appendix A: Alternative Flexibility Design, Admission Control, and Scheduling Policy

### A.1. Overloaded Systems

For overloaded systems, the flexibility structure design in Algorithm 2 and the admission control and scheduling policy in Definition 2 are not the only ones that can achieve an optimality gap of $\mathcal{O}(1)$. In this section, we propose an alternative flexibility structure design in Algorithm 4 and admission control and scheduling policy in Definition 4. Note that Algorithm 4 and Definition 4 are designed to be used together, just as Algorithm 2 and Definition 2 are intended to be jointly implemented.

**Algorithm 4** Generating Flexibility Structure in an Overloaded System (arc number is at most $I+J+\min\{I,J\}-2$)

**Input:** Sort the demand classes $\{i \mid i=1,\dots,I\}$ in descending order of revenue rate. Define $i^*=\arg\min_i \sum_{k=1}^{i}\lambda_i(n) \geq \sum_{j=1}^{J} s_j(n)$

Initialize $i=0$, $j=1$, $\omega=s_1(n)$, $\mathcal{G}=\emptyset$, $\delta$, where $\delta$ is an arbitrary number satisfying $0<\delta<\min\{\min_{i=1,\dots,i^*}\lambda_i(n), \min_{j=1,\dots,J} s_j(n)\}$.

**while** $i<i^*$ or $j<J$ **do**

  **if** $\omega>\delta$ or $j=J$ **then**

    $i=i+1$, $\mathcal{G}=\mathcal{G}\cup\{(i,j)\}$, $\omega=\omega-\lambda_i(n)$

  **else if** $\omega<0$ or $i=I$ **then**

    $j=j+1$, $\mathcal{G}=\mathcal{G}\cup\{(i,j)\}$, $\omega=\omega+s_j(n)$

  **else**

    $\mathcal{G}=\mathcal{G}\cup\{(i,j+1)\}\cup\{(i+1,j)\}\cup\{(i+1,j+1)\}$, $\omega=\omega+s_{j+1}(n)-\lambda_{i+1}(n)$, $i=i+1$, $j=j+1$

  **end if**

**end while**

If the system is preemptive, $\mathcal{G}=\mathcal{G}\cup_{i=i^*}^{I}\{(i,J)\}$; if the system is non-preemptive, $\mathcal{G}=\mathcal{G}\cup\{(i^*,J)\}$.

**Return** $\mathcal{G}$

Definition 4. When a server becomes available and multiple eligible demands are waiting to be served, we prioritize the demand in ascending order of their indices (i.e., descending order of their revenue rates). For demands in the same class, we adopt the first-come-first-served rule. When a demand arrives and there are multiple capable servers available, we prioritize the server in ascending order of their indices. Prioritization can be done either preemptively or non-preemptively, depending on the system dynamics. Ties can be broken arbitrarily. For any demand $i<i^*$, it is always admitted. Furthermore, the subsequent policies are implemented depending on whether preemption is allowed:

- If the system is preemptive: For demand $i<i^*$, upon its arrival, if all of its eligible servers are busy and some of them are occupied by demands of class $j$, $j\geq i^*$, then the demand with the highest index

among those with $j \geq i^*$ and occupying a server in class-$i$ demand's eligible server pools is kicked out of (i.e., removed from) the system, resulting in lost demand. Moreover, for any demand class $i$, $i \geq i^*$, it is blocked if no server from its eligible server pools is available upon its arrival. Since the system is preemptive, services can be interrupted or "reshuffled," namely, a job already in service can be moved to another capable server. Thus, at the arrival epoch of a job, the determination of whether there is an available server among all the eligible ones is made after potentially reshuffling the existing jobs. In particular, at each decision epoch $t$, following the admission, blocking, or removal decisions, we solve the following integer program for scheduling decisions to maximize the number of high-revenue jobs in service. That is, this integer program determines how many class $i$ jobs (among those admitted and not removed) are to be assigned to server pool $j$, i.e., $Z(t) := (Z_{ij}(t), 1 \leq i \leq I, 1 \leq j \leq J)$:

$$\begin{aligned}
\max_{Z(t)} \quad & \sum_{i=1}^{I}\sum_{j=1}^{J}(M-i-j)Z_{ij}(t) \\
\text{s.t} \quad & \sum_{j\in\Gamma_{\mathcal{G}}(i)} Z_{ij}(t) \leq X_i(t), \quad 1 \leq i \leq I \\
& \sum_{i\in\Xi_{\mathcal{G}}(j)} Z_{ij}(t) \leq s_j(n), \quad 1 \leq j \leq J \\
& Z_{ij}(t) \in \mathbb{Z}^+ \text{ and } Z_{ij}(t) > 0 \text{ only if } (i,j) \in \mathcal{G}, \quad 1 \leq i \leq I,\ 1 \leq j \leq J,
\end{aligned}$$

where $M > I + J$ is an arbitrarily large constant.

- If the system is non-preemptive: For any demand class $i$, $i \geq i^*$, it is blocked if no server from its eligible server pools is available upon its arrival.

To test the system performance under this alternative system design and control, we consider a sequence of systems with the same primitives as those studied in Section 5.2.1.

Note that if the system is preemptive, Algorithm 4 results in a structure that has one additional edge, $\{3, 2\}$, than the one in Figure 2. In the resulting network, server pool 1 is eligible to serve demands from classes 1 and 2, while server pool 2 is eligible to serve demands from all classes. Additionally, demands from classes 2 and 3 are admitted into the system only if an eligible server is available at the time of their arrival. Moreover, if all eligible servers are occupied when a demand from class 1 arrives, and some of these servers are occupied by class-2 or class-3 demands, a class-2 or class-3 demand (with a higher class index) is removed from the system.

If the system is non-preemptive, Algorithm 4 generates the same flexibility structure as that in Figure 2, regardless of the $\delta$ value used. (Note that for other problem instances, different values of $\delta$ may result in different flexibility designs.) In terms of the admission control and scheduling policy, demands from class 2 are admitted into the system only if an eligible server is available at the time of their arrival. All demands from class 3 are blocked and lost. No demand from class 1 is blocked and class-1 demands are prioritized for service.

Table 9 lists the simulated costs under the alternative flexibility design and policy, both preemptively and non-preemptively. The alternative policy without preemption achieves the same performance as the non-preemptive policy in Table 2. This is because the two have the same flexibility design and very similar control. The alternative policy with preemption achieves only a slightly better performance than the preemptive policy in Table 2, even for $n = 10$.

| $n$ | Lower bound | Alternative policy (preemptive) | | Alternative policy (non-preemptive) | |
|---|---|---|---|---|---|
| | Cost | Cost | Gap | Cost | Gap |
| 10 | 10 | 13.189 | 3.189 | 13.950 | 3.950 |
| 50 | 50 | 53.965 | 3.965 | 55.307 | 5.307 |
| 100 | 100 | 104.504 | 4.504 | 105.642 | 5.642 |
| 150 | 150 | 154.572 | 4.572 | 156.249 | 6.249 |
| 200 | 200 | 204.746 | 4.746 | 205.900 | 5.900 |
| 500 | 500 | 505.428 | 5.428 | 505.420 | 5.420 |
| 1000 | 1000 | 1005.828 | 5.828 | 1006.612 | 6.612 |

**Table 9 Cost Comparison for Overloaded Systems under Alternative Design and Control**

### A.2. Critically Loaded Systems

Similar to the underloaded and overloaded systems, the flexibility design and admission control and scheduling policy that achieve an $o(\sqrt{n})$ optimality gap are not unique for the critically loaded systems. In this section, we present an alternative flexibility structure generated by Algorithm 5 and an admission control and scheduling policy in Definition 5 below.

**Algorithm 5** Generating Flexibility Structure in a Critically Loaded System (Arc number is at most $I+J+\min\{I,J\}-2$.)

**Input:** Sort the demand classes $\{i \mid i=1,\dots,I\}$ in descending order of revenue rate and then move the demand class with the lowest cost of waiting to the last, i.e., $r_1 \geq r_2 \geq \cdots \geq r_{I-1}$ and $h_I = \min_{\{1\leq i\leq I\}} h_i$.

Initialize $i=0, j=1$, $\omega = s_1(n)$, $\mathcal{G}=\emptyset$, and set $\delta$ as is an arbitrary number satisfying $0<\delta<\min\{\min_{\{1\leq i\leq I\}}\lambda_i(n), \min_{\{1\leq j\leq J\}} s_j(n)\}$.

**while** $i<I$, or $j<J$ **do**

  **if** $\omega>\delta$ or $j=J$ **then**

    $i=i+1$, $\mathcal{G}=\mathcal{G}\cup\{i,j\}$, $\omega=\omega-\lambda_i(n)$

  **else if** $\omega<0$ or $i=I$ **then**

    $j=j+1$, $\mathcal{G}=\mathcal{G}\cup\{i,j\}$, $\omega=\omega+s_j(n)$.

  **else**

    $\mathcal{G}=\mathcal{G}\cup\{i,j+1\}\cup\{i+1,j\}\cup\{i+1,j+1\}$, $\omega=\omega+s_{j+1}(n)-\lambda_{i+1}(n)$, $j=j+1$, $i=i+1$

  **end if**

**end while**

Definition 5. All server pools except the last one prioritize the demands in ascending order of the indices of the demand classes. That is, the server always prioritizes the higher-revenue-rate demands connected to it. There are infinite waiting spaces for demand class $1,2,\dots,I-2$. The last server pool prioritizes the demand classes in descending order of holding costs. When the total number of demands from the last two demand classes in the system exceeds $n+a^*\left(\min_{1\leq i\leq I}\{r_i\}, \min_{1\leq i\leq I}\{h_i\}\right)\cdot\sqrt{n}$, for $a^*(\cdot,\cdot)$ defined in (3), the system stops admitting the demand class with the lowest revenue rate, which would be either class $I-1$ or class $I$.

We conduct numerical experiments for systems under the alternative flexibility design and control characterized in Algorithm 5 and Definition 5. These experiments utilize the same problem instances and cost lower bounds as detailed in Section 5.3.1. Tables 10 and 11 present the simulated costs. We observe that under $\sqrt{n}$ scaling, the gap between the simulated costs and the lower bounds decreases as the system index $n$ increases. Note, however, that these alternative solutions are not performing better than our proposed ones in Tables 3 and 4.

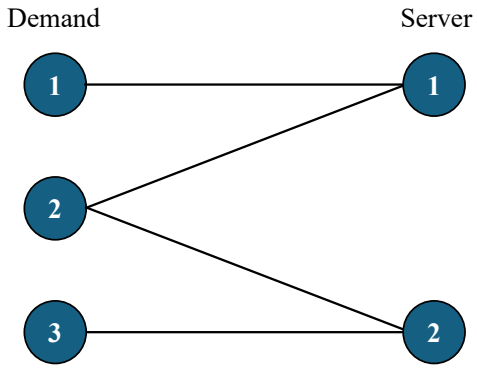


**Figure 4** **Alternative Flexibility Design for Critically Loaded Systems**

| $n$ | Lower bound | Alternative policy (preemptive) | | Alternative policy (non-preemptive) | |
|---|---|---|---|---|---|
| | Cost | Cost | Gap/$\sqrt{n}$ | Cost | Gap/$\sqrt{n}$ |
| 10 | 0.635 | 1.189 | 0.175 | 2.831 | 0.695 |
| 50 | 1.404 | 1.711 | 0.043 | 4.043 | 0.373 |
| 100 | 2.104 | 2.510 | 0.041 | 5.100 | 0.300 |
| 150 | 2.537 | 3.042 | 0.041 | 5.912 | 0.276 |
| 200 | 2.800 | 3.287 | 0.034 | 6.583 | 0.268 |
| 500 | 4.603 | 5.021 | 0.019 | 8.536 | 0.176 |
| 1000 | 6.682 | 6.997 | 0.010 | 10.192 | 0.111 |

**Table 10** **Cost Comparison for Critically Loaded Systems under Alternative Design and Control ($\beta = 0.75$)**

| $n$ | Lower bound | Alternative policy (preemptive) | | Alternative policy (non-preemptive) | |
|---|---|---|---|---|---|
| | Cost | Cost | Gap/$\sqrt{n}$ | Cost | Gap/$\sqrt{n}$ |
| 10 | 1.766 | 5.148 | 1.070 | 7.970 | 1.962 |
| 50 | 4.066 | 4.972 | 0.128 | 8.720 | 0.658 |
| 100 | 5.464 | 6.635 | 0.117 | 10.603 | 0.514 |
| 150 | 6.885 | 8.039 | 0.094 | 11.818 | 0.403 |
| 200 | 8.246 | 9.479 | 0.087 | 13.385 | 0.363 |
| 500 | 12.662 | 13.498 | 0.037 | 17.481 | 0.215 |
| 1000 | 18.278 | 18.415 | 0.004 | 22.527 | 0.134 |

**Table 11** **Cost Comparison for Critically Loaded Systems under Alternative Design and Control ($\beta = 0.05$)**

## Appendix B: Auxiliary Lemmas

### B.1. Stochastic Dominance

We first establish a stochastic dominance result that will be used in the proof of Proposition 3 and Theorem 3. Let $\{\Xi(t): t \geq 0\}$ denote a $d$-dimensional Markov chain. Define $h: \mathbb{Z}_{0+}^d \to \mathbb{Z}_{0+}$. We assume the transition of $\Xi(t)$ can be categorized into three types: *Birth type* that leads $h(\Xi(t))$ to increase by 1 and happens at rate $\lambda(\Xi(t))$; *Death type* that leads $h(\Xi(t))$ to decrease by 1 and happens at rate $\mu(\Xi(t))$; *Null type* that leads to no change in the value of $h(\Xi(t))$ and happens at rate $\gamma(X(t))$.

Let $\{Y(t): t \geq 0\}$ denote a one-dimensional birth and death process with birth rate $\tilde{\lambda}(Y(t))$ and death rate $\tilde{\mu}(Y(t))$.

LEMMA 1. *Suppose for any* $\xi \in \mathbb{Z}_{0+}^d$, $\tilde{\lambda}(h(\xi)) \geq \lambda(\xi)$ *and* $\tilde{\mu}(h(\xi)) \leq \mu(\xi)$. *Then,* $Y(\infty)$ *stochastically dominate* $h(\Xi(\infty))$

*Proof of Lemma 1.* We construct coupled processes $\{\Xi(t): t \geq 0\}$ and $\{Y(t): t \geq 0\}$ as follows. $\Xi(0)$ and $Y(0)$ are set such that $h(\Xi(0)) = Y(0)$. At time $t$, the next event happens with rate

$$\Lambda(t) = \max\{\lambda(\Xi(t)), \tilde{\lambda}(Y(t))\} + \max\{\mu(\Xi(t)), \tilde{\mu}(Y(t))\} + \gamma(X(t)).$$

At time $t$, we first sample the next event time, $\Delta$, according to an Exponential distribution with rate $\Lambda(t)$. We then sample a Uniform random variable $U$, and determine the next event for $\Xi$ according to the following:

- If
$$U \leq \frac{\lambda(\Xi(t))}{\Lambda(t)},$$
the next event for $\Xi$ is a *birth type* event and we update $\Xi(t+\Delta)$ accordingly.
- If
$$\frac{\max\{\lambda(\Xi(t)), \tilde{\lambda}(Y(t))\}}{\Lambda(t)} < U \leq \frac{\max\{\lambda(\Xi(t)), \tilde{\lambda}(Y(t))\} + \mu(\Xi(t))}{\Lambda(t)},$$
the next event for $\Xi$ is a *death type* event and we update $\Xi(t+\Delta)$ accordingly.
- If
$$U > \frac{\max\{\lambda(\Xi(t)), \tilde{\lambda}(Y(t))\} + \max\{\mu(\Xi(t)), \tilde{\mu}(Y(t))\}}{\Lambda(t)},$$
the next event of $\Xi$ is a *null type* event and we update $\Xi(t+\Delta)$ accordingly.
- Otherwise, $\Xi(t+\Delta) = \Xi(t)$.

Similarly, we determine the next event for $Y$ using the same $U$ according to the following:

- If
$$U \leq \frac{\tilde{\lambda}(Y(t))}{\Lambda(t)},$$
we have $Y(t+\Delta) = Y(t) + 1$.
- If
$$\frac{\max\{\lambda(\Xi(t)), \tilde{\lambda}(Y(t))\}}{\Lambda(t)} < U \leq \frac{\max\{\lambda(\Xi(t)), \tilde{\lambda}(Y(t))\} + \tilde{\mu}(Y(t))}{\Lambda(t)},$$
we have $Y(t+\Delta) = Y(t) - 1$.
- Otherwise, $Y(t+\Delta) = Y(t)$.

We next prove by induction that $h(\Xi(t)) \le Y(t)$ for any $t \ge 0$. Suppose $h(\Xi(s)) \le Y(s)$ for any $0 \le s \le t$. If $h(\Xi(t)) = Y(t)$, since $\tilde{\lambda}(h(\xi)) \ge \lambda(\xi)$ and $\tilde{\mu}(h(\xi)) \le \mu(\xi)$, according to the coupling construction, the next event can be categorized into one of the following five cases

Case I. $Y(t+\Delta) = Y(t)+1$ and $h(\Xi(t+\Delta)) = h(\Xi(t))+1$.

Case II. $Y(t+\Delta) = Y(t)+1$ and $h(\Xi(t+\Delta)) = h(\Xi(t))$.

Case III. $Y(t+\Delta) = Y(t)-1$ and $h(\Xi(t+\Delta)) = h(\Xi(t))-1$.

Case IV. $Y(t+\Delta) = Y(t)$ and $h(\Xi(t+\Delta)) = h(\Xi(t))-1$.

Case V. $Y(t+\Delta) = Y(t)+1$ and $h(\Xi(t+\Delta)) = h(\Xi(t))$.

Note that in all five cases, we have $h(\Xi(t+\Delta)) \le Y(t+\Delta)$.

Next, if $h(\Xi(t))+1 \le Y(t)$, according to the coupling construction, the next event can be categorized into one of the following seven cases

Case I. $Y(t+\Delta) = Y(t)+1$ and $h(\Xi(t+\Delta)) = h(\Xi(t))+1$.

Case II. $Y(t+\Delta) = Y(t)+1$ and $h(\Xi(t+\Delta)) = h(\Xi(t))$.

Case III. $Y(t+\Delta) = Y(t)$ and $h(\Xi(t+\Delta)) = h(\Xi(t))+1$.

Case IV. $Y(t+\Delta) = Y(t)-1$ and $h(\Xi(t+\Delta)) = h(\Xi(t))-1$.

Case V. $Y(t+\Delta) = Y(t)$ and $h(\Xi(t+\Delta)) = h(\Xi(t))-1$.

Case VI. $Y(t+\Delta) = Y(t)-1$ and $h(\Xi(t+\Delta)) = h(\Xi(t))$.

Case VII. $Y(t+\Delta) = Y(t)+1$ and $h(\Xi(t+\Delta)) = h(\Xi(t))$.

Note that in all seven cases, we have $h(\Xi(t+\Delta)) \le Y(t+\Delta)$.

We have thus proved that $h(\Xi(t)) \le Y(t)$ for any $t \ge 0$. Sending $t$ to infinity, we have $h(\Xi(\infty)) \le Y(\infty)$. This implies that $Y(\infty)$ stochastically dominate $h(\Xi(\infty))$. $\square$

### B.2. Optimality of Threshold Policy

In this subsection, we establish that the optimal admission control policy in a single-class queue is a threshold policy.

LEMMA 2. *In an $M/M/n$ queue with a single class of customers, the optimal admission control policy that minimizes the sum of long-run average revenue loss and waiting cost (1) is a threshold policy.*

*Proof.* Consider the value iteration where at iteration $t$ for state $k$, i.e., the number of demands in the system $k$, the relative value function update takes the form

$$\begin{aligned} J^t(k) = \frac{1}{\Lambda} \min_{\pi\in[0,1]} \big\{ & h\cdot\max\{k-n,0\} + \lambda r\pi/\mu - J^{t-1}(0) \\ & + \min\{n,k\}\mu J^{t-1}(k-1) + (1-\pi)\lambda J^{t-1}(k+1) + (\Lambda - \min\{n,k\}\mu - (1-\pi)\lambda) J^{t-1}(k) \big\}. \end{aligned}$$

Let $\pi^t(k)$ denote the minimizer of $J^t(k)$, i.e., $\pi^t(k)$ is the probability of blocking an incoming demand when the system is at state $k$.

$$\begin{aligned} \pi^t(k) &= \arg\min_{\pi\in[0,1]} \big\{ \lambda r\pi/\mu - \pi\lambda J^{t-1}(k+1) + \pi\lambda J^{t-1}(k) \big\} \\ &= \begin{cases} 0 & \text{if } r/\mu - J^{t-1}(k+1) + J^{t-1}(k) \ge 0 \\ 1 & \text{if } r/\mu - J^{t-1}(k+1) + J^{t-1}(k) < 0. \end{cases} \end{aligned} \tag{6}$$

We start by setting $J^0(k)=0, \forall k=0,1,2,\dots$. Note that $J^0$ is increasing and convex in $k$, i.e., $J^0(k+1)\geq J^0(k)$, $J^0(k+1)-J^0(k)\leq J^0(k+2)-J^0(k+1)$ for all $k=0,1,2,\dots$. We next show that by induction the monotonicity and convexity of $J^t$ hold for all $t\geq 0$. Suppose $J^{t-1}(k+1)\geq J^{t-1}(k)$, $J^{t-1}(k+1)-J^{t-1}(k)\leq J^{t-1}(k+2)-J^{t-1}(k+1)$ for all $k=0,1,2,\dots$. For $J^t$, we have

$$
\begin{aligned}
&\Lambda(J^t(k+1)-J^t(k))\\
=&h\cdot(\max\{k+1-n,0\}-\max\{k-n,0\})+\frac{\lambda r}{\mu}\big(\pi^t(k+1)-\pi^t(k)\big)\\
&+\min\{n,k+1\}\mu J^{t-1}(k)-\min\{n,k\}\mu J^{t-1}(k-1)\\
&+\lambda\big((1-\pi^t(k+1))J^{t-1}(k+2)-(1-\pi^t(k))J^{t-1}(k+1)\big)\\
&+(\Lambda-\min\{n,k+1\}\mu-\lambda(1-\pi^t(k+1)))J^{t-1}(k+1)-(\Lambda-\min\{n,k\}\mu-\lambda(1-\pi^t(k)))J^{t-1}(k)\\
=&h\cdot(\max\{k+1-n,0\}-\max\{k-n,0\})+\frac{\lambda r}{\mu}\big(\pi^t(k+1)-\pi^t(k)\big)\\
&+\min\{n,k+1\}\mu J^{t-1}(k)-\min\{n,k\}\mu J^{t-1}(k-1)\\
&+\lambda\big((1-\pi^t(k+1))J^{t-1}(k+2)-(1-\pi^t(k))J^{t-1}(k+1)\big)\\
&+(\Lambda-\min\{n,k+1\}\mu-\lambda(1-\pi^t(k)))\big(J^{t-1}(k+1)-J^{t-1}(k)\big)\\
&+\big(\lambda(1-\pi^t(k))-\lambda(1-\pi^t(k+1))\big)J^{t-1}(k+1)+(\min\{n,k\}\mu-\min\{n,k+1\}\mu)J^{t-1}(k)\\
=&h\cdot(\max\{k+1-n,0\}-\max\{k-n,0\})+\frac{\lambda r}{\mu}\big(\pi^t(k+1)-\pi^t(k)\big)+\min\{n,k\}\mu\big(J^{t-1}(k)-J^{t-1}(k-1)\big)\\
&+\lambda(1-\pi^t(k+1))\big(J^{t-1}(k+2)-J^{t-1}(k+1)\big)+(\Lambda-\min\{n,k+1\}\mu-\lambda(1-\pi^t(k)))\big(J^{t-1}(k+1)-J^{t-1}(k)\big)\\
\geq&0
\end{aligned}
$$

where the last inequality is due to $J^{t-1}(k)-J^{t-1}(k-1)\geq 0$, $J^{t-1}(k+1)-J^{t-1}(k)\geq 0$, and $J^{t-1}(k+2)-J^{t-1}(k+1)\geq 0$. Moreover, since $J^{t-1}(k+2)-J^{t-1}(k+1)\geq J^{t-1}(k+1)-J^{t-1}(k)$, we have that $\pi^t(k+1)-\pi^t(k)\geq 0$ according to the definition of $\pi^t(k)$ in (6). Hence, $J^t$ is also increasing in $k$.

For the convexity of $J^t(\cdot)$, plugging in $\pi^t(k)$'s, we have

$$
\begin{aligned}
&\Lambda\big[J^t(k+1))-2J^t(k)+J^t(k-1)\big]\\
=&\Lambda\big[J^t(k+1,\pi^t(k+1))-2J^t(k,\pi^t(k))+J^t(k-1,\pi^t(k-1))\big]\\
=&h\cdot\big(\max\{k+1-n,0\}-2\max\{k-n,0\}+\max\{k-n-1,0\}\big)+\frac{\lambda r}{\mu}\big(\pi^t(k+1)-2\pi^t(k)+\pi^t(k-1)\big)\\
&+\min\{n,k+1\}\mu J^{t-1}(k)-2\min\{n,k\}\mu J^{t-1}(k-1)+\min\{n,k-1\}\mu J^{t-1}(k-2)\\
&+(1-\pi^t(k+1))\lambda J^{t-1}(k+2)-2(1-\pi^t(k))\lambda J^{t-1}(k+1)+(1-\pi^t(k-1))\lambda J^{t-1}(k)\\
&+(\Lambda-\min\{n,k+1\}\mu-(1-\pi^t(k+1))\lambda)J^{t-1}(k+1)-2(\Lambda-\min\{n,k\}\mu-(1-\pi^t(k))\lambda)J^{t-1}(k)\\
&+(\Lambda-\min\{n,k-1\}\mu-(1-\pi^t(k-1))\lambda)J^{t-1}(k-1)\\
\geq&\frac{\lambda r}{\mu}\big(\pi^t(k+1)-2\pi^t(k)+\pi^t(k-1)\big)+\pi^t(k+1)\lambda J^{t-1}(k+1)-2\pi^t(k)\lambda J^{t-1}(k)+\pi^t(k-1))\lambda J^{t-1}(k-1)\\
&+\min\{n,k-1\}\mu\cdot\big(J^{t-1}(k)-2J^{t-1}(k-1)+J^{t-1}(k-2)\big)\\
&+\big(\min\{n,k+1\}-\min\{n,k-1\}\big)\mu J^{t-1}(k)-2\big(\min\{n,k\}-\min\{n,k-1\}\big)\mu J^{t-1}(k-1)\\
&+(1-\pi^t(k+1))\lambda J^{t-1}(k+2)-2(1-\pi^t(k))\lambda J^{t-1}(k+1)+(1-\pi^t(k-1))\lambda J^{t-1}(k)
\end{aligned}
$$

$$
\begin{aligned}
&+(\Lambda-\min\{n,k+1\}\mu-\lambda)\big(J^{t-1}(k+1)-2J^{t-1}(k)+J^{t-1}(k-1)\big)\\
&-2(\min\{n,k+1\}\mu-\min\{n,k\}\mu)J^{t-1}(k)+(\min\{n,k+1\}\mu-\min\{n,k-1\}\mu)J^{t-1}(k-1)\\
=&\min\{n,k-1\}\mu\cdot\big(J^{t-1}(k)-2J^{t-1}(k-1)+J^{t-1}(k-2)\big)\\
&+(\Lambda-\min\{n,k+1\}\mu-\lambda)\big(J^{t-1}(k+1)-2J^{t-1}(k)+J^{t-1}(k-1)\big)\\
&+\mu\big(2\min\{n,k\}-\min\{n,k+1\}-\min\{n,k-1\}\big)\big(J^{t-1}(k)-J^{t-1}(k-1)\big)\\
&+\frac{\lambda r}{\mu}\big(\pi^t(k+1)-2\pi^t(k)+\pi^t(k-1)\big)+\pi^t(k+1)\lambda J^{t-1}(k+1)-2\pi^t(k)\lambda J^{t-1}(k)+\pi^t(k-1))\lambda J^{t-1}(k-1)\\
&+(1-\pi^t(k+1))\lambda J^{t-1}(k+2)-2(1-\pi^t(k))\lambda J^{t-1}(k+1)+(1-\pi^t(k-1))\lambda J^{t-1}(k).
\end{aligned}
$$

First, due to the convexity and monotonicity of $J^{t-1}$, we have

$$
\begin{aligned}
&\min\{n,k-1\}\mu\cdot\big(J^{t-1}(k)-2J^{t-1}(k-1)+J^{t-1}(k-2)\big)\geq 0\\
&(\Lambda-\min\{n,k+1\}\mu-\lambda)\big(J^{t-1}(k+1)-2J^{t-1}(k)+J^{t-1}(k-1)\big)\geq 0\\
&\mu\big(2\min\{n,k\}-\min\{n,k+1\}-\min\{n,k-1\}\big)\big(J^{t-1}(k)-J^{t-1}(k-1)\big)\geq 0.
\end{aligned}
$$

Second, for the remaining terms, we consider the following cases:

1. When $\pi^t(k-1)=\pi^t(k)=\pi^t(k+1)=0$.

$$
\begin{aligned}
&+\frac{\lambda r}{\mu}\big(\pi^t(k+1)-2\pi^t(k)+\pi^t(k-1)\big)+\pi^t(k+1)\lambda J^{t-1}(k+1)-2\pi^t(k)\lambda J^{t-1}(k)+\pi^t(k-1))\lambda J^{t-1}(k-1)\\
&+(1-\pi^t(k+1))\lambda J^{t-1}(k+2)-2(1-\pi^t(k))\lambda J^{t-1}(k+1)+(1-\pi^t(k-1))\lambda J^{t-1}(k)\\
=&\lambda\Big[J^{t-1}(k+2)-2J^{t-1}(k+1)+J^{t-1}(k)\Big]\geq 0
\end{aligned}
$$

2. When $\pi^t(k-1)=\pi^t(k)=\pi^t(k+1)=1$.

$$
\begin{aligned}
&+\frac{\lambda r}{\mu}\big(\pi^t(k+1)-2\pi^t(k)+\pi^t(k-1)\big)+\pi^t(k+1)\lambda J^{t-1}(k+1)-2\pi^t(k)\lambda J^{t-1}(k)+\pi^t(k-1))\lambda J^{t-1}(k-1)\\
&+(1-\pi^t(k+1))\lambda J^{t-1}(k+2)-2(1-\pi^t(k))\lambda J^{t-1}(k+1)+(1-\pi^t(k-1))\lambda J^{t-1}(k)\\
=&\lambda\Big[J^{t-1}(k+1)-2J^{t-1}(k)+J^{t-1}(k-1)\Big]\geq 0
\end{aligned}
$$

3. When $\pi^t(k-1)=0,\pi^t(k)=\pi^t(k+1)=1$.

$$
\begin{aligned}
&+\frac{\lambda r}{\mu}\big(\pi^t(k+1)-2\pi^t(k)+\pi^t(k-1)\big)+\pi^t(k+1)\lambda J^{t-1}(k+1)-2\pi^t(k)\lambda J^{t-1}(k)+\pi^t(k-1))\lambda J^{t-1}(k-1)\\
&+(1-\pi^t(k+1))\lambda J^{t-1}(k+2)-2(1-\pi^t(k))\lambda J^{t-1}(k+1)+(1-\pi^t(k-1))\lambda J^{t-1}(k)\\
=&\lambda\Big[-\frac{r}{\mu}+J^{t-1}(k+1)-J^{t-1}(k)\Big]\geq 0,
\end{aligned}
$$

since $\pi^t(k)=1$ if and only if $\frac{r}{\mu}-J^{t-1}(k+1)+J^{t-1}(k)\leq 0$

4. When $\pi^t(k-1)=\pi^t(k)=0,\pi^t(k+1)=1$.

$$
\begin{aligned}
&+\frac{\lambda r}{\mu}\big(\pi^t(k+1)-2\pi^t(k)+\pi^t(k-1)\big)+\pi^t(k+1)\lambda J^{t-1}(k+1)-2\pi^t(k)\lambda J^{t-1}(k)+\pi^t(k-1))\lambda J^{t-1}(k-1)\\
&+(1-\pi^t(k+1))\lambda J^{t-1}(k+2)-2(1-\pi^t(k))\lambda J^{t-1}(k+1)+(1-\pi^t(k-1))\lambda J^{t-1}(k)\\
=&\lambda\Big[\frac{r}{\mu}-J^{t-1}(k+1)+J^{t-1}(k)\Big]\geq 0,
\end{aligned}
$$

since $\pi^t(k)=0$ if and only if $\frac{r}{\mu}-J^{t-1}(k+1)+J^{t-1}(k)\geq 0$.

Combing the above discussion, we have proved that $J^t(k+1))-2J^t(k)+J^t(k-1)\geq 0$, i.e., $J^t$ is convex. By the convergence of the value iteration algorithm in the Markov decision process (Puterman 1990), the optimal policy is a threshold policy. $\square$

## Appendix C: Proofs of Results in Section 4

*Proof of Proposition 1.* For $C^n \geq n$, we set $q_n = C^n - n$. Let

$$A_n := \sum_{k=0}^{n-1} \frac{\lambda(n)^k}{k!\mu^k} = \sum_{k=0}^{n-1} \frac{(n\rho_n)^k}{k!}$$

$$B_n := \sum_{k=n}^{C^n} \frac{\lambda(n)^k}{n^{k-n}n!\mu^k} = \frac{(n\rho_n)^n}{n!} \sum_{k=0}^{q_n} \rho_n^k = \frac{(n\rho_n)^n}{n!} R_n$$

$$U_n := \sum_{k=n}^{C^n} (k-n) \frac{\lambda(n)^k}{n^{k-n}n!\mu^k} = \frac{(n\rho_n)^n}{n!} \sum_{k=0}^{q_n} k\rho^k = \frac{(n\rho_n)^n}{n!} S_n,$$

where

$$R_n = \frac{1-\rho_n^{q_n+1}}{1-\rho_n} \text{ and } S_n = \frac{\rho_n \left(1-\rho_n^{q_n} - q_n\rho_n^{q_n}(1-\rho_n)\right)}{(1-\rho_n)^2}.$$

Define $N(n)$ as a Poisson random variable with rate $n\rho_n$.

**When $\boldsymbol{\rho < 1}$,** $C^n \geq n$. The probability of blocking takes the form

$$\begin{aligned}\mathbb{P}(X^n(\infty; C^n) = C^n) =& \frac{\rho_n^{q_n} \frac{(n\rho_n)^n}{n!} \exp(-n\rho_n)}{A_n \exp(-n\rho_n) + B_n \exp(-n\rho_n)} \\ =& \frac{\rho_n^{q_n} \exp(n(1-\rho_n+\log\rho_n)/(1+\mathcal{O}(1/n))}{\sqrt{2\pi n}\mathbb{P}(N(n)<n) + \exp(n(1-\rho_n+\log\rho_n)R_n/(1+\mathcal{O}(1/n))}\end{aligned}$$

by Sterling's approximation, i.e., $n! = \sqrt{2\pi n} \cdot (n/e)^n(1+\mathcal{O}(1/n))$. Since (1) $\rho_n \to \rho < 1$ as $n \to \infty$, (2) $\rho_n^{q_n} < 1$, and (3) $\mathbb{P}(N(n) < n)$ and $R_n$ both converges to an $\mathcal{O}(1)$ constant as $n \to \infty$, we have

$$\mathbb{P}(X^n(\infty; C^n) = C^n) = \mathcal{O}(\exp(n(1-\rho+\log\rho))/\sqrt{n}).$$

Then,

$$\lambda(n)\mathbb{P}(X^n(\infty; C^n) = C^n) = \mathcal{O}(\sqrt{n}\exp(n(1-\rho+\log\rho))) = o(1),$$

where $n(1-\rho+\log\rho) < 0$. Similarly, we can show that

$$\begin{aligned}\mathbb{E}[(X^n(\infty; C^n) - n)^+] =& \frac{U_n \exp(-n\rho_n)}{A_n \exp(-n\rho_n) + B_n \exp(-n\rho_n)} \\ =& \frac{\exp(n(1-\rho_n+\log\rho_n)S_n/(1+\mathcal{O}(1/n))}{\sqrt{2\pi n}\mathbb{P}(N(n)<n) + \exp(n(1-\rho_n+\log\rho_n)R_n/(1+\mathcal{O}(1/n))} \\ =& \mathcal{O}(\exp(n(1-\rho+\log\rho))/\sqrt{n}) = o(1).\end{aligned}$$

Above all, we have $\mathcal{L}_n(C^n) = o(1)$. Since $\mathcal{L}_n^* \geq 0$, we have $\mathsf{GAP}_n(C^n) = o(1)$.

**When $\boldsymbol{\rho > 1}$,** we set $C^n = n + \mathcal{O}(1)$. In this case, the waiting cost is $\mathcal{O}(1)$. For the blocking probability, since $\mathbb{P}(X^n(\infty; C^n) = C^n) \geq \mathbb{P}(X^n(\infty; n) = n)$, we establish a bound for $\mathbb{P}(X^n(\infty; n) = C^n)$ next.

$$\begin{aligned}\mathbb{P}(X^n(\infty; n) = n)^{-1} =& \frac{A_n + (n\rho_n^n)/n!}{(n\rho_n^n)/n!} \\ =& \sum_{i=0}^{n} \frac{n!}{(n-i)!i!} (n\rho_n)^{-i} i! \\ =& \sum_{i=0}^{n} \frac{n!}{(n-i)!i!} (n\rho_n)^{-i} \int_0^\infty t^i e^{-t} dt \\ =& \int_0^\infty \sum_{i=0}^{n} \frac{n!}{(n-i)!i!} (n\rho_n)^{-i} t^i e^{-t} dt \\ =& \int_0^\infty \left(1 + \frac{t}{n\rho_n}\right)^n e^{-t} dt \\ =& \int_0^\infty \frac{1}{1+\mathcal{O}(1/n)} \exp(t(1/\rho_n - 1)) dt = \frac{1}{1+\mathcal{O}(1/n)} \frac{1}{1-1/\rho_n}.\end{aligned}$$

Since $\rho_n \to \rho > 1$ as $n \to \infty$, then

$$\lambda(n)\mathbb{P}(X^n(\infty; n) = n) = \lambda(n) - n + \mathcal{O}(1).$$

Since the maximum rate demand can be served is $n$, $\mathcal{L}_n^* \geq \lambda(n) - n$. Thus, $\mathsf{GAP}_n(C^n) = \mathcal{O}(1)$.

**When $\boldsymbol{\rho = 1}$ and $\boldsymbol{\sqrt{n}(1-\rho_n) \to \beta}$ as $\boldsymbol{n \to \infty}$,** we set $C^n = n + a\sqrt{n}$ for $a = a^*(r, h)$. We first note that

$$A_n \exp(-n\rho_n) = \mathbb{P}(N(n) \leq n-1) \to \mathbb{P}(N(0,1) \leq \beta) =: \Phi(\beta)$$

as $n \to \infty$. In addition, by Sterling's approximation,

$$\begin{aligned} B_n \exp(-n\rho_n) &= \frac{(n\rho_n)^n e^{-n\rho_n}}{\sqrt{2\pi n} n^n e^{-n}(1 + \mathcal{O}(1/n))} \frac{1 - \rho_n^{a\sqrt{n}+1}}{1 - \rho_n} \\ &= \frac{e^{n(1-\rho_n) + n\log(\rho_n)} \left(1 - \rho_n^{a\sqrt{n}+1}\right)}{\sqrt{2\pi n}(1-\rho_n)(1 + \mathcal{O}(1/n))} \\ &\to \frac{e^{-\beta^2/2}\left(1 - e^{-a\beta}\right)}{\sqrt{2\pi}\beta} \text{ as } n \to \infty, \end{aligned}$$

and

$$\begin{aligned} \frac{1}{\sqrt{n}} U_n \exp(-n\rho_n) &= \frac{(n\rho_n)^n e^{-n\rho_n}}{\sqrt{2\pi n} n^n e^{-n}(1 + \mathcal{O}(1/n))} \frac{\rho_n \left(1 - \rho_n^{a\sqrt{n}} - a\sqrt{n}\rho_n^{a\sqrt{n}}(1-\rho_n)\right)}{\sqrt{n}(1-\rho_n)^2} \\ &= \frac{\exp(-\beta^2/2)}{\sqrt{2\pi}\beta^2}(1 - e^{-a\beta} - a\beta e^{-a\beta}) \end{aligned}$$

Next, for the steady-state blocking probability, we have

$$\begin{aligned} &\sqrt{n}\mathbb{P}\left(X_n(\infty; C^n) = n + a\sqrt{n}\right) \\ =&\sqrt{n}\frac{(n\rho_n)^n \rho_n^{a\sqrt{n}}}{n!} \frac{1}{A_n + B_n} \\ =&\sqrt{n}\frac{\rho_n^{n+a\sqrt{n}}}{\sqrt{2\pi n} e^{-n}(1 + \mathcal{O}(1/n))} \frac{\exp(-n\rho_n)}{(A_n + B_n)\exp(-n\rho_n)} \quad \text{by Stirling's formula} \\ \to& \left(\sqrt{2\pi} e^{a\beta + \beta^2/2}\right)^{-1} \left(\Phi(\beta) + \frac{e^{-\beta^2/2}(1 - e^{-a\beta})}{\sqrt{2\pi}\beta}\right)^{-1} \quad \text{as } n \to \infty. \end{aligned}$$

Meanwhile, for the steady-state average queue length, we have

$$\begin{aligned} &\frac{1}{\sqrt{n}}\mathbb{E}\left[(X_n(\infty; C^n) - n)^+\right] \\ =&\frac{U_n}{A_n + B_n} \\ \to& \left(\Phi(\beta) + \frac{e^{-\beta^2/2}(1 - e^{-a\beta})}{\sqrt{2\pi}\beta}\right)^{-1} \left(\sqrt{2\pi}\beta^2 e^{\beta^2/2}\right)^{-1} \left(1 - e^{-a\beta} - a\beta e^{-a\beta}\right) \quad \text{as } n \to \infty. \end{aligned}$$

Then, for the total cost rate, we have

$$\begin{aligned} \frac{1}{\sqrt{n}}\mathcal{L}_n(C^n) &= r\frac{\lambda(n)}{\sqrt{n}}\mathbb{P}\left(X_n(\infty; C^n) = n + a\sqrt{n}\right) + h\frac{1}{\sqrt{n}}\mathbb{E}\left[(X_n(\infty; C^n) - n)^+\right] \\ &\to g(a; r, h) \text{ as } n \to \infty. \end{aligned}$$

Suppose the true optimal capacity is $C_n^* = n + \tilde{a}\sqrt{n}$. Since $a = a^*(r, h) = \arg\min_a g(a; r, h)$,

$$\lim_{n\to\infty} \frac{\mathcal{L}_n(C^n = n + a\sqrt{n})}{\sqrt{n}} \leq \lim_{n\to\infty} \frac{\mathcal{L}_n(C^n = n + \tilde{a}\sqrt{n})}{\sqrt{n}}$$

Then,

$$\begin{aligned}\frac{\mathcal{L}_n(C^n = n + a^*\sqrt{n})}{\sqrt{n}} &\le \lim_{n\to\infty} \frac{\mathcal{L}_n(C^n = n + a^*\sqrt{n})}{\sqrt{n}} + o(1) \\ &\le \lim_{n\to\infty} \frac{\mathcal{L}_n(C^n = n + \tilde{a}\sqrt{n})}{\sqrt{n}} + o(1) \le \frac{\mathcal{L}_n(C^n = n + \tilde{a}\sqrt{n})}{\sqrt{n}} + o(1).\end{aligned}$$

Thus, $\mathcal{L}_n(C^n) = \mathcal{L}_n^* + o(\sqrt{n})$. □

*Proof of Proposition 2.* Let $X_\Sigma^n := X_1^n + X_2^n$ denote the total number of jobs in the system. The corresponding scheduling and admission control policy may vary from case to case.

**When $\rho < 1$,** the system is underloaded. Under any non-idling scheduling policy, the total demands in the system, $X_\Sigma^n$, is a birth-and-death process with birth rate $\lambda_1(n) + \lambda_2(n)$ and death rate $X_\Sigma^n \wedge n$. In this case, following the analysis in part 1 of Proposition 1, we can show that $\mathbb{E}[(X_\Sigma^n(\infty) - n)^+ = o(1)$. Since there is no revenue loss, $\mathsf{GAP}_n(\pi) = o(1)$.

**When $\rho > 1$,** the system is overloaded. A lower bound of the total cost under the optimal scheduling and admission policy is $r_2(\lambda_1(n) + \lambda_2(n) - n)$, i.e, the servers are busy all the time and revenue loss is incurred by the class with a lower revenue rate.

Since class 1 demand is prioritized, if we look at $X_1^n$, it is a birth and death process with birth rate $\lambda_1(n)$ and death rate $X_1^n \wedge n$. Since $\lim_{n\to\infty} \lambda_1(n)/n < 1$, it is an underloaded system. Following the analysis in part 1 of Proposition 1, the long-run average total cost (revenue loss plus waiting cost) incurred by class 1 demand is $o(1)$.

Class 2 demand is blocked when all servers are occupied, the total waiting cost in demand class 2 is generated by those demands that are interrupted and sent back to the queue. The total number of jobs in the system, $X_\Sigma^n$, is a birth and death process with birth rate $\lambda_1(n) + \lambda_2(n)\mathbb{1}\{X_\Sigma < n\}$ and death rate $X_\Sigma^n \wedge n$. Let

$$\begin{aligned}A_n &= \sum_{k=0}^{n} \frac{(\lambda_1(n) + \lambda_2(n))^k}{k!} = \frac{(\lambda_1(n) + \lambda_2(n))^n}{n!} \int_0^\infty \left(1 + \frac{t}{n\rho_n}\right)^n e^{-t} dt \\ &= \frac{(\lambda_1(n) + \lambda_2(n))^n}{n!} \frac{1}{1 + \mathcal{O}(1/n)} \frac{\lambda_1(n) + \lambda_2(n)}{\lambda_1(n) + \lambda_2(n) - n}, \\ B_n &= \sum_{k=n}^{\infty} \frac{(\lambda_1(n) + \lambda_2(n))^n \lambda_1(n)^{k-n}}{n^{k-n} n!} = \frac{(\lambda_1(n) + \lambda_2(n))^n}{n!} \sum_{k=0}^{\infty} \left(\frac{\lambda_1(n)}{n}\right)^k \\ &= \frac{(\lambda_1(n) + \lambda_2(n))^n}{n!} \frac{n}{n - \lambda_1(n)}.\end{aligned}$$

Then, for the blocking rate of class 2 demand, we have

$$\begin{aligned}\lambda_2(n)\mathbb{P}(X_\Sigma^n(\infty) \ge n) &= \lambda_2(n) \frac{B_n}{A_n + B_n - \frac{(\lambda_1(n)+\lambda_2(n))^n}{n!}} \\ &= \lambda_2(n) \frac{\frac{n}{n-\lambda_1(n)}}{\frac{1}{1+\mathcal{O}(1/n)} \frac{\lambda_1(n)+\lambda_2(n)}{\lambda_1(n)+\lambda_2(n)-n} + \frac{n}{n-\lambda_1(n)} - 1} \\ &= \lambda_1(n) + \lambda_2(n) - n + \mathcal{O}(1).\end{aligned}$$

For the total queue length, we have

$$\begin{aligned}\mathbb{E}[(X_\Sigma^n(\infty)-n)^+] &= \frac{\frac{(\lambda_1(n)+\lambda_2(n))^n}{n!}}{A_n+B_n-\frac{(\lambda_1(n)+\lambda_2(n))^n}{n!}}\cdot\sum_{k=1}^{\infty}k\left(\frac{\lambda_1(n)}{n}\right)^k\\ &= \frac{(\lambda_1(n)+\lambda_2(n)-n)(n-\lambda_1(n))}{\lambda_2(n)n}\cdot\frac{\lambda_1(n)n}{(n-\lambda_1(n))^2}\\ &= \frac{\lambda_1(n)+\lambda_2(n)-n}{\lambda_2(n)}\cdot\frac{\lambda_1(n)}{n-\lambda_1(n)}=\mathcal{O}(1).\end{aligned}$$

Above all,

$$\mathsf{GAP}_n(\pi)\le r_2(\lambda_1(n)+\lambda_2(n)-n+\mathcal{O}(1))+\mathcal{O}(1)-r_2(\lambda_1(n)+\lambda_2(n)-n)=\mathcal{O}(1).$$

**When $\rho=1$ and $\lim_{n\to\infty}\sqrt{n}(1-\rho_n)=\beta$,** the system is critically loaded. We first consider a single class system with arrival rate $\lambda_1(n)+\lambda_2(n)$ and $n$ servers. This demand class has a revenue rate of $\min\{r_1,r_2\}$ and a waiting cost rate of $\min\{h_1,h_2\}$. The optimal total cost achieved in this system is a lower bound of the total cost in the original V-system. According to part 3 of Proposition 1, the optimal total cost for the single class system is lower bounded by

$$\sqrt{n}\,g(a;\min\{r_1,r_2\},\min\{h_1,h_2\})+o(\sqrt{n}).$$

where $g(a;r,h)$ is defined in (2).

Under the scheduling and admission control policy $\pi$, there is no revenue loss incurred by class 1 demand, and according to part 1 in Proposition 1 the waiting cost generated incurred by the prioritized class, i.e., the class with a higher waiting cost, is $o(1)$.

For the rest of the analysis, we assume $h_1\ge h_2$. The case where $h_1<h_2$ follows similarly. For the total number of jobs in the system, $X_\Sigma$, it is again a birth and death process with birth rate $\lambda_1(n)+\lambda_2(n)\mathbb{1}\{X_\Sigma<a^*(r_2,h_2)\sqrt{n}\}$ and death rate $X_\Sigma\wedge n$. We set $a=a^*(r_2,h_2)$. With a little repetition of notations, we define

$$\begin{aligned}A_n &= \sum_{k=0}^{n-1}\frac{(n\rho_n)^k}{k!},\\ B_n &= \sum_{k=n}^{n+a\sqrt{n}}\frac{(n\rho_n)^n}{n!}\rho_n^{k-n}=\frac{(n\rho_n)^n}{n!}\sum_{k=0}^{a\sqrt{n}}\rho_n^k=\frac{(n\rho_n)^n}{n!}\frac{1-\rho_n^{a\sqrt{n}+1}}{1-\rho_n},\\ D_n &= \sum_{k=n+a\sqrt{n}+1}^{\infty}\frac{(n\rho_n)^n}{n!}\rho_n^{a\sqrt{n}}\left(\frac{\lambda_1(n)}{n}\right)^{k-n-a\sqrt{n}}=\frac{(n\rho_n)^n}{n!}\rho_n^{a\sqrt{n}}\frac{\lambda_1(n)}{n-\lambda_1(n)}.\end{aligned}$$

The rest of the analysis is similar to part 3 of Proposition 1. In particular, we have as $n\to\infty$,

$$\begin{aligned}A_n\exp(-n\rho_n)&\to\Phi(\beta)\\ B_n\exp(-n\rho_n)&\to\frac{e^{-\beta^2/2}\left(1-e^{-a\beta}\right)}{\sqrt{2\pi}\beta}\\ \sqrt{n}D_n\exp(-n\rho_n)&\to\frac{e^{-\beta^2/2}e^{-a\beta}}{\sqrt{2\pi}}\frac{\lambda_1}{1-\lambda_1}\end{aligned}$$

Then, for the blocking probability, which is incurred by class 2 demand only, we have

$$\begin{aligned}&\frac{\lambda_2(n)}{n}\sqrt{n}\mathbb{P}\left(X_\Sigma^n(\infty)\ge n+a\sqrt{n}\right)\\ =&\frac{\lambda_2(n)}{n}\sqrt{n}\frac{D_n+\frac{(n\rho_n)^n}{n!}\rho_n^{a\sqrt{n}}}{A_n+B_n+D_n}\\ \to&\left(\Phi(\beta)+\frac{e^{-\beta^2/2}\left(1-e^{-a\beta}\right)}{\sqrt{2\pi}\beta}\right)^{-1}\left(\sqrt{2\pi}e^{a\beta+\beta^2/2}\right)^{-1}\ \text{as } n\to\infty.\end{aligned}$$

For the queue length process, we have

$$\begin{aligned}
&\frac{1}{\sqrt{n}}\mathbb{E}\left[(X_\Sigma^n(\infty)-n)^+\right]\\
=&\frac{1}{\sqrt{n}}\frac{\frac{(n\rho_n)^n}{n!}\sum_{k=0}^{a\sqrt{n}}k\rho_n^k+\frac{(n\rho_n)^n}{n!}\rho_n^{a\sqrt{n}}\sum_{k=a\sqrt{n}+1}^{\infty}k\left(\frac{\lambda_1(n)}{n}\right)^{k-a\sqrt{n}}}{A_n+B_n+D_n}\\
\to&\left(\Phi(\beta)+\frac{e^{-\beta^2/2}\left(1-e^{-a\beta}\right)}{\sqrt{2\pi}\beta}\right)^{-1}\left(\sqrt{2\pi}\beta^2e^{\beta^2/2}\right)^{-1}\left(1-e^{-a\beta}-a\beta e^{-a\beta}\right)\ \text{as } n\to\infty.
\end{aligned}$$

Then,

$$\begin{aligned}
\mathcal{L}(\pi)&=r_2\lambda_2(n)\mathbb{P}\left(X_\Sigma^n(\infty)\geq n+a\sqrt{n}\right)+h_2\mathbb{E}\left[(X_\Sigma^n(\infty)-n)^+\right]\\
&=\sqrt{n}g(a;r_2,h_2)+o(\sqrt{n}).
\end{aligned}$$

Combing this with the lower bound of the optimal cost, we have,

$$\mathcal{L}(\pi)-L^*\leq\sqrt{n}g(a;r_2,h_2)+o(\sqrt{n})-\sqrt{n}g(a;r_2,h_2)=o(\sqrt{n}),$$

which completes the proof. □

*Proof of Proposition 3.* Let $X_\Sigma^n:=X_1^n+X_2^n$ denote the total number of jobs in the system. The corresponding scheduling and admission control policy may vary from case to case. Let $Q_i^n$ denote the queue length process of class $i$ demand, $i=1,2$, i.e., the number of class $i$ jobs waiting to be served.

**When $\rho<1$,** the system is underloaded. Under any non-idling scheduling policy, the total demands in the system, $X_\Sigma^n$, is a birth-and-death process with birth rate $\lambda_1(n)+\lambda_2(n)$ and death rate $X_\Sigma^n\wedge n$. In this case, following the analysis in part 1 of Proposition 1, we can show that $\mathbb{E}[(X_\Sigma^n(\infty)-n)^+=o(1)$. Since there is no revenue loss, $\mathsf{GAP}_n(\pi)=o(1)$.

**When $\rho>1$,** the system is overloaded. By Lemma 1 (where $\Xi(t)=(Q_1(t),X_1(t),X_2(t))$ and $h(X(t))=Q_1(t)$), $Q_1^n(t)$ is bounded by a coupled birth and death process, $Y_1^n(t)$, with birth rate $\lambda_1(n)$ and death rate $n\mathbb{1}\{Y_1^n(t)>0\}$, for $t\geq0$. In this case,

$$\mathbb{E}[Q_1^n(\infty)]\leq\mathbb{E}[Y_1^n(\infty)]=\frac{\lambda_1(n)}{n-\lambda_1(n)}=\mathcal{O}(1).$$

Next, following the analysis in part 2 of Proposition 2 we have

$$\lambda_2(n)\mathbb{P}(X_\Sigma^n(\infty)\geq n)=\lambda_1(n)+\lambda_2(n)-n+\mathcal{O}(1)$$

and

$$\mathbb{E}[(X_\Sigma^n(\infty)-n)^+]=\mathcal{O}(1).$$

Then,

$$\mathsf{GAP}_n(\pi)\leq r_2(\lambda_1(n)+\lambda_2(n)-n)+\mathcal{O}(1)-r_2(\lambda_1(n)+\lambda_2(n)-n)=\mathcal{O}(1).$$

**When $\rho=1$ and $\lim_{n\to\infty}\sqrt{n}(1-\rho_n)=\beta$,** the system is critically loaded. If $h_1\geq h_2$, by Lemma 1 (where $\Xi(t)=(Q_1(t),X_1(t),X_2(t))$ and $h(X(t))=Q_1(t)$), $Q_1^n(t)$ is bounded by a coupled birth and death process $Y_1^n(t)$, with birth rate $\lambda_1(n)$ and death rate $n\mathbb{1}\{Y_1^n(t)>0\}$ for $t\geq0$. Then,

$$\mathbb{E}[Q_1^n(\infty)]\leq\mathbb{E}[Y_1^n(\infty)]=\frac{\lambda_1(n)}{n-\lambda_1(n)}=\mathcal{O}(1).$$

If $h_1 < h_2$, by Lemma 1, $Q_2^n(t)$ is bounded by a coupled birth and death process $Y_2^n(t)$, with birth rate $\lambda_2(n)$ and death rate $n\mathbb{1}\{Y_2^n(t) > 0\}$, for $t \geq 0$. Then,

$$\mathbb{E}[Q_2^n(\infty)] \leq \mathbb{E}[Y_2^n(\infty)] = \frac{\lambda_2(n)}{n - \lambda_2(n)} = \mathcal{O}(1).$$

The rest of the analysis follows from the proof of part 3 of Proposition 2. In particular, if $h_1 \geq h_2$,

$$\frac{\lambda_2(n)}{n}\sqrt{n}\mathbb{P}\left(X_\Sigma^n(\infty) \geq n + a\sqrt{n}\right) \to \left(\Phi(\beta) + \frac{e^{-\beta^2/2}\left(1 - e^{-a\beta}\right)}{\sqrt{2\pi}\beta}\right)^{-1}\left(\sqrt{2\pi}e^{a\beta+\beta^2/2}\right)^{-1}$$

and

$$\frac{1}{\sqrt{n}}\mathbb{E}\left[(X_\Sigma^n(\infty) - n)^+\right] \to \left(\Phi(\beta) + \frac{e^{-\beta^2/2}\left(1 - e^{-a\beta}\right)}{\sqrt{2\pi}\beta}\right)^{-1}\left(\sqrt{2\pi}\beta^2 e^{\beta^2/2}\right)^{-1}\left(1 - e^{-a\beta} - a\beta e^{-a\beta}\right)$$

as $n \to \infty$. Then,

$$\begin{aligned}\mathsf{GAP}_n(\pi) &\leq \sqrt{n}\left(\frac{r_2\lambda_2(n)}{\sqrt{n}}\mathbb{P}\left(X_\Sigma^n(\infty) \geq n + a\sqrt{n}\right) + \frac{h_2}{\sqrt{n}}\mathbb{E}\left[(X_\Sigma^n(\infty) - n)^+\right]\right) - \sqrt{n}g(a; r_2, h_2) + o(\sqrt{n}) \\ &= \sqrt{n}g(a; r_2, h_2) + o(\sqrt{n}) - \sqrt{n}g(a; r_2, h_2) = o(\sqrt{n}).\end{aligned}$$

The analysis of the case where $h_1 < h_2$ follows similarly. □

## Appendix D: Proofs of Results in Section 5

*Proof of Theorem 1.* We first analyze the number of arcs of the structure generated by Algorithm 1. Note that in each iteration in Algorithm 1, the number of arcs increases by one. At the same time, in each iteration, either the demand class index $i$ increases by one or the server index $j$ increases by one. The algorithm starts with $i = j = 1$ and ends at $i = I$ and $j = J$, so the total number of arcs added in the while loop is $I + J - 2$. Together with the last arc added, $\{I, J\}$, the total number of arcs generated by Algorithm 1 is $I + J - 1$.

Next, we analyze the revenue loss and waiting cost for each demand class. First, since there is no blocking, there is no revenue loss. Second, for demand class $i$, the total number of servers reserved to it is at least $\delta + \lambda_i(n)$. Hence, the sub-system utilization satisfies

$$\lim_{n\to\infty}\frac{\lambda_i(n)}{\lambda_i(n) + \delta} = \frac{\lambda_i(n)}{\lambda_i(n) + \frac{(1/\rho - 1)\sum_{i'=1}^I \lambda_{i'}(n)}{I}} \leq \frac{\lambda_i(n)}{\lambda_i(n) + \frac{(1/\rho-1)\lambda_i(n)}{I}} = \frac{\rho I}{\rho I + 1 - \rho} < 1.$$

Then, following the analysis in part 1 of Proposition 1, the waiting cost is $o(1)$. Note that since each class has its reserved/dedicated servers, preemption or non-preemption does not matter. □

*Proof of Theorem 2.* We first study the number of arcs of the structure generated by Algorithm 2. Notice that in each iteration in Algorithm 2, either the demand class index $i$ increases by one or the server index $j$ increases by one. The algorithm starts with $i = j = 1$ and ends at $i = i^*$ and $j = J$. The total number of arcs added in the while loop is at most $i^* + J - 1$. Considering the last step in Algorithm 2, $\mathcal{G} = \mathcal{G} \cup (i^*, 1) \cup (i^*, 2) \cup \cdots \cup (i^*, J)$, the number of additional arcs is at most $J - 1$. Therefore, the total number of arcs generated by Algorithm 2 is at most $i^* + 2J - 2$ which is bounded from above by $I + 2J - 2$.

We next analyze the total cost generated by scheduling and admission control policy in Definition 2. The analysis can be decomposed into the $i^* - 1$ subsystems, i.e., $M/M/\tilde{s}_i(n)$, $i = 1, 2, \ldots, i^* - 1$. For $M/M/\tilde{s}_i(n)$, it serves two classes of demand, $i$ and $i^*$, with arrival rates $\lambda_i(n)$ and $\lambda_{i^*}(n)/(i^* - 1)$ respectively. The class

$i$ demand enjoys preemptive or non-preemptive priority over the class $i^*$ demand, and the arriving class $i^*$ demand is blocked if all servers are occupied. We denote $\tilde{X}_i^n$ as the total number of jobs in $M/M/\tilde{s}_i(n)$.

When looking at the subsystem $M/M/\tilde{s}_i(n)$, first, since there are infinite waiting spaces for the prioritized class, i.e., class $i$, there is no revenue loss for demand class $i$. Meanwhile, since $\lim_{n\to\infty}\lambda_i(n)/\tilde{s}_i(n)<1$, i) when preemption is allowed, the waiting cost for class $i$ demand is $o(1)$ as shown in the proof of part 2 of Proposition 2. ii) When preemption is not allowed, the waiting cost for class $i$ demand is $\mathcal{O}(1)$ as shown in the proof of part 2 of Proposition 3. Next, for class $i^*$ demand routed to $M/M/\tilde{s}_i(n)$, i) when preemption is allowed, the waiting cost for the class $i^*$ demand is $\mathcal{O}(1)$ and the blocking rate is

$$\Gamma_i(n):=\frac{\lambda_{i^*}(n)}{i^*-1}\mathbb{P}(\tilde{X}_i^n(\infty)\geq\tilde{s}_i(n))=\lambda_i(n)+\frac{\lambda_{i^*}(n)}{i^*-1}-\tilde{s}_i(n)+\mathcal{O}(1)$$

as shown in the proof of part 2 of Proposition 2. ii) When preemption is not allowed, the waiting cost for the class $i^*$ demand is 0 and the blocking probability is $\Gamma_i(n)$ as shown in the proof of part 2 of Proposition 3. Above all, combing the $i^*-1$ subsystems, we have

$$\begin{aligned}\mathsf{GAP}_n(\pi)&\leq r_{i^*}\sum_{i=1}^{i^*-1}\Gamma_i(n)+\mathcal{O}(1)-r_{i^*}\left(\sum_{i=1}^{i^*}\lambda_i(n)-n\right)+\mathcal{O}(1)\\&=r_{i^*}\left(\sum_{i=1}^{i^*}\lambda_i(n)-n\right)-r_{i^*}\left(\sum_{i=1}^{i^*}\lambda_i(n)-n\right)+\mathcal{O}(1)=\mathcal{O}(1),\end{aligned}$$

which completes the proof. □

*Proof of Theorem 3.* Analogous to the proof of Theorem 2, the total number of arcs is bounded from above by $I+2J-2$.

Let $X_\Sigma^n(t)=\sum_{i=1}^I X_i^n(t)$. To save space, we write $a^*=a^*(\min r_i,\min h_i)$, and suppress the dependence on $n$ in $X_i^n$, $X_\Sigma^n$, $\lambda_i(n)$, and $\tilde{s}_i(n)$.

If $i^*=I$, for demand class $i$ with $1\leq i\leq I-1$, since it has preemptive priority over class $I$ demand and it is blocked when the number of class $i$ jobs in the system is larger than or equal to $\tilde{s}_i$, its waiting cost is 0. In addition, since $\lambda_i=\tilde{s}_i-\delta$ with $\delta=\Theta(n)$, following the analysis in part 1 of Proposition 1, we have

$$r_i\lambda_i\mathbb{P}(X_i(t)=s_i)=o(1).$$

Next, $X_\Sigma(t)$ will increase by 1 at rate

$$\sum_{i=1}^{I-1}\lambda_i\mathbb{1}\{X_i(t)<\tilde{s}_i\}+\lambda_I\mathbb{1}\{X_\Sigma(t)<n+a^*\sqrt{n}\}\leq\sum_{i=1}^{I-1}\lambda_i+\lambda_I\mathbb{1}\{X_\Sigma(t)<n+a^*\sqrt{n}\},$$

and it will decrease by 1 at rate

$$\sum_{i=1}^{I-1}X_i(t)+\min\left\{X_I(t),\sum_{i=1}^{I-1}(s_i-X_i(t))\right\}=X_\Sigma(t)\wedge n.$$

Consider a birth and death process $\{Y(t),t\geq0\}$ with birth rate $\sum_{i=1}^{I-1}\lambda_i+\lambda_I\mathbb{1}\{Y(t)<n+a^*\sqrt{n}\}$ and death rate $Y(t)\wedge n$. Then, by Lemma 1 (where $\Xi(t)=(X_1(t),\ldots,X_I(t))$ and $h(X(t))=\sum_{i=1}^I X_i(t)$), $Y(\infty)$ stochastically dominates $X_\Sigma(\infty)$. Since

$$\mathbb{E}[Q_I(\infty)]\leq\mathbb{E}[(X_\Sigma(\infty)-n)^+]\leq\mathbb{E}[(Y(\infty)-n)^+]$$

and

$$\mathbb{P}(X_\Sigma(\infty) \geq n + a^*\sqrt{n}) \leq \mathbb{P}(Y(\infty) \geq n + a^*\sqrt{n}),$$

following the analysis in part 3 of Proposition 2, we have

$$\begin{aligned}
&r_I\lambda_I\mathbb{P}(X_\Sigma(\infty) \geq n + a^*\sqrt{n}) + h_I\mathbb{E}[Q_I(\infty)] \\
\leq &r_I\lambda_I\mathbb{P}(Y(\infty) \geq n + a^*\sqrt{n}) + h_I\mathbb{E}[(Y(\infty) - n)^+] = \sqrt{n}g(a^*) + o(\sqrt{n}).
\end{aligned}$$

Similarly, if $i^* = I-1$, for demand $i$ with $1 \leq i \leq I-1$, since it has preemptive priority over class $I$ demand and it is blocked when the number of class $i$ jobs in the system is larger than equal to $\tilde{s}_i$, its waiting cost is 0. For demand class $i$ with $1 \leq i \leq I-2$, since $\lambda_i = \tilde{s}_i - \delta$ with $\delta = \Theta(n)$, following the analysis in part 1 of Proposition 1, we have

$$r_i\lambda_i\mathbb{P}(X_i(t) = s_i) = o(1).$$

For demand class $I$, since no demand is blocked, the loss revenue is 0.

Next, $X_\Sigma(t)$ will increase by 1 at rate

$$\begin{aligned}
&\sum_{i=1}^{I-2}\lambda_i\mathbb{1}\{X_i(t) < \tilde{s}_i\} + \lambda_{I-1}\mathbb{1}\{X_{I-1}(t) < \tilde{s}_{I-1} \text{ and } X_\Sigma(t) < n + a^*\sqrt{n}\} + \lambda_I \\
\leq &\sum_{i=1}^{I-2}\lambda_i + \lambda_{I-1}\mathbb{1}\{X_\Sigma(t) < n + a^*\sqrt{n}\} + \lambda_I,
\end{aligned}$$

and it will decrease by 1 at rate

$$\sum_{i=1}^{I-1}X_i(t) + \min\left\{X_I(t), \sum_{i=1}^{I-1}(s_i - X_i(t))\right\} = X_\Sigma(t) \wedge n.$$

Consider a birth and death process $\{Y(t), t \geq 0\}$ with birth rate $\sum_{i=1}^{I-2}\lambda_i + \lambda_{I-1}\mathbb{1}\{Y(t) < n + a^*\sqrt{n}\} + \lambda_I$ and death rate $Y(t) \wedge n$. Then, by Lemma 1, $Y(\infty)$ stochastically dominates $X_\Sigma(\infty)$. Since

$$\mathbb{E}[Q_I(\infty)] \leq \mathbb{E}[(X_I(\infty) - n)^+] \leq \mathbb{E}[(Y(\infty) - n)^+]$$

and

$$\mathbb{P}(X_\Sigma(\infty) \geq n + a^*\sqrt{n}) \leq \mathbb{P}(Y(\infty) \geq n + a^*\sqrt{n}),$$

following the analysis in part 3 of Proposition 2, we have

$$\begin{aligned}
&r_{I-1}\lambda_{I-1}\mathbb{P}(X_\Sigma(\infty) \geq n + a^*\sqrt{n}) + h_I\mathbb{E}[Q_I(\infty)] \\
\leq &r_{I-1}\lambda_{I-1}\mathbb{P}(Y(\infty) \geq n + a^*\sqrt{n}) + h_I\mathbb{E}[(Y(\infty) - n)^+] = \sqrt{n}g(a^*) + o(\sqrt{n}).
\end{aligned}$$

Above all, we have that the total cost (revenue loss plus waiting cost) incurred is bounded by $\sqrt{n}g(a^*) + o(\sqrt{n})$. Thus the optimality gap is bounded by $o(\sqrt{n})$. $\square$