Truncating Multi-Dimensional Markov Chains with Accuracy Guarantee: Applications to Discriminatory Processor Sharing and Priority Queues

Gagan Somashekar[1] Mohammad Delasay[2] Anshul Gandhi[1]
{[1]PACE lab, Department of Computer Science,[2]College of Business}, Stony Brook University
[1]{gsomashekar,anshul}@cs.stonybrook.edu [2]mohammad.delasay@stonybrook.edu

***Abstract*—The ability to obtain the steady-state probability distribution of a Markov chain is invaluable for modern service providers who aim to satisfy arbitrary tail performance requirements. However, it is often challenging, and even intractable, to obtain the steady-state distribution for several classes of Markov chains, such as multi-dimensional and infinite state-space Markov chains with state-dependent transitions; two popular examples include the M/M/1 with Discriminatory Processor Sharing (DPS) and the preemptive M/M/c with multiple priority classes and customer abandonment. In this paper, we propose a Lyapunov function based state-space truncation technique for such Markov chains. Our technique leverages the available moments, or bounds on moments, of the state variables of the Markov chain to obtain tight truncation bounds while satisfying arbitrary probability mass guarantees for the truncated chain. We demonstrate the efficacy of our technique for the multi-dimensional DPS and M/M/c priority queue with abandonment, and highlight the significant reduction in state space (as much as 72%) afforded by our technique compared to the state-of-the-art.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Continuous-Time Markov chains (CTMCs) are widely used to model and analyze networked systems, such as processor-sharing (PS) systems and priority queue systems. A common approach to study the performance of these systems is to obtain the steady-state probability distribution of an underlying CTMC model. For example, to obtain *tail* measures (e.g., the tail queue length), which are the performance metric of choice for modern service operators such as those at Google [1] and Amazon [2], it is often necessary to first obtain the steady-state probability distribution by solving the balance equations governing the state transitions of a CTMC model.

Obtaining the exact steady-state probability distribution is not always practical or even possible. For many applications, the state space of the CTMC is infinite and multi-dimensional; exact analysis of such models is challenging. For CTMCs with specific structures, efficient numerical techniques exist for obtaining the exact steady-state distribution. For example, Matrix Analytic Methods are known to be efficient for solving CTMCs with a repeating pattern of transitions between adjacent states, including quasi-birth-and-death processes (QBDs, which are infinite state space multi-dimensional CTMCs in which states are organized into levels and transitions are skip-free between the levels) [3].

For chains with more general transitions, obtaining the exact steady-state probability distribution can be more challenging. For example, finding the distribution of the number of jobs in the system in the Discriminatory Processor Sharing (DPS) model (first introduced by Kleinrock in 1967 [4] and one of the models that we analyze in this paper) is still an open challenge. Likewise, obtaining the exact steady-state probability distributions of level-dependent QBDs (LDQBDs) is typically impossible [5, 6]. In fact, computing the steady-state probability distribution even for CTMCs with a finite but large state space can be computationally prohibitive [7]. For such chains, we instead resort to obtaining accurate approximations of the steady-state probability distribution.

An approach to approximate the steady-state probability distribution of multi-dimensional infinite CTMCs with general state transitions is to truncate their state space, in one or more dimensions, and then solve for the steady-state probability distribution of the *truncated* CTMC using existing analytical or numerical methods. Truncation algorithms have been proposed in the literature to carefully obtain truncation bounds such that the steady-state probability distribution of the truncated CTMC closely approximates that of the *original* infinite CTMC. For example, algorithms based on *Lyapunov* functions (see Section II) guarantee to provide truncation bounds that satisfy a desired accuracy with respect to the probability mass covered by the truncated chain [8]. In particular, if the maximum acceptable error due to truncation is $0 < \epsilon < 1$, the probability mass (or, sum of steady-state probabilities) of the states of the original (infinite) chain residing within the truncation bounds is guaranteed to be at least $(1 - \epsilon)$. An issue with such truncation techniques, as acknowledged by prior work [8], is that they lead to *loose truncation bounds*, which results in unnecessarily large truncated CTMCs and consequently, *expensive time and computational effort* to analyze them.

In this paper, we leverage available information about the moments (or bounds on moments) of the state variables of a CTMC to derive tighter truncation bounds while ensuring that these bounds satisfy the required probability mass guarantees. By leveraging the moments and using concepts from probability theory, specifically the Paley-Zygmund inequality [9],

we scale the *drift* of the Lyapunov function (the expected rate of change in its value) more efficiently to obtain much tighter truncation bounds without increasing the computational complexity, compared to the existing truncation techniques that are based on Lyapunov functions [8].

The design of our technique ensures that the resulted truncation bounds are, in theory, at least as tight as those obtained by the state-of-the-art. In practice, the truncation bounds obtained by our technique are significantly tighter than those obtained by existing techniques, allowing us to speed up the solving of the truncated CTMC (via solving of the relevant balance equations) by as much as $7\times$ when compared to the state-of-the-art.

We demonstrate the effectiveness of our proposed truncation technique by computing the steady-state probability distribution of the M/M/1-DPS system, which is a multi-class extension of the classic processor sharing system where the server capacity can be unequally shared, via user-specified weights, among different job or customer classes. The M/M/1-DPS system is known to be a "class of models notoriously hard to analyze in an exact manner" [10]. To the best of our knowledge, there has been no prior work on the approximation of the steady-state probability distribution of the DPS system with truncation error guarantee. We analyze the K-class DPS system (whose CTMC is K-dimensional) using our truncation technique leveraging the known moments of the queue-length distribution of the DPS system [11]. Across different parameter settings for $K = 2, 3, 4$ customer classes, we show that our technique achieves on average 34%, and up to 72%, tighter truncation bounds compared to those obtained when applying the state-of-the-art for the same desired accuracy.

We also apply our truncation technique to the M/M/c+M model (an M/M/c queue with exponential abandonment) with multiple priority classes and preemptive service policy, as an example of another system where our technique can be applied to obtain tighter truncation bounds. We show that our technique can reduce the size of the truncated state space by as much as 42%, while providing the same accuracy as the state-of-the-art. Interestingly, for the priority model, our truncation technique only requires the moments of the higher priority jobs, which are readily available [12].

We numerically validate the accuracy of our truncation technique, where possible. For example, the M/M/1-DPS with equal weights for customer classes reduces to the well-studied M/M/1-PS system. Likewise, the marginal distribution of the higher priority jobs under the M/M/c+M queue with preemptive priority reduces to the distribution of jobs in the standard M/M/c, which is known as the Erlang-A model. Across all validations, the maximum difference in per-state probability between our truncated CTMC and the original CTMC is about $3 \times 10^{-6}\%$ for the DPS system and about $9 \times 10^{-4}\%$ for the M/M/c+M queue with preemptive priority.

Finally, by leveraging our truncation technique, we conduct several performance analyses that are otherwise intractable, such as determining when the DPS system outperforms the PS system (and vice-versa) in terms of the tail of number of jobs in system, or comparing the tail performance of the M/M/1-FCFS system with that of the M/M/1-DPS system. Such performance analyses are crucial for designing customer-facing web applications that must meet strict tail performance targets [1, 2].

The rest of the paper is organized as follows. We discuss the background and related work, including the state-of-the-art, in Section II. We present our truncation technique in Section III, and apply it to obtain the steady-state probability distribution of the DPS system and the preemptive priority queue in Sections IV and V, respectively; we also present several use cases to demonstrate the performance analyses enabled by our truncation technique. We conclude this paper in Section VI.

II. Background and Prior Work

In this section, we first provide the necessary background and prior work on the two main approaches for state space reduction of infinite CTMCs, namely, state space *aggregation* and *truncation*. We then describe the state-of-the-art truncation technique of Dayar et al. [8]. When evaluating the efficiency of our technique in later sections, we primarily compare our truncation bounds with those obtained via Dayar et al.

A. State space reduction

When a CTMC with an infinite state space cannot be solved exactly, a natural alternative is to find an approximate solution by reducing the size of the state space. Prior research in this area has primarily focused on two key techniques: (i) state space *aggregation* (Section II-A1) and (ii) state space *truncation* (Section II-A2).

1) State space aggregation: The high-level idea in aggregation is to replace a subset of the state space of a CTMC with a single state. Courtois and Semal [13] present a *bounded aggregation* technique to find lower and upper bounds on the approximation error for individual state probabilities when the state space is large *but finite*. This technique approximates the transition rate matrix of a Markov chain, which is positive irreducible, using a block diagonal matrix. The eigenvectors of the blocks are then used to approximate the eigenvector of the original matrix.

The above work has been extended to analyze the availability of computer systems prone to failures [7, 14]. Muntz et al. [7] propose an aggregation technique motivated by the fact that for some Markov chains the majority of the probability mass is concentrated in a small subset of the state space. The authors thus aggregate the remaining states to obtain an accurate approximation and then derive a bound on the error resulting from this aggregation. Lui and Muntz [14] build on this approach and introduce an iterative algorithm that tightens the error bound.

Mahevas and Rubino [15] present a method to find bounds on various performance measures for complex computer systems by assuming that a partitioning of the state space into a small and a large subset is available. The large subset is aggregated into an auxiliary Markov chain with aggregated

transition rates. The authors then find lower and upper bounds on the steady-state performance measures of this new, reduced representation of the Markov chain.

Riska and Smirni [16] introduce a methodology to find the exact steady-state measures for M/G/1-type processes by aggregating the probability of finite subsets of the state space, which in turn is used to find the steady-state measures of various Markov reward functions. Buchholz [17] proposes an approach to find bounds on the steady-state performance measures by solving a finite subset of the original Markov chain exactly and aggregating the rest of the state space. However, this approach is only applicable to models that are overtake free, i.e., models for which the customers leave the system in the same order as they arrive.

A limitation of the above approaches (Lui and Muntz [14], Mahevas and Rubino [15], and Muntz et al. [7]) is the assumption that the state space can be decomposed by the user into two disjoint sets, with one set containing the states frequently visited by the system in steady state. Further, these approaches either help find the performance measures (Riska and Smirni [16]) or provide bounds on the performance measures (Buchholz [17], Lui and Muntz [14], Mahevas and Rubino [15], and Muntz et al. [7]) for Markov chains with infinite state space, but do not find bounds on the steady-state distribution. Thus, guarantees on the probability mass after aggregation cannot be immediately obtained.

2) State space truncation: A more popular approach for state space reduction is truncation, whereby the state space of a Markov chain is truncated along one or more dimensions. However, ad-hoc truncation can result in inaccurate approximations whereby the steady-state probability distribution and the associated performance measures (such as the mean queue length) of the truncated Markov chain are far from those of the original one.

There has been prior work on truncation techniques that provide some upper bound on the loss of accuracy due to truncation. Such techniques typically leverage the specific structure of the underlying CTMC to obtain an accurate truncation. Bright and Taylor [18] propose a numerical method to solve LDQBDs which involves iteratively finding a sufficiently large truncation level. However, the iterative method is computationally intensive, and, more concernedly, the authors explicitly state that the proposed method is not guaranteed to provide accurate results.

Lyapunov analysis has been often used in prior works to find bounds on the moments and tail probabilities for both discrete- and continuous-time Markov chains. Bertsimas et al. demonstrate how lower and upper bounds on the moments and tail probabilities of a discrete-time Markov chain can be obtained provided that a suitable Lyapunov function can be found [19]. One of the conditions on the Lyapunov function is a finite "jump size," i.e., the requirement that the maximum change in the value of the drift function is bounded. Maguluri and Srikant build on prior works [19, 21] to find an upper bound on tail probabilities of CTMCs [20]. However, their approach requires the drift of the Lyapunov function to have a finite lower *and* upper bound, thus restricting the applicability of the approach given the difficulty of finding Lyapunov functions even without these additional constraints [8, 22]. By contrast, our technique does not impose any requirements on the Lyapunov function. In fact, for the Lyapunov functions we employ throughout this paper, the drift is unbounded from below.

B. State-of-the-art in CTMC truncation

The state-of-the-art in terms of truncation methods is the work of Dayar et al. from 2011 [8]. Dayar et al. is based on leveraging Lyapunov functions to obtain truncation bounds with probability mass guarantees for LDQBDs. The central idea is to identify a subset of states towards which the CTMC drifts, and then truncate the infinite state space to ensure that this subset is part of the truncated CTMC. Our truncation technique builds on this work, and so we provide an overview of Dayar et al. below.

Let $\{N(t), t \geq 0\}$ be an ergodic k-dimensional CTMC with state space S and generic state $n = (n_1, n_2, \ldots, n_k)$. Let N_i denote the random variable corresponding to n_i, with $N(t) = (N_1(t), \ldots, N_k(t))$. Let $\pi(n) = \pi(n_1, n_2, ..., n_k)$ denote the steady-state probability of being in state n, and let Q denote the infinitesimal generator matrix. Without loss of generality, assume that the CTMC is infinite in the first m-dimensions and finite in the remaining $(k - m)$ dimensions.

The stability of a Markov chain can be established if a ***Lyapunov function*** that maps the state space to positive real numbers is found such that its ***drift*** (the expected rate of change in the value of the Lyapunov function in a state) is negative outside a finite subset of the state space and is bounded in this finite subset; such a finite subset is referred to as the ***attractor set***, C. Formally, if $N(t)$ is ergodic, there exists a Lyapunov function $g : S \to R_{\geq 0}$ and a set $C \subset S$ such that the following conditions hold for some $\gamma > 0$ [8]:

(i) $d(n) \leq -\gamma$, $\forall n \in \overline{C}$, where $\overline{C} = S \setminus C$,
(ii) $d(n) < \infty$, $\forall n \in C$, and
(iii) $\{n \in S \mid g(n) \leq r\}$ is finite, $\forall r < \infty$,

where $d(n)$ denotes the value of the drift function in state n:

$$d(n) = (\mathrm{d}/\mathrm{d}t)\ E[g(N(t)) \mid N(t) = n], \tag{1}$$

where $E[X]$ denotes the expectation of X. Dayar et al. use the above conditions to derive an upper bound on the probability mass (sum of steady-state probabilities of all states) in $\overline{C}$. The authors define a function $g^*(n) = g(n)/(c + \gamma)$, where $c = \sup_{n \in S} d(n)$ (note that c is finite from condition (ii)) and γ is as defined in condition (i). Figure 1 illustrates the above concepts pictorially, where the x-axis and y-axis correspond to the state variable and the value of the drift function, respectively. The parameter γ that partitions the state space into the attractor set and its complement is also shown in the figure. All the states whose value of the drift function is above the line corresponding to $-\gamma$ can be grouped into an attractor set.

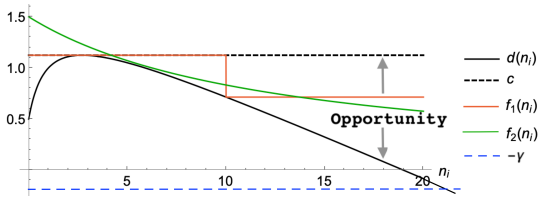


Fig. 1: Illustration of the drift ($d(n_i)$), the supremum of the drift (c), the parameter γ, and our state-dependent drift bounds ($f_1(n_i)$ and $f_2(n_i)$).

Using conditions (i)–(iii), the authors derive an upper bound on the probability mass outside the attractor set C, thereby providing a lower bound on the probability mass inside C. Conditions (i) and (ii) and the fact that $c = \sup_{n\in S} d(n)$ yield:

$$d^*(n) = \frac{d(n)}{c+\gamma} \leq \frac{c}{c+\gamma}, \qquad \forall n \in C, \tag{2}$$

$$d^*(n) \leq -\frac{\gamma}{c+\gamma}, \qquad \forall n \in \overline{C}. \tag{3}$$

Combining Eqs. (2) and (3) gives:

$$d^*(n) \leq \frac{c}{c+\gamma} - I_{\overline{C}}\ , \tag{4}$$

where $I_{\overline{C}} = 1$ if $n \in \overline{C}$ and 0 otherwise. If d, g, and π are the vectors of drift function values, Lyapunov function values, and steady-state probabilities for all states, respectively, by the definition of drift in Eq. (1) and the fact that $\pi Q = 0$ [23], we have:

$$d^T = Qg^T \implies \pi d^T = \pi Q g^T = 0 \implies \pi d^{*T} = \pi Q g^{*T} = 0. \tag{5}$$

Using Eqs. (4) and (5), a bound on the probability mass in $\overline{C}$ is obtained as follows:

$$0 = \sum_{n\in S} d^*(n)\cdot\pi(n) \leq \sum_{n\in S}\pi(n)\cdot\frac{c}{c+\gamma} - \sum_{n\in\overline{C}}\pi(n)$$
$$\implies \sum_{n\in\overline{C}}\pi(n) \leq \sum_{n\in S}\pi(n)\cdot\frac{c}{c+\gamma} = \frac{c}{c+\gamma}. \tag{6}$$

This guarantees that the probability mass in C is at least $1 - c/(c+\gamma)$. Hence, the value of γ obtained by solving $c/(c+\gamma) = \epsilon$, where $0 < \epsilon < 1$, guarantees that a truncated CTMC containing C has at least $(1-\epsilon)$ fraction of the probability mass, and thus loses at most ϵ fraction of the probability mass after truncation. Once γ is found, the set C can be found as follows:

$$C = \{n \in S \mid d(n) > -\gamma\}. \tag{7}$$

Omitting the states outside the attractor set C truncates the CTMC from "below" and from "above." For example, for a k-dimensional CTMC with positive state variables (i.e., $n_i \geq 0$ for $i = 1, 2, \ldots, k$), C might only contain states $n = (n_1, n_2, \ldots, n_k)$ such that $10 \leq n_i \leq 100$ for state variable N_i.

Note that Eq. (6) only provides an upper bound on the probability mass in $\overline{C}$; the actual probability mass in $\overline{C}$ could be much smaller than $c/(c+\gamma)$, as Dayar et al. acknowledge in their work [8]. Indeed, our experiments in Section IV show that the truncation bounds obtained via the above technique are quite loose. The goal of our work is to address this issue and provide tighter truncation bounds.

III. Our truncation technique

This section presents our truncation technique. We start with an overview of our technique in Section III-A. We provide specific truncation bounds to obtain probability mass guarantees by leveraging the state variable moments (or their bounds) of the Markov chain along one or multiple dimensions in Sections III-B and III-C. For reference, we illustrate in Appendix A the step-by-step algorithm of our truncation technique with application to the priority queue (the system that we will investigate in Section V).

A. Overview of the technique

Figure 1 illustrates the high-level idea of our state space truncation technique for the Discriminatory Processor Sharing (DPS) system that we analyze later in Section IV. The solid black line is the drift as a function of the state variable, $d(n_i)$. The state-of-the-art obtains truncation bounds by bounding the drift function with the trivial upper bound of $c = \sup_{n\in S} d(n)$ (the dashed black line). The advantage of using the supremum is that the bound on $\sum_{n\in\overline{C}}\pi(n)$ in Eq. (6) can be easily obtained as $\sum_{n\in S}\pi(n)\cdot c/(c+\gamma) = c/(c+\gamma)$. However, there is clearly a *gap* between the drift and the supremum, which tends to grow larger for higher values of n_i, as highlighted in Figure 1. The drift is a state-dependent function, but the supremum is a fixed function that does *not* adapt to changes in the state variate, thus making it a loose upper bound of the drift.

The key idea in our technique is to employ a *state-dependent bounding function* that mimics, to some extent, the changes in the drift function in response to the state variable to provide tighter upper bounds of the drift function; examples of such state-dependent bounding functions include a step function and a decaying function (e.g., $f_1(n_i)$ and $f_2(n_i)$ in Figure 1). However, when using a generic state-dependent bounding function, $f(n)$, in place of c in Eq. (6), the upper bound on the probability mass in $\overline{C}$ may not be easily obtained in closed-form, making it difficult to solve for the set C. We formalize this challenge below.

Generic bounding functions: Consider a generic state-dependent bounding function $f(n)$ that bounds the drift $d(n)$, i.e., $f(n) \geq \max(d(n), 0)$, $\forall n \in S$. Expanding Eq. (5) gives us:

$$0 = \sum_{n\in S}\pi(n)\cdot d(n) = \sum_{n\in C}\pi(n)\cdot d(n) + \sum_{n\in\overline{C}}\pi(n)\cdot d(n)$$
$$\implies 0 \leq \sum_{n\in C}\pi(n)\cdot d(n) - \gamma\sum_{n\in\overline{C}}\pi(n) \quad \because d(n) \leq -\gamma \quad \forall n \in \overline{C}$$
$$\implies \sum_{n\in\overline{C}}\pi(n) \leq \frac{\sum_{n\in C}\pi(n)\cdot f(n)}{\gamma} \quad \because f(n) \geq \max(d(n), 0). \tag{8}$$

The truncated CTMC consisting of the set C can now be obtained by setting the right-hand-side of Eq. (8) to ϵ, solving for γ, and then using Eq. (7). However, this requires knowing the exact value or an upper bound of the term $\sum_{n\in C}\pi(n)\cdot f(n)$. We show, in subsections III-B and III-C, examples of one-dimensional and two-dimensional generic bounding functions $f(n)$ that result in simple expressions for $\sum_{n\in C}\pi(n)\cdot f(n)$. In all the cases, the final expressions are in terms of the moments of the state variables.

Reliance on the CTMC moments: We acknowledge the reliance of our technique on the knowledge of the moments of the original CTMC's state variables. However, our technique can also be applied to CTMCs for which a lower bound on the moments is known. In general, knowing the moments is not enough to obtain the steady-state probability distribution; the DPS system is an example of a CTMC for which the moments of the number of jobs are known [11], but its exact steady-state distribution is as yet unknown. The preemptive M/M/c priority queue with abandonment is another CTMC for which some of the moments are known (e.g., for the highest priority jobs, since they constitute the classic M/M/c queue), but obtaining the steady-state probability distribution is computationally prohibitive, especially for $c > 2$ [24, 25].

The steady-state probability distribution of a CTMC can be invaluable to practitioners. When designing performant computing systems, it is not enough to simply focus on the expected value of the performance metric [1, 26]. For modern customer-facing online services, e.g., Amazon and Google, performance metrics typically take the form of tail probabilities [2, 27], thus requiring the steady-state probability distribution. In fact, to achieve predictably good performance, it is necessary to know the entire distribution of the performance metrics (e.g., the queue size distribution) [28].

B. Truncation bounds based on the moments

We now employ a specific function, a simple *step function*, along one of the dimensions of a CTMC to bound its drift function. Consider a 1-D step function (depicted in Figure 1 as $f_1(n)$) that initially is equal to the supremum and then drops to a lower value, c_1, along one dimension of the state space, resulting in a tighter bounding of the drift for larger values of the state variate. Note that the step function subsumes the supremum function used by Dayar et al. [8].

Consider a k-dimensional CTMC with state space S with a generic state denoted by $n = (n_1, n_2, \ldots, n_k)$, and let the CTMC be infinite in m-dimensions and finite in the remaining $(k-m)$ dimensions. We denote with N_j the random variable corresponding to the j^{th} dimension of the state space, n_j. We improve the upper bound of the drift along an arbitrary infinite dimension, say dimension i, corresponding to N_i. We formally define the step function, which drops to $c_1 \le c$ for $R = \{n \mid n_i > \overline{n}\}$, where $\overline{n}$ is a parameter, as:

$$f_1(n) = \begin{cases} c & = \sup_{\forall n\in S} d(n), \quad \forall n \in \overline{R}, \\ c_1 & = \sup_{\forall n\in R} d(n), \quad \forall n \in R \end{cases} \tag{9}$$

Substituting $f_1(n)$ in place of $f(n)$ in Eq. (8) and rearranging the terms gives us:

$$\begin{aligned} \gamma \sum_{n\in\overline{C}}\pi(n) &\le c \sum_{n\in C\cap\overline{R}}\pi(n) + c_1 \sum_{n\in C\cap R}\pi(n) \\ &= c\sum_{n\in\overline{R}}\pi(n) + c_1\sum_{n\in R}\pi(n) - c\sum_{n\in\overline{C}\cap\overline{R}}\pi(n) - c_1\sum_{n\in\overline{C}\cap R} \\ &\le c\sum_{n\in\overline{R}}\pi(n) + c_1\sum_{n\in R}\pi(n) - c_1\sum_{n\in\overline{C}}\pi(n) \quad \because -c \le -c \\ \implies \sum_{n\in\overline{C}}\pi(n) &\le \frac{c}{c_1+\gamma}\sum_{n\in\overline{R}}\pi(n) + \frac{c_1}{c_1+\gamma}\sum_{n\in R}\pi(n) \\ &= \frac{c}{c_1+\gamma} - \frac{c-c_1}{c_1+\gamma}\sum_{n\in R}\pi(n) \end{aligned} \tag{10}$$

Recall from Section II-B that $\overline{C}$ is the set of states outside the attractor set. Thus, $\sum_{n\in\overline{C}}\pi(n)$ represents the probability mass outside the attractor set. To obtain the probability mass guarantee on $\sum_{n\in\overline{C}}\pi(n)$, we require a lower bound on the tail probability, $\sum_{n\in R}\pi(n)$. While any applicable lower bound can be employed, we leverage a generic lower bound.

Definition 1: Paley-Zygmund inequality [9]: For a positive random variable X with finite variance and $0 \le \theta \le 1$, $\Pr(X > \theta E[X]) \ge (1-\theta)^2 E[X]^2/E[X^2]$.

Applying the Paley-Zygmund inequality for N_i and setting $\overline{n} = \theta E[N_i]$, we have, from Eq. (10):

$$\sum_{n\in\overline{C}}\pi(n) \le \frac{c}{c_1+\gamma} - \frac{c-c_1}{c_1+\gamma}(1-\theta)^2\frac{E[N_i]^2}{E[N_i^2]}. \tag{11}$$

The above upper bound on the probability mass in $\overline{C}$ (or the lower bound on the probability mass inside C) results in a tighter truncation bound compared to Dayar et al. when $\epsilon \le \frac{E[N_i]^2}{E[N_i^2]}$. In fact, we can state the following lemma (proof deferred to Appendix B):

Lemma 1: The truncation obtained via our 1-D step bounding function (given in Eq. (9)) is at least as tight as that obtained via the supremum bounding function, as employed in Dayar et al., when $\epsilon \le \frac{E[N_i]^2}{E[N_i^2]}$, where i is an arbitrary infinite dimension of the CTMC.

The upper bound on the probability mass in $\overline{C}$ given in Eq. (11) depends on the first and second moments of the marginal distribution of N_i. We apply this upper bound in Sections IV and V to the DPS and M/M/c+M with priority models, respectively, for which such moments are readily available.

C. Tighter truncation bounds using joint moments of state variables

In general, since the drift function is defined on the state space of the CTMC, it can be multi-variate (the illustration of drift in Figure 1 appears one-dimensional as it has been projected onto a single dimension). By bounding the drift in one dimension, it is likely that we obtain loose truncation bounds. To obtain tighter truncation bounds, we now consider multi-dimensional bounding functions. In such cases,

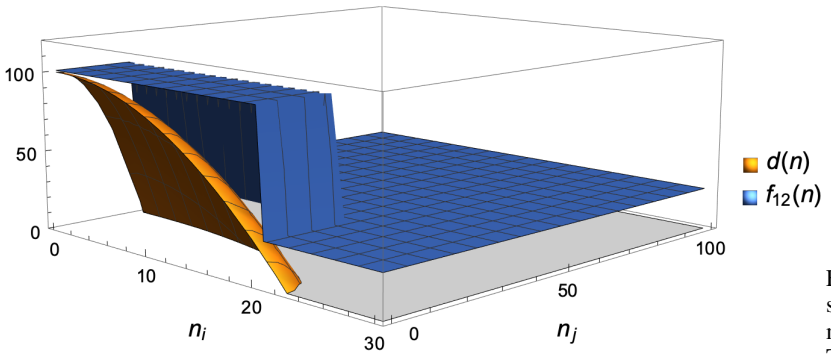


Fig. 2(a): Illustration of a 2-D drift bounding function.

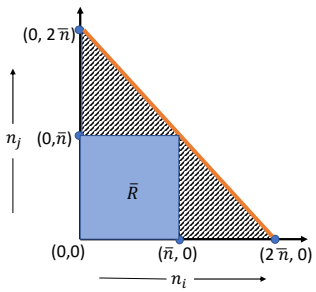


Fig. 2(b): Illustration of different subsets of the state space. The axes n_i and n_j are the two arbitrary dimensions along which the chain is being truncated. The shaded square corresponds to the set $\overline{R} = \{n \in S \mid \max\{n_i, n_j\} \leq \overline{n}\}$ and the region bounded by the orange line and the axes corresponds to the set $\overline{T} = \{n \in S \mid n_i + n_j \leq 2 \cdot \overline{n}\}$.

the joint moments of the state variables may be needed to obtain the truncation bounds. For simplicity, we consider two-dimensional bounding functions; however, our proposed technique can be extended to higher-dimensional functions.

1) Bounding the drift using a 2-D step function: Figure 2a plots a two-dimensional step function that bounds the drift; here, we assume that the CTMC is k-dimensional, $k > 1$, but we only show the bounding function for the two dimensions, say i and j, along which it takes a step. Mathematically, we define the 2-D step function as:

$$f_{12}(n) = \begin{cases} c & = \sup_{\forall n \in S} d(n), \quad \forall n \in \overline{R}, \\ c_1 & = \sup_{\forall n \in R} d(n), \quad \forall n \in R, \end{cases} \tag{12}$$

where $\overline{R} = \{n \in S \mid \max\{n_i, n_j\} \leq \overline{n}\}$ and $R = S \backslash \overline{R}$. Thus, the bounding function takes the value c in the "square" region (if projected onto two dimensions) defined by $n_i \leq \overline{n}$ and $n_j \leq \overline{n}$, and takes the value c_1 outside this region. Substituting Eq. (12) in the generic upper bound on the probability mass in $\overline{C}$ (Eq. (8)) gives us a result similar to Eq. (10):

$$\sum_{n \in \overline{C}} \pi(n) \leq \frac{c}{c_1 + \gamma} - \frac{c - c_1}{c_1 + \gamma} \sum_{n_i \in R} \pi(n) \tag{13}$$

To obtain a lower bound on the probability mass in R, we define the new set $\overline{T} = \{n \in S \mid n_i + n_j \leq 2 \cdot \overline{n}\}$ and $T = S \setminus \overline{T}$. Let $\pi(\mathcal{S})$ denote $\sum_{n \in \mathcal{S}} \pi(n)$. Since $T \subset R$, we have $\pi(R) \geq \pi(T)$. Figure 2b illustrates the different regions of the state space S. We now use the Paley-Zygmund inequality to find the lower bound on $\pi(T)$ by considering $Z = N_i + N_j$:

$$\begin{aligned} &P(Z > \theta E[Z]) \geq (1-\theta)^2 \frac{E[Z]^2}{E[Z^2]}, \text{ with } 2\overline{n} = \theta E[Z] \\ &\implies \pi(R) \geq \pi(T) = P(Z > \theta E[Z]) \geq (1-\theta)^2 \frac{E[Z]^2}{E[Z^2]}. \end{aligned} \tag{14}$$

Finally, substituting Eq. (14) in Eq. (13) gives us our upper bound on the probability mass in $\overline{C}$ as:

$$\sum_{n \in \overline{C}} \pi(n) \leq \frac{c}{c_1 + \gamma} - \frac{c - c_1}{c_1 + \gamma} (1-\theta)^2 \frac{E[Z]^2}{E[Z^2]} \tag{15}$$

The above bound is provably tighter compared to the state-of-the-art (proof deferred to Appendix B) when $\epsilon \leq \frac{E[Z]^2}{E[Z^2]}$:

Lemma 2: The truncation obtained via the 2-D step bounding function (given in Eq. (12)) is at least as tight as that obtained via the supremum bounding function, as employed in Dayar et al., when $\epsilon \leq \frac{E[Z]^2}{E[Z^2]}$ where $Z = N_i + N_j$ and N_i and N_j represent state variables corresponding to two arbitrary infinite dimensions of the CTMC.

2) Alternative approaches for obtaining the truncation bound for the 2-D step function: In the above, we obtained a lower bound on $\pi(R)$ via the lower bound on $\pi(T)$, where $T = \{n \in S \mid n_i + n_j > 2\overline{n}\}$. Alternatively, we considered the sets $\{n \in S \mid n_i^2 + n_j^2 > 2\overline{n}^2\}$ and $\{n \in S \mid n_i n_j > \overline{n}^2\}$ (in place of T). However, the higher order terms involved in the definition of these sets necessitated higher order moments in the Paley-Zygmund inequality, thereby resulting in a looser lower bound on $\pi(R)$.

We also considered using the fact that $\pi(R) = \pi(N_i > \overline{n}) + \pi(N_j > \overline{n}) - \pi(\min\{N_i, N_j\} > \overline{n})$. We again used the Paley-Zygmund inequality to obtain lower bounds on the first two terms and the two-variable Markov's inequality [29] to obtain an upper bound on the third term. However, the obtained truncation bounds for the DPS system, which we evaluate in Section IV, were not as tight as those derived from Eq. (15). Nonetheless, the above alternatives could be useful for other CTMCs; for example, if the problem at hand has a tighter upper bound on $\pi(\min\{N_i, N_j\} > \overline{n})$, the truncation bounds could be tighter than those obtained via Eq. (15).

IV. Applying our truncation technique to the DPS system

We now evaluate the efficacy of our truncation technique by applying the truncation bounds for the Discriminatory

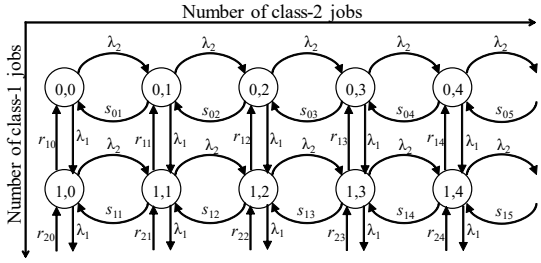


Fig. 3: M/M/1-DPS with two customer classes; for state (n_1, n_2), n_1 and n_2 are the number of class-1 and class-2 jobs, respectively.

Processor Sharing (DPS) system. We first consider the DPS system with two customer classes, for ease of exposition. However, our technique is not restricted to 2-dimensional CTMCs; we also apply our truncation technique to the DPS system with three and four customer classes in Section IV-B. We investigate the tightness of the drift bounding functions presented in Section III, and compare the resulting truncation bounds with those obtained from Dayar et al. [8]. We also present three use cases where we use the obtained steady-state probability distribution of the truncated DPS system to analyze its *tail* performance under different load conditions. Finally, we compare the tail performance of the DPS system with that of the M/M/1-PS and M/M/1-FCFS with priority, in order to determine the superior scheduling policy.

A. Overview of the DPS system

DPS was first introduced by Kleinrock as a generalisation to the processor sharing system to model time-sharing systems with priority [4]. While all customer classes receive equal server capacity in case of an egalitarian processor sharing policy, the server capacity under DPS is processor shared based on a given weight vector $\alpha = (\alpha_1, \alpha_2, ..., \alpha_k)$, where α_i is the weight associated with class-i customers. If there are N_i jobs of class-i, each class-j job gets a fraction $\alpha_j / \left(\sum_{i=1}^{k} \alpha_i N_i\right)$ of the server's capacity.

M/M/1-DPS system: Consider an M/M/1 system operating under the DPS policy with two customer classes, where the server's capacity is shared between two customer classes with the service priority expressed through weights α_1 and α_2. Arrivals for each customer class follow a Poisson distribution with mean $\lambda_i, i \in \{1, 2\}$, and the service times for each class follow an Exponential distribution with mean $1/\mu_i, i \in \{1, 2\}$.

Figure 3 shows the M/M/1-DPS CTMC in which a state is represented by the pair (n_1, n_2), where n_1 and n_2 are the number of jobs in system for classes 1 and 2, respectively; note that the CTMC is infinite in both dimensions. The transition rates from state (n_1, n_2) to $(n_1 - 1, n_2)$ and $(n_1, n_2 - 1)$ are $r_{n_1,n_2} = n_1\alpha_1\mu_1/(n_1\alpha_1 + n_2\alpha_2)$ and $s_{n_1,n_2} = n_2\alpha_2\mu_2/(n_1\alpha_1 + n_2\alpha_2)$, respectively.

Significance of the M/M/1-DPS system and the challenges in solving its underlying CTMC: The M/M/1-DPS system, with the proper choice of weights, has been shown to outperform the classical M/M/1-PS system for more than one customer class [30]. This has sparked interest in the community in the last decade to investigate the performance of DPS under various traffic regimes [31, 32, 33]. The DPS system can also be viewed as an idealization of the round-robin policy with priority groups where the time quantum shrinks to zero [4]. This interpretation makes it a natural candidate to model algorithms like Weighted Fair Queueing and Weighted Round Robin, further raising the interest in DPS models [34].

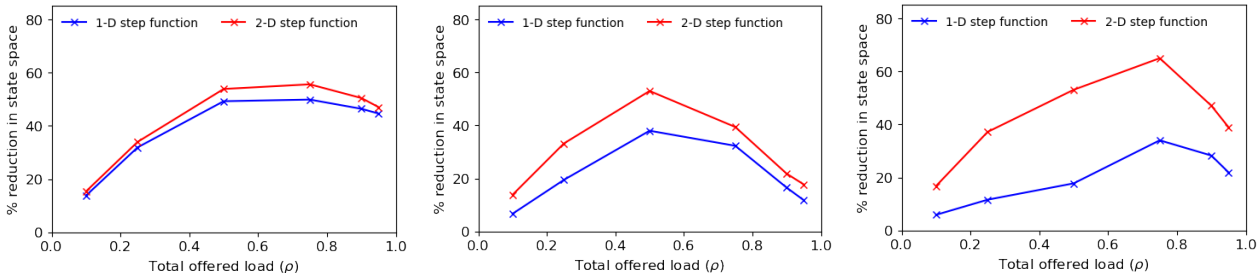
Despite the popularity of the DPS model, which was first introduced in the late 1960s [4], the exact steady-state probability distribution of its underlying CTMC continues to remain elusive. This is because of the complex and non-repeating structure of its multi-dimensional and infinite CTMC. Specifically, the per-class service rate transitions (r_{n_1,n_2} and s_{n_1,n_2} in Figure 3) depend on the current number of customers in each class. Exact analysis has been performed only for *finite* DPS queues [35]. To the best of our knowledge, there has been no prior work on the analysis of the *infinite* DPS system with truncation error guarantee.

B. Results for truncation bounds

Fortunately, the exact moments of M/M/1-DPS queue-length distribution are known [11], in terms of the solution to a system of linear equations. This allows us to apply our moment-based truncation bounds via Eqs. (11) and (15). To apply our truncation bounds (and the state-of-the-art truncation bounds [8] for comparison) to the M/M/1-DPS system, we employ the following feasible Lyapunov function, motivated by prior work in the stability literature [36]: $g(n_1, n_2) = (\alpha_1 n_1^2)/2\lambda_1 + (\alpha_2 n_2^2)/2\lambda_2$. The drift function for the M/M/1-DPS chain in state (n_1, n_2) is as follows:

$$d(n_1, n_2) = \lambda_1(g(n_1{+}1, n_2) - g(n_1, n_2)) + \lambda_2(g(n_1, n_2{+}1) - g(n_1, n_2)) + s_1(g(n_1{-}1, n_2) - g(n_1, n_2)) + s_2(g(n_1, n_2{-}1) - g(n_1, n_2)). \quad (16)$$

For the *1-D step* drift bounding function, we start by setting an appropriate value for $\overline{n}$ in Eq. (9), which in turn is determined via θ since $\overline{n} = \theta E[N_i]$ where $i \in \{1, 2\}$ is the dimension along which the drift bound is being improved. Noting that a smaller θ provides a tighter bound in Eq. (11), we set $\theta = 0.01$; this value of θ also satisfies the requirements of Lemmas 1 and 2 (see Appendix B). We then derive $\overline{n}$ by obtaining $E[N_i]$ via the known first moment of the i^{th} state variable of the M/M/1-DPS model [11]. We compute c and c_1 via Eq. (9). Using the known second moments [11], we compute the right-hand-side of Eq. (11); by setting this to ϵ (the tolerance for probability mass loss due to truncation), we solve for γ, which in turn gives us the attractor set C via Eq. (7). The chain is then truncated to include all states in C. We employ the step function over either dimension ($i = \{1, 2\}$ in Eq. (9)) and use the tighter of the two. Finally, the CTMC is truncated along the two dimensions at $m_1 = \max_{(n_1,n_2)\in C} n_1$ and $m_2 = \max_{(n_1,n_2)\in C} n_2$. A step-by-step analytical illustration of our technique is provided in Appendix A, for reference.



(a) Case 1: $\rho_1 < \rho_2$ $(\rho_1/\rho = 0.1)$. (b) Case 2: $\rho_1 = \rho_2$ $(\rho_1/\rho = 0.5)$. (c) Case 3: $\rho_1 > \rho_2$ $(\rho_1/\rho = 0.8)$.

Fig. 4: Reduction in state space afforded by our different bounding functions over state-of-the-art for the M/M/1-DPS system under different total offered loads (ρ) and class-1 load shares (ρ_1/ρ) for $\epsilon = 0.01$, $\alpha_1 = 0.2$, and $\alpha_2 = 1 - \alpha_1 = 0.8$.

For the *2-D step* drift bounding function, we again set θ to a low value and then set $\overline{n} = (\theta E[Z])/2$, where $Z = N_i + N_j$. The remaining steps are similar to the 1-D step bounding function discussed above.

Evaluation results: To evaluate the truncation improvement over the state-of-the-art Dayar et al. [8], we numerically experiment with different parameter values spanning the total offered load in the range $[0.1, 0.95]$; total offered load is expressed as $\rho = \rho_1 + \rho_2$, where $\rho_i = \lambda_i/\mu_i$. We find that the truncation bounds depend significantly on ρ, ρ_1, and ρ_2. We thus present our results along these parameters by setting $\mu_2 = 1$ and $\mu_1 = 1.2$ and varying λ_1 and λ_2. Figure 4 shows the reduction in the state space afforded by our drift bounding functions over the state-of-the-art as a function of the total offered load for different class-1 load shares when $\alpha_1 = 0.2$, $\alpha_2 = 1 - \alpha_1 = 0.8$, and $\epsilon = 0.01$. We see that the improvement is typically higher for moderate total offered loads ($\rho \approx 0.5$).

In general, the 2-D step function provides more improvement over state-of-the-art compared to the 1-D step function, with as much as 65% reduction in state space over state-of-the-art. In other words, by using our technique, the truncated DPS CTMC can be up to 65% smaller while providing the same probability mass accuracy guarantee ($\epsilon = 0.01$). For this peak reduction case, the state-of-the-art truncates the CTMC at $n_1 = 1358$ and $n_2 = 101$, whereas our 2-D step bounding function truncates at $n_1 = 801$ and $n_2 = 60$; the truncation bounds from "below" are $n_1 = 0$ and $n_2 = 0$ in both cases.

Across all experiments in Figure 4, the average improvements over state-of-the-art are around 39% and 27% for the 2-D and 1-D step functions, respectively. The corresponding average improvements for $0.5 \leq \rho \leq 0.95$ are 45% and 33%; since the truncated CTMC contains more states for higher loads, the absolute reduction in state space (the number of states) is much higher for this range.

Further analysis: We consider the better performing drift bounding function, the 2-D step function, and experiment with different α_1 values. Figure 5 shows the state space reduction over the state-of-the-art as a function of total offered load for different class-1 load shares and for different α_1 and ϵ values. As before, the improvement is higher for moderate offered loads. In general, the improvement increases as α_1 increases, except when the load share of class-1 is high, in which case the improvement tends to decrease as α_1 increases. The improvements are largely *insensitive* to the truncation error guarantee, ϵ; we also experimented with smaller ϵ values with similar insensitivity results.

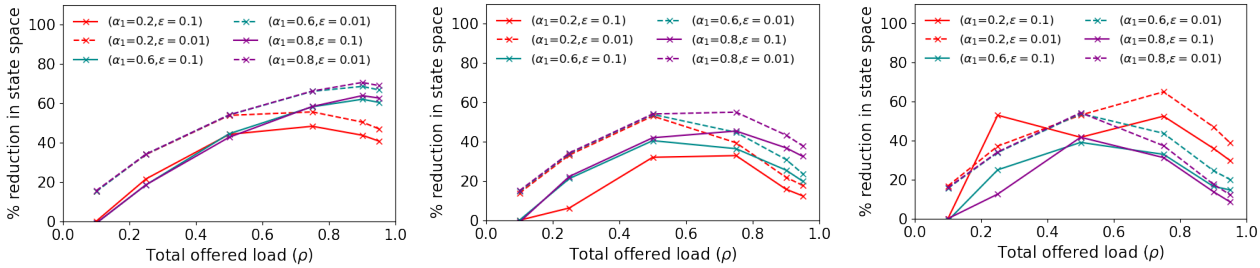
Across all experiments shown in Figure 5, the average improvement over state-of-the-art is around 33%, 34%, and 35% for $\alpha_1 = 0.2$, $\alpha_1 = 0.6$, and $\alpha_1 = 0.8$, respectively. The corresponding improvements for $0.5 \leq \rho \leq 0.95$ are 41%, 42%, and 44%. The maximum improvement is 71%, with the state-of-the-art truncating the CTMC at $n_1 = 108$ and $n_2 = 3292$, while our 2-D step bounding function truncates the CTMC at $n_1 = 65$ and $n_2 = 1610$ (here, the load share of class-2 was higher, so n_2 is truncated at a larger value than n_1).

By providing tighter truncation, our technique *significantly reduces the computational effort required* to solve the truncated CTMC. For example, averaged across all cases in Figure 5, solving the truncated CTMC (by obtaining the steady-state probabilities of all states in the truncated CTMC via solving of the relevant balance equations) is $3\times$ faster when employing our bounds as opposed to employing the bounds suggested by the state-of-the-art.

Validation results: For validation[1], we compare the obtained moments from the truncated CTMC with the exact moments provided in Rege et al. [11]. Using the 1-D step function and setting $\epsilon = 0.1$, the average difference between our results and the exact results for the first, second, and third moments of number of jobs in system (for either class) across various parameter settings is around $4 \times 10^{-4}\%$, $10^{-3}\%$, and $5 \times 10^{-3}\%$, respectively. We further validate our technique by comparing the steady-state distribution of the truncated CTMC for $\alpha_1 = \alpha_2 = 0.5$ with that of the classical processor sharing system (a DPS with $\alpha_1 = \alpha_2$). The maximum observed difference in per-state probability is only around $10^{-5}\%$.

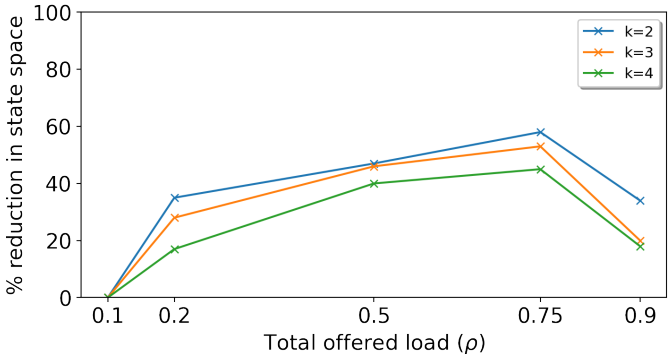
Application to higher-dimensional DPS chains: Our truncation technique is not limited to the 2-dimensional DPS

[1]In general, it may be difficult to numerically validate the truncation bounds since the CTMCs under consideration are intractable. However, under certain parameter settings, the underlying CTMC's exact steady-state probabilities can be obtained. Nonetheless, the theoretical guarantees presented in Section III always hold.

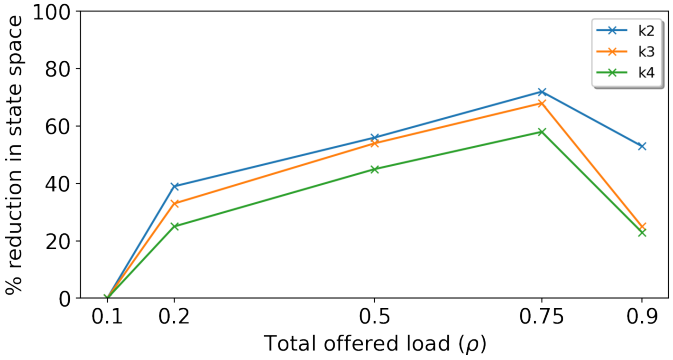


(a) Case 1: $\rho_1 < \rho_2$ $(\rho_1/\rho = 0.1)$. (b) Case 2: $\rho_1 = \rho_2$ $(\rho_1/\rho = 0.5)$. (c) Case 3: $\rho_1 > \rho_2$ $(\rho_1/\rho = 0.8)$.

Fig. 5: Reduction in state space afforded by our 2-D step bounding function over state-of-the-art for the M/M/1-DPS system under different DPS weights (α_1), total offered loads (ρ), and class-1 load shares (ρ_1/ρ).



(a) 1-D step function



(b) 2-D step function

Fig. 6: Reduction in state space afforded by our step bounding functions over state-of-the-art for the M/M/1-DPS system with k customer classes for $\epsilon = 0.1$.

chain. Since the exact moments of the M/M/1-DPS queue-length distribution are known for any number of customer classes, our truncation technique can be readily applied to the M/M/1-DPS CTMC for an arbitrary number of customer classes (along the same lines as for the 2-class M/M/1-DPS CTMC analyzed above). Note that for a k-class M/M/1-DPS system, the CTMC will be k-dimensional, and infinite in all k dimensions. We use the same form of Lyapunov function for the k-dimensional CTMCs as employed for the 2-dimensional CTMC; for example, the Lyapunov function we consider for the 3-class M/M/1-DPS is $g(n_1, n_2, n_3) = (\alpha_1 n_1^2)/2\lambda_1 + (\alpha_2 n_2^2)/2\lambda_2 + (\alpha_3 n_3^2)/2\lambda_3$.

Figure 6 shows the reduction in state space afforded by our truncation technique over the state-of-the-art as a function of the total offered load for the M/M/1-DPS with $k = 2, 3, 4$ customer classes; we show results for the 1-D and 2-D step drift bounding functions. We set $\epsilon = 0.1$ and consider equally-distributed load shares with $\alpha_i = 1/k$ for all k customer classes. We observe that the reduction in state space increases with k, with the 2-D step function generally providing better improvements.

Across all experiments shown in Figure 6, the average (and peak) reduction in state space afforded by our technique for $k = 4$, $k = 3$, and $k = 2$ is 44% (72%), 36% (68%), and 30% (58%), respectively, using the 2-D step bounding function. The corresponding improvements for the 1-D step bounding function are 35% (58%), 30% (53%), and 24% (45%). We note that the parameters corresponding to the total offered load (ρ) of 0.1 do not satisfy the conditions outlined in Lemmas 1 and 2. Thus, the truncation for $\rho = 0.1$ is the same as that of Dayar et al., resulting in 0 improvement. In terms of computational effort, across all experiments in Figure 6, solving the truncated CTMC is as much as 7× faster when employing our bounds as opposed to employing the bounds suggested by the state-of-the-art.

C. Applications of the truncated DPS CTMC

For modern web services, such as Amazon [2] and Google [1], *tail performance measures*, e.g., tail latency or tail queue length, are critical to provide acceptable performance to customers. Average measures, such as the mean response time or queue length, are now considered outdated when analyzing modern applications [26].

To analyze the M/M/1-DPS performance, we consider the 90$^{\text{th}}$ percentile of the number of jobs in system, denoted as P90. We employ our 2-D step drift bounding function to find the truncation bounds by setting $\epsilon = 0.01$. We then solve the balance equations for the truncated CTMC and obtain its steady-state probability distribution; we use the resulting probability distribution to compute the P90 values. Other tail measures can be similarly computed; for example, tail response time can be computed by truncating the CTMC and leveraging existing results for finite DPS queues [35]. We now present three use cases to illustrate the practical applicability of our truncation technique by analyzing the performance of the

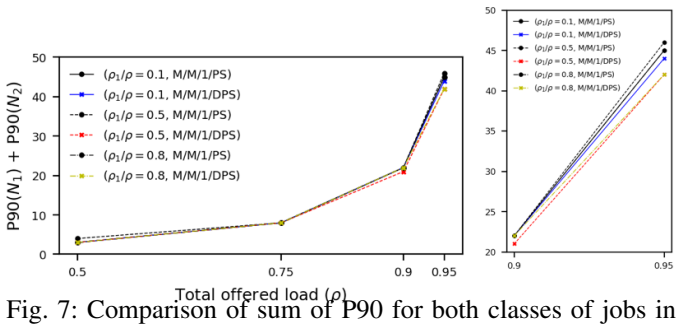


Fig. 7: Comparison of sum of P90 for both classes of jobs in system under the PS and DPS scheduling policies. Results in the $[0.9 \;\; 0.95]$ range are zoomed in for illustration.

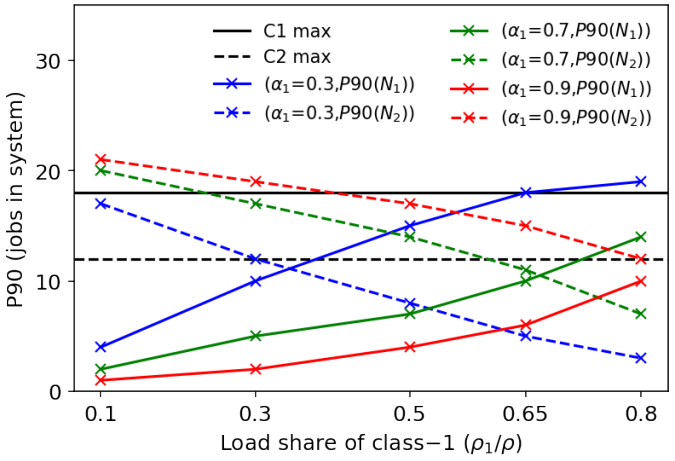
DPS system. For each use case, we consider a slightly different tail performance objective to highlight the applicability of our work.

1) Use case 1: DPS versus PS, for tail metrics: Prior work has shown that DPS can outperform, in terms of the mean queue length, the classical M/M/1-PS system with more than one customer class when a larger weight is assigned to the class with smaller mean service time [30]. We now investigate whether this result still holds for tail measures, such as the P90 of the number of customers in system. For both PS and DPS, we consider (the same) two customer classes with the offered load for class i being $\rho_i = \lambda_i/\mu_i$, and the total offered load $\rho = \rho_1 + \rho_2$.

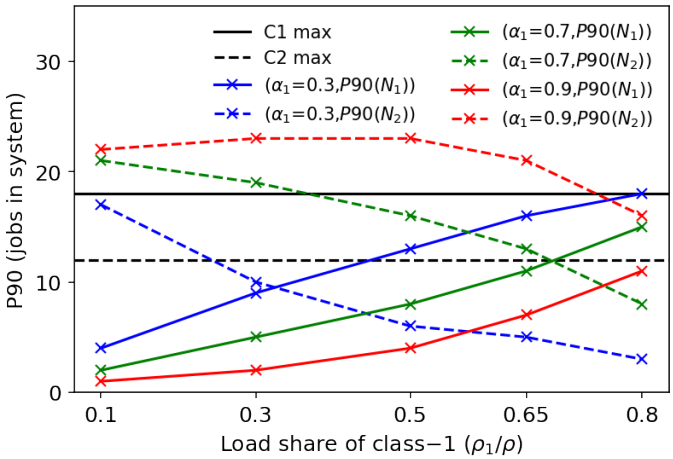
Figure 7 shows the summation of P90 values of the number of customers in system for both classes, P90(N_1) + P90(N_2), as a function of ρ for the M/M/1-PS (black line with circles) and different M/M/1-DPS systems with varying ρ_1 values; we set the parameters for the figure such that the mean service time of class-1 customers is lesser than that of class-2 customers ($1/\mu_1 < 1/\mu_2$) and set $\alpha_1 = 0.9$ to give preferential treatment to the class-1. We separately zoom in and plot the $[0.9 \;\; 0.95]$ x-axis range results for illustration. Note that the P90 values for M/M/1-PS for different values of ρ_1/ρ are quite similar and so appear as a single line.

We find that the DPS system outperforms the PS system for almost all parameter configurations shown in the figure, with more pronounced (and visible) improvements, ranging from 2%–9%, at higher offered loads (see zoomed in plot on the right of Figure 7). The average improvement over all cases shown in Figure 7 is about 4%. For the DPS cases in Figure 7, $\alpha_1 = 0.1$, and thus $\alpha_2 = 0.9 > \alpha_1$. By providing higher priority, or weights, for the class with smaller mean service time (with smaller jobs), DPS is able to achieve better performance, by as much as 9%. We also experimented with $\alpha_1 > \alpha_2$, and found that in this case, PS outperforms DPS.

2) Use case 2: Optimizing the weights of DPS: As seen in the previous use case, the weights of the two customer classes, α_1 and α_2, can significantly impact the performance of the M/M/1-DPS system. We now analyze how the optimal weights change as a function of various system parameters. For optimality, we consider the Service Level Objective (SLO) to be in the form of upper bounds on the P90 of number of jobs



(a) $\mu_1 = 0.6$



(b) $\mu_1 = 1.2$

Fig. 8: Dynamic weight selection to meet P90 SLOs when the total offered load, ρ is constant ($\rho = 0.9$).

in system for the two customer classes, denoted as $C1\ max$ and $C2\ max$.

Figure 8 shows the P90 value for number of customers in system under different α_1 and α_2 values as a function of the fractional load of class-1 customers, ρ_1/ρ; we fix $\rho = 0.9$. We set $\mu_2 = 1$ and experiment with two different μ_1 values, 0.6 and 1.2; for each setting of μ_1, we find the values of λ_1 and λ_2 such that the total load is 0.9 and the fractional load of class-1 is as shown on the x-axis. We only show specific α values in the graph for readability; note that $\alpha_2 = 1 - \alpha_1$. The SLOs for the two classes are set as $C1\ max = 12$ and $C2\ max = 18$, shown by the horizontal black lines.

As the fraction of class-1 load increases, the P90 value of class-1 customers, P90(N_1), denoted by solid lines, increases, and that of class-2 customers, P90(N_2), denoted by dashed lines, decreases. As α_1 increases, P90(N_1) decreases while P90(N_2) increases, in accordance with the DPS policy. For low values of ρ_1/ρ, we see that only $\alpha_1 = 0.3$ meets both SLOs; for higher values of α_1, the $C2\ max$ SLO is not met. For moderate values of class-1 load, $\alpha_1 = 0.7$ works well, whereas $\alpha_1 = 0.3$ violates the $C1\ max$ SLO. Finally, for larger values of class-1 load, α_1 must be high, as much as 0.9, for both

SLOs to be met. In Figure 8a, jobs of class-2 are smaller ($\mu_2 = 1 > \mu_1$), and so experience poor performance when their weight is low ($\alpha_1 = 0.9$). By contrast, in Figure 8b, jobs of class-1 are smaller, and thus experience poor performance when their weight is low ($\alpha_1 = 0.3$).

We also experimented with different values of total offered load, ρ, by fixing the ρ_1/ρ ratio. We again found that the optimal weights change with total offered load. These results highlight the importance of dynamically adapting the DPS weights in response to changes in load to comply with tail performance requirements.

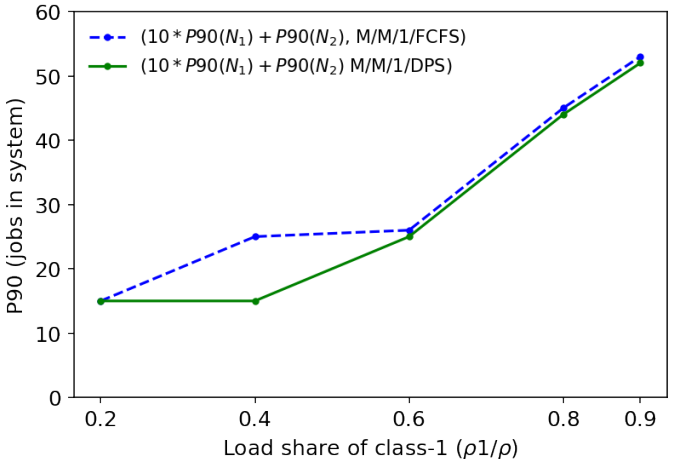
3) Use case 3: When is DPS better than FCFS with priority: Our last use case focuses on the long-standing debate between PS and FCFS policies [37, 38, 39]. In particular, we investigate the performance of M/M/1-DPS when compared with that of an M/M/1-FCFS with priority; as before, we consider two customer (priority) classes. For the SLO, we consider the weighted sum of tail of number of jobs in system: $10 \cdot P90(N_1) + P90(N_2)$. Of course, other weighted sums can be easily evaluated as well. For the M/M/1-FCFS with priority, we employ existing analytical results to obtain the P90 values [40].

We start by considering the non-preemptive version of the M/M/1-FCFS with priority. Figure 9 shows our results for the weighted metric as a function of ρ_1/ρ. To prioritize customer 1 jobs, we set $\alpha_1 = 0.95$ for DPS; results are qualitatively similar, but not as pronounced, under other $\alpha_1 > 0.5$ values. We set $\mu_1 = 0.6$ and $\mu_2 = 1$, and experiment with total offered load of $\rho = 0.7$ and $\rho = 0.9$. For each case, we find values of λ_1 and λ_2 such that the total load is ρ and the fractional load of customer 1 is as shown on the x-axis.

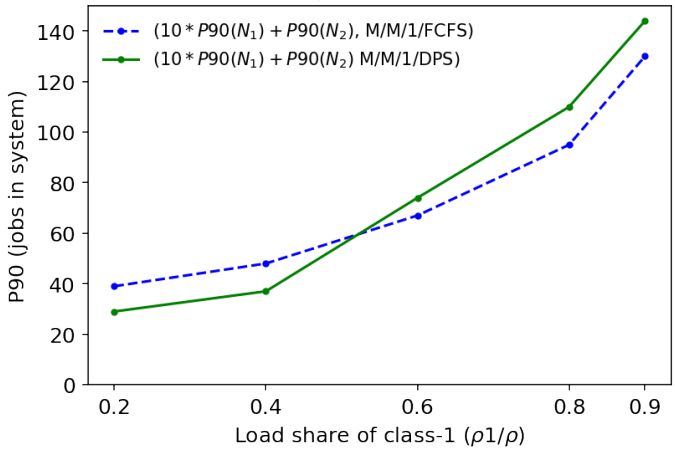
When $\rho = 0.7$, both policies perform similarly. However, when $\rho = 0.9$, we observe an interesting behavior. When the load share of customer class-1 is low, DPS performs better, whereas when the load share of class-1 is high, FCFS performs better. This is because for low load share of class-1, the load is higher for class-2 jobs, and due to the non-preemptive FCFS policy we consider, class-2 jobs can "hold up" incoming class-1 jobs, resulting in a high penalty under our metric that gives a higher weight to $P90(N_1)$. For high load share of class-1, FCFS outperforms DPS since FCFS provides strict priority, as opposed to the α_1-weighted DPS policy, which still provides a weight of $1 - \alpha_1 = 0.05$ for class-2 jobs.

We observed a similar trend for other values of μ_1 and μ_2 as well. Interestingly, under the DPS policy, the exact values of $P90(N_1)$ and $P90(N_2)$ can change for different arrival and service rates while the total offered load (ρ) and the per-class loads (ρ_1 and ρ_2) are kept constant. This is in contrast to the PS policy for which the tail of number of jobs in system only depends on ρ, ρ_1, and ρ_2, and not the individual arrival and service rates [41].

We also compared the performance of M/M/1-DPS with that of preemptive M/M/1-FCFS. However, in this case, across different parameter settings, the preemptive FCFS always outperforms M/M/1-DPS, with respect to the tail of number of jobs in system.



(a) $\rho = 0.7$



(b) $\rho = 0.9$

Fig. 9: Performance of M/M/1-DPS and non-preemptive M/M/1-FCFS with priority for different load conditions.

V. Applying our truncation technique to the preemptive M/M/c+M queue with two priority classes

In this section, we consider multi-server priority queues as another example of CTMCs for which our technique is applicable. Multi-server priority queues with preemption have been widely employed to model differentiated service for multiple customer classes in computer systems, call centers, and health care management [12, 24, 25]. For the case with customer abandonment (impatient customers), referred to as the M/M/c+M priority queue, the exact steady-state probability distribution for the low-priority customers is not known (the queue for the high-priority customers follows the M/M/c+M queue, also referred to as the Erlang-A model [12]). This is because the underlying CTMC, shown in Figure 10, is infinite in two dimensions and has state-dependent transitions—the upward and backward transitions in Figure 10 with rates $r_{i,j}$ and $s_{i,j}$, respectively. For the case with no abandonment, Wang et al. [24] recently introduced a technique to obtain the generating function for the number of low-priority customers in system when $c = 2$; for $c > 2$, the technique is cumbersome, so only moments were obtained.

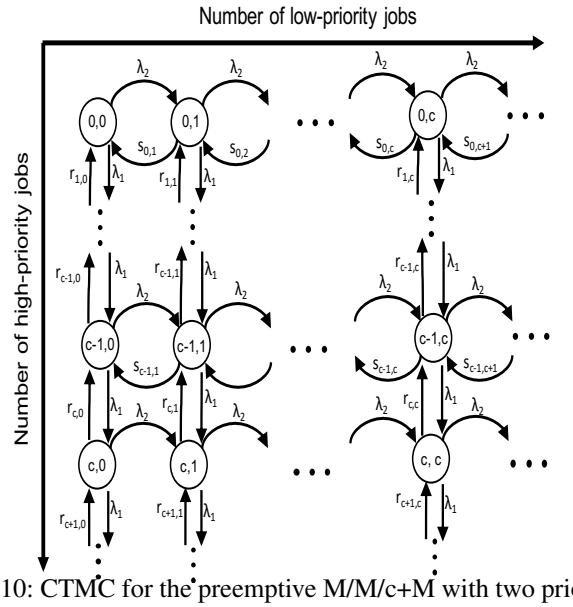
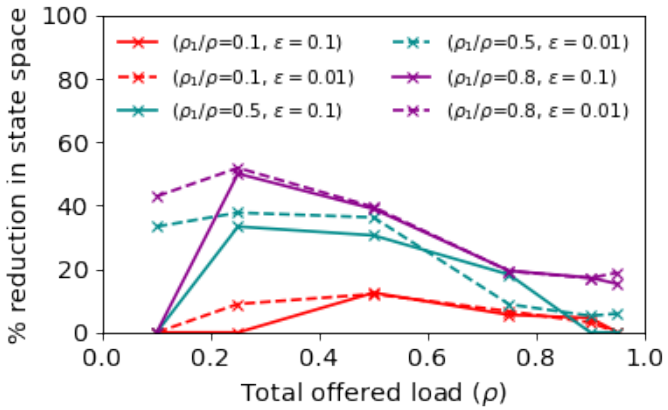


Fig. 10: CTMC for the preemptive M/M/c+M with two priority classes. For state (n_1, n_2), n_1 is the number of jobs of the high-priority class (class-1) and n_2 is the number of jobs of the low-priority class (class-2). The transition rates are $r_{n_1,n_2} = \mu_1 \cdot n_{1s} + \beta_1 \cdot (n_1 - n_{1s})$ and $s_{n_1,n_2} = \mu_2 \cdot n_{2s} + \beta_2 \cdot (n_2 - n_{2s})$, where $n_{1s} = min\{n_1, c\}$ and $n_{2s} = min\{c - n_{1s}, n_2\}$ are the number of high-priority jobs and low-priority jobs being served, respectively, and β_l is the abandonment rate of class-l jobs.

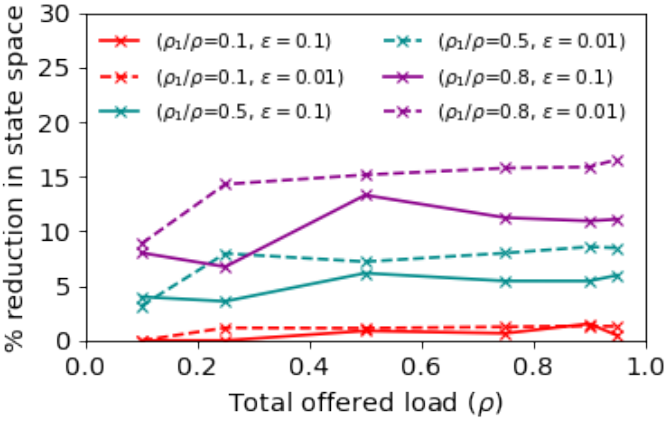
For both the with- and without-abandonment cases noted above, our truncation technique can be readily applied since the moments are available. Interestingly, for preemptive the M/M/c+M priority queue, only the moments of the high-priority jobs are known (via the M/M/c+M model [12]), but this suffices to obtain truncation bounds via our 1-D step bounding function from Section III-B. We find that both $(n_1 + n_2)$ and $(n_1 + n_2)^2$ are valid Lyapunov functions for the M/M/c+M priority queue. Our experiments with these two functions show that $(n_1 + n_2)^2$ generally provides tighter bounds compared to $(n_1 + n_2)$ for the M/M/c+M priority queue. We thus use $(n_1 + n_2)^2$ for the experiments in this section to compare the truncation bounds obtained from our technique and those obtained from Dayar et al. For reference, an analytical illustration of our truncation technique for the M/M/1+M priority queue based on the simpler Lyapunov function, $(n_1 + n_2)$, is provided in Appendix A.

Figure 11 shows the reduction in the state space achieved by our 1-D step bounding function over the state-of-the-art as a function of the total offered load for different class-1 load shares and different error thresholds, ϵ. As before, we set $\mu_1 = 1.2$ and $\mu_2 = 1$, and vary λ_1 and λ_2 to obtain the required total and per-class loads. The abandonment rate is set as $\beta_1 = 0.1\lambda_1$ and $\beta_2 = 0.01\lambda_2$, where β_i is the abandonment rate for class-i customers.

For the preemptive M/M/3+M queue with two priority classes, we achieve a significant reduction in state space compared to the state-of-the-art [8], with average reduction



(a) M/M/3+M



(b) M/M/100+M

Fig. 11: Reduction in state space for the preemptive M/M/c+M queue with two priority classes.

of about 28%, 18%, and 5% for class-1 load shares of 0.8, 0.5, and 0.1, respectively; the corresponding peak reductions are 52%, 38%, and 13%. For the preemptive M/M/100+M queue with two priority classes, we achieve moderate savings, with average and peak reduction ranging from 1%–13% and 2%–17%, respectively, across different class-1 load shares. In general, the reductions are higher for higher class-1 load shares (and for moderate offered loads). This is likely because the $E[N_i]^2/E[N_i^2]$ term, with $i = 1$, in Eq. (11) is larger for higher class-1 load shares, thus resulting in a tighter bound.

To validate our truncation accuracy, we leverage the fact that the higher priority class customers constitute an M/M/c+M queue, for which the steady-state probability distribution is known [12]. Comparing the marginal steady-state distribution of the higher priority class of the truncated M/M/3+M priority queue chain for $\epsilon = 0.1$ with the steady-state distribution of M/M/3+M, we find that the maximum per-state deviation across all states is about 3×10^{-5}%. For the M/M/100+M, the maximum deviation is about 8×10^{-4}%.

By leveraging the fact that only the moments of the highest priority class customer are required, our bounding technique can be extended to truncate the CTMCs of the M/M/c+M

priority systems with any number of priority classes. For the M/M/3+M system with three classes, with the underlying CTMC being three-dimensional, our 1-D step bounding function reduces the state space by about 21%, on average (across parameter settings similar to those in Figure 11a), and by up to 52%, compared to the state-of-the-art.

We also applied our truncation technique to the preemptive M/M/c priority queue with two priority classes but *without* abandonment, for which the analysis is known to be cumbersome when $c > 2$ [24]. For the preemptive M/M/3 queue with two priority classes, our 1-D step bounding function provides, on average, a 23% reduction in state space when compared to the state-of-the-art.

VI. Conclusion

In this paper, we present a Lyapunov function based technique to obtain tight truncation bounds with probability mass guarantees for multi-dimensional and infinite state-space CTMCs. By leveraging the known moments of the CTMC, compared to the state-of-the-art, our technique significantly truncates the state space, by around 34% on average and by as much as 72% in the case of M/M/1-DPS model, and by around 14% on average and by as much as 52% in the case of the M/M/c+M model. The truncated CTMC can then be easily and quickly solved (as much as $7\times$ speedup) to obtain the steady-state probability distribution. Importantly, we prove that the truncation guarantees a user-specified bound on loss in probability mass, thus allowing for arbitrary accuracy guarantees.

Our technique is valuable to researchers who deal with infinite-space, multi-dimensional CTMCs with state-dependent transitions, for which the steady-state probabilities are often challenging to obtain but moments of the state variables may be available; two such examples that we considered in this paper are the M/M/1-DPS and the preemptive M/M/c+M priority queue. Our technique is also valuable for the analysis of finite but large state-space CTMCs where the computation of the steady-state probabilities is computationally expensive, thus benefiting from the state-space reduction afforded by our technique. Finally, our technique is also applicable to CTMCs where the exact moments are not available but bounds on the moments are known via, for example, drift analysis [21].

References

[1] J. Dean and L. A. Barroso, "The Tail at Scale," *Communications of ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[2] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's Highly Available Key-value Store," in *Proceedings of Twenty-first ACM Symposium on Operating Systems Principles*, Stevenson, WA, USA, 2007, pp. 205–220.

[3] G. Latouche and V. Ramaswami, *Introduction to Matrix Analytic Methods in Stochastic Modeling*. Philadelphia, PA, USA: ASA-SIAM, 1999.

[4] L. Kleinrock, "Time-shared systems: A theoretical treatment," *J. ACM*, vol. 14, no. 2, pp. 242–261, 1967.

[5] H. Baumann and W. Sandmann, "Numerical solution of level dependent quasi-birth-and-death processes," *Procedia Comput. Sci.*, vol. 1, no. 1, pp. 1561–1569, 2010.

[6] J. P. Kharoufeh, "Level-dependent quasi-birth-and-death processes," in *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

[7] R. R. Muntz, E. de Souza e Silva, and A. Goyal, "Bounding availability of repairable computer systems," in *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems*, Oakland, CA, USA, 1989, pp. 29–38.

[8] T. Dayar, W. Sandmann, D. Spieler, and V. Wolf, "Infinite level-dependent QBD processes and matrix-analytic solutions for stochastic chemical kinetics," *Advances in Applied Probability*, vol. 43, no. 4, pp. 1005–1026, 2011.

[9] R. E. A. C. Paley and A. Zygmund, "On some series of functions," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 28, no. 2, pp. 190–205, 1932.

[10] P. Vis, "Performance analysis of multi-class queueing models," Ph.D. dissertation, Vrije Universiteit, Sep. 2017.

[11] K. M. Rege and B. Sengupta, "Queue-length distribution for the discriminatory processor-sharing queue," *Operations Research*, vol. 44, no. 4, pp. 653–657, 1996.

[12] A. Mandelbaum and S. Zeltyn, *Service Engineering in Action: The Palm/Erlang-A Queue, with Applications to Call Centers*. Springer, 2007, pp. 17–45.

[13] P.-J. Courtois and P. Semal, "Bounds for the positive eigenvectors of nonnegative matrices and for their approximations by decomposition," *J. ACM*, vol. 31, pp. 804–825, 1984.

[14] J. C. S. Lui and R. R. Muntz, "Computing bounds on steady state availability of repairable computer systems," *J. ACM*, vol. 41, no. 4, pp. 676–707, 1994.

[15] S. Mahevas and G. Rubino, "Bound computation of dependability and performance measures," *IEEE Transactions on Computers*, vol. 50, no. 5, pp. 399–413, May 2001.

[16] A. Riska and E. Smirni, "Exact aggregate solutions for m/g/1-type markov processes," *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, 08 2002.

[17] P. Buchholz, "Bounding stationary results of tandem networks with map input and ph service time distributions," *Perform. Eval. Rev.*, vol. 34, no. 1, pp. 191–202, 2006.

[18] L. Bright and P. G. Taylor, "Calculating the equilibrium distribution in level dependent quasi-birth-and-death processes," *Communications in Statistics. Stochastic Models*, vol. 11, pp. 497–525, 1995.

[19] D. Bertsimas, D. Gamarnik, and J. N. Tsitsiklis, "Geometric bounds for stationary distributions of infinite Markov chains via Lyapunov functions," MIT, Sloan School of Management, Tech. Rep. WP 4027-98, 1998.

[20] S. T. Maguluri and R. Srikant, "Heavy traffic queue length behavior in a switch under the maxweight algorithm," *Stochastic Systems*, vol. 6, pp. 211–250, 2016.

[21] H. Bruce, "Hitting-time and occupation-time bounds implied by drift analysis with applications," *Advances in*

Applied Probability, vol. 14, no. 3, pp. 502–525, 1982.
[22] H. Baumann and W. Sandmann, "Bounded truncation error for long-run averages in infinite markov chains," *J. Appl. Probab.*, vol. 52, no. 3, pp. 609–621, 2015.
[23] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013.
[24] J. Wang, O. Baron, and A. Scheller-wolf, "M/m/c queue with two priority classes," *Operations Research*, vol. 63, pp. 733–749, 05 2015.
[25] A. Rastpour, "Essays on health care operations management," Ph.D. dissertation, University of Alberta, 2016.
[26] J. Li, N. Sharma, D. Ports, and S. D. Gribble, "Tales of the tail: Hardware, os, and application-level sources of tail latency," in *Proceedings of the ACM Symposium on Cloud Computing*, Seattle, WA, USA, 2014, pp. 9:1–9:14.
[27] P. A. Misra, M. F. Borge, I. Goiri, A. R. Lebeck, W. Zwaenepoel, and R. Bianchini, "Managing Tail Latency in Datacenter-Scale File Systems Under Production Constraints," in *Proceedings of the 14th European Conference on Computer Systems*, Dresden, Germany, 2019, pp. 17:1–17:15.
[28] M. Mitzenmacher and R. Rajaraman, "Towards more complete models of tcp latency and throughput," *J. Supercomput.*, vol. 20, no. 2, pp. 137–160, 2001.
[29] H. Ogasawara, "The multivariate markov and multiple chebyshev inequalities," *Communications in Statistics - Theory and Methods*, vol. 49, no. 2, pp. 441–453, 2020.
[30] B. Kim and J. Kim, "Comparison of dps and ps systems according to dps weights," *IEEE Communications Letters*, vol. 10, no. 7, pp. 558–560, 2006.
[31] I. Verloop, U. Ayesta, and R. Nunez Queija, "Heavy-traffic analysis of a multiple-phase network with discriminatory processor sharing," *Operations Research*, vol. 59, pp. 648–660, 2011.
[32] P. Vis, R. Bekker, R. van der Mei, and R. Núñez Queija, "Heavy-traffic limits for discriminatory processor sharing models with joint batch arrivals," *Oper. Res. Lett.*, vol. 48, no. 2, p. 136–141, Mar. 2020. [Online]. Available: https://doi.org/10.1016/j.orl.2020.01.004
[33] A. Izagirre, U. Ayesta, and I. M. Verloop, "Sojourn time approximations for a discriminatory processor sharing queue," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, no. 1, pp. 5:1–5:31, 2016.
[34] K. Avrachenkov, U. Ayesta, P. Brown, and R. Nunez-Queija, "Discriminatory processor sharing revisited," in *Proceedings of the 24th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies*, Miami, FL, USA, 2005, pp. 784–795.
[35] O. J. Boxma, N. Hegde, and R. Nunez-Queija, "Exact and approximate analysis of sojourn times in finite discriminatory processor sharing queues," *AEU - International Journal of Electronics and Communications*, vol. 60, no. 2, pp. 109 – 115, 2006.
[36] G. De Veciana and T. Konstantopoulos, "Stability and performance analysis of networks supporting elastic services," *IEEE/ACM Trans. Netw.*, vol. 9, no. 1, pp. 2–14, 2001.
[37] M. Harchol-Balter, M. Crovella, and C. Murta, "To Queue or Not to Queue?: When FCFS is Better Than PS in a Distributed System," Boston U., Tech. Report., 1997.
[38] R. Hassin and M. Haviv, *To queue or not to queue: equilibrium behaviour in queuing systems*. Kluwer Academic Publishers, 2002.
[39] M. Haviv, "The M/G/1 Queueing Model with Preemptive Random Priorities," in *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools*, Bratislava, Slovakia, 2014, pp. 241–246.
[40] D. R. Miller, "Computation of steady-state probabilities for m/m/1 priority queues," *Operations Research*, vol. 29, no. 5, pp. 945–958, 1981.
[41] A. Zwart, *Sojourn times in a multiclass processor sharing queue*, ser. Memorandum COSOR. Technische Universiteit Eindhoven, 1998.

APPENDIX

A. Illustrating the application of our technique for a simple chain

This section provides a step-by-step pseudo-analytical procedure of applying our technique for the preemptive M/M/1+M priority queue with two classes of customers (with abandonment). The CTMC for this model follows Figure 10 (with $c = 1$), which is infinite in both dimensions. Let the average arrival rate, average service rate, average abandonment rate, and number of class $i \in \{1, 2\}$ customers be denoted as λ_i, μ_i, β_i, and n_i, respectively.

Our objective is to find a truncated version of this CTMC, which contains at least $(1-\epsilon)$ fraction of the probability mass of the original (infinite) CTMC. We achieve this by finding a subset C of the original CTMC's state space such that the probability mass outside C, $\sum_{n \in \overline{C}} \pi(n)$, is at most ϵ. We then truncate the original CTMC to contain at least the states in C, thus ensuring that the probability mass inside the truncated chain is at least $(1 - \epsilon)$.

Step 1: *Choose a suitable Lyapunov function $g(n)$.*
The first step is to choose a Lyapunov function for the CTMC in consideration. In general, finding the Lyapunov function is a non-trivial task [8, 22], but this is not the focus of our work. For our preemptive M/M/1+M priority queue chain, it can be easily verified that $g(n_1, n_2) = (n_1 + n_2)$ is a Lyapunov function. For ease of exposition of the steps of our truncation technique, we use this Lyapunov function in this section.

Step 2: *Calculate $d(n)$, the drift of $g(n)$.*
For a CTMC with state vector $n = (n_1, n_2, n_3, ..., n_k)$, we define pairs $(T_i(n), v_i)$, where $T_i : S \rightarrow R_{\geq 0}$, with i indexing all the possible transitions from state n, determines the transition rate in a state n, and $v_i \in Z^{1 \times k}$

is the state change vector. That is, the i^{th} transition from state n goes to state $n + v_i$ with rate $T_i(n)$. With these definitions, the drift of the Lyapunov function can be obtained as:

$$d(n) = \sum_{i=1}^{m} T_i(n) \cdot (g(n + v_i) - g(n)), \qquad (17)$$

where m is the total number of transitions from state n. In our specific example of the priority queue, the transition pairs for state (n_1, n_2) are $(\lambda_1, (1, 0))$, $(\lambda_2, (0, 1))$, $(r_{n_1,n_2}, (-1, 0))$, and $(s_{n_1,n_2}, (0, -1))$, where r_{n_1,n_2} and s_{n_1,n_2} are defined in Section V (with $c = 1$). Substituting these transition pairs in Eq. (17), we have $d(n_1, n_2) = \lambda_1 + \lambda_2 - r_{n_1,n_2} - s_{n_1,n_2}$. This drift function can be expressed as the below piece-wise function for easier analysis.

$$d(n_1, n_2) = \begin{cases} \beta_1 + \lambda_1 + \lambda_2 - \mu_1 - \beta_1 n_1 - \beta_2 n_2 & n_1 \geq 1 \\ \beta_2 + \lambda_1 + \lambda_2 - \mu_2 - \beta_2 n_2 & n_1 = 0 \text{ and } n_2 \geq 1 \\ \lambda_1 + \lambda_2 & n_1 = 0 \text{ and } n_2 = 0 \end{cases} \qquad (18)$$

Step 3: *Choose a drift bounding function, $f(n)$.*
For ease of illustration, we choose the 1-D step bounding function for the preemptive M/M/1+M priority queue:

$$f_1(n) = \begin{cases} c & = \sup_{\forall n \in S} d(n), \qquad \forall n \in S \setminus S', \\ c_1 & = \sup_{\forall n \in S'} d(n), \qquad \forall n \in S', \end{cases}$$

where we set $S' = \{n \mid n_1 > \overline{n}\}$, for some $\overline{n} \geq 0$. The supremum of $d(n)$ for this chain can be found (using Eq. (18)) as $c = \lambda_1 + \lambda_2$. For $\overline{n} \geq 0$, by definition, S' contains states with $n_1 > 0$, and so the supremum of $d(n)$ in S' is dictated by the first sub-function in Eq. (18), which gives us $c_1 = -(\lfloor \overline{n} + 1 \rfloor)\beta_1 + \beta_1 + \lambda_1 + \lambda_2 - \mu_1$.

Step 4: *Determine the upper bound on the probability mass in $\overline{C}$.*
Using the properties of the drift and the bounding function, we obtain an upper bound on the probability mass outside the attractor set, $\overline{C}$, via Eq. (8): $\sum_{n \in \overline{C}} \pi(n) \leq \frac{\sum_{n \in C} \pi(n) \cdot f(n)}{\gamma}$. Substituting the step function, $f_1(n)$, in place of $f(n)$, leads to Eq. (10): $\sum_{n \in \overline{C}} \pi(n) \leq \frac{c}{c_1 + \gamma} - \frac{c - c_1}{c_1 + \gamma} \sum_{n \in S'} \pi(n)$. To obtain the value of the upper bound on $\sum_{n \in \overline{C}} \pi(n)$, we need to find the value of $\sum_{n \in S'} \pi(n)$. If the exact value of $\sum_{n \in S'} \pi(n)$ cannot be found, a lower bound on this term can be used. For our example chain, we use the Paley-Zygmund inequality to come up with an upper bound given by Eq. (11) as $\frac{c}{c_1 + \gamma} - \frac{c - c_1}{c_1 + \gamma} \left(1 - \frac{\overline{n}}{E[N_1]}\right)^2 \frac{E[N_1]^2}{E[N_1^2]}$, where we set $\overline{n} = \theta E[N_1]$ in the Paley-Zygmund inequality. Substituting the values of c and c_1 from Step 3, we have an analytical expression for the upper bound on $\sum_{n \in C} \pi(n)$, say U: $\sum_{n \in C} \pi(n) \leq U = \frac{-\beta_1 \lfloor \overline{n} \rfloor (E[N_1] - \overline{n})^2 - \mu_1 E[N_1]^2 + 2\mu_1 E[N_1]\overline{n} + \lambda_1 E[N_1^2] + \lambda_2 E[N_1^2] - \mu_1 \overline{n}^2}{E[N_1^2](\gamma - \beta_1 \lfloor \overline{n} \rfloor + \lambda_1 + \lambda_2 - \mu_1)}$

Step 5: *Solve $U = \epsilon$ to find γ.*
We set the upper bound on the probability mass outside the attractor set to be the allowable error, ϵ, and we solve for γ to obtain the truncated chain (in the next step). For our example, equating the upper bound to ϵ gives a linear equation in γ. We obtain $E[N_1]$ and $E[N_1^2]$ for our chain based on results from prior work [12]. The obtained linear equation can be solved analytically to obtain the value of γ. In our case, $\gamma = \frac{-\beta_1 \lfloor \overline{n} \rfloor \left(E[N_1]^2 - 2E[N_1]\overline{n} - \epsilon E[N_1^2] + \overline{n}^2\right) - \mu_1 E[N_1]^2 + 2\mu_1 E[N_1]\overline{n} + \lambda_1 (E[N_1^2] - \epsilon E$
$\epsilon E[N_1^2]$

Step 6: *Find truncation bounds.*
We use the above value of γ to find the attractor set C via Eq. (7) as $C = \{n \in S \mid d(n) > -\gamma\}$. The truncated chain must at least contain C to provide the probability mass guarantee. For our example, we find the smallest values of k_1 and k_2 such that $d(n) \leq -\gamma$ for states $(k_1, 0)$ and $(0, k_2)$; note that for any state (n_1, n_2) with $n_1 \geq k_1$ and $n_2 \geq k_2$, we have, via Eq. (18), that $d(n) \leq -\gamma$. The values $k_1 = (\beta_1 + \gamma + \lambda_1 + \lambda_2 - \mu)/\beta_1$ and $k_2 = (\beta_1 + \gamma + \lambda_1 + \lambda_2 - \mu)/\beta_2$ satisfy the above requirements, giving us $C = \{n \in S \mid n_1 < k_1 \text{ and } n_2 < k_2\}$. We thus obtain our truncated chain by truncating the original chain along the first and second dimensions at $\lceil (k_1 - 1) \rceil$ and $\lceil (k_2 - 1) \rceil$, respectively.

The truncated chain can be solved to obtain the steady-state distribution (or associated performance measures). Note that other eligible Lyapunov functions and/or bounding functions can be used in the above example. The derivations in Steps 4 and 5 will change accordingly, but otherwise the high-level technique and sequence of steps will remain unchanged.

B. Proofs of Lemmas 1 and 2

We prove Lemma 1 by showing that $\gamma_O \leq \gamma_D$, provided $\epsilon \leq E[N_i]^2/E[N_i^2]$ where γ_O and γ_D denote the γ values obtained via our 1-D step drift bounding function and via the supremum drift bounding function (used in Dayar et al. [8]), respectively, and N_i represents an arbitrary infinite dimension of the CTMC. Recall that γ is the parameter that determines the minimum number of states that need to be part of the truncated chain via the attractor set, $C = \{n \in S \mid d(n) > -\gamma\}$.

Based on the supremum function (the drift bounding function used in Dayar et al. [8]), the upper bound on the probability mass of the states outside the attractor set is given by $c/(c + \gamma)$ (Eq. (6)). By equating this upper bound to ϵ, we have:

$$\gamma_D = \frac{c}{\epsilon} - c$$

Likewise, by equating the upper bound obtained via our 1-D step function (provided in Eq. (11)) to ϵ, we have:

$$\epsilon = \frac{c}{c_1 + \gamma_O} - \frac{c - c_1}{c_1 + \gamma_O} \cdot x \implies (c_1 + \gamma_O)\epsilon = c - (c - c_1) \cdot x \implies \gamma_O \qquad (19)$$

where $x = (1 - \theta)^2 y$ and $y = E[N_i]^2/E[N_i^2]$.

We thus have:

$$\begin{aligned}
\gamma_O \leq \gamma_D &\iff \frac{c}{\epsilon} - \frac{c - c_1}{\epsilon} \cdot x - c_1 \leq \frac{c}{\epsilon} - c \\
&\iff 0 \leq (c - c_1)(\frac{x}{\epsilon} - 1) \\
&\iff \epsilon \leq x \quad \because c - c_1 \geq 0 \\
&\iff \epsilon \leq (1 - \theta)^2 y \\
&\iff \theta \leq 1 - \sqrt{\epsilon / y} \qquad (20)
\end{aligned}$$

Recall, from Definition 1, that θ is a parameter in the Paley-Zygmund inequality that can be chosen to take any value in the range $[0, 1]$. Thus, if $\epsilon \leq y$, we have $(1 - \sqrt{\epsilon/y}) \geq 0$, and since $(1 - \sqrt{\epsilon/y}) \leq 1$, any $\theta \in [0, 1 - \sqrt{\epsilon/y}]$ will satisfy the condition in Eq. (20). For example, the value of $\theta = 0.01$ used in our experiments satisfied the condition from Eq. (20). Note that, in the above derivation, we have used the fact that $c \geq 0$ since $d(n) \geq 0$ for at least some states (otherwise, $\pi d^T < 0$). Hence, provided $\epsilon \leq y = \frac{E[N_i]^2}{E[N_i^2]}$, our truncation bounds are at least as tight as those obtained by Dayar et al. [8]. Since in practice the value of ϵ is usually small, we believe this is not too restrictive. In fact, all the parameters that we use in our experiments, except the ones corresponding to very low total offered loads, satisfy this condition.

A similar proof holds for the 2-D step function (Lemma 2) with the only change being $y = E[Z]^2 / E[Z^2]$, where Z is as given in Eq. (14).