Understanding and Optimizing Complex Stochastic Systems Through Simple Systems

Yige Hong

March 25, 2024

1 Introduction

Complex stochastic systems that consist of a large number of interacting components naturally arise in various research domains, such as resource allocation in computing systems, congestion control in networks, wireless communication, machine maintenance, clinical trials, etc. The scale of these systems, coupled with the interactions among the components, makes the dynamics within them highly complex. Consequently, decision-making problems for such stochastic systems are often highly challenging.

To understand and optimize these complex systems, we consider first solving a simple problem, and then converting the policy or performance bound obtained from the simple problem back to the complex problems. If the simple problem is properly constructed and the conversion is properly done, we can design a policy and prove its near-optimality. We explain this framework in more detail in Section 1.2.

This document investigates a set of example problems through some related simple systems. These problems are briefly introduced below and will be formally defined and studied in Section 2-5. Our studies demonstrate the effectiveness of this idea and its great potential for broader applicability. We are also working on a few new problems and plan to finish them before thesis defense. These problems are introduced in Section 6.

1.1 Complex stochastic systems studied in this document

Restless bandits. The restless bandit (RB) problem is a stochastic sequential decision-making problem, which consists of multiple arms, each being an identical Markov Decision process (MDP). In each time step, we pull a predetermined fraction of arms based on their states, which causes each arm to generate a certain amount of reward and have a state transition. Both the amount of reward and the distribution of the next state depend on the current state and whether the arm is pulled. The goal of the RB problem is to find a policy that maximizes the long-run average reward. The name "restless" comes from the fact that the state of each arm changes every time step no matter whether it is pulled or not.

The RB problem finds applications across a spectrum of domains, including wireless communication [3], congestion control [10], queueing models [6], crawling web content [11], machine maintenance [69], clinical trials [158], to name a few. To give a concrete example of the applications of RB, consider the problem of job scheduling with deadlines: suppose we have a finite-size queue and a few servers. Jobs arrive to the system over time, each associated with an unknown service time and a known deadline. We assume that the service times of the jobs are i.i.d. and geometrically

distributed. The scheduler needs to choose a subset of jobs to serve in each time step, with the goal of maximizing the long-run fraction of jobs completed by the deadline.

The problem of scheduling with deadlines can be seen as an RB problem in the following sense. Each position in the queue can be viewed as an arm. The state of the arm is whether there is a job in this position and the time until the deadline of the job. Pulling the arm means serving the job in this time step. The arms are restless because the times until deadlines change every time step.

In contrast to "restful" Markovian bandits whose optimal reward can be achieved with the famous Gittins index policy, solving the optimal policy of a restless bandit problem is PSPACE-hard in general [122]. Previous studies have focused on finding asymptotically optimal policies of restless bandits when the number of arms scales to infinity [161, 156, 59, 60]. However, even for asymptotic optimality, existing policies can only achieve it under a complicated assumption based on the global convergence of some non-linear ODE.

Stochastic bin-packing with time-varying item sizes. Consider a datacenter with a large number of servers for serving jobs submitted by users. The jobs arrive to the datacenter over time, each requesting a certain amount of resrouces like CPU, memory, network, etc. Each job is sent to a server by a dispatcher upon arrival, after which it immediately starts being served and remains in service for a random amount of time. The dispatcher seeks to collocate a set of jobs based on their resource requirements, so that the average utilization among the active servers (servers that are serving any job) is maximized.

Prior work often assumes the resource requirements of the jobs are static throughout, so the problem can be viewed as a stochastic bin-packing problem, where each server is a bin, each job is an item, and the items arrive and depart dynamically. In this setting, simple dispatching policies have been proved to be asymptotically optimal as the system scale becomes large [61, 145, 146, 147, 149].

However, in reality, the resource requirements can be time-varying. This necessitates a new model – stochastic bin-packing with time-varying item sizes. The utilization in this new model is significantly harder to optimize, because each dispatching decision can have a complicated effect on the future resource requirements on a server. Unlike in the static item-size setting, in the time-varying item-size setting, there is no intuitive heuristics policy that can be proved to be asymptotically optimal.

Optimal scheduling in complex queueing models. Consider a queueing system with a finite number of servers. The jobs arrive to the system over time, each with an unknown size sampled from a certain initial distribution. During the service of each job, additional information about its size is revealed, which causes the posterior distribution of the job size to update dynamically. The order in which the jobs are served is determined by a scheduling policy. Then which scheduling policy should we use to minimize the long-run expected queue length?

A policy called Gittins policy (also known as the Gittins index policy) adapted from the Markovian bandit setting [66] has been proven to be optimal in the M/G/1 system [65, 139]. For systems more complex than M/G/1, the optimal scheduling policy is unknown and is likely to be pretty hard to find. Then a natural question is, can we show that Gittins policy remains near-optimal in the queueing systems more complex than M/G/1?

Recent studies [137, 136] have proved that Gittins policy is heavy-traffic optimal in the M/G/k system, i.e., the system with Poisson arrivals and k servers. However, allowing multiple servers is just one way of generalizing the M/G/1 model. M/G/1 could also be generalized by allowing non-Poisson arrivals (G/G/1), having setup times (G/G/1/setup), or a combination of multiple complex features (e.g. G/G/k/setup). Given that Gittins policy itself is already complex, analyzing

it in all those complex queueing models has been considered to be pretty challenging.

Multiserver-job scheduling. Existing queueing models are mainly single-server-job models, where each job occupies one server during the service. However, in today's datacenters, lots of jobs require to be simultaneously served by multiple servers [151, 157, 100, 4]. Such jobs are called *multiserver jobs*.

To study the effect of multiserver jobs on the datacenter, consider the following queueing model with a centralized queue, n identical servers. In this model, jobs arrive over time according to a Poisson process with a certain arrival rate. Each job requires to be served for an exponentially distributed duration and occupies a predetermined number of servers (referred to as *server need*) during service. There is a finite number of types of jobs, each is characterized by its arrival rate, service rate, and server need. The order in which jobs of different types are served is determined by a scheduling policy.

Analyzing the performance or finding an optimal scheduling policy is more challenging in a multiserver-job system than in a traditional queueing models for many reasons. For instance, due to the packing effect, the multiserver-job system can waste service capacity even if there are enough jobs in the system. The capacity wasted in this way has a complex dependency on the scheduling policy. Moreover, the heterogeneity of the different types of jobs makes the system dynamics multi-dimensional. Although traditional queueing models can also be heterogeneous in service rates, the heterogeneous server needs in multiserver-job systems add another layer of complexity.

1.2 Near-optimal policy design through simple systems

In this section, we will set up a technical framework for designing policies and proving their near-optimality through understanding simple systems. This framework is underlying the technical approach of all of our papers included in this document and represents the main innovation of some of them.

Suppose the complex stochastic system is represented as a Markov decision process where each policy π induces a Markov process $\{X_t^\pi : t \in \mathcal{T}\}$, with $\mathcal{T} = \mathbb{R}_{\geq 0}$ or $\mathbb{N}$. Let X_∞^π have the steady-state distribution induced by π. Our goal is finding a policy π that maximizes $\mathbb{E}\left[h(X_\infty^\pi)\right]$ for some predefined function $h(\cdot)$ over the state space of $\{X_t\}$. The function $h(\cdot)$ represents a performance metric of the system under study – for example, the reward of bandits or the queue length of a queueing model.

Oftentimes, it is hard to exactly optimize $\mathbb{E}\left[h(X_\infty^\pi)\right]$, and only near-optimal policies are tractable. To find a near-optimal policy, we consider *jointly constructing* a policy $\hat{\pi}$ and a Markov process $\{Y_t : t \in \mathcal{T}\}$ such that

- $\mathbb{E}[h(X_\infty^\pi)] \leq \mathbb{E}\left[h(Y_\infty)\right]$ for any policy π, and
- $\left|\mathbb{E}\left[h(X_\infty^{\hat{\pi}})\right] - \mathbb{E}\left[h(Y_\infty)\right]\right|$ is small.

These two conditions should imply the near-optimality of $\hat{\pi}$. Intuitively speaking, $\{Y_t\}$ should be a system that operates under a more ideal situation than $\{X_t^\pi\}$, and the policy $\hat{\pi}$ should be designed to make $\{X_t^\pi\}$ mimic the behavior of Y_t. When $\{Y_t\}$ is properly constructed, we can often find a $\hat{\pi}$ that is both near-optimal and efficiently computable.

1.3 Techniques based on rate conservation

The framework in Section 1.2 requires calculating the steady-state expectation of a certain performance metric of a Markov process. Such calculations can be done via a set of techniques based

on *rate conservation*. The name "rate conservation" historically originates from Miyazawa's rate conservation law in palm calculus [115]. Here we use "rate conservation" to refer to the following fact that underpins a more general class of techniques for stochastic analysis: roughly speaking, for any Markov process $\{X_t\}$ and function $f(x)$ of Markov process' state x, under mild conditions, the long-run average rate that $f(X_t)$ increases is equal to the long-run average rate that $f(X_t)$ decreases.

Lyapunov drift analysis. Rate conservation in discrete-space Markov chains can be formalized using the notion of *drift*. Specifically, for a Markov chain $\{X_t\}$ and a function f of the Markov chain's state, the drift of f at state x, denoted by $G_X f(x)$, is defined as the instantaneous rate that $f(X_t)$ changes conditioned on $X_t = x$. The operator G_X that maps f to its drift is called the infinitesimal generator of $\{X_t\}$, whose form is available from the definition of $\{X_t\}$. Then under mild conditions on f and $\{X_t\}$, we have

$$\mathbb{E}\left[G_X f(X_\infty)\right] = 0, \tag{1}$$

i.e., the long-run average rate that $f_h(X_t)$ changes is zero.

To perform the Lyapunov drift analysis, we first construct a *Lyapunov function* or *test function* f and then plug it into (1) to get equalities about the steady-state expectations of the performance metrics under study. Lyapunov drift analysis is also known as the drift method and has been extensively used in queueing theory literature [see, e.g., 46, 109, 159].

Finding the right Lyapunov function is usually tricky. One possible idea is to consider a simpler Markov process $\{Y_t\}$ that captures some important features of $\{X_t\}$ and uses the dynamics of $\{Y_t\}$ to derive the Lyapunov function. We use this idea to study restless bandits in Section 2.5.2.

Stein's method. Stein's method is a powerful tool invented in [144] for bounding the difference between two distributions. There has been a large body of literature on different usages and variants of Stein's method [see, e.g., 21, 23, 19, 20, 170]. In this document, we only use Stein's method in a basic way, as briefly explained below.

For any two Markov chains $\{X_t\}$ and $\{Y_t\}$ on the same state space and any performance metric h, let f_h be a function of the state satisfying the Poisson equation:

$$\mathbb{E}\left[h(Y_\infty)\right] - h(y) = G_Y f_h(y). \tag{2}$$

f_h is called the relative value function. Under mild conditions, we have

$$\mathbb{E}\left[G_X f_h(X_\infty)\right] = 0, \tag{3}$$

Taking expectations in (2) with $y = X_\infty$ and subtracting (3), we get

$$\mathbb{E}\left[h(Y_\infty)\right] - \mathbb{E}\left[h(X_\infty)\right] = \mathbb{E}\left[(G_Y - G_X) f_h(X_\infty)\right]. \tag{4}$$

Equation (4) allows us to relate the complex system $\{X_t\}$ to a simple system $\{Y_t\}$, by comparing their generators $G_Y - G_X$ and analyzing properties of the relative value function f_h. Intuitively, $G_Y - G_X$ captures the rate that X_t deviates from Y_t, while f_h quantifies the consequence of per unit of deviation based on the dynamics of Y_t.

Roughly speaking, Stein's method can be interpreted as a special case of Lyapunov drift analysis that takes $f = f_h$ in (1). However, the two methods require insights into constructing different types of objects and have different workflows: when performing the Lyapunov drift analysis, we first

construct the Lyapunov function f and then manipulate $G_X f$ into a form that is related to the performance metric under study; when using Stein's method, we first construct the performance metric h and then analyze properties of f_h via the Poisson equation (2). Depending on the specific problem, one method could be more convenient than the other.

Rate conservation law as in Palm calculus. Rate conservation law is the generalization of Lyapunov drift analysis to Markov processes in general state spaces, where we could have both continuous changes and discrete jumps. It has been used a lot in recent analysis of queueing models with general inter-arrival times and service times [see, e.g., 116, 24, 22].

The key equation of rate conservation law is as follows: for a stochastic process Z_t, under mild conditions,

$$\mathbb{E}[Z'] + \lambda \mathbb{E}_0[\Delta Z] = 0, \tag{5}$$

where $\mathbb{E}[Z']$ denotes the long-run average continuous-changes of Z, $\mathbb{E}_0[\Delta Z]$ denotes the long-run average jumps of Z, and λ denotes the average rate of jumps.

To analyze a Markov process $\{X_t\}$ with rate conservation law, we let $Z_t = f(X_t)$ for some $f(\cdot)$ and apply (5) to get an equality. Similar to the Lyapunov drift analysis, we need to choose an appropriate f to extract useful information about $\{X_t\}$ from its continuous changes and jumps.

Little's Law. Little's law [92] is a fundamental tool in queueing theory. It characterizes the relationship among the average arrival rate, λ, average waiting time, $\mathbb{E}[W]$, and the average number of jobs, $\mathbb{E}[N]$, using the following equation

$$\mathbb{E}[N] = \lambda \mathbb{E}[W]. \tag{6}$$

Little's law also can be interpreted as a form of rate conservation: the long-run average rate of job arrivals, λ, is equal to the long-run average rate of job completions, $\mathbb{E}[N]/\mathbb{E}[W]$.

Little's law is useful even beyond queueing models, where it is applied to objects more abstract than jobs. See Section 2.5.1 for its usage in analyzing the coupling of arms in restless bandit problems.

1.4 Organization of the thesis proposal

In the rest of the thesis proposal, we will look into each example problem in Section 1.1. We set up the problem in more detail, overview our results, and explain how we find the near-optimal policies through simple systems. Each problem corresponds to one or multiple papers that we have already finished, as summarized below:

- In Section 2, we look into the restless bandit problem studied in [83, 84];
- In Section 3, we look into the stochastic bin-packing problem with time-varying resource requirements studied in [85];
- In Section 4, we look into the problem of optimal scheduling in G/G/k/setup studied in [81];
- In Section 5, we look into the multiserver scheduling problem studied in [82].

In Section 6, we discuss a few more problems which we plan to solve in the next year and include in the thesis:

- In Section 6.1, we propose the problem of achieving an exponentially small optimality gap in restless bandits. This project is 50% done.

- In Section 6.2, we propose the problem of restless bandits with asynchronous actions. This project is 30% done.
- In Section 6.3, we discuss the problem of improving the universal queue length bound for G/G/n under FCFS. This project is 80% done.

1.5 Timeline

- From April to June 2024, I will work on the problem in Section 6.1.
- From July to September 2024, I will prepare for getting onto the job market and work on the problem in Section 6.2.
- From October to December 2024, I will get onto the job market and continue working on the problem in Section 6.2.
- From January to March 2025, I will work on the problem in Section 6.3.
- From April to June 2025, I will write the thesis.

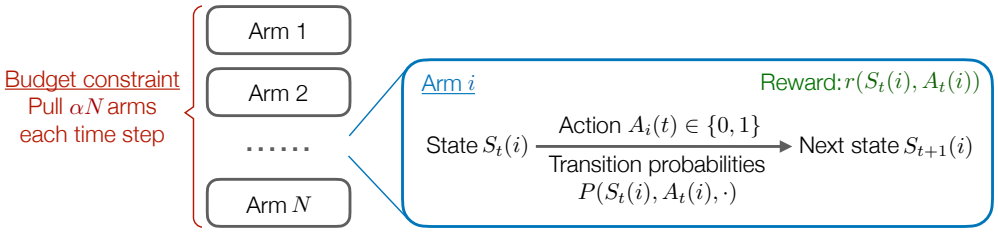


Figure 1: The restless bandit problem with N arms.

2 Restless bandits

2.1 Problem setup

We consider the discrete-time, infinite-horizon restless bandit problem with the average-reward criterion. The RB problem consists of N homogeneous arms and is henceforth referred to as the N-armed problem. Each arm is associated with an MDP called the single-armed MDP, which is defined by the tuple $(\mathbb{S}, \mathbb{A}, P, r)$. Here $\mathbb{S}$ is the state space, which is a finite set; $\mathbb{A} = \{0, 1\}$ is the action space, where the action 1 is interpreted as activating or pulling the arm; $P : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to [0, 1]$ is the transition kernel, where $P(s, a, s')$ is the probability of transitioning to state s' in the next time step conditioned on taking action a at state s in the current step; $r : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ is the reward function, where $r(s, a)$ is the expected reward for taking action a in state s. Let $r_{\max} = \max_{s\in\mathbb{S},a\in\mathbb{A}} |r(s, a)|$. The RB problem has a *budget constraint*, which requires that exactly αN arms must be pulled at every time step for some given constant $\alpha \in (0, 1)$. Here αN is assumed to be an integer for simplicity. We focus on the setting where all the model parameters, $\mathbb{S}, \mathbb{A}, P, r, \alpha$, are known. The problem is illustrated in Figure 1.

We index the arms in an N-armed bandit by $[N]$, where $[n] \triangleq \{1, 2, \ldots, n\}$. We refer to the index i of arm i as its *ID*, to avoid confusion with the Whittle index or other index notions.

A policy π for the N-armed problem chooses in each time step the action for each of the N arms, based on the current states of all arms and possibly an internal state maintained by the policy. All the policies considered in this paper have at most finitely many possible internal states.

Under a policy π, we use the *state vector* $\boldsymbol{S}_t^\pi \triangleq (S_t^\pi(i))_{i\in[N]} \in \mathbb{S}^N$ to represent the states of all arms, where $S_t^\pi(i) \in \mathbb{S}$ denotes the state of the i-th arm at time t. Similarly, the *action vector* is defined as $\boldsymbol{A}_t^\pi \triangleq (A_t^\pi(i))_{i\in[N]} \in \mathbb{A}^N$, where $A_t^\pi(i) \in \mathbb{A}$ denotes the action applied to the i-th arm at time t. We use $\boldsymbol{S}_\infty^\pi$ and $\boldsymbol{A}_\infty^\pi$ to denote random elements following the steady-state distributions of $\boldsymbol{S}_t^\pi$ and $\boldsymbol{A}_t^\pi$, respectively. Note that in later parts of the document, we will drop all the superscripts when the contexts are clear.

The objective of the RB problem is to find a policy that maximizes the long-run average of the expected reward from all N arms:

$$\underset{\text{policy } \pi}{\text{maximize}} \quad R(\pi, N) \triangleq \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \frac{1}{N} \sum_{i\in[N]} \mathbb{E}\left[r(S_t^\pi(i), A_t^\pi(i))\right] \tag{7}$$

$$\text{subject to} \quad \sum_{i\in[N]} A_t^\pi(i) = \alpha N, \quad \forall t \geq 0. \tag{8}$$

The objective can be equivalently written as $R(\pi, N) = \frac{1}{N} \sum_{i\in[N]} \mathbb{E}\left[r(S_\infty^\pi(i), A_\infty^\pi(i))\right]$. Let $R^*(N) \triangleq \sup_\pi R(\pi, N)$ denote the optimal value. The optimality gap of a policy π is defined as $R^*(N) - R(\pi, N)$. We say that a policy π is *asymptotically optimal* if its optimality gap vanishes as $N \to \infty$, i.e., $R^*(N) - R(\pi, N) = o(1)$.

Solving an exactly optimal policy for the RB problem is known to be PSPACE hard [122]. Therefore, in the rest of the section, we will focus on finding asymptotically optimal policies.

2.2 Related work

Restless bandits under infinite-horizon average-reward criterion. The seminal paper on restless bandits is the work by Whittle [167], who studied the restless bandits under infinite-horizon average-reward criterion and proposed the celebrated *Whittle index policy*. Whittle index policy is well-defined for RB instances that are *indexable*. For such RB instances, the asymptotic optimality for Whittle index policy was established in [161] under a *uniform global attractor property* (UGAP). Later, a more general class of policies called LP-Priority policies was proposed in [156]. LP-Priority policies are well-defined in general and they contain Whittle index policy as a special case for indexable RB problems. However, they still need UGAP to be asymptotically optimal.

More recent work on average-reward restless bandits studies the *rate* at which the optimality gap converges to zero. The work [59] and [60] prove a striking $O(\exp(-cN))$ optimality gap for the Whittle index policy and LP-Priority policies, respectively, where c is a constant. In addition to UGAP, these results require a non-singularity or non-degenerate condition.

As we can see from the review above, all the asymptotic optimality results in all prior work require the UGAP assumption. UGAP pertains to the mean-field/fluid limit of the restless bandit system in the asymptotic limit $N \to \infty$; it stipulates that the system's state distribution in the mean-field limit converges to the optimal state distribution attaining the maximum reward, from any initial distribution. It has been well recognized that UGAP is a highly technical assumption and challenging to verify: the primary way to test UGAP is numerical simulations. Moreover, there are documented RB instances where the Whittle-index and LP-priority policies fail to satisfy UGAP and are asymptotically suboptimal [161, 57, 83].

We finishing our review of infinite-horizon average-reward RB with a minor note: some of the papers discussed above actually study RB in the continuous-time setting. Specifically, [161, 156, 59] include results on the continuous-time setting, while [59, 60] include results on the discrete-time setting. Rigorously speaking, these two settings are not equivalent. However, the policies, results, and analysis in the two settings are similar in these previous papers, so we still view them as the same line of research.

Restless bandits under other reward criteria. There is a recent line of work on RBs with finite-horizon total-reward criteria. This line of work establish an $O(1/\sqrt{N})$ optimality gap [86, 171, 26, 62], and an $O(\exp(-cN))$ gap assuming non-degeneracy [172, 60]. Another line of work [173, 62] focus on the infinite-horizon discounted-reward setting, where they propose policies with $O(1/\sqrt{N})$ optimality gap without assuming indexability or UGAP.

Relationship with restful Markovian bandits. While the optimality of a general RB problem is intractable, there is a special case that has been solved optimally. Specifically, consider the case when an arm stays in the same state when it is not pulled, and only one arm is pulled at a time. The optimal policy for this special case is the celebrated Gittins index policy [67, 63]. A more recent reference on this topic is [64].

Relationship with other bandit problems. The RB problem falls within the broader class of bandit problems, for which different formulations exist including stochastic bandits, adversarial bandits, and Bayesian bandits. The common theme in these formulations is to find a reward-maximizing strategy of pulling arms in the presence of uncertainty in the arms' rewards; see the

book [95] for a comprehensive overview. Among these formulations, closely related to RBs is the Bayesian bandit problem, where Bayesian posteriors are used to model knowledge of unknown reward distributions. The Bayesian posterior can be seen as a state with known transition probabilities, hence the Bayesian bandit problem can be analyzed by applying tools from RBs. Examples demonstrating this connection can be found in [26].

2.3 Policy design through a simple system

In this section, we introduce several new policies for restless bandits developed in our two consecutive papers [83, 84]. Through these policies, we give an affirmative answer the following long-standing question: *Is it possible to efficiently find a policy that achieves asymptotic optimality in infinite-horizon, average-reward RBs under only standard unichain and aperiodicity assumptions, without imposing any additional conditions like UGAP?*

The section is organized as follows. In Section 2.3.1, we define a simple system using the so-called single-armed problem. In Section 2.3.2 and 2.3.3, we introduce the policies in each of the two papers, [83] and [84], where we demonstrate the idea of making the RB system mimic the behavior of the simple system to achieve asymptotic optimality.

2.3.1 Defining the simple system using the single-armed problem

Consider the *single-armed problem* that aims to optimize the long-run average reward in the single-armed MDP $(\mathbb{S}, \mathbb{A}, P, r)$ subject to a long-run average budget constraint. Formally,

$$\underset{\bar{\pi}}{\text{maximize}} \quad \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[r(S_t^{\bar{\pi}}, A_t^{\bar{\pi}})\right] \tag{9}$$

$$\text{subject to} \quad \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[A_t^{\bar{\pi}}\right] = \alpha. \tag{10}$$

Here $\bar{\pi} = (\bar{\pi}(a|s))_{s\in\mathbb{S}, a\in\mathbb{A}}$ denotes a randomized Markovian policy for the single-armed MDP, where $\bar{\pi}(a|s)$ is the probability of taking action a at state s; $S_t^{\bar{\pi}}$ and $A_t^{\bar{\pi}}$ denote the state and action of the arm at time t. The constraint (10) requires that the *average* rate of applying the active action is α.

The single-armed problem has been shown to be a relaxation of the N-armed problem (7)–(8) (see, e.g., [59]). Specifically, let R^{rel} be the optimal value of the single-armed problem and recall that $R^*(N)$ is the optimal value of the N-armed problem. Then we have $R^*(N) \le R^{\text{rel}}$. Consequently, the optimality gap of any N-armed policy π can be upper bounded as

$$R^*(N) - R(\pi, N) \le R^{\text{rel}} - R(\pi, N) \tag{11}$$

Therefore, to show that π is asymptotically optimal, it suffices to show that $R^{\text{rel}} - R(\pi, N) = o(1)$.

To understand the gap $R^{\text{rel}} - R(\pi, N)$, consider the simple system consisting of N independent copies of the single-armed system under any fixed optimal single-armed policy, $\bar{\pi}^*$, whose average steady-state-expected reward per arm is equal to R^{rel}. Then the gap $R^{\text{rel}} - R(\pi, N)$ can be viewed as the difference between the average reward of this simple system and the average reward of the N-armed RB system under the policy π.

In the next two sections, we will use the optimal single-armed policy $\bar{\pi}^*$ as a building block to construct RB policies, to match the dynamics of the RB system with the dynamics of the simple system defined above. This allows us to control $R^{\text{rel}} - R(\pi, N)$ and achieve asymptotic optimality.

To highlight the difference between our approach and the approach in prior work, we clarify that equivalent forms of the single-armed problem have been used in prior works to prove the asymptotic

Algorithm 1 FTVA($\bar{\pi}^*$)

Input: N-armed problem $(N, \mathbb{S}^N, \mathbb{A}^N, P, r, \alpha N)$, initial states $\boldsymbol{S}_0$, optimal single-armed policy $\bar{\pi}^*$
Initialize: Virtual states $\widehat{\boldsymbol{S}}_0$ are N i.i.d. samples following the stationary distribution of $\bar{\pi}^*$

for $t = 0, 1, 2, \ldots$ **do**
  Independently sample $\widehat{A}_t(i) \leftarrow \bar{\pi}^*(\cdot|\widehat{S}_t(i))$ for each arm $i \in [N]$ ▷ *Generate virtual actions*
  if $\sum_{i\in[N]} \widehat{A}_t(i) \geq \alpha N$ **then** ▷ *Select a set $\mathcal{A}$ of αN arms to activate*
    $\mathcal{A} \leftarrow$ a set of αN arms chosen from $\{i\colon \widehat{A}_t(i) = 1\}$ (any tie-breaking)
  else
    $\mathcal{B} \leftarrow$ a set of $\alpha N - \sum_{i\in[N]} \widehat{A}_t(i)$ arms chosen from $\{i\colon \widehat{A}_t(i) = 0\}$ (any tie-breaking)
    $\mathcal{A} \leftarrow \{i\colon \widehat{A}_t(i) = 1\} \cup \mathcal{B}$
  Apply $A_t(i) = 1$ and observe $S_{t+1}(i)$ for each arm $i \in \mathcal{A}$
  Apply $A_t(i) = 0$ and observe $S_{t+1}(i)$ for each arm $i \notin \mathcal{A}$
  for $i = 1, 2, 3, \ldots N$ **do** ▷ *Progress virtual processes*
    if $\widehat{S}_t(i) = S_t(i)$ and $\widehat{A}_t(i) = A_t(i)$ **then** ▷ *Couple virtual and real states*
      $\widehat{S}_{t+1}(i) \leftarrow S_{t+1}(i)$
    else
      Independently sample $\widehat{S}_{t+1}(i)$ from the distribution $P(\widehat{S}_t(i), \widehat{A}_t(i), \cdot)$

optimality of their policies [161, 156, 57, 58]. However, previous policies are constructed heuristically rather than based on the dynamics of the single-armed system. Consequently, these policies need the additional assumption, UGAP, to be asymptotic optimality.

2.3.2 Policy for the complex system: Follow-the-Virtual-Advice

In our first paper on restless bandits, [83], we propose `Follow-the-Virtual-Advice`, a simulation-based framework for converting a single-armed policy $\bar{\pi}$ into a N-armed policy, denoted as FTVA($\bar{\pi}$). In this section, we focus on FTVA($\bar{\pi}^*$), the N-armed policy converted from an optimal single-armed policy, $\bar{\pi}^*$.

The idea of FTVA($\bar{\pi}^*$) is to *explicitly simulate* the simple system that consists of N independent copies of the single-armed system under $\bar{\pi}^*$, and try to let the arms in the RB system take the same actions as the corresponding arms in the simple system.

Specifically, FTVA($\bar{\pi}^*$) has two main components:

- *Virtual single-armed processes.* Each arm i simulates a *virtual* single-armed process, whose state is denoted as $\widehat{S}_t(i)$, with action $\widehat{A}_t(i)$ chosen according to $\bar{\pi}$. To make the distinction conspicuous, we sometimes refer to the state $S_t(i)$ and action $A_t(i)$ in the original N-armed problem as the *real* state/action. The virtual processes associated with different arms are independent.

- *Follow the virtual actions.* At each time step t, we choose the real actions $A_t(i)$'s to best match the virtual actions $\widehat{A}_t(i)$'s, to the extent allowed by the budget constraint $\sum_{i\in[N]} A_t(i) = \alpha N$.

FTVA($\bar{\pi}^*$) is presented in detail in Algorithm 1. Note that we use an appropriate coupling in Algorithm 1 to ensure that the virtual processes $(\widehat{S}_t(i), \widehat{A}_t(i))$'s are independent and each follows the Markov chain induced by $\bar{\pi}^*$.

Algorithm 2 ID policy

Input: number of arms N, budget αN, an optimal single-armed policy $\bar{\pi}^*$, initial state vector $\boldsymbol{S}_0$

for $t = 0, 1, \ldots$ **do**
  Independently sample $\widehat{A}_t(i) \sim \bar{\pi}^*(\cdot|S_t(i))$ for $i \in [N]$ ▷ *Action sampling*
  if $\sum_{i\in[N]} \widehat{A}_t(i) \geq \alpha N$ **then** ▷ *Action rectification*
    $N_t^{\bar{\pi}^*} \leftarrow \max\{n \leq N\colon \sum_{i\in[n]} \widehat{A}_t(i) \leq \alpha N\}$
    $A_t(i) \leftarrow \widehat{A}_t(i)$ for $i \in [N_t^{\bar{\pi}^*}]$, $A_t(i) \leftarrow 0$ for $i \notin [N_t^{\bar{\pi}^*}]$
  else
    $N_t^{\bar{\pi}^*} \leftarrow \max\{n \leq N\colon \sum_{i\in[n]}(1 - \widehat{A}_t(i)) \leq (1-\alpha)N\}$
    $A_t(i) \leftarrow \widehat{A}_t(i)$ for $i \in [N_t^{\bar{\pi}^*}]$, $A_t(i) \leftarrow 1$ for $i \notin [N_t^{\bar{\pi}^*}]$
  Apply $A_t(i)$ for each arm $i \in [N]$ and observe $S_{t+1}(i)$

2.3.3 Policy for the complex system: ID policy

In our second paper on restless bandits, [84], we propose three policies: ID policy, set-expansion policy, and set-optimization policy. These three policies share similar ideas and are instances of a larger class of policies, focus-set policies. Here we focus on the simplest one, ID policy, which should be enough to illustrate the main idea of focus-set policies.

ID policy adopts a different idea from $\texttt{FTVA}(\bar{\pi}^*)$. In contrast to $\texttt{FTVA}(\bar{\pi}^*)$ which simulates virtual states $\widehat{S}_t(i)$'s and tries to follow the virtual actions sampled from these virtual states, ID policy directly samples some *ideal actions* based on the *real state* $S_t(i)$'s, and selects *a subset of arms* to follow the ideal actions. The selection of the subset is based on the IDs of the arms, prioritizing arms with smaller IDs. The detailed definition of ID policy is given in Algorithm 2.

ID policy matches the dynamics of the RB system with the simple system given in Section 2.3.1 in a very interesting way. Specifically, at any time step t, there is a *dynamic subset* D_t where the arms in the subset look like $|D_t|$ independent copies of the single-armed system under $\bar{\pi}^*$. This subset D_t is of the form $[n] \triangleq \{1, 2, \ldots, n\}$ and can be roughly seen as a subset of $[N_t^{\bar{\pi}^*}]$.

Although it seems that the RB system under ID policy only partially matches the dynamics of the RB with the dynamics of the simple system, we can show in the proof that D_t expands almost monotonically over time and covers most arms in the steady state. Dealing with the dynamic D_t to show asymptotic optimality is the most innovative part of ID policy's analysis, which we briefly discuss in Section 2.5.2.

2.4 Main results

In this section, we present the asymptotic optimality results for $\texttt{FTVA}(\bar{\pi}^*)$ [83] and ID policy [84].

Note that there are other results in [83] and [84] which we do not include in this document. Specifically, in [83], we bound the loss in long-run average reward after converting $\bar{\pi}$ to $\texttt{FTVA}(\bar{\pi})$ for any single-armed policy $\bar{\pi}$. We also consider continuous-time RB and propose a continuous-time analog of $\texttt{FTVA}(\bar{\pi}^*)$, which achieves asymptotic optimality under only the standard unichain condition. In [84], we prove that two other policies, set-expansion policy, and set-optimization policy, have similar optimality gap bounds as ID policy. Similar optimality gap bounds also hold more generally for all focus-set policies satisfying three certain conditions.

2.4.1 Asymptotic optimality result for Follow-the-Virtual-Advice

The asymptotic optimality result for `Follow-the-Virtual-Advice` relies on a new assumption named Synchronization Assumption(SA), which we state below. SA is more efficient to verify than UGAP and has some intuitive sufficient conditions. See [83] for a detailed discussion on this new assumption.

To state Synchronization Assumption, we first define a two-armed system called the *leader-and-follower* system, which consists of a *leader* arm and a *follower* arm following a given single-armed policy $\bar{\pi}$.

Definition 2.1 (Leader-and-follower system)**.** Consider a two-armed system, where each arm is associated with the MDP $(\mathbb{S}, \mathbb{A}, P, r)$. At each time step $t \geq 1$, the leader arm is in state $\widehat{S}_t$ and uses the policy $\bar{\pi}$ to chooses an action $\widehat{A}_t$ based on $\widehat{S}_t$; the follower arm is in state S_t, and it takes the action $A_t = \widehat{A}_t$ regardless of S_t. The state transitions of the two arms are coupled as follows. If $S_t = \widehat{S}_t$, then $S_{t+1} = \widehat{S}_{t+1}$. If $S_t \neq \widehat{S}_t$, then S_{t+1} and $\widehat{S}_{t+1}$ are sampled independently from $P(S_t, A_t, \cdot)$ and $P(\widehat{S}_t, \widehat{A}_t, \cdot)$, respectively. Note that once the states of the two arms become identical, they stay identical indefinitely.

Given the initial states and actions $(S_0, A_0, \widehat{S}_0, \widehat{A}_0) = (s, a, \widehat{s}, \widehat{a}) \in \mathbb{S} \times \mathbb{A} \times \mathbb{S} \times \mathbb{A}$, we define the *synchronization time* as the first time the two states become identical:

$$\tau^{\text{sync}}(s, a, \widehat{s}, \widehat{a}) \triangleq \min\{t \geq 0 \colon S_t = \widehat{S}_t\}. \tag{12}$$

Assumption 2.1 (Synchronization Assumption (SA) for a policy $\bar{\pi}$)**.** We say that a single-armed policy $\bar{\pi}$ satisfies the Synchronization Assumption (SA) if for any initial states and actions $(s, a, \widehat{s}, \widehat{a}) \in \mathbb{S} \times \mathbb{A} \times \mathbb{S} \times \mathbb{A}$, the synchronization time $\tau^{\text{sync}}(s, a, \widehat{s}, \widehat{a})$ is a stopping time and satisfies

$$\mathbb{E}\left[\tau^{\text{sync}}(s, a, \widehat{s}, \widehat{a})\right] < \infty. \tag{13}$$

Now we state Theorem 1 of [83], which shows that $\texttt{FTVA}(\bar{\pi}^*)$ achieves $O(1/\sqrt{N})$ optimality gap.

Theorem 2.1. *Consider an N-armed restless bandit problem with the single-armed MDP $(\mathbb{S}, \mathbb{A}, P, r)$ and budget αN for $0 < \alpha < 1$. Assume that the single-armed problem is unichain. Given any optimal single-armed policy $\bar{\pi}^*$ satisfying SA(Assumption 2.1), let policy π be $\texttt{FTVA}(\bar{\pi}^*)$ (Algorithm 1). For all any $N \geq 1$, the optimality gap of π is bounded as*

$$R^*(N) - R(\pi, N) \leq R^{\text{rel}} - R(\pi, N) \leq \frac{r_{\max} \overline{\tau}^{\text{sync}}_{\max}}{\sqrt{N}}, \tag{14}$$

where $r_{\max} \triangleq \max_{s \in \mathbb{S}, a \in \mathbb{A}} |r(s, a)|$ and $\overline{\tau}^{\text{sync}}_{\max} \triangleq \max_{(s,a,\widehat{s},\widehat{a}) \in \mathbb{S} \times \mathbb{A} \times \mathbb{S} \times \mathbb{A}} \mathbb{E}\left[\tau^{\text{sync}}(s, a, \widehat{s}, \widehat{a})\right]$.

2.4.2 Asymptotic optimality result for ID policy

For ID policy, we only need to assume the unichain and aperiodicity assumptions, stated below.

Assumption 2.2 (Unichain and aperiodicity)**.** The single-armed problem is unichain, i.e., any policy $\bar{\pi}$ of the single-armed problem induces a unichain $P_{\bar{\pi}}$ on $\mathbb{S}$. Moreover, any optimal policy $\bar{\pi}^*$ of the single-armed problem induces an aperiodic unichain $P_{\bar{\pi}^*}$ on $\mathbb{S}$.

Definition 2.2. Let W be an $|\mathbb{S}|$-by-$|\mathbb{S}|$ matrix given by

$$W = \sum_{k=0}^{\infty} (P_{\bar{\pi}^*} - \Xi)^k (P_{\bar{\pi}^*}^\top - \Xi^\top)^k, \tag{15}$$

where Ξ is an $|\mathbb{S}|$-by-$|\mathbb{S}|$ matrix with each row being μ^*. Let λ_W denote maximal eigenvalue of W.

Now we state Theorem 1 of [84], which gives an $O(1/\sqrt{N})$ bound for the optimality gap of ID policy. Remarkably, the bound is explicitly written out in terms of basic quantities like β, $r_{\max}$, $\mathbb{S}$, and λ_W, all of which are either given in the definition of the MDP or can be easily calculated from the single-armed Markov chain induced by $\bar{\pi}^*$.

Theorem 2.2 (Optimality gap of ID policy)**.** *Consider an N-armed restless bandit problem with the single-armed MDP $(\mathbb{S}, \mathbb{A}, P, r)$ and budget αN for $0 < \alpha < 1$. Assume that the single-armed problem is unichain and any optimal single-armed policy induces an aperiodic unichain (Assumption 2.2). Let π be ID policy (Algorithm 2). The optimality gap of π is bounded as*

$$R^*(N) - R(\pi, N) \le R^{\text{rel}} - R(\pi, N) \le \frac{672 r_{\max} \lambda_W^{5/2} |\mathbb{S}|^{3/2}}{\beta^3 \sqrt{N}}, \tag{16}$$

where $\beta = \min\{\alpha, 1-\alpha\}$, and λ_W is the largest eigenvalue of the matrix W (Definition 2.2).

2.5 Analysis of the policies

2.5.1 Follow-the-Virtual-Advice

Proof sketch for Theorem 2.1. Here we sketch the main ideas of the proof, whose key step involves bounding the gap loss $R^{\text{rel}} - R(\pi, N)$ using Little's Law [92]. Specifically, we start with the upper bound

$$\begin{aligned} R^{\text{rel}} - R(\pi, N) &= \frac{1}{N} \mathbb{E}\left[\sum_{i \in [N]} r\big(\widehat{S}_\infty(i), \widehat{A}_\infty(i)\big) - \sum_{i \in [N]} r\big(S_\infty(i), A_\infty(i)\big) \right] \\ &\le \frac{2 r_{\max}}{N} \mathbb{E}\left[\sum_{i \in [N]} \mathbb{1}\Big\{ \big(\widehat{S}_\infty(i), \widehat{A}_\infty(i)\big) \neq \big(S_\infty(i), A_\infty(i)\big) \Big\} \right], \end{aligned} \tag{17}$$

which holds since the virtual process $\big(\widehat{S}_t(i), \widehat{A}_t(i)\big)$ of each arm i follows the optimal single-armed policy $\bar{\pi}^*$. We say an arm i is a *bad arm* at time t if $\big(\widehat{S}_t(i), \widehat{A}_t(i)\big) \neq \big(S_t(i), A_t(i)\big)$, and a *good arm* otherwise. Then $\mathbb{E}\left[\sum_{i \in [N]} \mathbb{1}\Big\{ \big(\widehat{S}_\infty(i), \widehat{A}_\infty(i)\big) \neq \big(S_\infty(i), A_\infty(i)\big) \Big\}\right] = \mathbb{E}\big[\#\text{ bad arms}\big]$ in steady state.

By Little's Law, we have the following relationship:

$$\mathbb{E}\left[\#\text{ bad arms}\right] = (\text{rate of generating bad arms}) \times \mathbb{E}\left[\text{time duration of a bad arm}\right].$$

It suffices to bound the two terms on the right-hand side. Note that the virtual actions $\widehat{A}_t(i)$'s are i.i.d. with mean $\mathbb{E}[\widehat{A}_t(i)] = \alpha$; a standard concentration inequality shows that at most $\left| \sum_{i \in [N]} \widehat{A}_t(i) - \alpha N \right| \approx O(\sqrt{N})$ bad arms are generated per time slot. On the other hand, each bad arm stays bad until its real state becomes identical to its virtual state, which occurs in $O(1)$ time by virtue of SA.

2.5.2 ID policy

Detailed intuition for ID policy On a high level, at each time step, ID policy lets a subset of arms mimic the dynamics of i.i.d. copies the single-armed system under the optimal policy $\bar{\pi}^*$. The subset expands progressively and covers most of the arms in the N-armed system in the steady state, which allows the N-armed system to approach the optimal average reward of the single-armed system. Below we elaborate more on this intuition.

Consider the single-armed system under $\bar{\pi}^*$. Since the transition kernel $P_{\bar{\pi}^*}$ is an aperiodic unichain by Assumption 2.2, we know that starting from any initial distribution over $\mathbb{S}$, the state distribution of the Markov chain $P_{\bar{\pi}^*}$ converges to the steady-state distribution, denoted as μ^*.

Next, consider a subset of n arms, each taking actions according to $\bar{\pi}^*$. Given any set of initial states for these arms, the joint distribution of the states of these arms converges to n i.i.d. copies of μ^*. After reaching the steady state, the budget usage of each arm in any particular time step is a Bernoulli random variable with probability α, due to the budget constraint of the single-armed problem. Moreover, in the steady state, the total budget usage of the n arms concentrates around αn due to independence.

Finally, we consider the N-armed system under ID policy. Since the low-ID arms get priority in following the ideal actions, there should exist a n, such that the arms in $[n]$ can persistently follow $\bar{\pi}^*$ for a long time. Once these arms' state distributions converge to μ^*, their budget usage concentrates around αn, which leaves the budget to allow more arms to follow $\bar{\pi}^*$. This way, we progressively expand the subset of arms that can follow $\bar{\pi}^*$.

Notational preliminary for proof ideas Before describing the proof ideas, we introduce a useful quantity that is extensively used in the analysis. For each subset $D \subseteq [N]$, we define the *scaled state-count vector on* D as $X_t(D) = (X_t(D, s))_{s\in\mathbb{S}}$, where

$$X_t(D, s) = \frac{1}{N} \sum_{i\in D} \mathbb{1}\{S_t(i) = s\}.$$

Note that each entry of the vector $X_t(D)$ is the number of arms in D in a given state scaled by $1/N$. When $D = [N]$ is the set of all arms, we simply call $X_t([N])$ the *scaled state-count vector*.

Sometimes we view $X_t(D)$ as a vector-valued function of $D \subseteq [N]$. We refer to this function X_t as the *system state* at time t. The system state X_t contains the same information as the state vector $\boldsymbol{S}_t$ does; in particular, from X_t one can deduce the state of each arm.

Proof overview for Theorem 2.2 As previously mentioned in an intuitive discussion, to rigorously analyze ID policy, we need to characterize the dynamics of two processes that happen simultaneously: the expansion of the subset of arms that follow $\bar{\pi}^*$, and the convergence of state distribution for the arms on the subset. For the latter, we consider a class of functions of the system state x, $\{h(x, [n])\}_{n\in[N]}$. For each x and $[n]$, $h(x, [n])$ can be roughly seen as a weighted norm between $x([n])$ and $\frac{n}{N}\mu^*$, where the weight is carefully chosen based on the dynamics of the single-armed system under $\bar{\pi}^*$ such that

$$\mathbb{E}\big[h(X_1, [n]) \,\big|\, X_0 = x, A_0(i) \sim \bar{\pi}^*(\cdot|S_0(i))\, \forall i \in [n]\big] \le \rho_2 h(x, [n]) + \frac{K_{\text{drift}}}{\sqrt{N}}. \tag{18}$$

for some constants $\rho_2 \in (0, 1)$ and $K_{\text{drift}} > 0$. In other words, $h(X_t, [n])$ can witness the convergence of the scaled state-count vector $X_t([n])$ to $\frac{n}{N}\mu^*$ if all arms in $[n]$ follow $\bar{\pi}^*$ for a sufficiently long time.

To show that the subset of arms following $\bar{\pi}^*$ expand over time, we introduce another concept called *focus set*. The focus set for ID policy is a subset of the form $[n]$ dynamically chosen in each time step based on the states of the arms, denoted as D_t. The focus set is chosen to satisfy three conditions, informally described below:

- Majority conformity: all but an $O(1/\sqrt{N})$ fraction of arms in D_t follow $\bar{\pi}^*$.
- Almost non-shrinking: $D_t \backslash D_{t+1}$ contains no more than an $O(\sqrt{N})$ arms in expectation.

- Sufficient coverage: the fraction of arms not in D_t is bounded by a proportional of $h(X_t, D_t)$, the value of the subset Lyapunov function on the focus set.

The majority conformity condition implies that the subset Lyapunov function in the focus set converges towards a small neighborhood of zero, while the other two conditions capture the expansion of the focus set driven by the convergence of the subset Lyapunov function. For the precise definition of the focus set of ID policy and the proof that the focus set satisfies the three conditions, we refer the readers to [84] for details.

After defining the subset Lyapunov functions and the focus set, we are ready to prove Theorem 2.2. We conduct a Lyapunov drift analysis with the following Lyapunov function:

$$V(X_t, D_t) = h(X_t, D_t) + L_h(1 - m(D_t)), \tag{19}$$

where L_h is a Lipschitz constant of $h(x, [n])$ with respect to n/N, and $m(D_t) = |D_t|/N$ is the fraction of arms in D_t. By leveraging a relatively straightforward combination of the properties of subset Lyapunov functions and the focus set, we can show the following drift condition on $V(X_t, D_t)$:

$$\mathbb{E}\big[V(X_{t+1}, D_{t+1}) \,\big|\, X_t = x, D_t = D\big] \le \rho_1 V(x, D) + \frac{K_1}{\sqrt{N}}, \tag{20}$$

for some constants $\rho_1 \in (0, 1)$ and $K_1 > 0$. Then after taking expectation with $(x, D) \sim (X_t, D_t)$ and letting $t \to \infty$, we get

$$\mathbb{E}\left[V(X_\infty, D_\infty)\right] \le \frac{K_1}{\rho_1 \sqrt{N}}.$$

Moreover, we show that the optimality gap is bounded by the steady-state expectation of $\mathbb{E}\left[V(X_\infty, D_\infty)\right]$:

$$R^{\mathrm{rel}} - R(\pi, N) \le r_{\max} \left(\frac{1}{K_{\mathrm{dist}}} + \frac{2}{L_h} \right) \mathbb{E}\big[V(X_\infty, D_\infty)\big] + \frac{2 r_{\max} K_{\mathrm{conf}}}{\sqrt{N}}. \tag{21}$$

In this way, we can get the desired $O(1/\sqrt{N})$ bound on $R^{\mathrm{rel}} - R(\pi, N)$ as stated in Theorem 2.2.

3 Stochastic bin-packing with time-varying resource requirements

3.1 Prior work and motivation

In modern computing systems, a job often takes the form of a virtual machine (VM) or a container [30, 48]. Such a job comes with a resource requirement, such as a certain number of CPUs and amount of memory, while in service. Each server in the system offers a limited amount of these resources. When a job arrives at the system, the job dispatch policy needs to decide which server the job should be assigned to, given the job's resource requirement and servers' current job configurations. This job scheduling problem can be approached as a *Stochastic Bin-Packing (SBP) problem*, where jobs are viewed as items, job resource requirements as item sizes, and servers as bins. A traditional SBP setting considers a finite set of jobs that arrive online but do not depart from the system. The objective is to minimize the number of servers that have jobs on them, or 'non-empty bins', subject to the resource capacities of the servers. SBP, with a rich history in operations research and theoretical computer science [33, 32, 35], is a field of continuous developments and advancements [76, 50, 12].

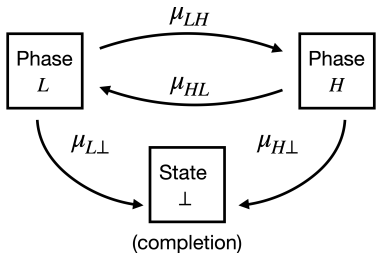
To incorporate job *departures* into the problem formulation, a setting referred to as *stochastic bin-packing in service systems* has been proposed recently [145, 147, 148, 146, 149, 61]. In this setting, jobs not only arrive but also depart over time. More specifically, jobs are assumed to arrive according to Poisson processes, and each job is assumed to stay in the system for an exponentially distributed service time. The service time of a job remains unknown until the job departs. Before delving further into SBP in service systems, it is worth mentioning that there is a parallel thread of research on the so-called dynamic bin-packing problem that also handles job departures (see, e.g., [31, 98, 27], and references therein), but it is primarily from a worst-case analysis perspective. Additionally, the virtual machine scheduling problem with objectives different from minimizing the number of active servers has also been widely studied (see, e.g., [110, 108, 111, 169, 127, 128, 129, 130]).

For SBP in service systems, the goal is to design a job dispatch policy σ that minimizes the expected number of active servers in *steady state*, denoted as $N(\sigma)$. The *optimality gap* of a policy σ is defined as $N(\sigma) - N(\sigma^*)$, where σ^* is the optimal policy. Since SBP in service systems aims to model today's large-scale computing systems, the optimality gap of a policy is usually studied in the regime where the total job arrival rate becomes large. As we scale up the total job arrival rate linearly with a scaling factor, r, the optimal value $N(\sigma^*)$ can be shown to be $\Theta(r)$. [1] Therefore, we say a policy is asymptotically optimal if its optimality gap is $o(r)$.

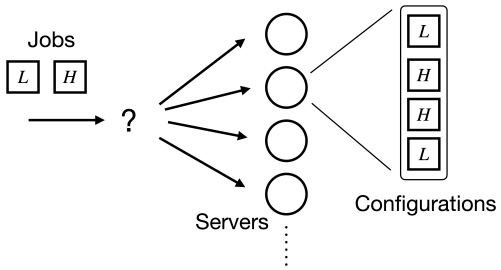
The optimality gap for SBP in service systems has been characterized in the line of work [145, 147, 148, 146, 149]. In particular, in [145] and [147], greedy policies are proposed and are proved to be asymptotically optimal, but the scheduler that executes these policies needs to know detailed state information, which is in a high-dimensional space. Later, policies that use much less state information are developed in [148] and [149], which achieve $\Theta(r)$ (with an arbitrarily small constant) and $o(r)$ optimality gaps, respectively.

While prior work on SBP in service systems has provided substantial insights into scheduling virtual-machine-type jobs, it primarily focuses on job resource requirements that remain fixed over time. However, in modern computing systems, jobs' resource requirements often vary over time [131, 41, 105, 151, 134, 14]. For example, when a job involves providing user-facing services, the instantaneous requirement on CPUs and memory depends on the service demand, which is subject to fluctuation over time [41, 105]. Time-varying job resource requirements pose significant challenges

[1] We use the standard Bachmann–Landau notation. Consider two functions $a(r)$ and $b(r)$, where $b(r)$ is positive for large enough r. Then $a = O(b)$ if $\limsup_{r\to+\infty} \frac{|a(r)|}{b(r)} < \infty$; $a = o(b)$ if $\lim_{r\to+\infty} \frac{a(r)}{b(r)} = 0$; $a = \Theta(b)$ if $a = O(b)$ and $b = O(a)$.



(a) A simplified version of our job model. Each job in service is in either an L phase or an H phase, associated with low and high resource requirements, respectively. When the job is completed, it is said to be in the state $\perp$. The job transitions between the two phases while in service until it is completed, following a continuous-time Markov chain with rates $\mu_{ii'}$, $i, i' \in \{L, H, \perp\}$.



(b) A system model with an infinite number of identical servers. As soon as a job arrives to the system, the job needs to be dispatched to a server to start service immediately. The configuration of each server is the number of jobs in each phase on the server.

Figure 2: Job model and system model.

in optimizing system efficiency, particularly when aiming to minimize the number of active servers, thereby improving server utilization. It is pertinent to note that low utilization has been recognized as a significant obstacle to the continued scaling of today's computing systems.

Motivated by this gap, in this paper, we propose a *new setting of stochastic bin-packing in service systems* that allows job resource requirements, or 'item sizes', to vary over time.

3.2 Problem formulation: a simplified version

We first describe our job model that features time-varying resource requirements. For ease of exposition, here we present a simplified setting where each job in service can be in one of the two *phases*, L and H, associated with *low* and *high* resource requirements, respectively. Our full model, presented in [85, Section 2] allows *more than two phases and more than one type of resources*. To model the temporal variation in the resource requirement, we assume that each job transitions between the two phases while in service until it is completed, following a continuous-time Markov chain illustrated in Figure 2a. We use an absorbing state $\perp$ to denote that the job is completed. A job can initialize in either phase L or phase H, and they are referred to as *type* L and *type* H jobs, respectively. Note that the setting where a job's resource requirement does not vary over time is a special case of our job model where the transition rates between phases are 0.

We consider a system with an infinite number of identical servers, illustrated in Figure 2b. We assume jobs arrive according to a Poisson process as existing work on SBP in service systems. In particular, we assume that the two types of jobs arrive at the system following two independent Poisson processes, with rates Λ_L and Λ_H, respectively; i.e., the interarrival times of type L and type H jobs are i.i.d. following exponential distributions with means $1/\Lambda_L$ and $1/\Lambda_H$, respectively. Upon arrival, a job needs to be dispatched to a server according to a *dispatch policy*, and the job enters service immediately. The goal is to design a policy σ to minimize the expected number of *active servers* (servers currently serving a positive number of jobs) in steady state, denoted as $N(\sigma)$.

As job resource requirements vary over time, situations can arise where the total job resource requirement on a server exceeds the server's resource capacity, resulting in *resource contention*. Modern computing systems can tolerate temporary overruns of resource capacity, though they often incur performance degradation or other costs [29, 49]. In our model, we incorporate a rate at which the cost accumulates due to resource contention. We first represent the state of a server by its

configuration, a vector $\boldsymbol{k} = (k_L, k_H)$ where k_L and k_H are the numbers of jobs in phase L and phase H, respectively. Then a *cost rate function* $h(\cdot)$ maps a server's configuration to a cost rate. We allow the cost rate function to be any function satisfying $h(\mathbf{0}) = 0$. In particular, it can be proportional to how much the total resource requirement of the jobs on the server exceeds this server's resource capacity. We assume that the resource contention does not affect the transition rates in the job model nor prompt jobs to be terminated, suitable for the application scenarios where the contention level is low and manageable. Let $C(\sigma)$ denote the ratio between the steady-state expected cost rate and the steady-state number of active servers.

Now our bin-packing problem can be formulated as follows:

$$\begin{aligned} \underset{\sigma}{\text{minimize}} \quad & N(\sigma) \\ \text{subject to} \quad & C(\sigma) \leq \epsilon, \end{aligned} \tag{22}$$

where $\epsilon > 0$ is a budget for the cost rate of resource contention. We denote the optimal value of this bin-packing problem as N^*.

We are interested in solving this problem in the asymptotic regime where the arrival rates (Λ_L, Λ_H) scale to infinity [147, 148, 146, 149], motivated by the ever-increasing computing demand that drives today's computing systems to be large-scale. Specifically, we assume $(\Lambda_L, \Lambda_H) = (\lambda_L r, \lambda_H r)$ for some fixed coefficients λ_L and λ_H and a scaling factor r, and we study the asymptotic regime where $r \to \infty$. We say a policy σ is *asymptotically optimal* if

$$N(\sigma) \leq \left(1 + O\left(r^{-0.5}\right)\right) \cdot N^* \tag{23}$$

$$C(\sigma) \leq \left(1 + O\left(r^{-0.5}\right)\right) \cdot \epsilon. \tag{24}$$

3.3 Policy design through a simple system

In this section, we construct an asymptotic optimal policy for our stochastic bin-packing problem set up in the last section. We break down the policy design into two steps: First, in Section 3.3.1, we define a simple system through a certain single-server problem. Then in Section 3.3.2, we construct a meta-policy called Join-Requesting-Server, which converts an optimal solution to the single-server problem to a policy in the original system to achieve asymptotic optimality.

3.3.1 Defining the simple system using the single-server problem

To facilitate proving asymptotic optimality, we want to define the simple system to be more "efficient" than original infinite-server system under any feasible policy. To do so, consider the following motivating question: suppose that our goal is to maximize the throughput of *one specific server* while keeping its steady-state expected cost rate of resource contention below ϵ, how should we send jobs to this server? Note that in this case, we can potentially send jobs to the server at almost an infinite rate when the scaling factor $r \to \infty$. However, the throughput of the server is limited by the phase-transition dynamics of the jobs and the structure of its cost rate function $h(\cdot)$.

To capture the throughput limit of one server under the steady-state expected cost-rate constraint, we define the following *single-server problem*, illustrated in Figure 3: There is an infinite supply of jobs of all types, so the server can start the service of any number of new jobs of any type at any time. We say the server *requests* a job from the infinite supply whenever it starts serving a new job. The job-requesting decisions are made by a *single-server policy*, denoted as $\bar{\sigma}$, under two constraints:

- The steady-state expected cost rate of resource contention is no more than ϵ.

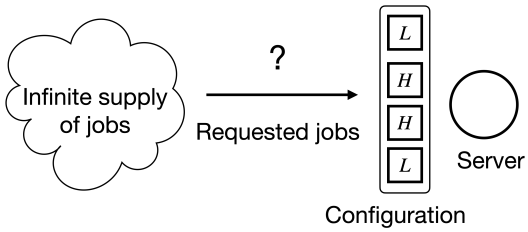


Figure 3: The single-server problem has an infinite supply of jobs. A single-server policy decides when to request jobs and how many jobs of each type to request.

- The long-run average rates of requesting each type of jobs, $(\bar{\lambda}_L, \bar{\lambda}_H)$, is proportional to the arrival rate vector $(\lambda_L r, \lambda_H r)$.

Let $\bar{N}$ be the positive number such that $(\bar{\lambda}_L \bar{N}, \bar{\lambda}_H \bar{N}) = (\lambda_L r, \lambda_H r)$. The objective of the single-server problem is to minimize $\bar{N}$. Let $\bar{N}^*$ be the optimal value of the single-server problem, and let $\bar{\sigma}^*$ be an optimal Markovian policy of the single-server problem.

Now we are ready to define the simple system. Consider the system with $\bar{N}^*$ servers [2], where each server has independent phase transitions and independently requests jobs using a optimal single-server policy $\bar{\sigma}^*$. Then in this system, the total rates of requesting each type of jobs is equal to $(\lambda_L r, \lambda_H r)$, and the steady-state expected cost rate per server is no more than ϵ. Moreover, since each server in this simple system requests jobs with the optimal single-server policy, and we can show that $N^* \geq \bar{N}^*$, i.e., the number of active servers in this simple system is a lower bound to our bin-packing problem (22) (Theorem 3.2).

In the next section, we will define a dispatching policy in the infinite-server system to mimic the dynamics of the simple system defined above. To see why this simple system serves as a reasonable target to mimic, we observe that the stochastic processes of when type i jobs arrive to the two systems are similar for each type i. Specifically, in the infinite-server system, the process is a *Poisson process with rate* $\lambda_i r$; in the simple system, the process is the *superposition of* $\bar{N}^*$ *independent point processes*, each corresponds to the job requests from one server and has the average rate $\bar{\lambda}_i = \lambda_i r / \bar{N}^*$. Intuitively, when r and $\bar{N}^*$ are large, these two processes should be close due to concentration of independent random variables. Therefore, intuitively, there should exists a dispatching policy in the infinite-server system that make the dynamics of the first $\bar{N}^*$ approximate the dynamics of the simple system.

3.3.2 Policy design in the complex system: Join-Requesting-Server

In this section, we define a dispatching policy that makes the infinite-server system mimic the simple system defined in the last section. This is done through a meta-policy named Join-Requesting-Server (JRS), which converts any single-server policy $\bar{\sigma}$ to a dispatch policy in the infinite-server system. We say that JRS takes $\bar{\sigma}$ as a subroutine. The subroutine of particular interest is the optimal single-server policy $\bar{\sigma}^*$, which will be converted to an asymptotically optimal dispatching policy. We will fix the subroutine as $\bar{\sigma}^*$ when we describe JRS in this section.

The basic idea of JRS is simple: Under JRS, each of the first $\bar{N}^*$ servers requests jobs using $\bar{\sigma}^*$, and each request for a type i job generates a *type i token*. When a type i job arrives, the dispatcher sends the job to any server with a type i token and removes the token; if there is no type i token in the system, the job goes to any server outside the first $\bar{N}^*$ servers. In this way, if each

[2] Because $\bar{N}^* \to \infty$ as $r \to \infty$, without loss of generality, we assume $\bar{N}^*$ to be an integer.

token is matched with a job soon after it is generated, the dynamics of the first $\bar{N}^*$ servers in the infinite-server system approximates the dynamics of the simple system.

Next, we state the definition of JRS. For the ease of presentation, we state a slightly simplified version under the assumption that $\bar{\sigma}^*$ induces a unichain in the single-server problem. The general definition of JRS can be found in [85, Appendix B.3].

Definition of Join-Requesting-Server The inputs of JRS include: (i) the optimal single-server policy $\bar{\sigma}^*$, (ii) the optimal value of the single-server problem $\bar{N}^*$, and (iii) the transition rates of the job model.

We first divide the servers into two sets based on the server's index ℓ. We call servers with index $\ell \leq \bar{N}^*$ *normal servers*; we call servers with index $\ell > \bar{N}^*$ *backup servers*.

The JRS is specified in two steps.

- **Step 1 (Job Requesting on a Normal Server):** We let each normal server request jobs using its $\bar{\sigma}^*$. The input to the policy $\bar{\sigma}^*$ is what we refer to as the *observed configuration* of the server, which will be further explained below. When $\bar{\sigma}^*$ requests $\boldsymbol{a} = (a_i)_{i\in\mathcal{I}}$ jobs, a_i type i *tokens* are generated for each $i \in \mathcal{I}$ to store the job requests. The server *pauses* the job requesting process if it already has any type of tokens, and resumes when all the tokens that it generated are removed.
- **Step 2 (Arrival Dispatching):**
  - **Real jobs.** When a type i job arrives, the dispatcher chooses a type i token uniformly at random, removes the token, and assigns the job to the corresponding server. When there are no type i tokens, the dispatcher sends the job to an idle backup server.
  - **Virtual jobs.** When the total number of type i tokens throughout the system exceeds the *token limit*, $\eta_{\max} = \lceil (\bar{N}^*)^{1/2} \rceil$, a type i *virtual arrival* is triggered, which causes the dispatcher to choose a type i token uniformly at random, remove the token, and assign a *virtual job* to the corresponding server. A virtual job has the same transition dynamics as a real job but does not consume physical resources.

The *observed configuration* of a normal server in Step 1 is the configuration resulting from real jobs and virtual jobs combined. That is, it is a vector whose i-th entry represents the total number of real and virtual jobs in phase i on this server. The observed configuration changes when there is a new real or virtual job arrival assigned to the server, or when a real or virtual job on the server has a phase transition or departs. We update the input to $\bar{\sigma}^*$ when the observed configuration changes.

Discussion on the detail of JRS. Note that rather than matching all tokens with job arrivals, JRS opts to convert some of the tokens into virtual jobs to keep the total number of tokens within an upper limit $\eta_{\max}$. By capping the number of tokens, JRS ensures that the job requests generated by each server get fulfilled quickly (either by a real job or a virtual job), and thus the dynamics of the observed configurations of the normal servers maintain proximity to the dynamics of the simple system that we want to match with.

The choice of the token limit $\eta_{\max} = \lceil (\bar{N}^*)^{1/2} \rceil$ balances two key considerations. On the one hand, a smaller $\eta_{\max}$ brings the observed configurations closer to the simple system. On the other hand, if $\eta_{\max}$ is overly small, the rate of generating virtual jobs becomes high and the probability for a job arrival to see no tokens is also high. As a result, the observed configurations, which include both real and virtual jobs, deviate from the real-job configurations.

See [85, Section 5.5] for a more in-depth discussion on the role of tokens and virtual jobs and whether they are fundamental.

3.4 Main result

In our main result is stated below as Theorem 3.1, where we show that JRS with the subroutine $\bar{\sigma}^*$, defined in Section 3.3.2, is asymptotically optimal. Theorem 3.1 corresponds to Theorem 1 in our full paper [85].

Theorem 3.1 (Asymptotic Optimality)**.** *Consider a stochastic bin-packing problem in service systems with time-varying job resource requirements stated in (22), with arrival rates $(\lambda_L r, \lambda_H r)$ and the cost rate budget $\epsilon > 0$. Let N^* be the optimal value of the problem. Let the policy σ be* Join-Requesting-Server *(JRS) with the subroutine $\bar{\sigma}^*$, where $\bar{\sigma}^*$ is an optimal single-armed policy. Then*

$$N(\sigma) \leq \left(1 + O\left(r^{-0.5}\right)\right) \cdot N^* \tag{25}$$

$$C(\sigma) \leq \left(1 + O\left(r^{-0.5}\right)\right) \cdot \epsilon. \tag{26}$$

3.5 Proof roadmap

Now we outline the main steps of our analysis.

In Theorem 3.2 below, we prove a lower bound to the bin-packing problem. Theorem 3.2 corresponds to Theorem 2 in our full paper [85].

Theorem 3.2 (Lower Bound)**.** *Consider a stochastic bin-packing problem in service systems with time-varying job resource requirements stated in (22), with arrival rates $(\lambda_L r, \lambda_H r)$ and the cost rate budget $\epsilon > 0$. Let N^* be the optimal value of the problem. Let $\bar{N}^*$ be the optimal value of the corresponding single-server problem. Then $N^* \geq \bar{N}^*$.*

Next, we show that the meta-policy JRS preserves the performance when converting $\bar{\sigma}^*$ to a policy in the infinite-server system, as stated in Theorem 3.3 below. Theorem 3.3 corresponds to Theorem 3 in our full paper [85].

Theorem 3.3 (Conversion Theorem)**.** *Consider a stochastic bin-packing problem in service systems with time-varying job resource requirements stated in (22), with arrival rates $(\lambda_i r)_{i \in \mathcal{I}}$ and the cost rate budget $\epsilon > 0$. Let $\bar{\sigma}^*$ be an optimal single-server policy with objective value $\bar{N}^*$. Let the infinite-server policy σ be JRS with the subroutine $\bar{\sigma}^*$. Then under σ, we have*

$$N(\sigma) \leq \left(1 + O\left(r^{-0.5}\right)\right) \cdot \bar{N}^*, \tag{27}$$

$$C(\sigma) \leq \left(1 + O\left(r^{-0.5}\right)\right) \cdot \epsilon. \tag{28}$$

Theorem 3.2 and 3.3 together imply Theorem 3.1. In addition, we show that an optimal single-server policy $\bar{\sigma}^*$ can be efficiently solved from a linear program, as stated in the Theorem 3.4 below. Theorem 3.4 corresponds to Theorem 4 in our full paper [85].

Theorem 3.4 (Solving the single-server problem, Informal)**.** *The single-server problem defined in Section 3.3.1 is equivalent to a linear program whose complexity is independent of the scaling factor r. Specifically, the optimal solution of the linear program can be used to construct an optimal policy of the single-server problem.*

The most interesting step in the analysis is the proof of Theorem 3.3, where we compare the infinite-server system under JRS with subroutine $\bar{\sigma}^*$ (the complex system) with the system where $\bar{N}^*$ servers independently request jobs using $\bar{\sigma}^*$ (the simple system). In the proof, we bound the Wasserstein distance between the steady-state distributions of the server-configurations in these two systems. We utilize some nice structures of JRS with a combination of multiple techniques, including Stein's method, Lyapunov drift analysis, and Little's law. Interested readers can see the proof in Section 5 of our full paper [85].

4 Optimal scheduling in G/G/k/setup

4.1 Background and motivation

We consider the classic problem of preemptively scheduling jobs in a queue to minimize mean number-in-system, or equivalently mean response time (a.k.a. sojourn time). Even in single-server queueing models, this can be a nontrivial problem whose answer depends on the information available to the scheduler. The simplest case is when the scheduler knows each job's size (a.k.a. service time), for which the optimal policy is Shortest Remaining Processing Time (SRPT) [135]: always serve the job of least remaining work.

In the more realistic case of scheduling with unknown or partially known job sizes, the optimal policy is only known for the M/G/1. It is called the *Gittins* policy (a.k.a. Gittins index policy) [1, 2, 65, 139]. Based on whatever service time information is available for each job, Gittins assigns each job a scalar *rank* (i.e. priority), then serves the job of least rank. For example, SRPT is the special case of Gittins where job sizes are known exactly, and a job's rank is its remaining work. More generally, a job's rank is, roughly speaking, an estimate of its remaining work based on whatever information is available.

The Gittins policy is known to be optimal in the M/G/1 [65, 139]. But plenty of systems and models have more complex features, including:

(a) *Multiple servers*, such as the M/G/k.

(b) *Non-Poisson arrival processes*, such as the G/G/1 (more specifically, the GI/GI/1).

(c) *Periods of server unavailability*, such as models with setup times.

Either (a) or (b) alone makes optimal scheduling intractable. Combining all three, as in the G/G/k with setup times (G/G/k/setup), only adds to the challenge.

With optimality out of reach, we are left to find a tractable near-optimal policy. We thus ask:

> How well does Gittins perform in systems with features (a), (b), and (c) like the G/G/k/setup?

Gittins policy is a natural candidate because its definition naturally generalizes beyond the M/G/1, even if its optimality proof does not [65]. For instance, in a G/G/k, Gittins policy simply serves the k jobs of k least ranks, or all jobs if there are fewer than k.

Only feature (a) has been addressed in full generality in prior work [137, 74, 136], where Gittins policy is proved to be heavy-traffic optimal as the load $\rho \to 1$, with explicit bounds on the optimality gap. Analyzing Gittins in models with features (b) and (c) has been open and cannot be directly done using existing techniques.

4.2 Related work

Gittins policy in single-server models The Gittins policy was originally conceived to solve the Markovian multi-armed bandit problem [65, 66], but it was soon adapted to also solve the problem of scheduling in an M/G/1 to minimize mean number of jobs and similar metrics. See [139] and the references therein for a review of Gittins in the M/G/1. However, aside from some particular cases [135, 132], the degree to which Gittins performs well in the G/G/1 or G/G/1/setup was previously unknown.

The SOAP technique of [140] can be used to analyze the performance of the Gittins policy in the M/G/1. However, while SOAP is convenient for analyzing any fixed size distribution (e.g.

numerically), using it to prove theorems that hold for all size distributions is cumbersome [141, Section 1.1]. Moreover, SOAP is limited to the M/G/1 and, thanks to an extension by [155], the M/G/1/setup. Analyzing Gittins with G/G arrivals or multiple servers seems to be beyond SOAP [138, Appendix B].

Gittins policy in multiserver models Gittins is known to be suboptimal with multiple servers [65], but researchers have studied the extent to which the optimality gap is large or small. The earliest results of this type analyzed an M/M/k with Bernoulli feedback [70] and *nonpreemptive* M/G/k with Bernoulli feedback [68]. These results proved (in the latter case, under an additional assumption) constant optimality gaps for Gittins in these systems. But both models are somewhat restrictive, excluding, for instance, heavy-tailed job size distributions that are common in computer systems [79, 78, 34, 123, 125]. More recent work, which we discussed in Section 1, overcomes these limitations to bound the performance of Gittins in the M/G/k for general job sizes, including heavy-tailed sizes [142, 74, 136]. However, all of the above work assumes M/G arrivals with no server unavailability.

Setup times in single-server models Single-server models with setup times has been extensively studied and are relatively well-understood [162, 42, 17, 28, 80, 96, 114, 52, 51, 43, 152]. See [43] for a survey of the work before 1986 and [152] for a more recent survey. These works consider various arrival and service processes, as well as other types of server unavailability in addition to setup times.

However, none of the above works discuss optimal scheduling in the presence of setup times. Progress was made by [155], who obtains the mean response time of Gittins in the M/G/1/setup as a special case of a more general analysis. But the analysis does not show that other policies might outperform Gittins, nor does it apply to the G/G/1/setup.

Setup times in multiserver models Compared with single-server models, multiserver models with setup times are less well-understood. A significant line of previous work has studied the M/M/k/setup with exponential setup times and FCFS scheduling [7, 54, 56, 55, 53, 124]. Among those works, [54] and [55] also demonstrate that their results generalize to M/G/k/setup with exponential setup times via simulation or analyzing special examples. Recently, [168] go beyond exponential setup times, studying M/M/k/setup with deterministic setup times and FCFS scheduling. However, none of these prior works apply to general setup times, non-Poisson arrivals, or scheduling policies beyond FCFS.

We note that [68], who studies the nonpreemptive M/G/k with Bernoulli feedback, actually studies a more general model that allows for certain types of server unavailability, such as server breakdowns. However, setup times are not covered by [68]. It is likely that more general future work could simultaneously cover setup times, server breakdowns, and other types of server unavailability. See [81, Section 10.3] for an additional discussion.

Work decomposition law An important ingredient of our analysis is a new work decomposition law, which we explain in Section 4.5. There is a long tradition of proving work decomposition laws for queueing systems [51, 52, 18, 114, 70, 68, 137, 136, 44]. Most of these laws take the form

$$\mathbb{E}\left[\text{work in complex system with M/G arrivals}\right] = \mathbb{E}\left[\text{work in M/G/1}\right] + \mathbb{E}\left[\text{extra work due to complexity}\right].$$

For example, if the complex system is an M/G/1/setup, the extra work from complexity depends on the setup time distribution. Most work decomposition laws are actually even stronger, holding *distributionally* instead of just in expectation.

We need a work decomposition law where the complexity includes, among other factors, having multiple servers. Such a result for M/G arrivals is relatively recent [137, 136], and no such result exists for G/G arrivals. While there are work decomposition laws for G/G arrivals in the literature [42, 44, 114], to the best of our knowledge, they apply only to single-server models with vacations. To the best of our knowledge, we prove the first work decomposition law for G/G arrivals that holds for multiserver systems like the G/G/k.

4.3 Model

4.3.1 Core queueing models: G/G/k, G/G/1, M/G/k, and M/G/1

We consider a G/G/k queueing model with a single central queue and k identical servers. The system experiences *G/G arrivals*: jobs arrive one-by-one with i.i.d. *interarrival times*, and each job has an i.i.d. *size*, or service requirement. Interarrival times and job sizes are independent of each other. We denote a generic random interarrival time by A and a generic random job size by S.[3]

At any moment of time, a job in the system can be served by one server. Any jobs not in service wait in the queue. Once a job's service is finished, it departs. We follow the convention that each of the k servers has service rate $1/k$. A job of size S thus requires kS time in service to finish. This convention gives all systems we study the same maximum total service rate, namely $k \cdot 1/k = 1$, and thereby the same stability condition.

The name "G/G/k" denotes the fact that the system has G/G arrivals and k servers. When A is exponentially distributed, we write M/G in place of G/G, as in "M/G/k".

Scheduling policies The scheduling policy decides, at every moment in time, which job is in service at which server. We consider a preempt-resume model where preemption occurs without delay or loss of work.

The scheduling objective is minimizing the *mean number of jobs* in the system. We denote the mean number of jobs in system SYS under scheduling policy π by $\mathbb{E}[N]^{\pi}_{\text{SYS}}$, omitting the "SYS" and/or "π" if there is no ambiguity. By Little's law [92], minimizing $\mathbb{E}[N]$ is equivalent to minimizing *mean response time*, the average amount of time a job spends between its arrival and departure.

We assume that the job sizes are unknown to the scheduler, so the scheduler makes decisions based on only the amount of service each job has received, which we refer to the job's state and denote as x. In our full paper [81], we consider a more general job model where the scheduler could have partial size information.

We restrict attention to *non-idling* policies, which are those that never unnecessarily leave servers idle. Nevertheless, our results have implications even for idling policies.

Load and stability We write $\lambda = 1/\mathbb{E}[A]$ for the average arrival rate and $\rho = \lambda\mathbb{E}[S]$ for the system's *load*, or utilization. One can think of ρ as the average fraction of servers that are busy. It is clear that $\rho < 1$ is a necessary condition for stability (unless both A and S are deterministic), so we assume this throughout.

Some of our results are stated for the *heavy-traffic limit*. For our purposes, this limit, denoted $\rho \to 1$, refers to a limit as the job size distribution S remains constant, and the interarrival time

[3] This arrival process is often referred to more specifically as GI/GI arrivals, with the "I" emphasizing the independence assumption. Under this convention, G/G arrivals include even more general stationary arrival processes where independence does not hold. In this work, we focus only on the independent case, so we write simply "G/G" instead of "GI/GI" for brevity.

distribution A is scaled uniformly down with its mean approaching the mean job size. That is, the system with load ρ has interarrival time $A_\rho = A_1/\rho$ for some fixed distribution A_1, where $\mathbb{E}[A_1] = \mathbb{E}[S]$.

It seems intuitive that $\rho < 1$ should be sufficient for stability under non-idling policies, and it is in the G/G/1 [107]. But to the best of knowledge, there are no results characterizing stability of the G/G/k under complex scheduling policies. Even under FCFS, proving stability of the G/G/k is not simple, because the system can be stable even when it never empties [90, 165, 143, 118]. Setup times further complicate the matter.

We consider the question of proving stability of the G/G/k/setup under arbitrary non-idling scheduling policies to be outside the scope of this work, so we simply assume stability when $\rho < 1$. We expect this is indeed the case, giving the initial steps of a proof sketch in the appendix of [81].

Assumption 4.1. For all $\rho < 1$, the G/G/k/setup is stable under all non-idling scheduling policies, including the Gittins policy.

Additional assumption on interarrival times Our results for G/G arrivals depend on "how non-Poisson" arrival times are, which we quantify using the following assumption.

Assumption 4.2. There exist $A_{\min}, A_{\max} \in \mathbb{R}_{\geq 0}$ such that $\mathbb{E}[A - a \mid A > a] \in [A_{\min}, A_{\max}]$ for all $a \geq 0$. That is, letting the *interarrival age* A_{age} be the time since the last arrival and *residual interarrival time* A_{res} be the amount of time until the next arrival, we have

$$\mathbb{E}[A_{\text{res}} \mid A_{\text{age}}] \in [A_{\min}, A_{\max}] \quad \text{with probability 1.}$$

One may always use $A_{\min} = \inf_{a \geq 0} \mathbb{E}[A - a \mid A > a]$ and $A_{\max} = \sup_{a \geq 0} \mathbb{E}[A - a \mid A > a]$, so this assumption boils down to the latter being finite.

Our results use Assumption 4.2 via the quantity $\lambda(A_{\max} - A_{\min})$, which we can think of as measuring "how non-Poisson" arrival times are. In the Poisson case, one may use $A_{\min} = A_{\max} = 1/\lambda$, so $\lambda(A_{\max} - A_{\min}) = 0$.

Many interarrival distributions A satisfy Assumption 4.2, such as all phase-type distributions. One can also think of Assumption 4.2 as a relaxation of the well-known *New Better than Used in Expectation* (NBUE) property, which is the special case where $A_{\max} = \mathbb{E}[A]$. The main distributions ruled out by Assumption 4.2 are various classes of heavy-tailed distributions, e.g. power-law tails.

4.3.2 Setup times

In addition to the basic G/G/k model defined above, we also consider models in which servers require *setup times* to transition from idle to busy. We denote these models with an extra "/setup", as in G/G/k/setup. Whenever a server switches from idle to busy, it must first complete an i.i.d. amount of *setup work*, denoted U. Like work from jobs, servers complete setup work at rate $1/k$, so setup *work* U results in setup *time* kU. Setup work amounts are independent of interarrival times and job sizes.

For the purposes of stating our results and proofs in a unified manner, we consider the G/G/k without setup times to be the special case of the G/G/k/setup where $U = 0$ with probability 1.

In our model, a server can be in one of three states:

- *Setting up*, i.e. doing setup work.
- *Busy*, i.e. serving a job.

- *Idle*, i.e. neither serving a job nor doing setup work.

In the G/G/k/setup, a server transitions

- from setting up to busy when it finishes its setup work,
- from busy to idle when the system has fewer jobs than busy servers, and
- from idle to setting up when the system has fewer busy or setting up servers than jobs.

Note that once a setup time begins, it is never canceled, even if the job whose arrival triggered the setup time begins service at another server. Unless another job arrives during the setup time, the server will transition from setting up to busy, then immediately back to idle.

4.3.3 Gittins policy

The scheduling policy we focus on in this work is the *Gittins* policy (a.k.a. *Gittins index* policy). Gittins is primarily known for the fact that it minimizes $\mathbb{E}[N]$ in the M/G/1 [65, 139]. In formulas, we abbreviate Gittins to "Gtn", as in $\mathbb{E}[N]^{\text{Gtn}}_{G/G/k/setup}$.

The Gittins policy assigns each job a numerical priority, called a *rank*, where lower rank is better. Gittins always serves the job or jobs of least rank,[4] and it is non-idling, serving as many jobs as the number of available servers allows. Gittins policy determines ranks using a *rank function* [65]

$$\text{rank}_{\text{Gtn}}(x) = \inf_{y>x} \frac{\mathbb{E}[\min\{S, y\} - x \mid S > x]}{\mathbb{P}(S \le y \mid S > x)},$$

assigning $\text{rank}_{\text{Gtn}}(x)$ to a job in state $x \in \mathbb{X}$.

4.4 Main results

In this section, we present our main results: the optimality gap bounds and the asymptotic optimality of Gittins policy in G/G/k/setup.

All of our results hold under the assumptions of Section 4.3, and in particular Assumption 4.1 and 4.2. As in Section 1, we can view a G/G/k/setup system, or any special case thereof, by whether it has (a) multiple servers, (b) non-Poisson arrivals, and (c) setup times. Our bounds use the quantities

$$\begin{aligned}
\ell_{(\text{a})} &= C(k-1)\log\frac{1}{1-\rho},\\
\ell_{(\text{b})} &= \lambda(A_{\max} - A_{\min}),\\
\ell_{(\text{c})} &= \mathbb{1}\{\mathbb{P}(U>0)>0\}\left(2(k-1) + \lambda(A_{\max} + k\mathbb{E}[U_{\text{e}}])\right),
\end{aligned}$$

where $C = \frac{9}{8\log 1.5} + 1 \approx 3.775$, U_e denotes the excess distribution of the setup work U. The idea is that $\ell_{(\text{a})}$ is the loss due to feature (a), as it is nonzero only for systems with $k \ge 2$ servers, and similarly for $\ell_{(\text{b})}$ and $\ell_{(\text{c})}$.

Theorem 4.1. *The performance gap between the Gittins policy in G/G/k/setup and the optimal policy in G/G/1 is bounded by*

$$\mathbb{E}[N]^{\text{Gtn}}_{G/G/k/setup} - \inf_{\pi} \mathbb{E}[N]^{\pi}_{G/G/1} \le \ell_{(a)} + \ell_{(b)} + \ell_{(c)}.$$

[4] Much literature on the Gittins policy uses the opposite convention, where higher numbers are better. These works typically call a job's priority its *index* [65, 1, 2], which is the reciprocal of its rank [139].

Note that although Theorem 4.1 is not directly about the optimality gap of Gittins policy in G/G/k/setup, it still provides an upper bound on the optimality gap, because the optimal performance of G/G/1 is a lower bound to G/G/k/setup. This is because servers in G/G/k/setup have speed $1/k$ (Section 4.3.1), so the G/G/1 can mimic any policy in the G/G/k/setup through processor sharing and idling.

With that said, in the special case of the non-idling G/G/1/setup, we can prove a stronger result that drops the $\ell_{(c)}$ term by comparing to a G/G/1/setup instead of a G/G/1.

Theorem 4.2. *In the G/G/1/setup, the performance gap between the Gittins policy and the optimal non-idling policy is bounded by*

$$\mathbb{E}[N]^{\text{Gtn}}_{G/G/1/setup} - \inf_{\pi} \mathbb{E}[N]^{\pi}_{G/G/1/setup} \leq \ell_{(b)}.$$

In particular, in the M/G/1/setup, the Gittins policy minimizes $\mathbb{E}[N]$ among non-idling policies.

The optimality gap in Theorem 4.1 is constant when $k = 1$ and $O\big(\log \frac{1}{1-\rho}\big)$ when $k \geq 2$. In both cases, the gap grows more slowly in the $\rho \to 1$ limit than $\mathbb{E}[N]^{\pi}_{G/G/1}$, implying heavy-traffic optimality.

Theorem 4.3. *In the G/G/k/setup, if either $k = 1$ or $\mathbb{E}\left[S^2(\log S)^+\right] < \infty$, and if either S or A is not deterministic, the Gittins policy is heavy-traffic optimal. Specifically,*

$$\lim_{\rho \to 1} \frac{\mathbb{E}[N]^{\text{Gtn}}_{G/G/k/setup}}{\inf_{\pi} \mathbb{E}[N]^{\pi}_{G/G/1}} = 1.$$

4.5 Proof overview and technical contributions

In this section, we give an overview of the proofs of our main results: bounds on Gittins's optimality gap (Theorems 4.1 and 4.2) and Gittins's heavy-traffic optimality (Theorem 4.3). Note that each of these results had already been stated in the form of a comparison between a complex system (G/G/k/setup) and a simpler system (G/G/1 or G/G/1/setup). Next, we outline how we prove these bounds that involve the comparisons between complex and simple systems.

Our proofs work by combining two queueing identities: *Work Integral Number Equality (WINE)*, which is from prior work; and a novel *work decomposition law*, which is built on similar decomposition results from prior work (Section 4.2).

The first tool, WINE, expresses the mean number-in-system in terms of *mean r-work* $\mathbb{E}[W_r]$ [136, 137, 139]:

$$\mathbb{E}[N] = \int_0^{\infty} \frac{\mathbb{E}[W_r]}{r^2} \, \mathrm{d}r.$$

A system's r-work, W_r, is the total service required to serve all jobs in the system until they all either complete or reach a rank greater than r, as determined by rank_{Gtn}. For example, ∞-work is the total remaining work of all jobs, which we call *total work* or simply *work*.

The second tool, the work decomposition law, bounds the difference in $\mathbb{E}[W_r]$ between G/G/k/setup under Gittins, and G/G/1 (or G/G/1/setup) under any policy. Combining this with WINE yields bounds on $\mathbb{E}[N]$. Our proof thus boils down to three steps:

- Proving the work decomposition law (Section 4.5.1).
- Using the work decomposition law to bound Gittins's optimality gap (Section 4.5.2).
- Using the optimality gap bounds to show Gittins is heavy-traffic optimal (Section 4.5.3).

4.5.1 New tool: work decomposition law for G/G arrivals

Our work decomposition law characterizes mean r-work $\mathbb{E}[W_r]$ in the G/G/k/setup. For simplicity of presentation, below we focus on the special case of mean total work $\mathbb{E}[W]$. We refer the readers to Theorem 7.2 of our full paper [81] for the full version of our work decomposition law.

Our work decomposition law implies that in the G/G/k/setup under any policy π,

$$\mathbb{E}[W]^{\pi} - \mathbb{E}[W]_{G/G/1} \leq \frac{\mathbb{E}[J_{\text{idle}}W]^{\pi}}{1-\rho} + \frac{\mathbb{E}[J_{\text{setup}}W]^{\pi}}{1-\rho} + \rho(A_{\max} - A_{\min}).$$

Above, $\mathbb{E}[W]_{G/G/1}$ is the mean work in a non-idling G/G/1, which is policy-invariant; and J_{idle} and J_{setup} are the fraction of idle and setting-up servers, respectively. Flipping the sign on the $\rho(A_{\max} - A_{\min})$ term yields a lower bound instead of an upper bound.

The work decomposition law decomposes work $\mathbb{E}[W]^{\pi}$ into the policy-invariant term $\mathbb{E}[W]_{G/G/1}$, plus error terms that can depend on the policy π. Each error term characterizes the consequence of a complicating factor that G/G/k/setup has on the top of the G/G/1 system: [(a)] The first term is due to having multiple servers. It vanishes when $k = 1$, as then $J_{\text{idle}} = 0$ if $W > 0$.[5]

- The second term is due to the setup time. It vanishes if servers do not need setup, as then $J_{\text{setup}} = 0$.
- The third term is due to non-Poisson arrivals. It vanishes for Poisson arrivals, as then $A_{\max} = A_{\min}$.

How we prove the work decomposition law The proof of work decomposition laws in prior work involves viewing W as a process in the steady state and analyzing its continuous changes and jumps. This strategy works well in M/G systems, because all times have an equal chance of seeing W jump up due to an arrival. But in G/G systems, the chance of having an arrival in the next moment depends on A_{age}, the amount of time since the previous arrival. The jumps of W are thus more complicated to analyze.

The key idea in our proof is to smooth out the non-constant jumping rate of W. Specifically, we consider the process $W - \rho A_{\text{res}}$, which only differs from W by one interarrival time. This process decreases at a constant rate of $1 - \rho$. When an arrival happens, the process jumps, but the expected change is $\mathbb{E}[S] - \rho\mathbb{E}[A] = 0$. Therefore, arrivals only have a "second-order" effect on W, which makes them easier to analyze. This idea builds upon similar smoothing approaches in recent queueing literature [24, 116].

4.5.2 From work decomposition to optimality gap bounds

We focus here on proving Theorem 4.1, commenting only briefly on the similar proof of Theorem 4.2.

Combining our work decomposition law with WINE gives a formula for Gittins's optimality gap that has the same types of error terms as (4.5.1). Each error term in the work decomposition law will result in one term in the optimality gap $\ell_{(a)} + \ell_{(b)} + \ell_{(c)}$ in Theorem 4.1, after doing the integration and applying some additional treatments that are specific to each term.

Among the three error terms, $\ell_{(a)}$ can be derived similarly to prior work on the M/G/k [137, 136, 74], and $\ell_{(b)}$ follows from Assumption 4.2. But the term corresponding to setup, $\ell_{(c)}$,

[5] When generalizing Section 4.5.1 from total work to r-work, there are actually two terms due to having multiple servers. But both vanish when $k = 1$.

requires a new analysis. We demonstrate the intuition by bounding $\mathbb{E}[J_{\text{setup}}W]$ in Section 4.5.1. First, we write $\mathbb{E}[J_{\text{setup}}W] = \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[J_{\text{setup},i}W]$, where $J_{\text{setup},i} = \mathbb{1}\{\text{server } i \text{ is setting up}\}$. Observe that

$$\mathbb{E}[J_{\text{setup},i}W] = \mathbb{P}(J_{\text{setup},i} = 1)\,\mathbb{E}[W \mid J_{\text{setup},i} = 1].$$

Intuitively, $\mathbb{P}(J_{\text{setup},i} = 1)$ should be diminishing as the load gets heavy because the queue length will get longer the server i will be turned off less frequently. The second factor, $\mathbb{E}[W \mid J_{\text{setup},i} = 1]$, should be bounded because given that the server i is setting up, the work in the system should be no more than the work that arrives during the setup, plus the work that already exists when the setup happens.

For the proof of Theorem 4.2, which gives a tighter bound for the single-server case, we apply WINE and work decomposition law in the same way as above. We will get an expression for $\mathbb{E}[N]^{\text{Gtn}}_{G/G/1/setup}$ in terms of one $G/G/1$ term, and two error terms corresponding to non-Poisson arrivals and setup times. Instead of analyzing the setup term as in the proof of Theorem 4.1, we make the simple observation that the setup term is the same for all non-idling policies, so it does not contribute to the optimality gap.

4.5.3 From optimality gap bounds to heavy-traffic optimality

Theorem 4.1 provides an upper bound on the optimality gap of Gittins policy in G/G/k/setup. To show that the optimality gap is small compared with $\inf_\pi \mathbb{E}[N]^{\pi}_{G/G/k/setup}$ and establish heavy-traffic optimality of the Gittins policy, we need a lower bound on $\inf_\pi \mathbb{E}[N]^{\pi}_{G/G/k/setup}$. This lower bound can be obtained by analyzing $\mathbb{E}[N]^{\text{SRPT}}_{G/G/1}$ because SRPT gives the optimal number-in-system in G/G/1 with known job sizes [135], which is no more than the optimal number-in-system in G/G/k/setup achievable by a policy that does not know the job size.

We use WINE and work decomposition law, in a similar way as in the proofs in the optimality gaps, to connect SRPT's performance in the G/G/1 to its performance in the M/G/1. Our end result ([81, Theorem 9.1]) shows that $\mathbb{E}[N]^{\text{SRPT}}_{G/G/1}$ is a constant factor away from $\mathbb{E}[N]^{\text{SRPT}}_{M/G/1}$ as $\rho \to 1$. This lets us to use the known heavy-traffic asymptotics of SRPT in the M/G/1 [99] to lower bound $\mathbb{E}[N]^{\text{SRPT}}_{G/G/1}$ and thus show Gittins's heavy-traffic optimality in the G/G/k/setup.

5 Multiserver-job scheduling

5.1 Background and motivation

In today's large-scale computing clusters behind cloud platforms, *multiserver jobs* have become increasingly prevalent, where a multiserver job is a job that demands to occupy multiple "servers" (which can be multiple physical servers, multiple CPU cores, etc.) simultaneously during its runtime [151, 157, 100, 4]. For example, cloud platforms allow users to specify the number of CPU cores in their virtual machines or containers, and this information can be utilized by centralized schedulers to make scheduling decisions [see, e.g. 157, 71]. Moreover, the number of "servers" that a multiserver job requests, which we refer to as the *server need*, is becoming increasingly large. This trend is driven by machine learning jobs from applications like TensorFlow in [4], where the jobs are highly parallel and require synchronization. According to the statistics from Google's Borg Scheduler in [157], the server needs in Borg can vary across six orders of magnitudes.

In this work, we study the impact of multiserver jobs on the delay performance of large-scale computing systems using queueing models. Queueing models with multiserver jobs have been studied in the literature, but quantifying the delay performance is notoriously hard. Exact steady-state distributions can only be derived in highly simplified settings with two servers [25, 47], while the majority of prior work has focused on characterizing stability conditions [117, 133, 5, 72, 121]. However, even for stability, exact conditions are known only for the special cases where all jobs have the same service rate or where there are two job classes. We comment that concurrent to our work [82], [73] and [75] study the delay performance of multiserver jobs in the traditional heavy-traffic regime. A more detailed review of related work is provided in Section 5.3.

A recent advance in understanding the delay of multiserver jobs is a characterization of the *queueing probability* in a *large system* in [160], where the queueing probability is the probability that an arriving job has to queue rather than entering service immediately. Specifically, they consider a multiserver job system with n servers, and study the asymptotic scaling regimes where n becomes large. The scaling regimes allow different job types to have different arrival rates, server needs and service rates. Among those parameters, server needs and arrival rates can scale up with n. Such scaling regimes capture the trend that different multiserver jobs can be highly heterogeneous, especially in terms of server needs. They establish an upper bound on the queueing probability, based on which they give a sufficient condition for the queueing probability to diminish as $n \to \infty$.

Although the work of [160] identifies when the queueing probability diminishes in large systems, which is a much desirable operating scenario, it does not provide much insight for differentiating between scheduling policies. In particular, their queueing probability upper bound holds for any scheduling policy that is reasonably work-conserving (although the bound is presented only for the First-Come-First-Serve policy). Moreover, the queueing probability does not directly translate to the delay of jobs.

In this work, we focus on the *waiting time* of jobs, which is the total time a job spends waiting in the queue (not receiving any service), under various scheduling policies. The waiting time is a performance metric that is directly related to job delay. Our goal is to establish bounds on the mean waiting time that are *order-wise tight* as the number of servers, n, scales. Such tight bounds will enable us to differentiate between policies based on their delay performance. We comment that there has been a line of work in the literature [101, 104, 103, 102, 153, 163, 164] that focuses on quantifying *when* the mean waiting time diminishes in large systems for various queueing models. However, little is known on *how fast* the mean waiting time diminishes due to the lack of lower bounds. Our results provide the rate of diminishing when the mean waiting time does diminish, but our tight bounds on the mean waiting time are not limited to the "diminishing" scenario.

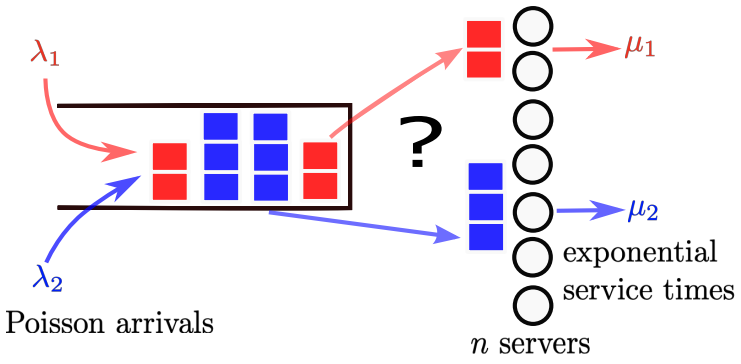


Figure 4: A multiserver-job system with two types of jobs. Type 1 jobs have arrival rate λ_1, service rate μ_1, and server need $\ell_1 = 2$. Type 2 jobs have arrival rate λ_2, service rate μ_2, and server need $\ell_2 = 3$.

Since the First-Come-First-Serve (FCFS) policy is widely used as a default policy in practice and also receives the most attention from theoretical studies of multiserver jobs [25, 47, 72, 5, 117, 133], in this work, we will first examine FCFS and understand the exact order of the mean waiting time under it. Then a natural question that arises is: *can any policy outperform FCFS in terms of the mean waiting time?* More generally, we aim to answer the following fundamental questions:

- *What is the optimal order of the mean waiting time as the system scales?*
- *Which policy achieves the optimal order?*

5.2 Model

We consider a system that consists of n servers and I types of jobs. An example is illustrated in Figure 4. Suppose type i jobs need the simultaneous service of ℓ_i servers. We sort the job types such that their *server needs* ℓ_i's satisfy $\ell_1 \le \ell_2 \le \cdots \le \ell_I$. Let the *maximal server need* $\ell_{\max}$ be $\ell_{\max} = \max_{i\in\{1,2,\ldots,I\}} \ell_i = \ell_I$, and we call type I jobs the *maximal-need jobs*.

The dynamics of the system are as follows. For each $i = 1, 2, \ldots, I$, type i jobs arrive to the system following a Poisson process with *arrival rate* λ_i. Upon arrival, a job either starts service immediately or waits in a centralized queue. When a type i job starts service, it leaves the queue and makes exclusive use of ℓ_i servers. The job leaves the system after receiving enough service. The service time of a type i job follows an exponential distribution with *service rate* μ_i. The service times and arrival events are independent. During the operation of the system, a scheduling policy is used to determine which set of jobs to serve at any time. The scheduling policy is allowed to be preemptive, i.e., we can put a job in service back to the queue and resume its service later.

We measure the performance of our scheduling policy based on *mean waiting time* as defined below: let $T_i^w(\infty)$ denote the expected waiting time of type i jobs in steady-state, then the mean waiting time is defined as the steady-state expected waiting time averaged over all job types, i.e.,

$$\mathbb{E}\big[T^w(\infty)\big] = \frac{1}{\lambda}\sum_{i=1}^{I} \lambda_i \mathbb{E}\left[T_i^w(\infty)\right],$$

where $\lambda \triangleq \sum_{i=1}^{I} \lambda_i$ is the total arrival rate.

Scheduling policies. A scheduling policy decides which jobs to put into service at any moment of time. We are interested in the following two policies:

- *First-Come-First-Serve (FCFS)*: Jobs are placed onto servers in a First-Come-First-Serve fashion until either the next job in queue does not fit or all the jobs are in service.
- *Smallest-Need-First (SNF)*: Recall that the job types are indexed in a way such that $\ell_1 \leq \ell_2 \leq \cdots \leq \ell_I$. We assign priorities to job types such that a smaller index has a higher priority. Whenever there is a job arrival or departure, SNF preempts all the jobs in service and determines a new schedule from scratch. SNF starts from job type 1 and places as many type 1 jobs as possible onto servers. After this, if there are still servers available, SNF goes to the next priority level, type 2, and places as many type 2 jobs as possible onto servers. This procedure continues until no more jobs in the queue can fit into the servers.

Scaling regimes. We consider scaling regimes where the number of servers, n, goes to infinity, and the arrival rates λ_i and server needs ℓ_i are allowed to scale with n, *while the service rate μ_i and the number of job types I stay constant.* The scaling regimes are specified by the slack capacity $\delta \triangleq n - \sum_{i=1}^{I} \frac{\lambda_i \ell_i}{\mu_i}$, the maximal server need $\ell_{\max} \triangleq \max_{i\in[I]} \ell_i = \ell_I$ and another parameter called the *work variability*: $\sigma^2 \triangleq \sum_{i=1}^{I} \frac{\lambda_i \ell_i^2}{\mu_i^2}$. Work variability reflects the variability of the "work" caused by job arrivals in terms of server–time product, which is $\frac{\ell_i}{\mu_i}$ in expectation for each type i job. To help later presentation, we define the *load* ρ as $\rho = \sum_{i=1}^{I} \frac{\lambda_i \ell_i}{n\mu_i}$; we define the *load brought by type i jobs* ρ_i as $\rho_i = \frac{\lambda_i \ell_i}{n\mu_i}$.

We state our assumptions below. Throughout the section, $\log n$ denotes natural logarithm.

Assumption 5.1 (Heavy traffic assumption)**.** The slack capacity δ is small relative to $\sqrt{\sigma^2}$:

$$\delta = o\left(\frac{\sqrt{\sigma^2}}{\log n}\right). \tag{29}$$

Assumption 5.2 (Maximal server need assumption)**.** There exists a constant $\epsilon_0 \in (0,1)$ s.t.

$$\ell_{\max} \leq \epsilon_0 \delta. \tag{30}$$

Assumption 5.3 (Commonness assumption)**.** The load brought by the maximal-need jobs is not too small (note that $\ell_1 \leq \ell_2 \leq \cdots \leq \ell_I = \ell_{\max}$):

$$\rho_I \triangleq \frac{\lambda_I \ell_I}{n\mu_I} = \omega\left(\sqrt{\frac{\delta \log n}{\sqrt{\sigma^2}} \cdot \frac{\ell_{\max}}{n}} \log n\right). \tag{31}$$

Assumption 5.1 guarantees that the traffic is not too light, while Assumption 5.2 guarantees that the system is stable under FCFS and SNF. Assumption 5.3 states that the load brought by the maximal-need jobs are not too small. To understand the right hand side expression in Assumption 5.3, note that it is automatically satisfied when $\rho_I = \omega\left(\sqrt{\frac{\ell_{\max}}{n}} \log n\right)$ due to Assumption 5.1. For example, when $\ell_{\max} = \Theta\left(\sqrt{n}\right)$, then it suffices to have $\rho_I = \omega\left(n^{-1/4} \log n\right)$.

To have an intuitive view of the magnitudes of the parameters, we give the following asymptotics: $\sigma^2 = O\left(n\ell_{\max}\right)$, $\delta = o\left(\frac{n}{(\log n)^2}\right)$, and $\ell_{\max} \leq \epsilon_0 \delta = o\left(\frac{n}{(\log n)^2}\right)$. They can be verified using the definitions and assumptions.

The asymptotic regimes that we consider can be viewed as the many-server heavy-traffic scalings in traditional multiclass M/M/n models, where the load $\rho \to 1$ and the number of servers $n \to \infty$ jointly. In particular, when we set $\ell_i = 1$ and let $\lambda_i = \Theta(n)$ for all $i \in \Theta(n)$, because $\delta = n(1-\rho)$, one can verify that the Assumptions 5.1, 5.2, and 5.3 are equivalent to

$$1 - \rho = o\left(\frac{1}{\sqrt{n}\log n}\right) \text{ and } 1 - \rho = \Omega\left(\frac{1}{n}\right).$$

In other words, in this special case, our scaling regimes are lighter than the Non-Degenerate Slowdown (NDS) regime introduced in [9], and at least as heavy as the Halfin-Whitt (HW) regime introduced in [77].

Subsystems. When we analyze SNF and prove the lower bound for all policies, we frequently use the concept of the i-th *subsystem*, which is the system that has all type j jobs in the original system with $j \le i$ and removes all type k jobs with $k \ge i$. In the i-th subsystem, the slack capacity becomes $\delta_i = n - \sum_{j=1}^{i} \frac{\lambda_j \ell_j}{\mu_j}$, and the work variability becomes $\sigma_i^2 = \sum_{j=1}^{i} \frac{\lambda_j \ell_j^2}{\mu_j^2}$. Note that $\delta = \delta_I$ and $\sigma_I^2 = \sigma^2$. The maximal server need in the i-th system is ℓ_i since $\ell_1 \le \ell_2 \le \cdots \le \ell_i$.

As i increases, the load of the i-th subsystem gets heavier since δ_i becomes smaller. There is a *critical index* i^* such that

$$i^* = \min\left\{ i \in [I] \,\middle|\, \delta_i = o\left(\frac{\sqrt{\sigma_i^2}}{\log n}\right) \right\}, \tag{32}$$

i.e., the i^*th subsystem is the smallest subsystem whose traffic regime is as heavy as that of the original system. Recall that we have assumed $\delta = o\left(\frac{\sqrt{\sigma^2}}{\log n}\right)$, so the set in (32) contains at least the index I and thus i^* is well-defined. Note that δ_i is monotonically decreasing while σ_i^2 is monotonically increasing. Thus the index i^* serves as a division point: for any i with $i^* \le i \le I$, we have $\delta_i = o\left(\frac{\sqrt{\sigma_i^2}}{\log n}\right)$, resulting in a heavier traffic regime; and for any i with $1 \le i < i^*$, we have $\delta_i = \Omega\left(\frac{\sqrt{\sigma_i^2}}{\log n}\right)$, resulting in a lighter traffic regime.

5.3 Related work

In this section, we give a more detailed review of the prior work on multiserver-job models as well as some related models that are not covered in the introduction.

Multiserver-job model.

As mentioned in the introduction, the majority of prior work on the multiserver-job model has either focused on characterizing stability conditions [117, 133, 5, 72, 121], or been restricted to the highly specialized settings with two servers [25, 47]. There are relatively fewer papers that study the delay of the multiserver-job model, with different performance metrics. [160] characterizes the queueing probability in a large multiserver-job model. [175] studies the optimal cumulative holding cost in a finite-horizon setting. The papers whose performance metrics are closest to ours are [73] and [75]. In [73], the mean response time in a multiserver-job model under a policy called ServerFilling is characterized. In [75], a variant of ServerFilling called ServerFilling-SRPT is proposed, which optimizes the mean response time in the traditional heavy traffic regime. The biggest distinction

between their work and our work is the scaling regimes: in their work, the analysis of mean response time is asymptotically tight when the load of the system approaches one and the number of servers remains *fixed*; in contrast, we consider the scaling regimes where the load, number of servers and server needs *scale jointly*. Another distinction is the distributional assumptions on the server needs: their work assumes that the server needs can divide the total number of servers, while our work assumes that the maximal server need is small compared with the slack capacity.

Virtual machine (VM) scheduling.

A problem related to the multiserver-job scheduling problem studied in this work is the virtual machine (VM) scheduling problem (see, e.g., [110, 108, 111, 169, 127, 128, 149]). For the VM scheduling problem, typically the system consists of multiple servers, where each server has certain units of each type of resource (e.g., CPU, memory, storage). A VM job demands to occupy multiple units of each type of resource. Each VM job will be served on a single server. Some results for the VM scheduling problem in the traditional heavy-traffic regime can be specialized to the multiserver job scheduling problem. To see this, consider a VM scheduling problem where the system consists of a single server and there is a single resource type. Then each unit of resource can be viewed as a server in the multiserver-job scheduling problem. With this specialization, the results in [111] provide bounds on a linear combination of the queue lengths of different types of jobs. However, these bounds do not directly translate into bounds on mean job response time.

Multitask job model.

A multitask job is a job that consists of a batch of tasks that can run on servers in parallel, which is similar to a multiserver job in that both can occupy multiple servers at the same time. However, unlike a multiserver job, the tasks of a multitask job can have different runtimes and do not need to be executed simultaneously. Multitask job model has been considered under a wide variety of settings, and is sometimes referred to as batch arrival model [see, e.g., 113, 40, 38, 39]. Recently, multitask job model is also extensively studied under the setting of parallel computing due to the popularity of large-scale data processing systems such as MapReduce, Apache Hadoop and Apache Spark. [see, e.g., 163, 174]. The work closest to our work is [163], which shows diminishing queueing time for multi-server jobs in a load-balancing system where tasks of a job need to be dispatched to the queues at the servers upon arrival.

Dropping model.

When the multiserver-job system does not have any queueing space and allows incoming jobs to be dropped, it becomes a model that has been studied in the literature and we refer to it as the *dropping model*. In this model, one can design a dropping policy that decides whether to drop an incoming job or not based on the types of the incoming job and of the jobs currently in service. Under the policy that drops an incoming job only when it cannot fit into the servers, i.e., when its server need is larger than the number of available servers, the stationary distribution has a product form under exponentially distributed service times, as observed in [8]. The results have been generalized by [166] to allow jobs to demand multiple resource types (e.g., both CPU and I/O) and by [154] to allow general service time distributions. In [150], aspects of [166] and [154] are further combined. Different dropping policies, which mostly fall within the class of trunk reservation policies, have been designed to minimize the cost associated with dropping [87, 15, 88].

Streaming model.

The *streaming model* for a communication network resembles the multiserver-job model in many aspects. In a streaming model, the "servers" correspond to the bandwidth in the network and the "jobs" are data flows such as audio or video flows. Then flows that require a fixed amount of bandwidth [112, 37, 16, 126], sometimes referred to as streaming flows, can be viewed as multiserver jobs. However, a communication network also features a network structure that the multiserver-job model does not have. A communication network usually has both streaming flows and flows that are flexible in their bandwidth needs, and streaming flows again operate in the dropping model. The performance metric in such a system typically combines the cost associated with dropping for streaming flows and the cost associated with delay for other flows.

5.4 Main results

In this section, we present our main results under the scaling regimes we specify in Section 5.2 as Theorems 5.1, 5.2, and 5.3.

Theorem 5.1 (Mean waiting time under FCFS)**.** *Consider the multiserver-job system with n servers satisfying Assumptions 5.1 and 5.2. Under the FCFS policy, for each $i \in [I]$, the expected waiting time of type i jobs satisfies*

$$\mathbb{E}\big[T_i^w(\infty)\big]^{\mathrm{FCFS}} \geq \frac{\sigma^2}{n(\delta + \ell_{\max})} \cdot (1 - o(1))\,, \tag{33}$$

$$\mathbb{E}\big[T_i^w(\infty)\big]^{\mathrm{FCFS}} \leq \frac{\sigma^2}{n(\delta - \ell_{\max})} \cdot (1 + o(1))\,. \tag{34}$$

Consequently,

$$\mathbb{E}\big[T^w(\infty)\big]^{\mathrm{FCFS}} = \Theta\left(\frac{\sigma^2}{n\delta}\right), \quad \mathbb{E}\big[T_i^w(\infty)\big]^{\mathrm{FCFS}} = \Theta\left(\frac{\sigma^2}{n\delta}\right). \tag{35}$$

Theorem 5.2 (Mean waiting time lower bound)**.** *Consider the multiserver-job system with n servers satisfying Assumptions 5.1 and 5.2. Under any policy, the mean waiting time is lower bounded as*

$$\mathbb{E}\big[T^w(\infty)\big] \geq \max_{i^* \leq i \leq I} \frac{1}{\lambda} \frac{\mu_{\min}\sigma_i^2}{\ell_i \delta_i} \cdot (1 - o(1)) = \Omega\left(\max_{i^* \leq i \leq I} \frac{1}{\lambda} \frac{\sigma_i^2}{\ell_i \delta_i}\right) \tag{36}$$

where i^* is the critical index defined in (32) and $\mu_{\min} = \min_{i \in [I]} \mu_i$, and the expression represented by $o(1)$ is independent of the policies.

Theorem 5.3 (Mean waiting time under SNF)**.** *Consider the multiserver-job system with n servers satisfying Assumptions 5.1, 5.2, and 5.3. Under the SNF policy, the mean waiting time satisfies*

$$\mathbb{E}\big[T^w(\infty)\big]^{\mathrm{SNF}} \leq \frac{1}{\lambda} \sum_{i=i^*}^{I} \frac{\mu_{\max}\sigma_i^2}{\ell_i(\delta_i - \ell_i)} \cdot (1 + o(1)) = O\left(\max_{i^* \leq i \leq I} \frac{1}{\lambda} \frac{\sigma_i^2}{\ell_i \delta_i}\right), \tag{37}$$

where i^* is the critical index defined in (32) and $\mu_{\max} = \max_{i \in [I]} \mu_i$. Consequently, the SNF policy achieves the optimal order of the mean waiting time.

5.5 Key ideas in the analysis

In this section, we give an overview of our analysis of the multiserver-job system, centered around a key quantity called normalized work. The consideration of the normalized work is loosely related to the idea of understanding a complex system through a simple system.

Below we first introduce some preliminaries; then we define the normalized work and comment on its relationship with the mean waiting time; finally, we give some intuitions on the dynamics of normalized work to show why it is simple.

Preliminaries. We first define a few necessary notations. For any time t and for each job type $i = 1, \ldots, I$, let $X_i(t)$ denote the steady-state number of type i jobs in the system; let $Z_i(t)$ denote the steady-state number of type i jobs in service; let $Q_i(t)$ denote the steady-state number of type i jobs waiting in the queue. We use $X_i(\infty)$, $Q_i(\infty)$ and $Z_i(\infty)$ to denote the steady-state distributions of $X_i(t)$, $Q_i(t)$, and $Z_i(t)$.

By the definitions of $X_i(t)$, $Z_i(t)$ and $Q_i(t)$, we have the relation $X_i(t) = Q_i(t) + Z_i(t)$. In addition, note that since the total number of servers in use cannot exceed n, and we cannot serve more jobs than there are in the system, we have the following constraints:

$$\begin{aligned} &\sum_{i=1}^{I} \ell_i Z_i(t) \le n \quad \text{for all } t \ge 0, \\ &Z_i(t) \le X_i(t) \quad \text{for all } t \ge 0, i = 1, 2, \ldots, I. \end{aligned} \tag{38}$$

We will consider a class of policies called $\ell_{\max}$-work-conserving policies, which keeps no more than $\ell_{\max}$ servers idle whenever there are enough jobs in the system. Formally, under a $\ell_{\max}$-work-conserving policy, we have $\sum_{i=1}^{I} \ell_i Z_i(t) \ge n - \ell_{\max}$ whenever $\sum_{i=1}^{I} \ell_i X_i(t) \ge n - \ell_{\max}$.

Normalized work and mean waiting time. A key quantity in our analysis is the normalized work, $\overline{W}(t)$, given by

$$\overline{W}(t) \triangleq \sum_{i=1}^{I} \frac{\ell_i}{\mu_i} \left(X_i(t) - \frac{\lambda_i}{\mu_i} \right).$$

The normalized work is closely related to the mean waiting time. To see this, we first show that $\mathbb{E}[Z_i(\infty)] = \lambda_i/\mu_i$. Then the expectation of the normalized work is equal to

$$\mathbb{E}\left[\overline{W}(\infty)\right] = \sum_{i=1}^{I} \frac{\ell_i}{\mu_i} \mathbb{E}[Q_i(\infty)].$$

On the other hand, by Little's law,

$$\mathbb{E}\left[T^w(\infty)\right] = \frac{1}{\lambda} \sum_{i=1}^{I} \mathbb{E}[Q_i(\infty)].$$

As we can see, $\mathbb{E}\left[\overline{W}(\infty)\right]$ and $\mathbb{E}\left[T^w(\infty)\right]$ are both linear combinations of $\mathbb{E}[Q_i(\infty)]$'s.

The benefit of considering $\overline{W}(t)$ is that it has simple dynamics under a large class of policies, allowing its steady-state distribution to be characterized in a relatively precise way. In our analysis in [82], the characterization of $\overline{W}(\infty)$ serves as a basis and starting point for the analysis of other more complicated quantities, like total server needs of the jobs, the total queue length and $\mathbb{E}\left[T^w(\infty)\right]$. We refer the readers to [82] for details of going from normalized work to mean waiting times.

Dynamics of $\overline{W}(t)$. We show that under a $\ell_{\max}$-work-conserving policy, the dynamics of $\overline{W}(t)$ is *approximately* 1-*dimensional* in some sense. To see this, consider the drift of $\overline{W}(t)$, which we informally write as $G\overline{W}(t)$. Because type i jobs arrive at the rate λ_i and complete at the rate $\mu_i Z_i$, the drift of $\overline{W}(t)$ should be

$$G\overline{W}(t) = \sum_{i=1}^{I} \frac{\ell_i}{\mu_i}(\lambda_i - \mu_i Z_i(t)) = \sum_{i=1}^{I} \frac{\lambda_i \ell_i}{\mu_i} - \sum_{i=1}^{I} \ell_i Z_i(t).$$

Note that $\sum_{i=1}^{I} \ell_i Z_i(t)$ on the right-hand side equals the number of busy servers at time t. On the one hand, we have $\sum_{i=1}^{I} \ell_i Z_i(t) \le n$, so the drift of $\overline{W}(t)$ is at least

$$G\overline{W}(t) \ge \sum_{i=1}^{I} \frac{\lambda_i \ell_i}{\mu_i} - n = -\delta, \tag{39}$$

by the definition of δ. On the other hand, under a $\ell_{\max}$-work-conserving policy, when $\sum_{i=1}^{I} \ell_i X_i(t) \ge n + \delta$, the drift of $\overline{W}(t)$ is at most

$$G\overline{W}(t) \le \sum_{i=1}^{I} \frac{\lambda_i \ell_i}{\mu_i} - (n - \ell_{\max}) = -\delta + \ell_{\max}. \tag{40}$$

By Assumption 5.2, $\ell_{\max} \le \epsilon_0 \delta$ for some $\epsilon_0 < 1$. Therefore, $\overline{W}(t)$ decreases at the rate approximately equal to δ when $\sum_{i=1}^{I} \ell_i X_i(t) \ge n + \delta$, under a $\ell_{\max}$-work-conserving policy.

Moreover, through some state-space concentration arguments, we can show that in the steady-state, $\sum_{i=1}^{I} \ell_i X_i(t) \ge n + \delta$ is roughly equivalent to $\overline{W}(t) \ge \bar{r}$ for some threshold $\bar{r}$. We can also show that $\overline{W}(t)$ will not be too much smaller than the threshold $\bar{r}$ with high probability in the steady state, again through a state-space concentration argument.

Given the above intuitions and facts, we can approximately view $\overline{W}(t)$ as a one-dimensional quantity that decreases at a constant rate δ whenever it is larger than the threshold $\bar{r}$. The formal analysis of $\overline{W}(t)$ is through Lyapunov drift analysis and iterative state-space concentration. We refer the readers to [82] for details.

6 Proposed work

6.1 Achieving exponential optimality gap for restless bandits without UGAP

6.1.1 Motivation

In our work on restless bandits included in Section 2, we have constructed policies that achieve asymptotic optimality under only minimal conditions (namely, unichain and aperiodicity). Moreover, we have shown that the optimality gap converges to zero at the rate $O(1/\sqrt{N})$ as the number of arms $N \to \infty$. In this work, we plan to investigate if $O(1/\sqrt{N})$ is the "optimal convergence rate", in an appropriate sense that we elaborate below.

Recall that in Section 2, we construct a simple system using the single-armed problem (Section 2.3.1) to facilitate proving optimality gaps and constructing policies. In particular, we consider the optimal value of the single-armed problem, R^{rel}, which is an upper bound of the optimal value of the N-armed problem, $R^*(N)$, under mild conditions. We relax the optimality gap $R^*(N) - R(\pi, N)$ to $R^{\text{rel}} - R(\pi, N)$ and show that a policy π under study satisfies $R^{\text{rel}} - R(\pi, N) = O(1/\sqrt{N})$.

In this work, we plan to use the same approach to bound the optimality gap of a policy. In this case, the best possible optimality gap that we can prove is no less than $R^{\text{rel}} - R^*(N)$. Naturally, we could ask the following questions:

How large is the gap $R^{\text{rel}} - R^*(N)$*?*
Can we tightly characterize the order of $R^{\text{rel}} - R^*(N)$*?*

In the literature, it has been proved that in general, the gap $R^{\text{rel}} - R^*(N)$ is at most $O(1/\sqrt{N})$ under unichain and aperiodicity conditions [59, Theorem 1]. With two additional assumptions, UGAP and non-degeneracy (Assumption 6.2), [59, 60] show that $R^{\text{rel}} - R^*(N)$ is $O(\exp(-CN))$ for some constant $C > 0$.

Based on these existing results, a natural next step is to study the order of $R^{\text{rel}} - R^*(N)$ in the cases without UGAP or non-degeneracy. For these cases, we can either prove that $R^{\text{rel}} - R^*(N)$ is exponentially small, or figure out the exact order of $R^{\text{rel}} - R^*(N)$ as $N \to \infty$.

In addition to understanding the order of $R^{\text{rel}} - R^*(N)$, we will also consider the following question:

Can we efficiently compute the policy π *such that*
$R^{\text{rel}} - R(\pi, N)$ *has the same order as* $R^{\text{rel}} - R^*(N)$*?*

Note that the requirement of "efficiently computing the policy" prohibits directly using the exact optimal policy, which is PSPACE hard to find. Answering this question under general conditions not only improves our fundamental understanding of restless bandits, but also leads to some new policies that could potentially be useful in practice.

6.1.2 Expected result

We consider the N-armed restless bandit problem with the single-armed MDP $(\mathbb{S}, \mathbb{A}, P, r)$ and budget αN for $0 < \alpha < 1$. We fix an optimal policy for the single-armed problem $\bar{\pi}^*$. The policy $\bar{\pi}^*$ induces a Markov chain with transition kernel, $P_{\bar{\pi}^*}$.

We consider the following three assumptions.

Assumption 6.1 (Unichain and aperiodicity for the optimal single-armed policy)**.** Assume that $\bar{\pi}^*$ induces an aperiodic unichain $P_{\bar{\pi}^*}$ on $\mathbb{S}$.

Under Assumption 6.1, the Markov chain $P_{\bar{\pi}^*}$ has a unique stationary distribution, which we can represent as a row vector, $\mu^* = (\mu^*(s))_{s\in\mathbb{S}}$, where $\mu^*(s)$ is the stationary probability of state s in the Markov chain. Let $y^* \triangleq (y^*(s,a))_{s\in\mathbb{S},a\in\mathbb{A}}$, where $y^*(s,a)$ is the stationary probability of being in state s and taking action a in this Markov chain.

Assumption 6.2 (Non-degeneracy)**.** There exists a unique state $\tilde{s}$ s.t. $y^*(\tilde{s},1) > 0$ and $y^*(\tilde{s},0) > 0$.

Assumption 6.3 (Local stability)**.** When non-degeneracy (Assumption 6.2) holds, let

$$\Phi = P_{\bar{\pi}^*} - \mathbb{1}^\top \mu^* - (c_{\bar{\pi}^*} - \alpha\mathbb{1})^\top (P_1(\tilde{s}) - P_0(\tilde{s})),$$

where $c_{\bar{\pi}^*} \triangleq (\bar{\pi}^*(1|s))_{s\in\mathbb{S}}$ and $P_a(\tilde{s}) \triangleq (P_a(\tilde{s},s))_{s\in\mathbb{S}}$ are both row vectors; $\mathbb{1}$ is the all-one row vecctor. We assume that the modulus of each eigenvalue of Φ is strictly less than 1.

Remark 6.1. The emphasis of Assumption 6.2 is on the existence of a state $\tilde{s}$ such that $y^*(\tilde{s},1) > 0$ and $y^*(\tilde{s},0) > 0$. In fact, one can always find an optimal single-armed policy $\bar{\pi}^*$ such that the corresponding y^* satisfies $y^*(\tilde{s},1) > 0$ and $y^*(\tilde{s},0) > 0$ for at most one $\tilde{s} \in \mathbb{S}$ [60, Proposition 2].

Remark 6.2. Assumption 6.3 is a necessary condition for the LP-Priority policy to satisfy UGAP and plays an important role in the proof that the LP-Priority policy achieves an exponentially small optimality gap [60]. In particular, Φ defines the local dynamics of the mean-field difference equation induced by the LP-Priority policy around the fixed point of the difference equation.

We conjecture that Assumption 6.1, 6.2 and 6.3 are sufficient for efficiently finding a policy with an exponentially small optimality gap.

Conjecture 6.1. *Consider an N-armed restless bandit problem with the single-armed MDP $(\mathbb{S},\mathbb{A},P,r)$ and budget αN for $0 < \alpha < 1$. Assuming Assumption 6.1, 6.2 and 6.3, we can construct a policy π such that*

$$R^{\text{rel}} - R(\pi,N) = O\big(\exp(-CN)\big), \tag{41}$$

for some $C > 0$. The complexity for finding π is similar to solving the single-armed problem (9)-(10).

Moreover, we conjecture that the set of assumptions, Assumption 6.1, 6.2 and 6.3, are necessary for achieving $o(1/\sqrt{N})$ optimality gap.

Conjecture 6.2. *Consider an N-armed restless bandit problem with the single-armed MDP $(\mathbb{S},\mathbb{A},P,r)$ and budget αN for $0 < \alpha < 1$. Without any of Assumption 6.1, 6.2 or 6.3, one can find an RB instance such that for any policy π,*

$$R^{\text{rel}} - R(\pi,N) = \Omega\Big(\frac{1}{\sqrt{N}}\Big). \tag{42}$$

6.1.3 Status and timeline

Here is a summary of the status of the project by the time the thesis proposal is written.

- We have a proof sketch for Conjecture 6.2. Specifically, we are working on proving a lower bound for $R^{\text{rel}} - R(\pi,N)$. Once the lower bound is proved, it should imply $R^{\text{rel}} - R(\pi,N) = \Omega\big(1/\sqrt{N}\big)$ whenever Assumption 6.2 or 6.3 does not hold.
- We have a proof sketch for Conjecture 6.1. Specifically, we have designed a policy that generalizes the focus-set policies in [84]. We wrote a proof outline to decompose the optimality gap analysis of the policy into verifying three conditions, with a similar structure as our analysis of the ID policy (Section 2.5.2). To finish the proof, we only need to fill in the proofs for each condition.

We plan to finish the project in the next three months, by the end of June 2024.

6.2 Restless bandits with asynchronous actions.

6.2.1 Motivation

Restless bandits can be viewed as a set of independent MDPs coupled together by a budget constraint on their joint actions. In discrete-time restless bandits, the actions are required to be *synchronous* – exactly αN arms are pulled at the same time in each time step. For continuous-time restless bandit problems, although they are referred to as being asynchronous in some literature [e.g. 59] because different arms transition at different time points, they are still synchronous in the sense of actions – the decision maker needs to keep αN arms activated at the same time.

However, in many resource allocation problems, sometimes it makes more sense to model the actions as *asynchronous* rather than synchronous. Specifically, we can consider the following variant of the standard restless bandit problem: Suppose we have a set of N arms. Budgets arrive unit by unit following a stochastic point process in continuous time. When a unit of budget arrives, *exactly one arm should be pulled*, which causes the state of the arm to change immediately according to a predefined transition probability matrix. For arms that are not pulled, they could jump to another state according to some predefined transition rates. The state transition of different arms is independent of each other. We call this model *restless bandits with asynchronous actions* since the arms are pulled one at a time.

To give an example of restless bandits with asynchronous actions, consider the basic continuous-time load-balancing model in queueing theory, where jobs arrive over time according to a Poisson process and require exponentially distributed service times. Each job needs to be dispatched to a suitable queue upon arrival, with the goal of minimizing the steady-state expected queue length. We can view this load-balancing model as a special case of restless bandits with asynchronous actions, where each queue is an arm whose state is the number of jobs in the queue, and each job arrival is a unit of budget that allows you to change the state of an arm. The state of each arm can also change on its own due to job completions.

The formulation of restless bandits with asynchronous actions allows us to study many problems with similar structures in a unified way. For example, it allows us to directly study load balancing whose service time distribution is an arbitrary phase-type distribution, by incorporating the service phase as part of the arms' state. It also allows servers to have setup times, again by incorporating the relevant information into the arms' state. Even the stochastic bin-packing problem with time-varying resource requirements (Section 3) can be viewed as a type of restless bandits with asynchronous actions, after doing appropriate modifications.

Similar to standard restless bandits, the goal of studying restless bandits with asynchronous actions is to identify asymptotic optimal policies and find algorithms for computing these policies efficiently. We will precisely define the notion of asymptotic optimality when we formally set up the model in the next section.

6.2.2 Model

A restless bandit problem with asynchronous actions bandit problem consists of N homogeneous arms. Each arm can be viewed as a continuous-time MDP with impulsive controls (see, e.g., [45]). Specifically, each arm is specified by the tuple $(\mathbb{S}, \mathbb{A}, Q, P, r)$, where $\mathbb{S}$ is a finite state space, $\mathbb{A} = \{0, 1\}$ denotes the action space, $Q = (Q(s, s'))_{s,s' \in \mathbb{S}}$ is a transition rate matrix, $P = (P(s, s'))_{s,s' \in \mathbb{S}}$ is a transition probability matrix, and $r = (r(s))_{s \in \mathbb{S}}$ is a reward function. The action 1 corresponds to pulling the arm, and is referred to as the *impulse action*; the action 0 corresponds to not pulling the arm and is referred to as the *passive action*. Each impulse action only lasts for an instance and causes the state of the arm to transition *immediately* according to the transition probability matrix

P; a passive action is applied whenever the impulse action is not applied, under which the state of the arm transitions according to the transition rate matrix Q. When an arm is in state s, it generates rewards at the rate $r(s)$.

We assume that budgets arrive over time following a Poisson process with rate $N\lambda$ for some constant $\lambda > 0$. When each unit of budget arrives, the decision maker needs to immediately choose an arm and apply the impulse action to the arm. The decision is made based on the states of all arms, using a certain *policy*, whose performance is quantified by its long-run average reward. Let $R(\pi, N)$ be the long-run average reward per arm and per unit time under a policy π, and let $R^*(N)$ be the maximal value of $R(\pi, N)$ over all possible policies. We call $R^*(N) - R(\pi, N)$ the optimality gap of π, and call a policy π asymptotically optimal if $R^*(N) - R(\pi, N) \to 0$ as $N \to \infty$.

6.2.3 Expected results

We plan to find asymptotic optimal policies for restless bandits with asynchronous actions in a similar way to how we do it in discrete-time restless bandits (Section 2). Specifically,

1. We will first define a single-armed problem and optimal single-armed policy, $\bar{\pi}^*$.
2. Then we can try to construct a policy π using $\bar{\pi}^*$ as building blocks, perhaps in a way similar to the FTVA policy or ID policy defined in Algorithm 2).
3. Finally, we bound the optimality gap for the policy π is asymptotic optimal, under certain conditions.
$$R^{\text{rel}} - R(\pi, N) = O\left(\frac{1}{\sqrt{N}}\right). \tag{43}$$

6.2.4 Status of the project

Here we give a summary of the status of this project.

- We have defined the single-armed problem and optimal single-armed policy.
- We have constructed a policy inspired by `FTVA`(Algorithm 1).
- We are writing the proof that the policy has $O\left(1/\sqrt{N}\right)$ optimality gap assuming a certain unichain condition. The proof is almost complete, though some details need to be double-checked.

While it seems that we have already got most of the results we want, the definitions of the policy and the proofs are complicated and notationally heavy. Therefore, in the next step, we plan to try a different policy adapted from ID policy, which we expect to be simpler to define and analyze. We plan to work on the project starting from July 2024 and finish it by December 2024.

6.3 Improving the universal queue length bound for G/G/n

6.3.1 Motivation

In the study of multi-server queueing models, one of the most basic and extensively studied objects is the GI/GI/n (or simply G/G/n) model. The G/G/n model has one centralized queue and n identical servers; jobs arrive following a renewal process with a general interarrival time distribution, whose service times are also i.i.d. with a general distribution. The jobs in the queue are served in a First-Come-First-Served order. The performance of a G/G/n model is often measured in terms of

its steady-state queue length, denoted as Q, which is the number of jobs waiting in the queue (not in service) in the steady state.

Although the G/G/n model has been studied for decades, there are still some fundamental mysteries about the steady-state distribution of its queue length – even the first moment, $\mathbb{E}[Q]$, does not have a very tight upper bound. The current state-of-the-art bounds on $\mathbb{E}[Q]$ are proved in [97], which show that

$$\mathbb{E}[Q] \le \frac{C}{1-\rho}, \tag{44}$$

for some $C > 0$. Notably, the numerator C in (44) is independent of the number of servers n and the load ρ, so the bound in (44) scales as $1/(1-\rho)$ universally in any scaling regimes where $\rho \to 1$ or $n \to \infty$. The $1/(1-\rho)$-scaling matches with the tightest bounds or approximations in a few special cases, for example, the Kingman's bound for G/G/1 [91], the results for G/G/n when $\rho \to 1$ and n remains fixed [93, 94, 106, 119, 120, 89], and the results in Halfin-Whitt regime [see, e.g. 36, 21, 23, 13] (We refer the readers to [97] for a more comprehensive review of these prior works).

Although the bound in (44) achieves the universal $1/(1-\rho)$-scaling in any limiting regimes, the constant C can be huge. Specifically, it is proved in Corollary 3 of [97] that

$$\mathbb{E}[Q] \le 10^{450}\big(\mathbb{E}\left[(S\mu_S)^3\right]\mathbb{E}\left[(A\mu_A)^3\right]\big)^3 \times \frac{1}{1-\rho}, \tag{45}$$

where S and A are the service time distribution and inter-arrival time distribution; $\mu_S = 1/\mathbb{E}[S]$ and $\mu_A = 1/\mathbb{E}[A]$. While [97] also provides a set of other bounds of the form (44), those bounds have similarly large constants.

In this project, we plan to improve upon the work of [97], proving a bound on $\mathbb{E}[Q]$ with $1/(1-\rho)$-scaling in any scaling regimes while making the constant C smaller. We also hope to give a simpler proof than the one given in [97], so that we can get a more intuitive understanding of the G/G/n model. To set a tractable goal, we focus on a less general but not too restrictive setting where the service times S and inter-arrival times A are "light-tailed" in the sense stated in Assumption 6.4.

6.3.2 Expected results

Assumption 6.4 (Bounded expected remaining times)**.** We assume that there exists a constant $M > 0$ such that

$$\sup_{t\ge 0} \mathbb{E}[A\mu_A - t \,|\, A\mu_A \ge t] \le M \tag{46}$$

$$\sup_{t\ge 0} \mathbb{E}[S\mu_S - t \,|\, S\mu_S \ge t] \le M. \tag{47}$$

Note that Assumption 6.4 is not a very restrictive assumption. For example, it holds for all phase-type distributions.

We plan to prove the following bound on $\mathbb{E}[Q]$:

Conjecture 6.3. *Consider a G/G/n model satisfying Assumption 6.4. We have*

$$\mathbb{E}[Q] \le \frac{Var[A\mu_A] + Var[S\mu_S] + 2M}{2(1-\rho)} + \frac{1}{2} + M, \tag{48}$$

where A denotes the inter-arrival time distribution, S denotes the service time distribution, $\mu_A = 1/\mathbb{E}[A]$, $\mu_S = 1/\mathbb{E}[S]$, $\rho = \mu_A/(n\mu_S)$, M is given in Assumption 6.4.

6.3.3 Status and timeline

We have written a proof sketch for Conjecture 6.3, using rate conservation law and coupling with some simpler systems. The proof sketch is close to a full proof, except that some Palm expectation arguments need to be more rigorously written. We plan to write a paper from January 2025 to March 2025.

References

[1] S. Aalto, U. Ayesta, and R. Righter. On the Gittins index in the $M/G/1$ queue. *Queueing Systems*, 63(1-4):437–458, Dec. 2009.

[2] S. Aalto, U. Ayesta, and R. Righter. Properties of the Gittins index with application to optimal scheduling. *Probability in the Engineering and Informational Sciences*, 25(3):269–288, July 2011.

[3] S. Aalto, P. Lassila, and I. Taboada. Whittle index approach to opportunistic scheduling with partial channel information. *Perform. Eval.*, 136:102052, 2019.

[4] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *Proc. USENIX Conf. Operating Systems Design and Implementation (OSDI)*, pages 265–283, Savannah, GA, Nov. 2016.

[5] L. Afanaseva, E. Bashtova, and S. Grishunina. Stability analysis of a multi-server model with simultaneous service and a regenerative input flow. *Methodol. Comp. Appl. Probab.*, pages 1–17, 2019.

[6] T. W. Archibald, D. P. Black, and K. D. Glazebrook. Indexability and index heuristics for a simple class of inventory routing problems. *Oper. Res.*, 57(2):314–326, 2009.

[7] J. R. Artalejo, A. Economou, and M. J. Lopez-Herrero. Analysis of a multiserver queue with setup times. *Queueing Systems*, 51(1-2):53–76, Oct. 2005.

[8] E. Arthurs and J. Kaufman. Sizing a message store subject to blocking criteria. In *Proc. Int. Symp. Computer Performance, Modeling, Measurements and Evaluation (IFIP Performance)*, pages 547–564, 1979.

[9] R. Atar. A diffusion regime with nondegenerate slowdown. *Oper. Res.*, 60(2):490–500, 2012.

[10] K. Avrachenkov, U. Ayesta, J. Doncel, and P. Jacko. Congestion control of tcp flows in internet routers by means of index policy. *Computer Networks*, 57(17):3463–3478, 2013.

[11] K. Avrachenkov and V. S. Borkar. A learning algorithm for the whittle index policy for scheduling web crawlers. In *Proc. Ann. Allerton Conf. Communication, Control and Computing*, pages 1001–1006, 2019.

[12] N. Ayyadevara, R. Dabas, A. Khan, and K. V. N. Sreenivas. Near-Optimal Algorithms for Stochastic Online Bin Packing. In *Proc. Int. Conf. Automata, Languages and Programming (ICALP)*, volume 229, pages 12:1–12:20, 2022.

[13] C. Bandi, D. Bertsimas, and N. Youssef. Robust queueing theory. *Operations Research*, 63(3):676–700, 2015.

[14] N. Bashir, N. Deng, K. Rzadca, D. Irwin, S. Kodak, and R. Jnagal. Take it to the limit: Peak prediction-driven resource overcommitment in datacenters. In *Proc. European Conf. Computer Systems (EuroSys)*, page 556–573, Online Event, United Kingdom, 2021.

[15] N. G. Bean, R. J. Gibbens, and S. Zachary. Asymptotic analysis of single resource loss systems in heavy traffic, with applications to integrated networks. *Adv. Appl. Probab.*, 27(1):273–292, Mar. 1995.

[16] N. Benameur, S. Fredj, F. Delcoigne, S. Oueslati-Boulahia, and J. Roberts. Integrated admission control for streaming and elastic traffic. In *Int. Workshop Quality of Future Internet Services (QofIS)*, COST 263, pages 69–81, Berlin, Heidelberg, 2001.

[17] W. Bischof. Analysis of $M/G/1$-queues with setup times and vacations under six different service disciplines. *Queueing Systems*, 39(4):265–301, 2001.

[18] O. J. Boxma and W. P. Groenendijk. Pseudo-conservation laws in cyclic-service systems. *Journal of Applied Probability*, 24(4):949–964, 1987.

[19] A. Braverman. The prelimit generator comparison approach of stein's method. *Stoch. Syst.*, 12(2):181–204, 2022.

[20] A. Braverman. The join-the-shortest-queue system in the halfin-whitt regime: Rates of convergence to the diffusion limit. *Stochastic Systems*, 13(1):1–39, 2023.

[21] A. Braverman and J. Dai. Stein's method for steady-state diffusion approximations of $m/Ph/n+m$ systems. *Ann. Appl. Probab.*, 27:550–581, Feb. 2017.

[22] A. Braverman, J. Dai, and M. Miyazawa. The bar-approach for multiclass queueing networks with sbp service policies. *arXiv preprint arXiv:2302.05791*, 2023.

[23] A. Braverman, J. G. Dai, and J. Feng. Stein's method for steady-state diffusion approximations: an introduction through the Erlang-A and Erlang-C models. *Stoch. Syst.*, 6(2):301–366, 2017.

[24] A. Braverman, J. G. Dai, and M. Miyazawa. Heavy traffic approximation for the stationary distribution of a generalized Jackson network: The BAR approach. *Stochastic Systems*, 7(1):143–196, June 2017.

[25] P. H. Brill and L. Green. Queues in which customers receive simultaneous service from a random number of servers: A system point approach. *Manage. Sci.*, 30(1):51–68, 1984.

[26] D. B. Brown and J. E. Smith. Index policies and performance bounds for dynamic selection problems. *Management Science*, 66(7):3029–3050, 2020.

[27] N. Buchbinder, Y. Fairstein, K. Mellou, I. Menache, and J. S. Naor. Online virtual machine allocation with lifetime and load predictions. *ACM SIGMETRICS Perform. Evaluation Rev.*, 49(1):9–10, May 2021.

[28] G. Choudhury. On a batch arrival Poisson queue with a random setup time and vacation period. *Computers & Operations Research*, 25(12):1013–1026, Dec. 1998.

[29] G. Cloud. Overcommitting CPUs on sole-tenant VMs. https://cloud.google.com/compute/docs/nodes/overcommitting-cpus-sole-tenant-vms, 2023.

[30] G. Cloud. Virtual machine instances. https://cloud.google.com/compute/docs/instances, 2023.

[31] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Dynamic bin packing. *SIAM J. Comput.*, 12(2):227–258, 1983.

[32] C. Courcobetis and R. Weber. Stability of on-line bin packing with random arrivals and long-run-average constraints. *Probab. Eng. Inf. Sci.*, 4(4):447–460, 1990.

[33] C. Courcoubetis and R. R. Weber. Necessary and sufficient conditions for stability of a bin-packing system. *J. Appl. Probab.*, 23(4):989–999, 1986.

[34] M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, Dec./1997.

[35] J. Csirik, D. S. Johnson, C. Kenyon, J. B. Orlin, P. W. Shor, and R. R. Weber. On the sum-of-squares algorithm for bin packing. *J. ACM*, 53(1):1–65, Jan. 2006.

[36] J. G. Dai, A. Dieker, and X. Gao. Validity of heavy-traffic steady-state approximations in many-server queues with abandonment. *Queueing Systems*, 78:1–29, 2014.

[37] A. Dasylva and R. Srikant. Bounds on the performance of admission control and routing policies for general topology networks with multiple call centers. In *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*, volume 2, pages 505–512, New York, NY, 1999.

[38] A. Daw, B. Fralix, and J. Pender. Non-stationary queues with batch arrivals. *arXiv preprint arXiv:2008.00625*, 2020.

[39] A. Daw, R. C. Hampshire, and J. Pender. How to staff when customers arrive in batches. *arXiv preprint arXiv:1907.12650*, 2019.

[40] A. Daw and J. Pender. On the distributions of infinite server queues with batch arrivals. *Queueing Syst.*, 91(3–4):367–401, apr 2019.

[41] C. Delimitrou and C. Kozyrakis. Quasar: Resource-efficient and QoS-aware cluster management. In *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, page 127–144, Salt Lake City, UT, 2014.

[42] B. T. Doshi. A note on stochastic decomposition in a $GI/G/1$ queue with vacations or set-up times. *Journal of Applied Probability*, 22(2):419–428, June 1985.

[43] B. T. Doshi. Queueing systems with vacations — a survey. *Queueing Systems*, 1(1):29–66, June 1986.

[44] B. T. Doshi. Generalizations of the stochastic decomposition results for single server queues with vacations. *Communications in Statistics. Stochastic Models*, 6(2):307–333, Jan. 1990.

[45] F. Dufour and A. B. Piunovskiy. Impulsive control for continuous-time markov decision processes. *Adv. Appl. Probab.*, 47(1):106–127, 2015.

[46] A. Eryilmaz and R. Srikant. Asymptotically tight steady-state queue length bounds implied by drift conditions. *Queueing Syst.*, 72(3-4):311–359, Dec. 2012.

[47] D. Filippopoulos and H. Karatza. A two-class parallel queue with pure space sharing among rigid jobs and general service times. In *Proc. EAI Int. Conf. Performance Evaluation Methodologies and Tools (VALUETOOLS)*, pages 2–es, New York, NY, 2006.

[48] A. S. Foundation. Apache mesos: Containerizers. https://mesos.apache.org/documentation/latest/containerizers/, 2023.

[49] A. S. Foundation. Apache mesos: Oversubscription. https://mesos.apache.org/documentation/latest/oversubscription/, 2023.

[50] D. Freund and S. Banerjee. Good prophets know when the end is near. *Available at SSRN: https://ssrn.com/abstract=3479189*, Nov. 2019.

[51] S. W. Fuhrmann. A note on the $M/G/1$ queue with server vacations. *Operations Research*, 32(6):1368–1373, 1984.

[52] S. W. Fuhrmann and R. B. Cooper. Stochastic decompositions in the $M/G/1$ queue with generalized vacations. *Operations Research*, 33(5):1117–1129, Oct. 1985.

[53] A. Gandhi, S. Doroudi, M. Harchol-Balter, and A. Scheller-Wolf. Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward. *Queueing Systems*, 77(2):177–209, June 2014.

[54] A. Gandhi and M. Harchol-Balter. M/G/k with exponential setup. Technical Report CMU-CS-09-166, Carnegie Mellon University, Pittsburgh, PA, Sept. 2009.

[55] A. Gandhi and M. Harchol-Balter. $M/G/k$ with staggered setup. *Operations Research Letters*, 41(4):317–320, July 2013.

[56] A. Gandhi, M. Harchol-Balter, and I. Adan. Server farms with setup costs. *Performance Evaluation*, 67(11):1123–1138, Nov. 2010.

[57] N. Gast, B. Gaujal, and C. Yan. Exponential convergence rate for the asymptotic optimality of whittle index policy. *arXiv:2012.09064 [cs.PF]*, 2020.

[58] N. Gast, B. Gaujal, and C. Yan. LP-based policies for restless bandits: Necessary and sufficient conditions for (exponentially fast) asymptotic optimality. *arXiv:2106.10067 [math.OC]*, 2022.

[59] N. Gast, B. Gaujal, and C. Yan. Exponential asymptotic optimality of whittle index policy. *Queueing Systems*, 104(1):107–150, 2023.

[60] N. Gast, B. Gaujal, and C. Yan. Linear program-based policies for restless bandits: Necessary and sufficient conditions for (exponentially fast) asymptotic optimality. *Math. Oper. Res.*, 2023.

[61] J. Ghaderi, Y. Zhong, and R. Srikant. Asymptotic optimality of bestfit for stochastic bin packing. *ACM SIGMETRICS Perform. Evaluation Rev.*, 42(2):64–66, Sept. 2014.

[62] A. Ghosh, D. Nagaraj, M. Jain, and M. Tambe. Indexability is not enough for whittle: Improved, near-optimal algorithms for restless bandits. In *Proc. Int. Conf. Autonomous Agents and Multiagent Syst.*, page 1294–1302, Richland, SC, 2023.

[63] J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164, 1979.

[64] J. C. Gittins, K. D. Glazebrook, and R. Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[65] J. C. Gittins, K. D. Glazebrook, and R. R. Weber. *Multi-Armed Bandit Allocation Indices*. Wiley, Chichester, UK, second edition, 2011.

[66] J. C. Gittins and D. M. Jones. A dynamic allocation index for the sequential design of experiments. In J. M. Gani, K. Sarkadi, and I. Vincze, editors, *Progress in Statistics*, number 9 in Colloquia Mathematica Societatis János Bolyai, pages 241–266. North-Holland, Amsterdam, The Netherlands, 1974.

[67] J. C. Gittins and D. M. Jones. A dynamic allocation index for the sequential design of experiments. In J. Gani, editor, *Progress in Statistics*, pages 241–266. North-Holland, Amsterdam, 1974.

[68] K. D. Glazebrook. An analysis of Klimov's problem with parallel servers. *Mathematical Methods of Operations Research*, 58(1):1–28, Sept. 2003.

[69] K. D. Glazebrook, H. M. Mitchell, and P. S. Ansell. Index policies for the maintenance of a collection of machines by a set of repairmen. *European Journal of Operational Research*, 165(1):267–284, 2005.

[70] K. D. Glazebrook and J. Niño-Mora. Parallel scheduling of multiclass $M/M/m$ queues: Approximate and heavy-traffic optimization of achievable performance. *Operations Research*, 49(4):609–623, Aug. 2001.

[71] Google. Google Kubernetes Engine. https://cloud.google.com/kubernetes-engine, 2022.

[72] I. Grosof, M. Harchol-Balter, and A. Scheller-Wolf. Stability for Two-class Multiserver-job Systems. *arXiv:2010.00631 [cs.PF]*, Oct. 2020.

[73] I. Grosof, M. Harchol-Balter, and A. Scheller-Wolf. Wcfs: a new framework for analyzing multiserver systems. *Queueing Systems*, 102(1):143–174, 2022.

[74] I. Grosof, Z. Scully, M. Harchol-Balter, and A. Scheller-Wolf. Optimal scheduling in the multiserver-job model under heavy traffic. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6(3):1–32, Dec. 2022.

[75] I. Grosof, Z. Scully, M. Harchol-Balter, and A. Scheller-Wolf. Optimal scheduling in the multiserver-job model under heavy traffic. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(3), dec 2022.

[76] V. Gupta and A. Radovanović. Interior-point-based online stochastic bin packing. *Oper. Res.*, 68(5):1474–1492, 2020.

[77] S. Halfin and W. Whitt. Heavy-traffic limits for queues with many exponential servers. *Oper. Res.*, 29(3):567–588, 1981.

[78] M. Harchol-Balter. *Network Analysis without Exponentiality Assumptions*. PhD thesis, University of California, Berkeley, Berkeley, CA, Aug. 1996.

[79] M. Harchol-Balter and A. B. Downey. Exploiting process lifetime distributions for dynamic load balancing. *ACM Transactions on Computer Systems*, 15(3):253–285, Aug. 1997.

[80] Q.-M. He and E. Jewkes. Flow time in the $MAP/G/1$ queue with customer batching and setup times. *Communications in Statistics. Stochastic Models*, 11(4):691–711, Jan. 1995.

[81] Y. Hong and Z. Scully. Performance of the gittins policy in the g/g/1 and g/g/k, with and without setup times. *Performance Evaluation*, 163:102377, 2024.

[82] Y. Hong and W. Wang. Sharp waiting-time bounds for multiserver jobs. In *ACM Int. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc)*, Seoul, South Korea, 2022.

[83] Y. Hong, Q. Xie, Y. Chen, and W. Wang. Restless bandits with average reward: Breaking the uniform global attractor assumption. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 12810–12844. Curran Associates, Inc., 2023.

[84] Y. Hong, Q. Xie, Y. Chen, and W. Wang. Unichain and aperiodicity are sufficient for asymptotic optimality of average-reward restless bandits. *arXiv preprint arXiv:2402.05689*, 2024.

[85] Y. Hong, Q. Xie, and W. Wang. Near-optimal stochastic bin-packing in large service systems with time-varying item sizes. *Proc. ACM Meas. Anal. Comput. Syst.*, 7(3), dec 2023.

[86] W. Hu and P. Frazier. An asymptotically optimal index policy for finite-horizon restless bandits. *arXiv:1707.00205 [math.OC]*, 2017.

[87] P. J. Hunt and T. G. Kurtz. Large loss networks. *Stoch. Proc. Appl.*, 53(2):363 – 378, 1994.

[88] P. J. Hunt and C. N. Laws. Optimization via trunk reservation in single resource loss systems under heavy traffic. *Ann. Appl. Probab.*, 7(4):1058–1079, Nov. 1997.

[89] D. P. Kennedy. Rates of convergence for queues in heavy traffic. ii: Sequences of queueing systems. *Advances in Applied Probability*, 4(2):382–391, 1972.

[90] J. Kiefer and J. Wolfowitz. On the theory of queues with many servers. *Transactions of the American Mathematical Society*, 78(1):1–18, Jan. 1955.

[91] J. F. Kingman. On queues in heavy traffic. *Journal of the Royal Statistical Society: Series B (Methodological)*, 24(2):383–392, 1962.

[92] L. Kleinrock. *Queueing Systems*. John Wiley & Son, 1975.

[93] J. Köllerström. Heavy traffic theory for queues with several servers. i. *Journal of Applied Probability*, 11(3):544–552, 1974.

[94] J. Köllerström. Heavy traffic theory for queues with several servers. ii. *Journal of Applied Probability*, 16(2):393–401, 1979.

[95] T. Lattimore and C. Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[96] H. Li and Y. Zhu. A new approach to the $G/G/1$ queue with generalized setup time and exhaustive service. *Journal of Applied Probability*, 31(4):1083–1097, Dec. 1994.

[97] Y. Li and D. A. Goldberg. Simple and explicit bounds for multi-server queues with universal $1/(1-\rho)$ scaling, June 2017.

[98] Y. Li, X. Tang, and W. Cai. On dynamic bin packing for resource allocation in the cloud. In *Proc. Ann. ACM Symp. Parallelism in Algorithms and Architectures (SPAA)*, page 2–11, Prague, Czech Republic, 2014.

[99] M. Lin, A. Wierman, and B. Zwart. Heavy-traffic analysis of mean response time under shortest remaining processing time. *Performance Evaluation*, 68(10):955–966, Oct. 2011.

[100] S.-H. Lin, M. Paolieri, C.-F. Chou, and L. Golubchik. A model-based approach to streamlining distributed training for asynchronous sgd. In *IEEE Int. Symp. Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 306–318, Milwaukee, WI, 2018.

[101] X. Liu. *Steady State Analysis of Load Balancing Algorithms in the Heavy Traffic Regime*. PhD thesis, Arizona State University, 2019.

[102] X. Liu, K. Gong, and L. Ying. Steady-state analysis of load balancing with coxian-2 distributed service times. *Nav. Res. Log.*, 69(1):57–75, 2022.

[103] X. Liu and L. Ying. Steady-state analysis of load-balancing algorithms in the sub-Halfin–Whitt regime. *J. Appl. Probab.*, 57(2):578–596, 2020.

[104] X. Liu and L. Ying. Universal scaling of distributed queues under load balancing in the super-halfin-whitt regime. *IEEE/ACM Trans. Netw.*, 30(1):190–201, 2022.

[105] D. Lo, L. Cheng, R. Govindaraju, P. Ranganathan, and C. Kozyrakis. Heracles: Improving resource efficiency at scale. In *Prof. ACM/IEEE Ann. Int. Symp. Computer Architecture (ISCA)*, pages 450–462, 2015.

[106] R. Loulou. Multi-channel queues in heavy traffic. *Journal of Applied Probability*, 10(4):769–777, 1973.

[107] R. M. Loynes. The stability of a queue with non-independent inter-arrival and service times. *Mathematical Proceedings of the Cambridge Philosophical Society*, 58(3):497–520, July 1962.

[108] S. T. Maguluri and R. Srikant. Scheduling jobs with unknown duration in clouds. In *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*, pages 1887–1895, Turin, Italy, 2013.

[109] S. T. Maguluri and R. Srikant. Heavy traffic queue length behavior in a switch under the maxweight algorithm. *Stoch. Syst.*, 6(1):211–250, 2016.

[110] S. T. Maguluri, R. Srikant, and L. Ying. Stochastic models of load balancing and scheduling in cloud computing clusters. In *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*, pages 702–710, Orlando, FL, 2012.

[111] S. T. Maguluri, R. Srikant, and L. Ying. Heavy traffic optimal resource allocation algorithms for cloud computing clusters. *Perform. Eval.*, 81:20–39, 2014.

[112] A. Melikov. Computation and optimization methods for multiresource queues. *Cybern. Syst. Anal.*, 32(6):821–836, 1996.

[113] R. G. Miller. A contribution to the theory of bulk queues. *J. Roy. Statist. Soc. Ser. B*, 21(2):320–337, 1959.

[114] M. Miyazawa. Decomposition formulas for single server queues with vacations : A unified approach by the rate conservation law. *Communications in Statistics. Stochastic Models*, 10(2):389–413, Jan. 1994.

[115] M. Miyazawa. Rate conservation laws: A survey. *Queueing Systems*, 15(1):1–58, Mar. 1994.

[116] M. Miyazawa. Diffusion approximation for stationary analysis of queues and their networks: A review. *Journal of the Operations Research Society of Japan*, 58(1):104–148, 2015.

[117] E. Morozov and A. S. Rumyantsev. Stability analysis of a MAP/M/s cluster model by matrix-analytic method. In *European Workshop Computer Performance Engineering (EPEW)*, volume 9951, pages 63–76, Chios, Greece, Oct. 2016.

[118] E. Morozov and B. Steyaert. *Stability Analysis of Regenerative Queueing Models: Mathematical Methods and Applications*. Springer, Cham, Switzerland, 2021.

[119] S. V. Nagaev. On the speed of convergence in a boundary problem. i. *Theory of Probability & Its Applications*, 15(2):163–186, 1970.

[120] S. V. Nagaev. On the speed of convergence in a boundary problem. ii. *Theory of Probability & Its Applications*, 15(3):403–429, 1970.

[121] D. Olliaro, M. Ajmone Marsan, S. Balsamo, and A. Marin. The saturated multiserver job queuing model with two classes of jobs: Exact and approximate results. *Perform. Eval.*, 162:102370, 2023.

[122] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of optimal queuing network control. *Math. Oper. Res.*, 24(2):293–305, 1999.

[123] K. Park and W. Willinger, editors. *Self-Similar Network Traffic and Performance Evaluation*. Wiley, New York, NY, 2000.

[124] J. Pender and T. Phung-Duc. A law of large numbers for M/M/c/Delayoff-Setup queues with nonstationary arrivals. In S. Wittevrongel and T. Phung-Duc, editors, *Analytical and Stochastic Modelling Techniques and Applications*, volume 9845, pages 253–268. Springer, Cham, Switzerland, 2016.

[125] D. L. Peterson. Data center I/O patterns and power laws. In *22nd International Computer Measurement Group Conference*, pages 1034–1045, San Diego, CA, Dec. 1996. Computer Measurement Group.

[126] L. Ponomarenko, C. S. Kim, and A. Melikov. *Performance analysis and optimization of multi-traffic on communication networks*. Springer Science & Business Media, 2010.

[127] K. Psychas and J. Ghaderi. On non-preemptive VM scheduling in the cloud. In *Proc. ACM SIGMETRICS Int. Conf. Measurement and Modeling of Computer Systems*, page 67–69, Irvine, CA, 2018.

[128] K. Psychas and J. Ghaderi. Scheduling jobs with random resource requirements in computing clusters. In *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*, pages 2269–2277, 2019.

[129] K. Psychas and J. Ghaderi. High-throughput bin packing: Scheduling jobs with random resource demands in clusters. *IEEE/ACM Trans. Netw.*, 29(1):220–233, 2021.

[130] K. Psychas and J. Ghaderi. A theory of auto-scaling for resource reservation in cloud services. *Stoch. Syst.*, 12(3):227–252, 2022.

[131] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proc. ACM Symp. Cloud Computing (SoCC)*, San Jose, CA, 2012.

[132] R. Righter and J. G. Shanthikumar. Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures. *Probability in the Engineering and Informational Sciences*, 3(3):323–333, July 1989.

[133] A. Rumyantsev and E. Morozov. Stability criterion of a multiserver model with simultaneous service. *Ann. Oper. Res.*, 252(1):29–39, 2017.

[134] K. Rzadca, P. Findeisen, J. Swiderski, P. Zych, P. Broniek, J. Kusmierek, P. Nowak, B. Strack, P. Witusowski, S. Hand, and J. Wilkes. Autopilot: Workload autoscaling at Google. In *Proc. European Conf. Computer Systems (EuroSys)*, Heraklion, Greece, 2020.

[135] L. E. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3):687–690, June 1968.

[136] Z. Scully. *A New Toolbox for Scheduling Theory*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, Aug. 2022.

[137] Z. Scully, I. Grosof, and M. Harchol-Balter. The Gittins policy is nearly optimal in the M/G/k under extremely general conditions. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(3), Nov. 2020.

[138] Z. Scully, I. Grosof, and M. Harchol-Balter. Optimal multiserver scheduling with unknown job sizes in heavy traffic. *Performance Evaluation*, 145, Jan. 2021.

[139] Z. Scully and M. Harchol-Balter. The Gittins policy in the M/G/1 queue. In *19th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2021)*, pages 248–255, Philadelphia, PA, Oct. 2021. IFIP.

[140] Z. Scully, M. Harchol-Balter, and A. Scheller-Wolf. SOAP: One clean analysis of all age-based scheduling policies. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(1), Apr. 2018.

[141] Z. Scully, M. Harchol-Balter, and A. Scheller-Wolf. Simple near-optimal scheduling for the M/G/1. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(1), May 2020.

[142] Z. Scully, L. van Kreveld, O. J. Boxma, J.-P. Dorsman, and A. Wierman. Characterizing policies with optimal response time tails under heavy-tailed job sizes. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(2), June 2020.

[143] K. Sigman. One-dependent regenerative processes and queues in continuous time. *Mathematics of Operations Research*, 15(1):175–189, 1990.

[144] C. Stein. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*, volume 6, pages 583–603. University of California Press, 1972.

[145] A. L. Stolyar. An infinite server system with general packing constraints. *Oper. Res.*, 61(5):1200–1217, 2013.

[146] A. L. Stolyar. Large-scale heterogeneous service systems with general packing constraints. *Adv. Appl. Probab.*, 49:61–83, Mar. 2017.

[147] A. L. Stolyar and Y. Zhong. A large-scale service system with packing constraints: Minimizing the number of occupied servers. *ACM SIGMETRICS Perform. Evaluation Rev.*, 41(1):41–52, June 2013.

[148] A. L. Stolyar and Y. Zhong. Asymptotic optimality of a greedy randomized algorithm in a large-scale service system with general packing constraints. *Queueing Syst.*, 79:117–143, June 2015.

[149] A. L. Stolyar and Y. Zhong. A service system with packing constraints: Greedy randomized algorithm achieving sublinear in scale optimality gap. *Stoch. Syst.*, 11:83–111, June 2021.

[150] O. M. Tikhonenko. Generalized Erlang problem for service systems with finite total capacity. *Probl. Inf. Transm.*, 41(3):243–253, 2005.

[151] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes. Borg: The next generation. In *Proc. European Conf. Computer Systems (EuroSys)*, Heraklion, Greece, 2020.

[152] S. Upadhyaya. Queueing systems with vacation: An overview. *International Journal of Mathematics in Operational Research*, 9(2):167, 2016.

[153] M. van der Boor, M. Zubeldia, and S. Borst. Zero-wait load balancing with sparse messaging. 48(3):368–375, 2020.

[154] N. M. van Dijk. Blocking of finite source inputs which require simultaneous servers with general think and holding times. *Oper. Res. Lett.*, 8(1):45 – 52, Feb. 1989.

[155] R. van Vreumingen. Queueing systems with non-standard service policies and server vacations. Master's thesis, University of Amsterdam, Amsterdam, The Netherlands, Nov. 2019.

[156] I. M. Verloop. Asymptotically optimal priority policies for indexable and nonindexable restless bandits. *Ann. Appl. Probab.*, 26(4):1947–1995, 2016.

[157] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes. Large-scale cluster management at Google with Borg. In *Proc. European Conf. Computer Systems (EuroSys)*, page 18, Bordeaux, France, Apr. 2015.

[158] S. S. Villar, J. Bowden, and J. Wason. Multi-armed Bandit Models for the Optimal Design of Clinical Trials: Benefits and Challenges. *Statistical Science*, 30(2):199 – 215, 2015.

[159] W. Wang, S. T. Maguluri, R. Srikant, and L. Ying. Heavy-traffic insensitive bounds for weighted proportionally fair bandwidth sharing policies. *Math. Oper. Res.*, 47(4):2691–2720, 2022.

[160] W. Wang, Q. Xie, and M. Harchol-Balter. Zero queueing for multi-server jobs. *Proc. ACM Meas. Anal. Comput. Syst.*, 5(1), Feb. 2021.

[161] R. R. Weber and G. Weiss. On an index policy for restless bandits. *J. Appl. Probab.*, 27(3):637–648, 1990.

[162] P. D. Welch. On a generalized M/G/1 queuing process in which the first customer of each busy period receives exceptional service. *Operations Research*, 12(5):736–752, Oct. 1964.

[163] W. Weng and W. Wang. Achieving zero asymptotic queueing delay for parallel jobs. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(3), Nov. 2020.

[164] W. Weng, X. Zhou, and R. Srikant. Optimal load balancing with locality constraints. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(3), nov 2020.

[165] W. Whitt. Existence of limiting distributions in the $GI/GI/s$ queue. *Mathematics of Operations Research*, 7(1):88–94, Feb. 1982.

[166] W. Whitt. Blocking when service is required from several facilities simultaneously. *AT&T Tech. J.*, 64:1807 – 1856, 1985.

[167] P. Whittle. Restless bandits: activity allocation in a changing world. *J. Appl. Probab.*, 25:287 – 298, 1988.

[168] J. K. Williams, M. Harchol-Balter, and W. Wang. The M/M/k with deterministic setup times. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6(3):1–45, Dec. 2022.

[169] Q. Xie, X. Dong, Y. Lu, and R. Srikant. Power of d choices for large-scale bin packing: A loss model. In *Proc. ACM SIGMETRICS Int. Conf. Measurement and Modeling of Computer Systems*, pages 321–334, Portland, OR, 2015.

[170] L. Ying. Stein's method for mean field approximations in light and heavy traffic regimes. *Proc. ACM SIGMETRICS Int. Conf. Measurement and Modeling of Computer Systems*, 1(1):12:1–12:27, June 2017.

[171] G. Zayas-Cabán, S. Jasin, and G. Wang. An asymptotically optimal heuristic for general nonstationary finite-horizon restless multi-armed, multi-action bandits. *Advances in Applied Probability*, 51:745–772, 2019.

[172] X. Zhang and P. I. Frazier. Restless bandits with many arms: Beating the central limit theorem. *arXiv:2107.11911 [math.OC]*, 2021.

[173] X. Zhang and P. I. Frazier. Near-optimality for infinite-horizon restless bandits with many arms. *arXiv:2203.15853 [cs.LG]*, 2022.

[174] M. Zubeldia. Delay-optimal policies in partial fork-join systems with redundancy and random slowdowns. *Proc. ACM SIGMETRICS Int. Conf. Measurement and Modeling of Computer Systems*, 4(1), May 2020.

[175] N. Zychlinski, C. W. Chan, and J. Dong. Managing queues with different resource requirements. *Operations Research*, 71(4):1387–1413, 2023.