Analysis of Multi-server Systems via Dimensionality Reduction of Markov Chains

Takayuki Osogami

June, 2005
CMU-CS-05-136

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Thesis Committee:
Mor Harchol-Balter, Chair
Hui Zhang
Bruce M. Maggs
Alan Scheller-Wolf (Tepper School of Business, Carnegie Mellon University)
Mark S. Squillante (T. J. Watson Research Center, IBM Research)

This research was supported by grant sponsorship from IBM Japan and IBM Tokyo Research Laboratory.

Keywords: Performance analysis, Multi-server system, Markov chain, Resource allocation, Cycle stealing, Priority, Task assignment, Phase type, Dimensionality reduction, Moment matching.

Abstract

The performance analysis of multiserver systems is notoriously hard, especially when the system involves resource sharing or prioritization. We provide two new analytical tools for the performance analysis of multiserver systems: moment matching algorithms and dimensionality reduction of Markov chains (DR).

Moment matching algorithms allow us to approximate a general distribution with a phase type (PH) distribution. Our moment matching algorithms improve upon existing ones with respect to the computational efficiency (we provide closed form solutions) as well as the quality and generality of the solution (the first three moments of almost any nonnegative distribution are matched). Approximating job size and interarrival time distributions by PH distributions enables modeling a multiserver system by a Markov chain, so that the performance of the system is given by analyzing the Markov chain. However, when the multiserver system involves resource sharing or prioritization, the Markov chain often has a multidimensionally infinite state space, which makes the analysis computationally hard.

DR allows us to closely approximate a multidimensionally infinite Markov chain with a Markov chain on a one-dimensionally infinite state space, which can be analyzed efficiently. We validate the accuracy of DR against simulation. Further, we formally define two classes of multidimensionally infinite Markov chains, called recursive foreground-background processes and generalized foreground-background processes (RFB/GFB processes), and analyze the RFB/GFB process via DR. The definition of the RFB/GFB process enables one to easily identify whether a given multiserver system can be analyzed via DR, and our analysis of the RFB/GFB process enables the immediate analysis of a multiserver system by simply modeling it as an RFB/GFB process.

These new analytical tools enable us to analyze the performance of many multiserver systems with resource sharing or prioritization for the first time. For example, we study the benefit and penalty of cycle stealing, the effectiveness of prioritization and threshold-based resource allocation policies for multiserver systems, and the impact of job size variability and irregularity of arrival processes on the response time in multiserver systems. Our analysis results in lessons on the design of good resource allocation policies for multiserver systems.

Acknowledgements

I joined Computer Science Department at Carnegie Mellon University as a graduate student in the fall of 2001. Mor Harchol-Balter has been my adviser since then. First of all, I would like to thank Mor for her advice on all aspects of my graduate study.

One of my first accomplishments as a graduate student was an analysis of the size-based task assignment with cycle stealing under immediate dispatching, `SBCS-ID` (Chapter 5). In the summer of 2002, I derived a closed form solution for the mean response time of the long jobs via a virtual waiting time analysis (Theorem 9). This small step turned out to be a "big" step in my graduate research. Mor Harchol-Balter, Alan Scheller-Wolf, and Mark S. Squillante had been taking a different approach in the analysis of `SBCS-ID`, which was later developed into dimensionality reduction (DR) as in this thesis. With the virtual waiting time analysis, I joined the project of `SBCS-ID` analysis. It was very fortunate that I could join the project of developing DR in its early stage.

In parallel to `SBCS-ID`, we analyzed the size-based task assignment with cycle stealing under central queue, `SBCS-CQ` (Chapter 5). The analysis of `SBCS-CQ` was more difficult, and we needed to introduce an additional approximation together with the renaming scheme. Ironically, it turns out that `SBCS-CQ` can be analyzed without the additional approximation by DR (when it is extended as in this thesis) if the renaming scheme is not used; however, DR does not apply without an additional approximation if the renaming scheme is used.

In the fall of 2002, we summarized `SBCS-ID` in [68] and `SBCS-CQ` in [67], and Mor, Alan, and I moved on to the analysis of a threshold-based policy for reducing switching costs in cycle stealing (Chapter 6). Retrospectively, it is quite surprising that we were able to analyze this system at that time, when the DR was not fully generalized as in this thesis. This system can be modeled as a GFB process where the background process is a QBD process. At that time, we did not know how to analyze the inter-level passage time in a QBD process (Section 3.7). Nevertheless, we were able to combine various busy periods in M/G/1 queues with setup times to represent the inter-level passage time in a QBD process in this special case. We summarized the analysis for the case without the donor size threshold in [151], and the full analysis in [152]. The use of generalized vacation as in Theorem 11 was suggested by an anonymous reviewer of Performance Evaluation, and this simplified the analysis of the mean response time of donor jobs.

In the analysis of `SBCS-ID`, `SBCS-CQ`, and the threshold-based policy for reducing switching costs in cycle stealing, I had occasionally encountered a problem in approximating a busy period duration by a two-phase phase type (PH) distribution. The parameters of the two-phase PH distributions sometimes became complex numbers! This motivated me to investigate which distributions can be well-represented (Definition 1) by an n-phase PH distribution for $n = 2, 3, 4, ...$ (Chapter 2). This characterization was summarized in [147] with Mor Harchol-Balter.

This characterization gave me a great insight into *how* a general distribution can be well-

represented by a PH distribution with nearly minimal number of phases (Chapter 2). Discovery of the properties of function ϕ (Theorem 3) was the key in the development of our moment matching algorithms. The moment matching algorithms were summarized in [145] with Mor Harchol-Balter. The two papers [145, 147] were selected among the best papers in TOOLS 2003, and combined into [148] for a special edition in Performance Evaluation. Miklos Telek and anonymous reviewers of TOOLS 2003 and Performance Evaluation gave us very detailed and relevant comments and suggestions for [147, 145, 148], and the quality of these papers and Chapter 2 of this thesis was greatly improved by these comments and suggestion. Also, the `Positive` solution (Section 2.8) was motivated by a discussion with Miklos Telek and Armin Heindl at TOOLS 2003. Applications of moment matching algorithms in inventory system analysis were suggested by Geert-Jan van Houtum (Section 2.1), and the benefit of acyclic PH distribution over cyclic one was suggested by Sindo Nunez Queija (Section 2.9) during my visit at Eindhoven University of Technology in July 2004.

DR was extended quite a bit in the summer of 2003, utilizing Neuts' algorithm for computing the inter-level passage time in a QBD process [134, 137]. The extended DR was then applied to an analysis of multiserver systems with multiple priority classes (Chapter 4), which was summarized in [156, 155, 70] with Adam Wierman, Mor Harchol-Balter, and Alan Scheller-Wolf.

DR was also applied to an analysis of threshold-based policies for the Beneficiary-Donor model (Chapter 7) in the spring of 2004. This interesting problem was inspired by a talk by Li Zhang at our weekly seminar, SQUALL, and Li Zhang, Mor Harchol-Balter, Alan Scheller-Wolf, and I started to analyzed the performance of various threshold-based policies. The analysis was summarized in an invited paper at Allerton Conference [153]. The extensive analysis of the threshold-based policies revealed robustness of these threshold-based policies. Mor Harchol-Balter, Alan Scheller-Wolf, and I continued to investigate the robustness of threshold-based policies for the Beneficiary-Donor model, and the results were summarized in [149]. some of the results in [149] were also presented at the MAMA workshop [69].

DR was further extended and formalized in the summer and fall of 2004. I defined a class of Markov chains called RFB/GFB processes, and the RFB/GFB process was analyzed via DR (Chapter 3). The RFB/GFB process can model all the multiserver systems that we had analyzed by then [67, 68, 69, 70, 149, 151, 153, 155, 156] as well as more multiserver systems such as multiserver systems with *nonpreemptive* priority classes. Also, in [67, 68, 69, 70, 149, 151, 153, 155, 156], only the performance of "beneficiary jobs" or "low priority jobs" was analyzed via DR, and the performance of "donor jobs" or "high priority jobs" needed to be analyzed independently with different techniques for different multiserver systems. By contrast, in the analysis of RFG/GFB process, the performance of "donor jobs" or "high priority jobs" is also analyzed with a unified framework of DR. This extension and formalization were inspired by a discussion with Ivo Adan, Andrei Sleptchenko, and Onno Boxma during my visit at Eindhoven University of Technology in July 2004. A part of the results in Chapter 3 was summarized in a technical report [143].

I would like to thank Mor Harchol-Balter, Mark S. Squillante, Varun Gupta, Alan Scheller-Wolf, and Adam Wierman for reading an earlier version of this thesis. Their comments and suggestions significantly helped improving the quality of the thesis. Also, the thesis would not be completed if it was not for their encouragement. Further, I would like to thank IBM Japan and IBM Tokyo Research Laboratory for their full support during the four years of my graduate study. Finally, I would like to thank Adam Wierman for making my graduate life productive and fun.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

How can we analyze the performance of multiserver systems efficiently and accurately?

1.1 Motivation

The use of multiple servers (such as computers, operators, and machines) is ubiquitous. In our everyday life, we frequently see more than one teller serving at a bank, more than one checker at a supermarket, and more than one cashier at a fast food restaurant, in situations where a single teller, checker, or cashier is insufficient to handle the volume of customers. Similarly, the use of multiple computers is popular in computer systems such as web server farms and supercomputing centers, because the use of multiple computers is a cost-effective and scalable solution for achieving high performance and dependability (reliability).

When we design a multiserver system (or even a single-server system), an important step is to analyze and understand their performance. The study of system performance analysis dates back to the seminal work of A. K. Erlang in 1917 [50], where he provided formulas that can be used to evaluate the performance at a telephone exchange. Erlang's formulas were heavily used by telephone companies to provide necessary and sufficient resources, which led to the successful and rapid growth of telephone networks. Erlang's work was followed by many researchers and developed into a theory known today as queueing theory.

Queueing theory was introduced to computer scientists in 1960s by L. Kleinrock and others, and contributed to the successful growth of computer systems such as the Internet and time-sharing systems. For example, J. R. Jackson's theory for networks of queues [83, 84] was used to study the performance of the ARPANET, a precursor of the Internet. Understanding how the performance is affected by various parameters was an important necessary step for optimizing the network topology, capacity assignment, and flow control of the ARPANET [101, 102].

Although Erlang's formulas and Jackson's theory are defined for (simple) multiserver systems, much of queueing theory has been developed ironically for *single*-server systems. In earlier days (prior to the 1970s), researchers in queueing theory provided beautiful and simple formulas that can be used directly to evaluate the performance of single-server systems and some of the simplest models of multiserver systems such as the ones studied by Erlang and Jackson. These explicit formulas not only can be evaluated numerically but also can give us good intuition on how the system parameters affect the performance. However, as researchers moved on to more complex models, the formulas that they obtained became more complex. This was particularly true for multiserver systems. As a result, although classical queueing theory provides beautiful formulas that allow us to evaluate the performance of complicated single-server systems, it provides much less for multiserver systems. (See [189] for more details of a history of queueing theory.)

Today's multiserver systems can employ more complex configurations and/or more complex resource allocation towards more efficient and more profitable operations. For example, advancement of operating systems and communication technology allows sharing resources among tasks or users for utilizing otherwise idle CPU cycles. Also, advancement of technology such as information retrieval and data mining enables one to identify customers (jobs, or tasks) who are likely to bring more revenue to a company, and today's routers allow a sophisticated assignment of customers to servers based on the available information on the customers (e.g. based on the priority of the customers).

For the performance evaluation of these multiserver systems with resource sharing or prioritization, analytical tools are largely unavailable. The lack of analytical tools is unfortunate, since analytical tools have been proved to be effective in allocating necessary and sufficient resources, for example, at telephone exchanges and the ARPANET. Such capacity planning remains difficult for multiserver systems with resource sharing or prioritization.

Also, due to the lack of analytical tools, it is not well understood what is a good configuration and what is a good resource allocation policy for a multiserver system with resource sharing or prioritization. It is also not well understood how different configurations and different resource allocation policies compare with respect to their performance in these multiserver systems. In fact, many fundamental questions regarding the performance of multiserver systems, of importance to system designers and service providers, are largely unanswered.

For example, when a high volume web site (web server farm) receives requests of web pages or files, each request must be dispatched to exactly one of the web servers to be served. Similarly, when a supercomputing center receives jobs (tasks), each job must be dispatched to exactly one of the host machines for processing (see Figure 1.1(a)). The rule for assigning requests/jobs to host servers/machines is known as the *task assignment policy*. The choice of the task assignment policy can have a significant effect on the performance perceived by users of web sites and supercomputing centers. Designing a distributed server system such as web server farms and supercomputing centers thus often comes down to choosing the "best" possible task assignment policy for the given system and user requirements.

However, analyzing even the simplest task assignment policies can prove to be difficult. As a result, fundamental questions regarding the optimal task assignment policy and relative performance among various task assignment policies are largely open.

> *Which job should be assigned to which server? What is the best task assignment policy, and how does its performance compare to other task assignment policies?*

Networks of workstations can benefit tremendously from allowing a user to take advantage of machines other than her own, at times when those machines are idle (see Figure 1.1(b)). This notion is often referred to as *cycle stealing*. Cycle stealing has been implemented in various workload management systems for networks of workstations such as Butler [139], Condor [115, 116], Benevolent Bandit [54], Sprite [48], Stealth Distributed Scheduler [109], and Linger-Longer [170]. Cycle stealing is also popular in Internet computing, which allows one to steal idle cycles of personal computers at home [41]. Internet computing has been used for the purpose of searching for extraterrestrial

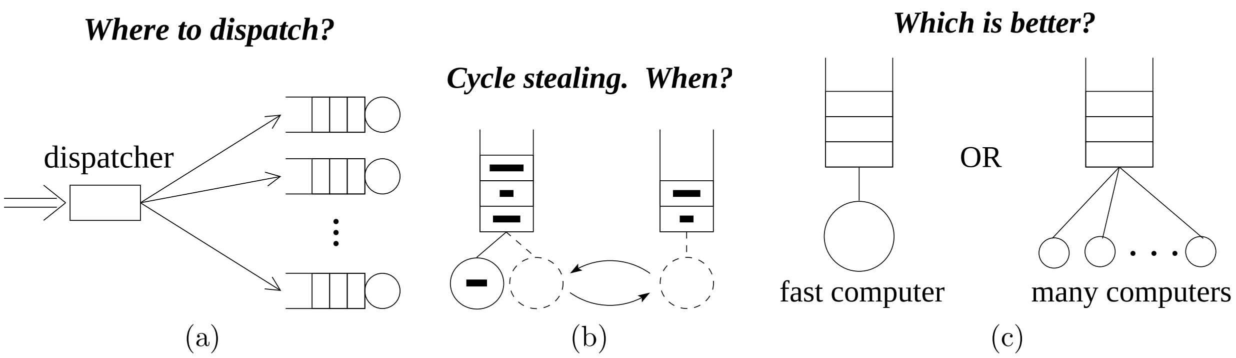


Figure 1.1: *Fundamental questions in multiserver systems.*

intelligence, protein folding research, cancer research, AIDS research, finding large prime numbers, etc. More generally, cycle stealing enables one to use multiple servers as a single resource, and has been receiving attention in the context of grid computing [14, 40, 56].

Despite the popularity of cycle stealing, the performance benefit or penalty of cycle stealing is not well understood, largely due to the lack of analytical tools for multiserver systems with cycle stealing. For example, when is cycle stealing a good idea?

> *When does cycle stealing pay, and how much? Should we always steal idle cycles? Should we steal only idle cycles?*

Analytical tools can also be used to gain insights into the physical nature of a system such as how the performance of the system is affected by the number of servers, system loads, job size distributions, and arrival processes of jobs. Such insights into the physical nature of a system are useful in designing good resource allocation policies for the system and in configuring the system effectively. However, due to the lack of analytical tools for multiserver systems with resource sharing and job prioritization, the physical nature of these multiserver systems is not well understood. For example, a single fast computer of speed s is more expensive than k slow computers, each of speed s/k, but does this mean that a single fast computer is better (see Figure 1.1(c))?

> *How many servers are best? Does the answer change depending on system loads, job size distributions, and how the jobs are processed (e.g., first-come-first-served or with priority)? How much is the performance different between a single fast computer and k slow computers? What is the optimal k?*

The goal of this thesis is thus twofold. First, we will develop new analytical tools for analyzing the performance of multiserver systems with resource sharing or job prioritization. Second, we will apply the new analytical tools to analyze various multiserver systems, addressing the fundamental questions regarding their performance. Our analysis leads to many lessons and guidelines useful in configuring multiserver systems and designing good resource allocation policies for multiserver systems.

1.2 Difficulty of performance analysis of multiserver systems

We begin by specifying two fundamental problems that we will address towards the first goal of this thesis, i.e. new analytical tools for the performance analysis of multiserver systems with resource sharing or job prioritization. To motivate these fundamental problems, it helps to first review some major attempts to overcome the limitations of classical queueing theory in the performance analysis of complex (multiserver) systems.

Since classical approaches in queueing theory lead to complex expressions or do not apply for complex (multiserver) systems, at least two main streams of research emerged: computational probability and heavy traffic approximations. The theme of computational probability is to give *numbers* efficiently. Computational probability provides ways (algorithms) to *compute* performance measures rather than explicit (closed form) expressions. The need for computational probability was advocated by M. F. Neuts and others. On the other hand, the theme of heavy traffic approximations is to provide explicit expressions that are accurate in the heavy traffic limit (when the servers are almost always busy). Heavy traffic approximations originates from the work of J. F. C. Kingman in 1961 [99], where he provided a heavy traffic approximation for a single-server system via the central limit theorem. Since computational approaches are subject to inaccuracy and/or slow convergence in the heavy traffic limit, heavy traffic approximations complement computational probability.

However, even computational probability usually does not apply directly to multiserver systems with resource sharing or job prioritization. In this thesis, we will address two fundamental problems towards an analysis of multiserver systems with resource sharing or job prioritization via computational probability. The first problem involves approximating a general probability distribution by a collection of exponential distributions. This is an important step in an analysis of (multiserver) systems via computational probability. The second problem involves a multidimensionally infinite state space (multidimensional state space) that is needed to capture the behavior of a multiserver system with resource sharing or prioritization. The difficulty of a performance analysis via computational probability stems from the multidimensional state space.

Approximating general distributions

Typically, a first step in the analysis via computational probability is to model the behavior of the system as a Markov chain. (For an introduction to Markov chains, we refer readers to [169, 167, 210].) By analyzing the Markov chain, we can understand the performance of the system, for example with respect to the mean response time[1] (see Figure 1.2). Modeling as a Markov chain is possible when system parameters such as service demand (time needed to complete a job) and interarrival time (time between two consecutive arrivals of jobs) are exponential random variables or some combinations of exponential random variables (for example, a sum of exponential random variables)[2]. However, these system parameters are not usually given as combinations of exponential random variables, but they are rather given as some general probability distributions. Thus, an important step in modeling system behavior as a Markov chain is to map these general probability distributions into some combinations of exponential distributions. Here, we face a fundamental problem in Markov chain modeling.

How can we map a general distribution into a combination of exponential distributions?

[1]The response time of a job refers to the time from when the job arrives until it is completed.
[2]We consider only continuous time systems.

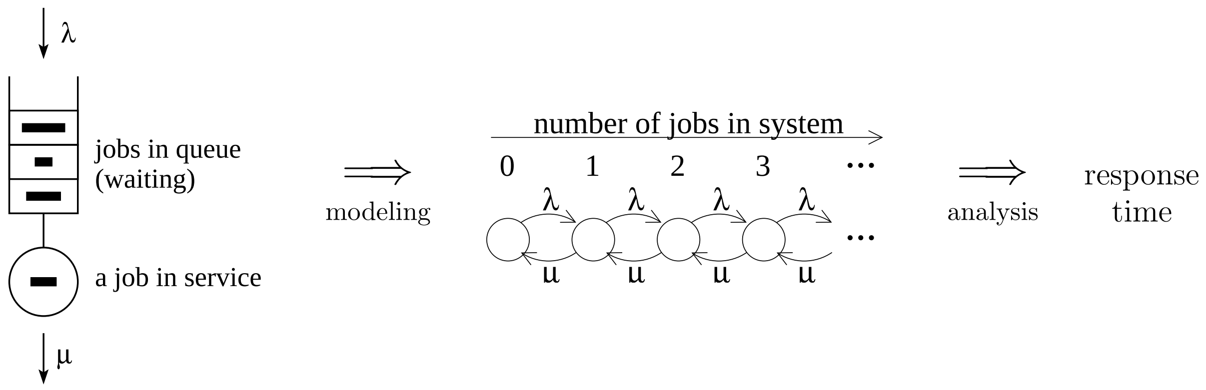


Figure 1.2: *An example of Markov chain modeling: an M/M/1/FCFS queue. The system is modeled as a Markov chain, and the analysis of the Markov chain gives (mean, variance, etc. of) the response time of the system. Here, jobs arrive according to a Poisson process with rate λ (i.e., the interarrival time between two consecutive job arrivals has an exponential distribution with rate λ). The jobs are served in the order of their arrivals (first-come-first-served, FCFS). The service demand of a job has an exponential distribution with rate μ.*

Multidimensional Markov chains

One of the major achievements in computational probability is the development of algorithmic methods, which are known as matrix geometric methods [136] and matrix analytic methods [111]. Matrix geometric/analytic methods allow us to efficiently analyze a Markov chain on a state space that is infinite in one dimension and finite in another dimension (1D Markov chain), provided that the Markov chain has a certain structure (Figure 1.3 shows an example of such a 1D Markov chain). This is in contrast to classical approaches (that provide exact and explicit expressions), which result in very complicated expressions that are difficult to evaluate numerically, for the general 1D Markov chains.

The difficulty in the analysis of many multiserver systems stems from the fact that the Markov chain that models the behavior of the system often has a state space that is infinite in multiple dimensions (multidimensional Markov chain). Most multidimensional Markov chains cannot be analyzed efficiently via matrix analytic methods or other computational methods. Unfortunately, the multidimensional Markov chain appears quite frequently in the analysis of multiserver systems with resource sharing or job prioritization, as these multiserver systems involve multiple classes/types of jobs.

For example, consider two servers, Betty and Dan, each serving its own queue, but Dan (donor) allows Betty (beneficiary) to steal his idle cycles. That is, when Dan has no jobs in his queue (and Betty has more than one job in her queue), we allow Dan to process Betty's job. We can model the behavior of this simple cycle stealing system by a Markov chain, but the Markov chain has a state space that is infinite in two dimensions (2D Markov chain). This is illustrated in Figure 1.4. Since there is an inherent dependency between the number of Dan's jobs and the number of Betty's jobs, the 2D Markov chain cannot simply be decomposed into two 1D Markov chains. As a result, the performance analysis of this cycle stealing system requires an analysis of the multidimensional Markov chain. Here, a fundamental problem arises in the performance analysis of multiserver systems.

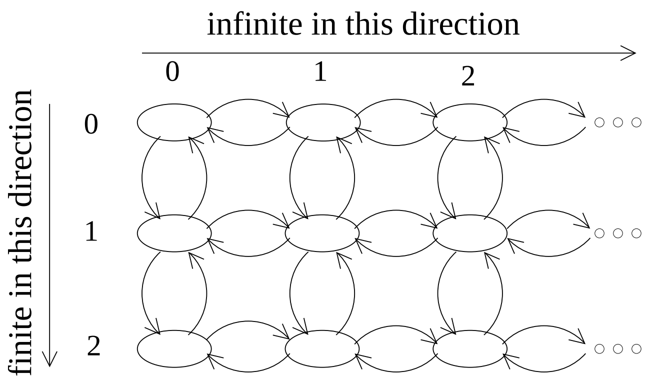


Figure 1.3: *A 1D Markov chain.*

How can we analyze Markov chains on multidimensionally infinite state spaces?

1.3 Road map

This thesis is organized into three parts. In Part I, we will develop new analytical tools for the performance analysis of multiserver systems. In Part II, we will address fundamental questions regarding the performance of multiserver systems. In Part III, we will further discuss applications of the results in this thesis, and conclude. Although the analytical tools that we provide in Part I are heavily used to derive results in Part II, Part II can be read independently without any difficulty. Alternatively, a reader can obtain basic ideas of our analytical tools by reading "Overview" sections at the beginning of each chapter in Part I, before proceeding to Part II. If, instead, a reader is interested in applying or building upon our analytical tools, he/she can read the entire Part I, which provides detailed information on our analytical tools.

1.3.1 Synopsis of Part I: Analytical tools for multiserver systems

Part I is organized into two chapters (see Figure 1.5). In Chapter 2, we address the first fundamental problem in an analysis of multiserver systems, namely "How can we map a general distribution into a combination of exponential distributions?" Specifically, we develop *moment matching algorithms*, which allow us to map a general probability distribution into a combination of exponential distributions. Chapter 3 is the heart of this thesis, and here we address the second fundamental problem in an analysis of multiserver systems, namely "How can we analyze Markov chains on multidimensionally infinite state spaces?" Specifically, we introduce *dimensionality reduction* (DR), which allows us to analyze a class of multidimensional Markov chains that can model many multiserver systems with resource sharing or job prioritization. The moment matching algorithm is also used in a key step of DR.

Moment matching algorithm (Chapter 2)

In Chapter 2, we develop *moment matching algorithms*, which allow us to map a general probability distribution into a combination of exponential distributions, known as a phase type (PH) distribution. Specifically, a moment matching algorithm maps an input (general) distribution, G, into a PH

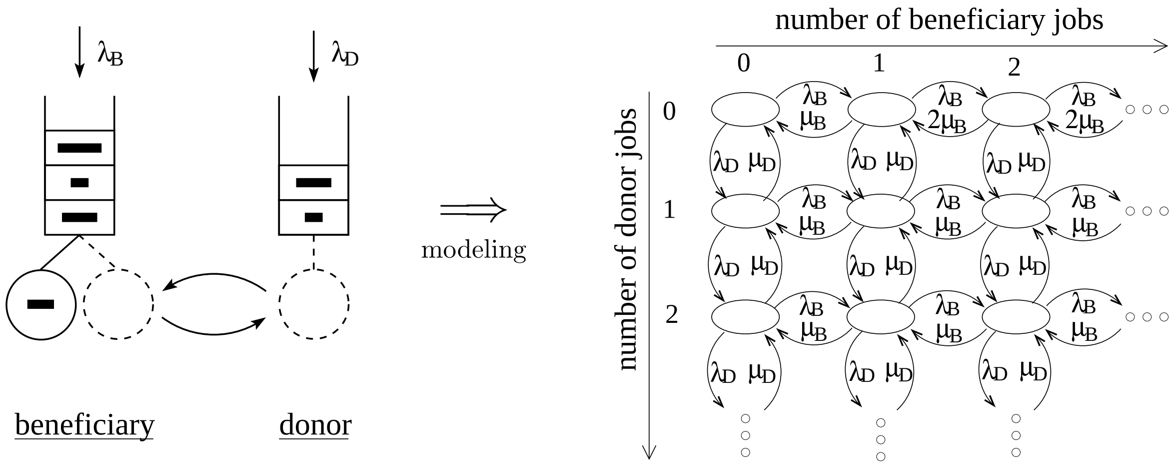


Figure 1.4: *A multidimensional Markov chain that appears in a multiserver analysis. Here, the beneficiary jobs arrive according to a Poisson process with rate* λ_B*, and the donor jobs arrive according to a Poisson process with rate* λ_D*. The service demands of these jobs have exponential distributions. The beneficiary jobs are usually served by the beneficiary server with rate* μ_B*, and the donor jobs are served by the donor server with rate* μ_D*. When there are no donor jobs and there are at least two beneficiary jobs, the donor server helps the beneficiary server (i.e., the donor server processes a beneficiary job), and hence the beneficiary jobs are completed with rate* $2\mu_B$*.*

distribution, P, such that some moments of G and P agree. For example, if the first moment of G and P agree, then the corresponding random variables have the same mean. If the first two moments of G and P agree, then the corresponding random variables have the same mean and variance. In practice, matching more moments leads to a better approximation. Moment matching algorithms can be evaluated along four different measures:

1. **The number of moments matched**: In general matching more moments is desirable.

2. **The computational efficiency of the algorithm**: It is desirable that the algorithm have short running time.

3. **The generality of the solution**: Ideally the algorithm should work for as broad a class of distributions as possible.

4. **The minimality of the number of phases**: It is desirable that the matching PH distribution, P, have a small number of exponential distributions (phases).

We will provide moment matching algorithms which perform very well along all four of these measures. (i) Our moment matching algorithms match first three moments; (ii) their running time does not depend on the input distribution (we provide closed form solutions for the parameters of the approximate PH distribution); (iii) they apply to the widest possible set of input distributions; (iv) they require a nearly minimal number of phases.

To prove that our moment matching algorithms use a nearly minimal number of phases, we need to know the minimal number of phases needed to approximate (by matching first three moments) an input distribution by a PH distribution. We will therefore start by providing a formal characterization

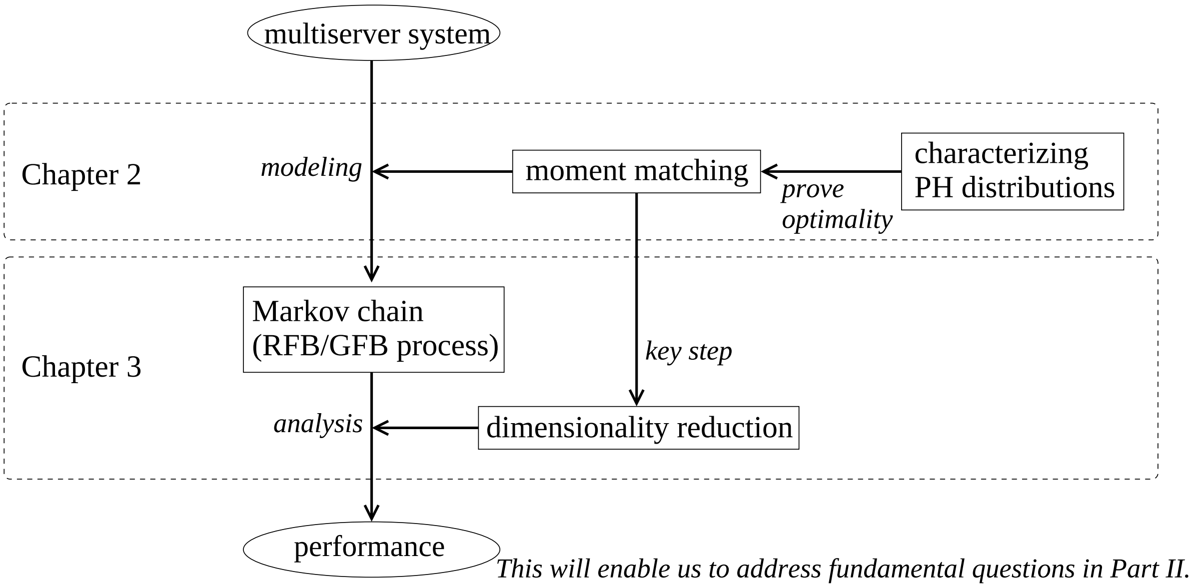


Figure 1.5: *Organization of Part I: Analytical tools for multiserver systems.*

of the set of distributions that are approximated by an n-phase PH distribution, for each $n = 1, 2, 3, ...$. This characterization is used to prove the minimality of the number of phases used in our moment matching algorithms.

Dimensionality reduction of Markov chains (Chapter 3)

Instead of analyzing individual multiserver systems separately, we will define a broad class of multidimensional Markov chains that allow us to model many multiserver systems with resource sharing or job prioritization, and show an (approximate) analysis of these Markov chains. We will refer to these multidimensional Markov chains as RFB/GFB (recursive foreground-background or generalized foreground-background) processes. Also, we will refer to our new analytical method for analyzing the RFB/GFB process as dimensionality reduction (DR). Our analysis of the RFB/GFB process via DR will enable one to analyze the performance of many multiserver systems by simply modeling them as RFB/GFB processes. We will provide many examples of such multiserver systems.

The key idea in the analysis of the RFB/GFB process via DR is the dimensionality reduction of a multidimensional Markov chain: DR reduces (approximates) a multidimensional Markov chain, which is hard to analyze, into a 1D Markov chain, which is easy to analyze. Figure 1.6 illustrates the idea of DR for a simple 2D Markov chain. The 1D Markov chain will be carefully constructed such that the 1D Markov chain well approximates the multidimensional Markov chain.

Since the analysis of the RFB/GFB process via DR involves approximations, we will extensively validate the accuracy of DR via simulation. We will find that the error in DR is typically within 5% (and often within 1%) with respect to first order metrics (such as mean response time and mean queue length), and within 10% with respect to second order metrics (such as the variance of the queue length).

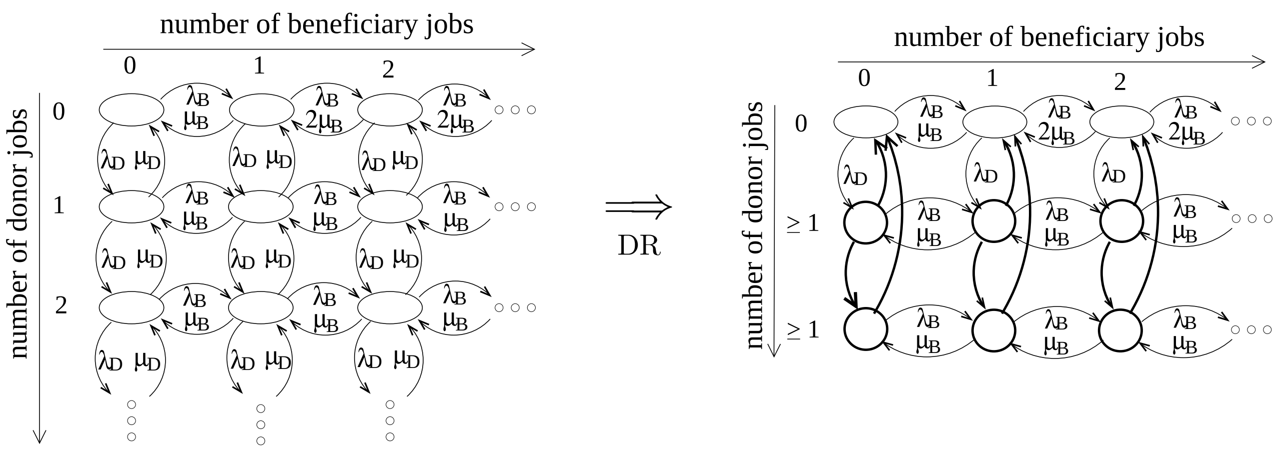


Figure 1.6: *The key idea in DR: dimensionality reduction (2D → 1D).*

1.3.2 Synopsis of Part II: Performance analysis of multiserver systems

Part II is organized into four chapters (see Figure 1.7). In each chapter, we analyze the performance of particular multiserver systems via DR, and address fundamental questions regarding the multiserver systems, as we elaborate below. Our analysis leads to lessons and guidelines that are useful in configuring multiserver systems and designing resource allocation policies for multiserver systems.

Configuring multiserver systems with multiple priority classes (Chapter 4)

In Chapter 4, we address a fundamental question in designing multiserver systems, namely "How many servers are best to minimize mean response time?" As may be expected, this question has a long history in the literature; however, it has not been fully resolved. As early as 1958, Morse observed that for an M/M/k/FCFS system[3], the optimal number of servers is one ($k = 1$) [131]. In 1970, Stidham formalized and generalized the observation by Morse [188]; Stidham showed that a single server is optimal in a G/Er/k/FCFS system, namely when the arrival process is a general process and the service demand has an Erlang distribution (i.e., when the service demand is a sum of i.i.d., or independent identically distributed, exponential random variables). Stidham's result is extended by Brumelle to the system where the service demand is at most as variable as the exponential random variable; i.e., a single server is optimal in a G/G/k/FCFS system where the service demand distribution has the squared coefficient variation at most one ($C^2 \leq 1$) [31]. For general service demand distributions, only results for limiting cases are known. Specifically, the work by Reiman and Simon [161] can be used to show that a single server ($k = 1$) is optimal in the light traffic limit (when servers are almost always idle), and the work by Iglehart and Whitt [79] implies that a multiserver system (GI/GI/k/FCFS) and its single server counterpart (GI/GI/1/FCFS) coincide in the heavy traffic limit (when servers are almost always busy). In fact, based on these results in the light and heavy traffic limits, Mandelbaum and Reiman state "One expects, therefore, that the difference in performance between the systems[4] is maximal in light traffic, and it diminishes as

[3]An M/M/k/FCFS system has a single queue and k servers, where jobs arrive according to a Poisson process, and the service demand has an exponential distribution. The jobs are served in the order of their arrivals (first-come-first-served, FCFS).

[4]These systems refer to a single server system and a multiserver system with the same service capacity.

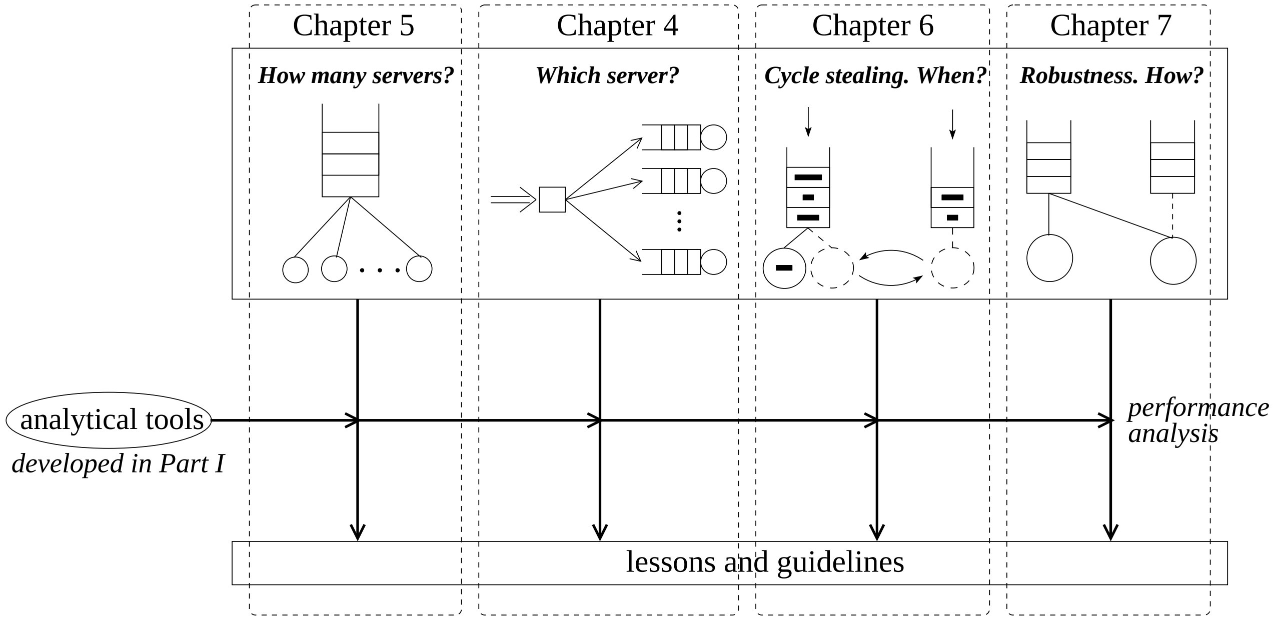


Figure 1.7: *Organization of Part II: Performance analysis of multiserver systems.*

utilization increases to heavy traffic." [120].

Contrary to the prior work that suggests the optimality of single server systems in special/limiting cases, we will see that a multiserver system often provides lower mean response time than its single server counterpart. Specifically, we will see that the optimal number of servers increases as the load and/or the service demand variability increases. Intuition behind this finding is that multiple servers allow short jobs to avoid queueing behind long jobs; by contrast, a long job blocks all the other jobs in a single server system, while being processed.

Above we assume that jobs are served in the order of their arrivals (FCFS); however, real world systems are not always FCFS. Rather, there is often inherent scheduling in the system to allow high priority jobs to move ahead of low priority jobs (priority scheduling). For example, the high priority jobs may carry more importance, e.g., representing users who pay more. Alternatively, high priority jobs may simply consist of those jobs with small service demand, since giving priority to short jobs reduces mean response time overall.

The prevalence of priority scheduling motivates us to study the question "How many servers are best?" under priority scheduling. In particular, how does the optimal number of servers under priority scheduling compare to that under FCFS, and how is the answer affected by system parameters such as loads and service demand distributions? To answer these questions, we analyze the mean response time in a multiserver systems with multiple priority classes via DR, as its analysis involves a multidimensional Markov chain.

Our analysis shows that the optimal number of servers can be quite different between FCFS and priority scheduling. Intuition behind this finding is seen by considering an example where shorter jobs have higher priority: in this case, the benefit of multiple servers (i.e., allowing short jobs to avoid queueing behind long jobs) is smaller under priority scheduling, since priority scheduling by itself allows small jobs to jump ahead of long jobs. We will also study how the optimal number of servers differs for different priority classes, and how the mean response time is affected by choosing a non-optimal number of servers. We will also evaluate the accuracy of various existing approximations for multiserver systems with multiple priority classes, and see that the error in the mean response time can be as high as 50% or more.

Improving traditional task assignment policy (Chapter 5)

In Chapter 5, we consider a fundamental question in multiserver systems: "Which job should be assigned to which server?" Specifically, we study task assignment policies in a server farm architecture shown in Figure 1.1(a), where our objective is to minimize the mean response time. We assume that the execution of a job cannot be interrupted and subsequently resumed; i.e., a job is run to completion (nonpreemptive). Our model is motivated by servers at supercomputing centers and other systems, where jobs are typically run to completion.

When service demands (job sizes) have high variability, it has been shown analytically and empirically that a size-based task assignment policy, `Dedicated`, far outperforms other common task assignment policies[5] [65, 174]. In the `Dedicated` policy, some servers are designated as the "short servers" and others as the "long servers." Short jobs are always sent to the short servers and long jobs to the long servers. The idea behind `Dedicated` is that, under highly variable service demands, it is important to isolate short jobs from the long jobs, as having short jobs wait behind long jobs is very wasteful. Given the extremely high variability of service demands under many computer workloads [46, 47, 66, 113, 174], `Dedicated` is preferable to other policies.

However `Dedicated` is still *not* optimal. One problem is that `Dedicated` can lead to situations where the servers are not fully utilized: five consecutive short jobs may arrive, with no long job, resulting in an idle long server. This is especially likely in common computer workloads, where there are many short jobs and just a few very long jobs, resulting in longer idle periods between the arrivals of long jobs.

The possible low utilization of `Dedicated` motivates us to study a new task assignment policy, the size-based task assignment with cycle stealing under immediate dispatching, `SBCS-ID` (see Figure 1.8). The idea behind `SBCS-ID` is that we would like to segregate jobs by size so as to provide isolation for short jobs, but during times when the long job server is free, we would like to *steal* the long server's idle cycles and use those to serve incoming short jobs. We will see that `SBCS-ID` provides significant improvement over `Dedicated`.

However, under `SBCS-ID`, only those short jobs arriving *after* the long server has entered an idle period can steal idle cycles of the long server. Hence, if a short job arrives just before the long server enters an idle period, the short job is not eligible for stealing idle cycles of the long server.

This motivates us to study another new task assignment policy, the size-based task assignment with cycle stealing under central queue, `SBCS-CQ` (see Figure 1.9). Under `SBCS-CQ`, arriving jobs are held in a central queue, instead of being dispatched immediately. The central queue delays the decision of which jobs should be processed by which server, and this enables more short jobs to utilize the idle cycles of the long server.

Although cycle stealing is an old concept, the performance analysis of policies with cycle stealing has eluded researchers as it involves multidimensional Markov chains. We will analyze the performance of `SBCS-ID` and `SBCS-CQ` via DR. Our analysis shows that the short jobs can benefit by an order of magnitude under both `SBCS-ID` and `SBCS-CQ` as compared to `Dedicated`, while the long

[5] These common task assignment policies include the round robin policy, the shortest queue policy, and the least remaining work policy.

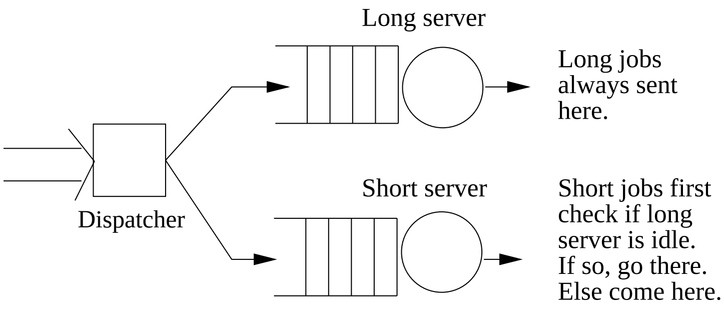


Figure 1.8: *The* SBCS-ID *policy. An arriving long job is always dispatched to the long server. An arriving short job first checks to see if the long server is idle. If so, the short job is dispatched to the long server; otherwise, the arriving short job is dispatched to the short server. Jobs at each server are serviced in FCFS order.*

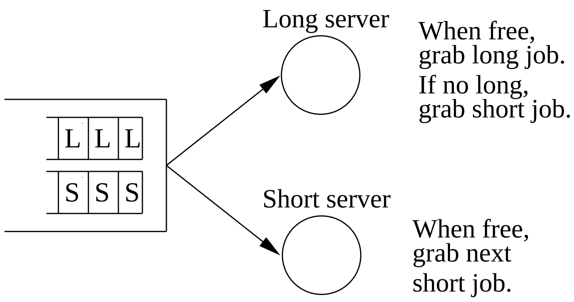


Figure 1.9: *The* SBCS-CQ *policy. All jobs are held in a central queue. Whenever the short server becomes idle, it picks the first short job in the queue to run. Whenever the long server becomes idle, it picks the first long job in the queue. However, if there is no long job, the long server picks the first short job in the queue.*

jobs are penalized only by a small percentage. We also find that SBCS-CQ is a superior strategy to SBCS-ID from the perspective of both the short jobs and the long jobs. We will also study the effect of service demand variability, the effect of increasing the arrival rate of the short jobs and the number of short servers, and the effect of stealing idle cycles of *short* servers.

Reducing switching costs in cycle stealing (Chapter 6)

In Chapter 6, we consider the benefit and penalty of cycle stealing in the context of a network of workstations, where the donor (Dan) allows the beneficiary (Betty) to steal his idle cycles (recall Figure 1.4). Although cycle stealing provides obvious benefits to the beneficiary, these benefits come at some cost to the donor. Typically, there is a switching time, K_{sw}, required for the donor server to start working on the beneficiary's jobs, as well as a switching back time, K_{ba}. For example, the beneficiary's job may have to be checkpointed and the donor's working set memory reloaded before the donor server can resume, delaying the resumption of processing of donor jobs. In our model, we refer to these additional costs associated with cycle stealing as switching costs (or switching times).

Due to non-zero switching costs, the donor server's switching may pay only when the beneficiary's queue length, N_B, is sufficiently long, and the donor server's switching back may pay only when the

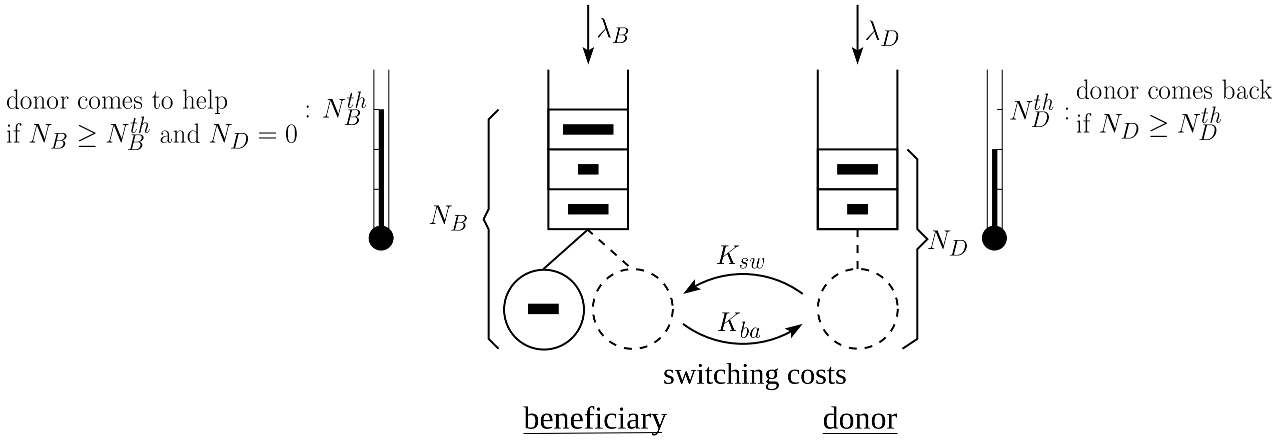


Figure 1.10: *A threshold-based policy for reducing switching costs in cycle stealing. If $N_B \geq N_B^{th}$ and $N_D = 0$, the donor server starts switching to the beneficiary's queue. After K_{sw} time, the donor server is available to work on the beneficiary's jobs. When N_D reaches N_D^{th}, the donor server starts switching back to its own queue. After K_{ba} time, the donor server resumes working on its own jobs.*

donor's queue length, N_D, is sufficiently long. This motivates us to introduce a threshold-based policy, whereby whether the donor server should switch (and switch back) is decided based on a queue length, i.e. whether the queue length exceeds a threshold (see Figure 1.10).

Our primary goal is to understand the benefit of cycle stealing for the beneficiary and the penalty to the donor, as a function of switching times. A secondary goal is to derive parameter settings for cycle stealing. We seek to understand the optimal threshold values, N_B^{th} and N_D^{th}. More broadly, we seek general insights into which system parameters have the most significant impact on the effectiveness of cycle stealing.

To quantitatively understand the benefit and penalty of cycle stealing as a function of switching times, we need to be able to analyze the mean response time under cycle stealing with switching times. However, an exact analysis for this system is not known in the literature (even without switching times), since it would involve multidimensional Markov chains. We will analyze the mean response time under the threshold-based policy for reducing switching costs in cycle stealing via DR. Our analysis yields many interesting results concerning cycle stealing.

Designing robust resource allocation policies (Chapter 7)

A common problem in multiserver systems is deciding how to allocate resources (e.g. operators, CPU time, and bandwidth) among jobs. A good resource allocation policy that maximizes system performance often has parameters (e.g., thresholds) that need to be tuned. Since the optimal settings of the parameters typically depend on environmental conditions such as system loads, a resource allocation policy whose parameters are chosen to achieve the best performance under a certain load can provide poor performance when the load fluctuates or when the estimation/prediction of the load is wrong.

Designing resource allocation policies that are robust against load fluctuations and misestimation is important, since irregularity of arrival processes has been observed in various systems in various ways, including long range dependence, self-similarity, time of day effect, and flash crowd (also known

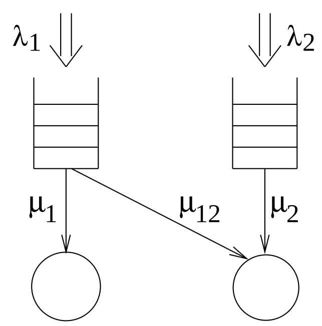


Figure 1.11: *The Beneficiary-Donor model. Jobs arrive at queue 1 and queue 2 with average arrival rates λ_1 and λ_2, respectively. Jobs have exponentially distributed service demands; however, the running time of a job may also depend on the affinity between the particular server and the particular queue. Hence, we assume that server 1 processes jobs in queue 1 with rate μ_1 (jobs/sec), while server 2 can process jobs in queue 1 with rate μ_{12} (jobs/sec) and can process jobs in queue 2 with rate μ_2 (jobs/sec).*

as slashdot effect). Recent studies show long-range dependence or self-similarity in arrival processes in various computer and communication systems, including the Internet [46, 114, 158, 206, 207], supercomputing centers [183], web servers [80, 81], and MPEG streams [110]. Squillante et al. show that correlation in arrival process can have a big impact on the system performance [183]. Time of day effect [35] has been observed in web servers [10, 81, 80] and supercomputing centers [53, 78, 208]. Flash crowd [10] and Slashdot effects [199] refer to the phenomenon that it is not uncommon to experience a more than 100-fold increase in demand when a site becomes popular, and these effects are often observed in web servers.

In Chapter 7, we will design and study characteristics of various resource allocation policies for the Beneficiary-Donor model (see Figure 1.11), which frequently comes up in the literature [4, 17, 58, 61, 71, 105, 125, 176, 181, 182, 185, 186, 205]. Whereas the standard metric for evaluating resource allocation policies for the Beneficiary-Donor model is the (weighted) mean response time, we choose to consider an additional metric, which we refer to as *robustness*. We introduce two types of robustness: *static robustness* and *dynamic robustness*. Static robustness measures robustness against *misestimation* of load. To evaluate static robustness, we analyze the mean response time of resource allocation polices for a range of loads to see how a policy tuned for *one* load behaves under different loads. Dynamic robustness measures robustness against *fluctuations* in load. To evaluate dynamic robustness, we analyze the mean response time of resource allocation policies under fluctuating load.

Although various resource allocation policies in the Beneficiary-Donor model have been studied in the literature, the majority of the prior work has investigated only the optimality of these resource allocation policies in limiting or special cases. With respect to the analysis of mean response time, only coarse approximations exist for most of the resource allocation policies in the Beneficiary-Donor model, since an exact analysis would involve multidimensional Markov chains. DR allows us to analyze the mean response time, as well as the static and dynamic robustness, of these and other resource allocation policies.

We will see that single-threshold (resource) allocation policies have either poor mean response time or poor static robustness. This motivates us to introduce a multi-threshold allocation policy, whose threshold value is self-adapted to the load, and show this has significant advantage over single-threshold allocation policies with respect to achieving both low mean response time and good static robustness. Surprisingly, we will find that the use of multiple thresholds offer only small advantage over a single-threshold with respect to *dynamic* robustness. That is, a (single-threshold) policy that has poor static robustness can excel in dynamic robustness.

1.3.3 Synopsis of Part III: Further discussion and conclusion

Part III is organized into two chapters. In Chapter 8, we discuss how the analytical tools developed in Part I and the results of our analysis or lessons learned in Part II may help in designing and operating real-world systems in general and contact centers (call centers) in particular. Chapter 9 concludes the thesis.

Applications in contact center design (Chapter 8)

In today's competitive business, it becomes increasingly important to provide *accessibility* to the company, which has a direct impact on the customers' perception of quality, in order to acquire, keep, and grow customers [8, 58, 123]. A contact center provides such accessibility for customers who buy or reserve products or services and for customers who have questions, complaints, or need for technical supports and other customer services. Although a contact center is a significant revenue generator, its operating expenses are also huge, and human resource costs are estimated to account for 50-75% of the total operating expenses [58, 123].

Thus, a major challenge in contact centers is determining the number of agents to be assigned (capacity planning) such that a right level of customer satisfaction is achieved at right operational cost. Designing good routing policies (that determine which customer is served by which agent) is also important, since good routing policies can increase the accessibility or shorten the waiting time without increasing the number of agents. As a result, a significant amount of research has addressed analytical tools and simulation techniques that support capacity planning and routing policy design at contact centers [119].

However, operations at contact centers have become increasingly complex, and the increased complexity makes it difficult to apply existing analytical tools to capacity planning and routing policy design at today's contact centers. We will discuss specifically how the analytical tools developed and lessons learned in this thesis may help in capacity planning and routing policy design at contact centers today.

Part I

Analytical tools for multiserver systems

Chapter 2

Moment matching algorithm

How can we map a general distribution into a combination of exponential distributions?

Much of queueing theory revolves around the exponential distribution, since the memoryless nature of the exponential distribution enables an analysis via Markov chains. Unfortunately, real-world workloads such as CPU service requirements, file sizes, durations of FTP transfers, etc. do not have exponential distributions, but rather follow various general distributions. A way for us to deal with workloads with general distributions is to model these general distributions by a combination of exponential distributions, known as a phase type (PH) distribution, first introduced by M. F. Neuts [133, 136]. The PH distribution then fits nicely into the Markov chain.

A popular approach in mapping a general probability distribution, G, into a phase type (PH) distribution, P, is to choose P such that some moments of P and G agree. Matching the first moment of any nonnegative distribution is possible by a single exponential distribution. Matching the first moment is certainly important, but unfortunately it is often not sufficient, as ignoring the higher moments can result in misleading conclusions.

Thus, it is desirable to match more moments of the input distribution G by P. Matching more moments may be possible if we are allowed to use many exponential distributions (phases). However, the use of many phases in the PH distribution increases the complexity of the Markov chain, and makes its analysis hard. Matching many moments using a small number of phases may be possible if we are allowed to use unbounded computational resources or if we limit the class of input distributions. However, these limitations are not desirable. Achieving all the four desirable properties is the challenge in designing a moment matching algorithm:

1. It is desirable that more moments of the input distribution, G, and the matching PH distribution, P, agree.

2. It is desirable that P have a small number of phases.

3. It is desirable that the algorithm be defined for a broad class of input distributions.

4. It is desirable that the algorithm have short running time.

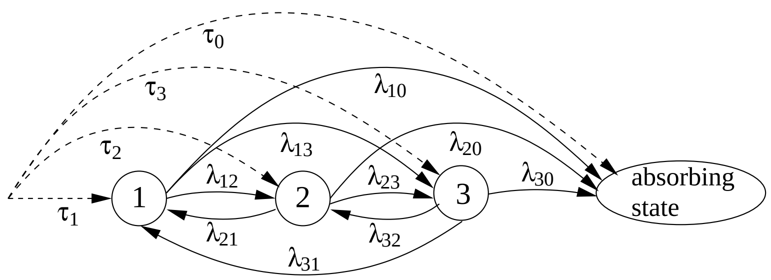


Figure 2.1: *A PH distribution is shown as the distribution of the time until absorption in a continuous time Markov chain. At time 0, the Markov chain is in state i with probability τ_i for $0 \le i \le 3$, where state 0 denotes the absorbing state. The time that the Markov chain enters the absorbing state is said to have a PH distribution. See Section 2.2 for a more detailed explanation of the PH distribution.*

2.1 Overview

In this chapter, we will design moment matching algorithms. To prove the optimality of our moment matching algorithms with respect to the number of phases used in the solution, we will also provide a characterization of the general probability distribution with respect to the necessary number of phases in the approximate PH distribution. In this section, we briefly summarize these results. Details will be provided in later sections.

2.1.1 Moment matching algorithms

A moment matching algorithm maps a general probability distribution, G, into a PH distribution, P. A PH distribution is a combination of exponential distributions with a certain structure. Examples of PH distributions include an exponential distribution, the convolution of exponential distributions, and a mixture of exponential distributions. More formally, a PH distribution is defined as the distribution of the time until absorption in a Markov chain (see Figure 2.1). Thus, essentially, a moment matching algorithm takes a general probability distribution, G, as an input, and outputs a Markov chain with an absorbing state together with the probability vector for the initial state, such that some moments of the distribution of the absorption time in the Markov chain (the PH distribution, P) agree with those of G. By convention, when the Markov chain has n states, we say that the PH distribution has n *phases*.

Key idea

The general approach in designing moment matching algorithms in the literature is to start by defining a subset $\mathcal{S}$ of the PH distributions, and then map each input distribution G into a distribution in $\mathcal{S}$. The reason for limiting the solution to a distribution in $\mathcal{S}$ is that this narrows the search space and thus improves the computational efficiency of the algorithm. One has to be careful in defining the subset $\mathcal{S}$, however. If $\mathcal{S}$ is too small, it may exclude solutions with a minimal number of phases. Also, if $\mathcal{S}$ is too small, it may limit the class of input distributions, or it may limit the number of moments that can be matched.

The key idea in our moment matching algorithm is the way that we define the subset, $\mathcal{S}$, of PH distributions, which we call EC (Erlang-Coxian) distributions. The EC distribution has only six free

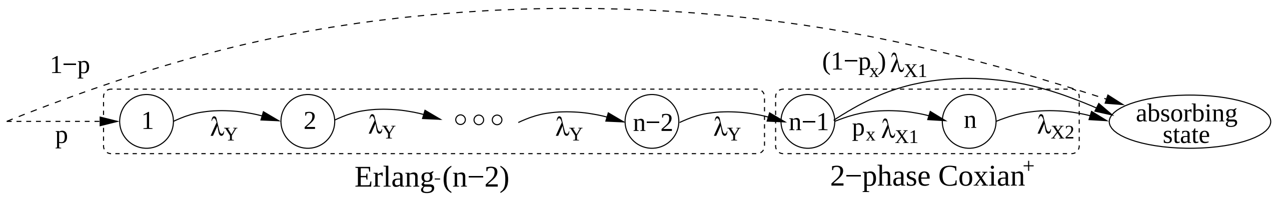


Figure 2.2: *The Markov chain whose absorption time defines an n-phase EC distribution. The first box above depicts the Markov chain that defines an Erlang-$(n-2)$ distribution, and the second box depicts the Markov chain that defines a two-phase Coxian$^+$ PH distribution.*

parameters, regardless of the number of phases, which allows us to derive a closed form solution for these parameters in terms of the moments of the input distribution. This is in contrast to the general PH distribution, as an n-phase PH distribution has $\Theta(n^2)$ free parameters. Furthermore, the class of EC distributions is broad enough to excel in the other three desired properties. In particular, the EC distribution allows us to match the first *three* moments of the input distribution, and it applies to the broadest possible class of input distributions (almost all nonnegative distributions). Also, the number of phases used in the output EC distribution is nearly minimal. Below, we say that distribution G is *well-represented* by P if P and G agree on their first three moments.

Definition 1 *A distribution G is* **well-represented** *by a distribution F if F and G agree on their first three moments.*

Figure 2.2 shows the Markov chain whose absorption time defines an n-phase EC distribution. Here, at time zero, the Markov chain is in state 1 with probability p and in the absorbing state with probability $1-p$. If the absorbing state is the initial state, the absorption time is zero. When the Markov chain is in state i, it stays in the state for a random time having an exponential distribution with rate λ_Y, and then transitions to state $i+1$, for $1 \le i \le n-2$. (The time it takes to transition from state 1 to state $n-1$ is said to have an Erlang-$(n-2)$ distribution or an $(n-2)$-phase Erlang distribution, E_{n-2}). When the Markov chain is in state $n-1$, it stays in the state for a random time having an exponential distribution with rate λ_{X1}, and then transitions to state n with probability p_X and to the absorbing state with probability $1-p_X$. When the Markov chain is in state n, it stays in the state for a random time having an exponential distribution with rate λ_{X2}, and then transitions to the absorbing state. (The time it takes to transition from state $n-1$ to the absorbing state is said to have a two-phase Coxian$^+$ PH distribution (as we will define in Section 2.2).) The time until the Markov chain enters the absorbing state is said to have an n-phase EC distribution.

Definition 2 *An n-phase EC (Erlang-Coxian) distribution is the convolution of an $(n-2)$-phase Erlang distribution and a two-phase Coxian$^+$ distribution, where the EC distribution may have mass probability at zero.*

We now provide some intuition behind the creation of the EC distribution. We will see that a Coxian$^+$ PH distribution is very good for approximating a distribution with high variability. In particular, we will see that a two-phase Coxian$^+$ PH distribution can well-represent any distribution that has high second and third moments. However, we will also see that a Coxian$^+$ PH distribution requires many more phases for approximating distributions with lower second and third moments. The large number of phases needed implies that many free parameters must be determined, implying high computational cost to yield an exact (optimal) solution. By contrast, an n-phase Erlang

	`Simple`	`Complete`	`Positive`
number of moments matched	3	3	3
running time	closed form	closed form	closed form
input distributions	almost all $\mathcal{PH}_3$	all $\mathcal{PH}_3$	almost all $\mathcal{PH}_3$
number of phases	$\leq$OPT(G)+1	$\leq$OPT(G)+1	$\leq$OPT(G)+1
possible mass probability at zero	Yes	Yes	No
output distribution	EC	EC	extended EC
solution	Figure 2.11	Figure 2.13	Figure 2.15

Table 2.1: *A summary of three closed form solutions for moment matching.*

distribution has only two free parameters and is also known to have the least variability among all the n-phase PH distributions [5, 142]. As we will see, however, the Erlang distribution is limited in the set of distributions which it can well-represent.

By combining the Erlang distribution with the two-phase Coxian$^+$ PH distribution, we can represent distributions with all ranges of variability, while using only a small number of phases. Furthermore, the fact that the EC distribution has a small number of parameters, (n, p, λ_Y, λ_{X1}, λ_{X2}, p_X), allows us to obtain closed form expressions for these parameters that well-represent any given distribution in $\mathcal{PH}_\ni$.

Summary of results

We present three variants of closed form solutions for the free parameters of the EC distribution: the `Simple` solution, the `Complete` solution, and the `Positive` solution. Each of the three solutions achieves slightly different goals. The `Simple` solution has the advantage of simplicity and readability. For any input distribution G, the `Complete` solution requires the smallest number of phases among the three solutions. The `Positive` solution achieves an additional property that it does not have mass probability at zero. The `Positive` solution can be used to construct yet another solution, `ZeroMatching`, that matches not only matches the first three moments but also the mass probability at zero of the input distribution, as we will discuss in Section 2.9.

Table 2.1 summarizes the characteristics of the three solutions. In all the three solutions, the first three moments of the input distribution are matched, and the parameters of the output EC distribution are provided in closed form. Since the parameters of the solutions are given in closed form, the running time does not depend on the number of necessary number of phases and is bounded by some constant time.

To characterize the class of input distributions that are defined for our solutions, we introduce set, $\mathcal{PH}_3$.

Definition 3 *$\mathcal{PH}_3$ refers to the set of distributions that are well-represented by a PH distribution.*

The set $\mathcal{PH}_3$ is quite broad, and it contains almost all the nonnegative distributions. More precisely,

Proposition 1 *Set $\mathcal{PH}_3$ is dense in the set of all the nonnegative distributions.*

The `Simple` and `Positive` solutions are defined for almost all the distributions in $\mathcal{PH}_3$, while the `Complete` solution is defined for all the distributions in $\mathcal{PH}_3$. Although the `Simple` and `Positive` solutions are not defined for a very small subset in $\mathcal{PH}_3$, this is not a problem in practice, since a distribution in the small subset can be perturbed to be moved out of the subset[1].

To characterize the optimality of the number of phases used in our solutions, we define OPT(G) as follows:

Definition 4 *$OPT(G)$ is defined to be the minimum number of phases in an acyclic PH distribution, allowing a mass probability at zero, that well-represents a distribution G.*

All of the `Simple`, `Complete`, and `Positive` solutions require at most OPT(G) + 1 phases, but the number of phases in the `Complete` solution is always at most those in the `Simple` and `Positive` solutions. In the definition of OPT($\cdot$), we limit our focus to the set of *acyclic* PH distributions. A PH distribution is called an acyclic PH distribution if the Markov chain whose absorption time defines the PH distribution does not have a cycle (i.e., any state is never visited more than once). It is not clear whether restricting our search space to the set of *acyclic* PH distributions is limiting. While it is theoretically possible that the minimum phase solution is cyclic, in practice we have not been able to find a situation where the minimal solution requires cycles.

The `Simple` and `Complete` solutions can have mass probability at zero (i.e. $p < 1$), but the `Positive` solution has no mass probability at zero. In some applications, mass probability at zero is not an issue. Such applications include approximating busy period distributions in the analysis of multiserver systems via dimensionality reduction (see Chapter 3) and approximating shortfall distributions in inventory system analysis [197, 198]. However, there are also applications where a mass probability at zero increases the computational complexity or even makes the analysis intractable. For example, a PH/PH/1/FCFS queue can be analyzed efficiently via matrix analytic methods (see Section 3.2) when the PH distributions have no mass probability at zero; however, no simple analytical solution is known when the PH distributions have nonzero mass probability at zero.

The key idea in the design of the `Positive` solution is to match the input distribution by a mixture of an EC distribution (with no mass probability at zero) and an exponential distribution. The use of this type of distribution makes intuitive sense, since it can approximate the EC distribution with mass probability at zero arbitrarily closely by letting the rate of the exponential distribution approach infinity. It turns out, however, that it is not always easy (or even possible) to find a *closed form* expression for the parameters of the EC distribution and the exponential distribution. We find that in such cases the *convolution* of an EC distribution and an exponential distribution leads to tractability. We refer to the output distribution of the `Positive` solution as an extended EC distribution. Figure 2.3 shows the Markov chain whose absorption time defines an extended EC distribution.

2.1.2 Characterizing phase type distributions

To prove that our moment matching algorithm results in a nearly minimal number of phases, we need to know the minimal number of phases needed to well-represent an input distribution, G, by an acyclic PH distribution (namely, OPT(G)). Thus, we will provide a formal characterization of the set of distributions, $\mathcal{S}^{(n)}$, that are well-represented by an n-phase acyclic PH distribution for each $n = 1, 2, 3, ...$

[1] Likewise, the `Complete` and `Positive` solutions are not numerically stable for a small subset of input distributions, but this is not a problem in practice, since distributions lying in the small subset can be perturbed to move out of the subset. (In [146, 145], we also provide a version of the `Complete` solution that is numerically stable for all input distributions.)

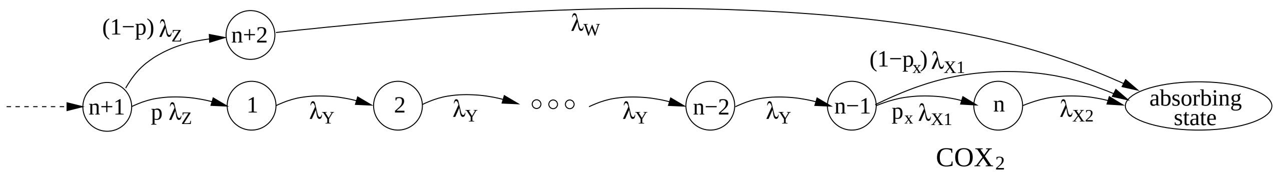


Figure 2.3: *The Markov chain whose absorption time defines an extended EC distribution.*

Definition 5 *Let $\mathcal{S}^{(n)}$ denote the set of distributions that are well-represented by an n-phase acyclic PH distribution for positive integer n.*

Key idea

While our goal is to characterize the set $\mathcal{S}^{(n)}$, this characterization turns out to be ugly. One of the key ideas is that there is a set $\mathcal{T}^{(n)} \supset \mathcal{S}^{(n)}$ which is very close to $\mathcal{S}^{(n)}$ in size, such that $\mathcal{T}^{(n)}$ has a simple specification. To provide a simple specification of $\mathcal{T}^{(n)}$, we define an alternative to the standard moments, which we refer to as *normalized moments*:

Definition 6 *Let μ_k^F be the k-th moment of a distribution F for $k = 1, 2, 3$. The* ***normalized k-th moment*** *m_k^F of F for $k = 2, 3$ is defined to be*

$$m_2^F = \frac{\mu_2^F}{(\mu_1^F)^2} \quad \textit{and} \quad m_3^F = \frac{\mu_3^F}{\mu_1^F \mu_2^F}.$$

Notice the relationship between the normalized moments and the coefficient of variability C_F and the skewness γ_F of F:

$$m_2^F = C_F^2 + 1 \quad \text{and} \quad m_3^F = \nu_F \sqrt{m_2^F},$$

where $\nu_F = \mu_3^F/(\mu_2^F)^{3/2}$. ($\nu_F$ and γ_F are closely related, since $\gamma_F = \bar{\mu}_3^F/(\bar{\mu}_2^F)^{3/2}$, where $\bar{\mu}_k^F$ is the centralized k-th moment of F for $k = 2, 3$.)

Now, $\mathcal{T}^{(n)}$ can be defined via normalized moments (see Figure 2.4).

Definition 7 *For integers $n \geq 2$, let $\mathcal{T}^{(n)}$ denote the set of distributions, F, that satisfy exactly one of the following two conditions:*

$$\textit{(i)}\ m_2^F > \frac{n+1}{n} \quad \textit{and} \quad m_3^F \geq \frac{n+2}{n+1} m_2^F, \ \textit{or}$$
$$\textit{(ii)}\ m_2^F = \frac{n+1}{n} \quad \textit{and} \quad m_3^F = \frac{n+2}{n}.$$

Summary of results

Figure 2.5 illustrates our characterization of general distributions as a nested relationship between $\mathcal{S}^{(n)}$ and $\mathcal{T}^{(n)}$ for all $n \geq 2$. Observe that $\mathcal{S}^{(n)}$ is a proper subset of $\mathcal{S}^{(n+1)}$, and likewise $\mathcal{T}^{(n)}$ is a proper subset of $\mathcal{T}^{(n+1)}$ for all integers $n \geq 2$. Formally, the nested relationship between $\mathcal{S}^{(n)}$ and $\mathcal{T}^{(n)}$ is characterized in the next theorem.

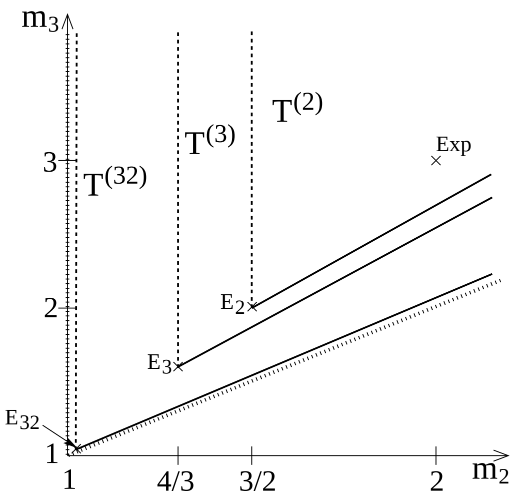


Figure 2.4: *Set $\mathcal{T}^{(n)}$ is depicted as a function of the normalized moments. Sets $\mathcal{T}^{(n)}$ are delineated by solid lines, which include the border, and dashed lines, which do not include the border ($n = 2, 3, 32$). Here, Exp denotes the exponential distribution, and E_n denotes the Erlang-n distribution. Observe that all possible nonnegative distributions lie within the region delineated by the two dotted lines: $m_2 \geq 1$ and $m_3 \geq m_2$ [96]. Also, a distribution G is in $\mathcal{PH}_3$ iff its normalized moments satisfy $m_3^G > m_2^G > 1$ [88].*

Theorem 1 *For all integers $n \geq 2$, $\mathcal{S}^{(n)} \subset \mathcal{T}^{(n)} \subset \mathcal{S}^{(n+1)}$.*

The property $\mathcal{S}^{(n)} \subset \mathcal{T}^{(n)}$ is important because it will allow us to prove that the EC distribution produced by our moment matching algorithm uses a nearly minimal number of phases. The property $\mathcal{T}^{(n)} \subset \mathcal{S}^{(n+1)}$ is important in completing our characterization of $\mathcal{S}^{(n)}$. The latter property will follow immediately from our construction of the `Complete` solution. Note that it is important that the `Complete` solution is defined for *all* $\mathcal{PH}_3$, to prove $\mathcal{T}^{(n)} \subset \mathcal{S}^{(n+1)}$.

In [147], we also provide examples of common, practical distributions in set $\mathcal{S}^{(n)}$. These distributions include the Pareto distribution, the Bounded Pareto distribution (as defined in [64]), and the uniform distribution. We show conditions under which these and other distributions are in $\mathcal{S}^{(n)}$ for each $n \geq 2$.

2.1.3 Organization of this chapter

The rest of this chapter is organized as follows. Section 2.2 provides a brief tutorial on the PH distribution. For the purpose of understanding the rest of the chapter, one only needs to understand the definitions of the PH distribution and its subclasses. In Section 2.3, we review the state of the art in the moment matching algorithm and characterization of $\mathcal{S}^{(n)}$.

Sections 2.4-2.8 are devoted to the characterization of $\mathcal{S}^{(n)}$ and three variants of moment matching algorithms. The characterization of $\mathcal{S}^{(n)}$ is covered primarily in Section 2.4. In Section 2.5, we study properties of the EC distribution in depth. In particular, we will find that for the purpose of moment matching it suffices to narrow down the set of EC distributions further from six free parameters to five

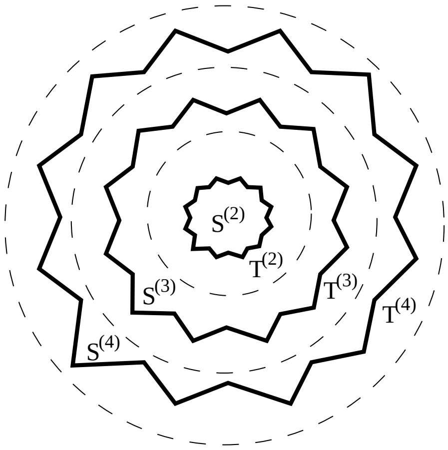


Figure 2.5: *Characterizing $\mathcal{S}^{(n)}$ via $\mathcal{T}^{(n)}$. Solid lines delineate $\mathcal{S}^{(n)}$ (which is irregular) and dashed lines delineate $\mathcal{T}^{(n)}$ (which is regular – has a simple specification). Observe the nested structure of $\mathcal{S}^{(n)}$ and $\mathcal{T}^{(n)}$: $\mathcal{S}^{(n)} \subset \mathcal{T}^{(n)} \subset \mathcal{S}^{(n+1)}$ for all integers $n \geq 2$.*

free parameters, by optimally fixing one of the parameters. Section 2.6 provides the `Simple` solution, Section 2.7 provides the `Complete` solution, and Section 2.8 provides the `Positive` solution.

2.2 Brief tutorial on phase type distributions

In this section, we provide a brief tutorial on the PH distribution, limiting our discussion to the *continuous time* PH distribution. For the purpose of understanding this chapter, one only needs to know the definitions of the PH distribution and its subclasses. We also provide some basic properties of the PH distribution (e.g., its density and distribution functions), which we will need later in Chapter 3. These properties can also be used to further study the characteristics of our solutions (e.g., the density and distribution functions of the EC distribution).

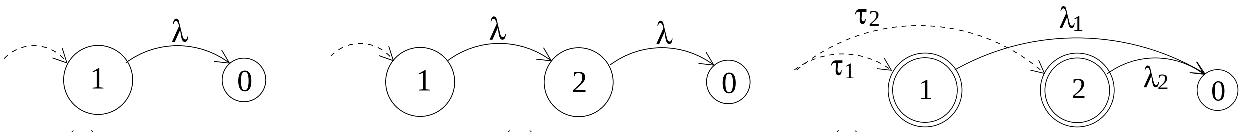
2.2.1 Examples of PH distributions

We start by providing canonical examples of PH distributions. Here, we provide both pictorial explanation and more formal explanation. Pictorial explanation gives intuitive understanding of the PH distribution, and more formal explanation allows us to get used to the notation that we use later.

First, an exponential distribution is a PH distribution. Figure 2.6(a) illustrates an exponential distribution as the absorption time in a (continuous time) Markov chain[2]. At time 0, we start at state 1. We stay in this state for a random time having an exponential distribution with rate λ, and then transition to state 0, the absorbing state. The time until we enter the absorbing state is, of course, an exponential distribution. More formally, an exponential distribution with rate λ is the distribution of the time until absorption into state 0 in a Markov chain on the states $\{0,1\}$ with initial probability vector $(0,1)$ and infinitesimal generator:

$$\begin{pmatrix} 0 & 0 \\ \lambda & -\lambda \end{pmatrix}.$$

[2]Throughout, a Markov chain is assumed to be a *continuous time* Markov chain.



(a) exponential (b) Erlang-2 (c) two-phase hyperexponential

Figure 2.6: *Examples of PH distributions.*

Second, a convolution of two independent identical exponential distributions is a PH distribution (i.e., the sum of two i.i.d. exponential random variables has a PH distribution); this distribution is called an Erlang-2 distribution. Figure 2.6(b) illustrates an Erlang-2 distribution as the absorption time in a Markov chain. At time 0, we start at state 1. After a random time having an exponential distribution with rate λ, we transition to state 2. We stay in state 2 for a random time having an exponential distribution with rate λ, and then transition to state 0, the absorbing state. The time until we enter the absorbing state has an Erlang-2 distribution. More formally, an Erlang-2 distribution with parameter λ is the distribution of the time until absorption into state 0 in a Markov chain on the states $\{0, 1, 2\}$ with initial probability vector $(0, 1, 0)$ and infinitesimal generator:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & -\lambda & \lambda \\ \lambda & 0 & -\lambda \end{pmatrix}.$$

Third, a mixture of two exponential distributions is a PH distribution; this distribution is called a two-phase hyperexponential distribution, H_2. Figure 2.6(c) illustrates an H_2 distribution as the absorption time in a Markov chain. At time 0, we start at state 1 with probability τ_1 and at state 2 with probability τ_2. If we start at state 1 (respectively, state 2), we stay there for a random time having an exponential distribution with rate λ_1 (respectively, λ_2), and then transition to state 0, the absorbing state. The time until we enter the absorbing state has an H_2 distribution. More formally, an H_2 distribution with parameter $(\tau_1,\tau_2,\lambda_1,\lambda_2)$ is the distribution of the time until absorption into state 0 in a Markov chain on the states $\{0, 1, 2\}$ with initial probability vector $(0, \tau_1, \tau_2)$ and infinitesimal generator:

$$\begin{pmatrix} 0 & 0 & 0 \\ \lambda_1 & -\lambda_1 & 0 \\ \lambda_2 & 0 & -\lambda_2 \end{pmatrix}.$$

2.2.2 Definition of PH distribution

In general, a PH distribution is the distribution of the time until absorption into state 0 in a Markov chain.

Definition 8 *A PH distribution with parameter $(\vec{\tau}, \mathbf{T})$, PH($\vec{\tau}, \mathbf{T}$), is the distribution of the time until absorption into state 0 in a Markov chain on the states $\{0, 1, ..., n\}$ with initial probability vector*

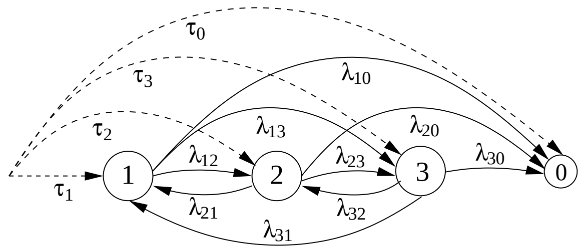


Figure 2.7: *A three-phase PH distribution.*

$(\tau_0, \overrightarrow{\tau})$ *and infinitesimal generator*

$$\begin{pmatrix} 0 & \overrightarrow{0} \\ \overrightarrow{t} & \mathbf{T} \end{pmatrix},$$

where $\overrightarrow{t} = -\mathbf{T}\overrightarrow{1}$ *and* $\tau_0 = 1 - \overrightarrow{\tau}\overrightarrow{1}$, *where* $\overrightarrow{1}$ *is a column vector of 1's.*

Figure 2.7 illustrates a three phase PH distribution having parameters

$$\overrightarrow{\tau} = (\tau_1, \tau_2, \tau_3) \quad \text{and} \quad \mathbf{T} = \begin{pmatrix} -\sigma_1 & \lambda_{12} & \lambda_{13} \\ \lambda_{21} & -\sigma_2 & \lambda_{23} \\ \lambda_{31} & \lambda_{32} & -\sigma_3 \end{pmatrix},$$

where $\sigma_i = \sum_{j \neq i} \lambda_{ij}$ for $i = 1, 2, 3$. Note that the initial state can be the absorption state (state 0) with positive probability. If the initial state is the absorption state, the absorption time is zero.

2.2.3 Subclasses of PH distribution

By restricting the structure of the Markov chain, we can define a subclass of the PH distribution. Below, we define subclasses of the PH distribution that come up in this thesis.

First, if the Markov chain whose absorption time defines a PH distribution is acyclic (i.e., any state is never visited more than once in the Markov chain), the PH distribution is called an **acyclic PH distribution**. A three-phase acyclic PH distribution is illustrated in Figure 2.8(a) as the absorption time in a Markov chain.

An acyclic PH distribution is called a **Coxian PH distribution** if the Markov chain whose absorption time defines the acyclic PH distribution has the following two properties: (i) the initial non-absorbing state is unique (i.e., the initial state is either the unique non-absorbing state or the absorbing state), and (ii) for each state, the next non-absorbing state is unique (i.e., the next state is the unique non-absorbing state or the absorbing state). A three-phase Coxian PH distribution is illustrated in Figure 2.8(b). When a Coxian PH distribution does not have mass probability at zero (i.e., the initial state is not the absorbing state), we refer to the Coxian PH distribution as a **Coxian$^+$ PH distribution**.

An acyclic PH distribution is called a **hyperexponential distribution** if the Markov chain whose absorption time defines the acyclic PH distribution has the following property: for any state, the next state is the absorbing state. That is, a mixture of exponential distributions is a hyperexponential distribution.

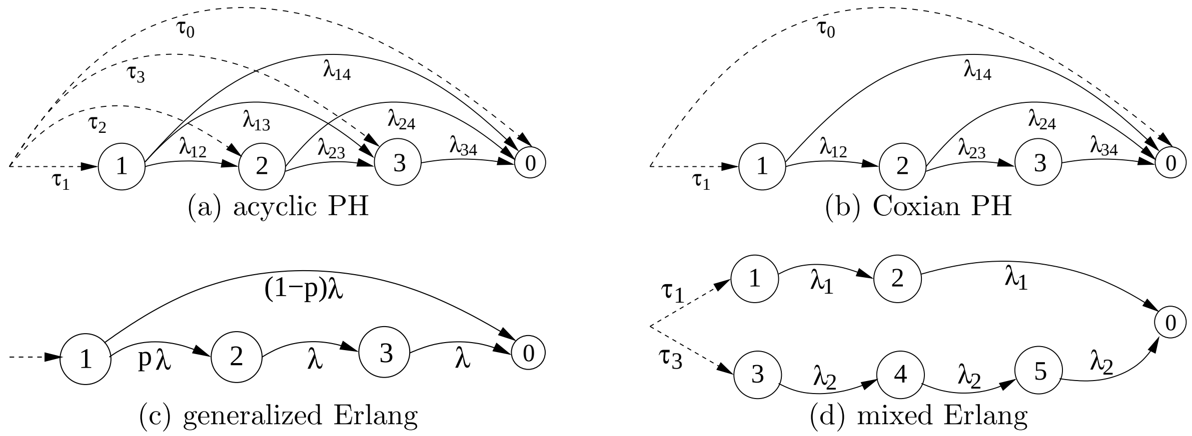


Figure 2.8: *Subclasses of the PH distribution.*

An acyclic PH distribution is called an **Erlang distribution** if the Markov chain whose absorption time defines the acyclic PH distribution has the following three properties: (i) the initial state is a unique non-absorbing state, (ii) for each state, the next state is unique, and (iii) the sojourn time distribution at each state is identical. That is, the sum of n i.i.d. exponential random variables has an n-phase Erlang distribution. An n-phase Erlang distribution is also called an **Erlang-n** distribution. An Erlang distribution is generalized to a **generalized Erlang distribution** [122] by allowing a transition from the initial state to the absorbing state. A three-phase generalized Erlang distribution is illustrated in Figure 2.8(c).

A mixture of Erlang distributions is called a **mixed Erlang distribution** [87, 88]. A mixed Erlang distribution is illustrated in Figure 2.8(d), where an Erlang-2 distribution and an Erlang-3 distribution are mixed. A mixture of Erlang distributions with the same number of phases, is called a **mixed Erlang distribution with common order** [87, 88].

2.2.4 Properties of PH distributions

Below, we summarize some of the basic properties of the PH distribution. First, the set of PH distributions is quite broad and, in theory, any nonnegative distribution can be approximated arbitrarily closely by a PH distribution.

Proposition 2 *[132] The set of PH distributions is dense in the set of nonnegative distributions (distributions with support on $[0, \infty)$).*

Observe that Proposition 1 follows immediately from Proposition 2.

Second, the set of PH distributions is closed under some operations. In particular, a mixture of independent PH distributions is a PH distribution, and the convolution of independent PH distributions is a PH distribution.

Proposition 3 *[111] Consider two PH distributions: PH($\vec{\tau_1}, \mathbf{T_1}$) with distribution function $F_1(\cdot)$ and PH($\vec{\tau_2}, \mathbf{T_2}$) with distribution function $F_2(\cdot)$. A* **mixture** *of the two PH distribution, which has* *distribution function* $pF_1(\cdot) + (1-p)F_2(\cdot)$*, is a PH distribution, PH(*$\overrightarrow{\tau}, \mathbf{T}$*), where*

$$\overrightarrow{\tau} = (p\overrightarrow{\tau_1}, (1-p)\overrightarrow{\tau_2}) \quad \textit{and} \quad \mathbf{T} = \begin{pmatrix} \mathbf{T_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{T_2} \end{pmatrix}.$$

Here, $\mathbf{0}$ *denotes a zero matrix.*

The **convolution** *of the two PH distributions, PH(*$\overrightarrow{\tau_1}, \mathbf{T_1}$*) and PH(*$\overrightarrow{\tau_2}, \mathbf{T_2}$*), is a PH distribution, PH(*$\overrightarrow{\tau}, \mathbf{T}$*), where*

$$\overrightarrow{\tau} = (\overrightarrow{\tau_1}, \tau_{10}\overrightarrow{\tau_2}) \quad \textit{and} \quad \mathbf{T} = \begin{pmatrix} \mathbf{T_1} & \overrightarrow{t_1}\overrightarrow{\tau_2} \\ \mathbf{0} & \mathbf{T_2} \end{pmatrix}.$$

Here, $\overrightarrow{t_1} = -\mathbf{T_1}\overrightarrow{1}$ *and* $\tau_{10} = 1 - \overrightarrow{\tau_1}\overrightarrow{1}$*, where* $\overrightarrow{1}$ *is a column vector of 1's.*

To shed light on the expression $F(\cdot) = pF_1(\cdot) + (1-p)F_2(\cdot)$, consider a random variable V_1 whose distribution function is $F_1(\cdot)$ and a random variable V_2 whose distribution function is $F_2(\cdot)$. Then, random variable

$$V = \begin{cases} V_1 & \text{with probability } p \\ V_2 & \text{with probability } 1-p \end{cases}$$

has distribution function $F(\cdot)$. Below, unless otherwise stated, we denote the (cumulative) distribution function of a distribution, X, by $X(\cdot)$.

Definition 9 *Let* V_X *be a random variable having a distribution* X*. We denote the cumulative distribution function by* $X(\cdot)$*, namely*

$$X(x) \equiv Pr(V_X \leq x).$$

Finally, the distribution function, the density function, the moments, and the Laplace transform of a PH distribution have simple mathematical expressions.

Proposition 4 *[111] The distribution function of PH(*$\overrightarrow{\tau}, \mathbf{T}$*) is given by*

$$F(x) = 1 - \overrightarrow{\tau}\exp(\mathbf{T}x)\overrightarrow{1}$$

for $x \geq 0$*, where the matrix exponential is defined by* $\exp(\mathbf{X}) = \sum_{i=0}^{\infty}\frac{1}{i!}\mathbf{X}^i$*. The density function of PH(*$\overrightarrow{\tau}, \mathbf{T}$*) is given by*

$$f(x) = \overrightarrow{\tau}\exp(\mathbf{T}x)\overrightarrow{t}$$

for $x \geq 0$*, where* $\overrightarrow{t} = -\mathbf{T}\overrightarrow{1}$*.*

Let X *be a random variable with the PH(*$\overrightarrow{\tau}, \mathbf{T}$*) distribution. Then,*

$$\mathsf{E}\left[X^i\right] = i!\overrightarrow{\tau}(-\mathbf{T}^{-1})^i\overrightarrow{1}$$

for $i \geq 1$*. The Laplace transform of PH(*$\overrightarrow{\tau}, \mathbf{T}$*) is given by*

$$\widetilde{X}(s) = \tau_0 + \overrightarrow{\tau}(s\mathbf{I} - \mathbf{T})^{-1}\overrightarrow{t},$$

where $\tau_0 = 1 - \overrightarrow{\tau}\overrightarrow{1}$ *and* $\mathbf{I}$ *is an identity matrix.*

2.3 State of the art in moment matching algorithms

In this section, we review prior work on moment matching algorithms and characterization of the PH distribution.

Moment matching algorithms

Prior work has contributed a large number of moment matching algorithms. While all of these algorithms excel with respect to some of the four measures mentioned earlier (number of moments matched; minimality of the number of phases; generality of the solution; computational efficiency of the algorithm), they all are deficient in at least one of these measures as explained below.

In cases where matching only two moments suffices, it is possible to achieve solutions which perform very well along all the other three measures. Sauer and Chandy [171] provide a closed form solution for matching two moments of a general distribution with squared coefficient of variation $C^2 > 0$ (i.e., any distribution in $\mathcal{PH}_3$). They use a two-phase hyper-exponential distribution, H_2, for matching distributions with squared coefficient of variability $C^2 > 1$, and a generalized Erlang distribution for matching distributions with $C^2 < 1$. Marie [122] provides a closed form solution for matching two moments of a general distribution with $C^2 > 0$. He uses a two-phase Coxian$^+$ PH distribution for distributions with $C^2 > 1$, and a generalized Erlang distribution for distributions with $C^2 < 1$.

If one is willing to match only a subset of distributions, then again it is possible to achieve solutions which perform very well along the remaining three measures. Whitt [200] and Altiok [6] focus on the set of distributions with $C^2 > 1$ and sufficiently high third moment. They obtain a closed form solution for matching three moments of any distribution in this set. Whitt matches to an H_2, and Altiok matches to a two-phase Coxian$^+$ PH distribution. Telek and Heindl [190] focus on the set of distributions with $C^2 \geq \frac{1}{2}$ and various constraints on the third moment. They obtain a closed form solution for matching three moments of any distribution in this set, by using a two-phase acyclic PH distribution with no mass probability at zero.

Johnson and Taaffe [87, 88] come closest to achieving all four measures. They provide a closed form solution for matching the first three moments of any distribution $G \in \mathcal{PH}_3$. They use a mixed Erlang distribution with common order. Unfortunately, this mixed Erlang distribution requires $2\mathrm{OPT}(G) + 2$ phases in the worst case.

In complementary work, Johnson and Taaffe [89, 90] again look at the problem of matching the first three moments of any distribution $G \in \mathcal{PH}_3$, this time using three types of PH distributions: a mixture of Erlang distributions, a Coxian$^+$ PH distribution, and a general PH distribution. Their solution is nearly minimal in that it requires at most $\mathrm{OPT}(G) + 2$ phases. Unfortunately, their algorithm requires solving a nonlinear programming problem and hence is computationally inefficient, requiring time exponential in $\mathrm{OPT}(G)$.

Above we have described the prior work focusing on moment matching algorithms, which is the focus of this chapter. There is also a large body of work focusing on fitting the *shape* of an input distribution using a PH distribution. Recent research has looked at fitting heavy-tailed distributions to PH distributions [55, 76, 77, 98, 162, 187]. There is also work which combines the goals of moment matching with the goal of fitting the shape of the distribution [86, 173]. The work above is clearly broader in its goals than simply matching three moments. Unfortunately there is a tradeoff: obtaining a more precise fit requires more phases, and it can sometimes be computationally inefficient [86, 173].

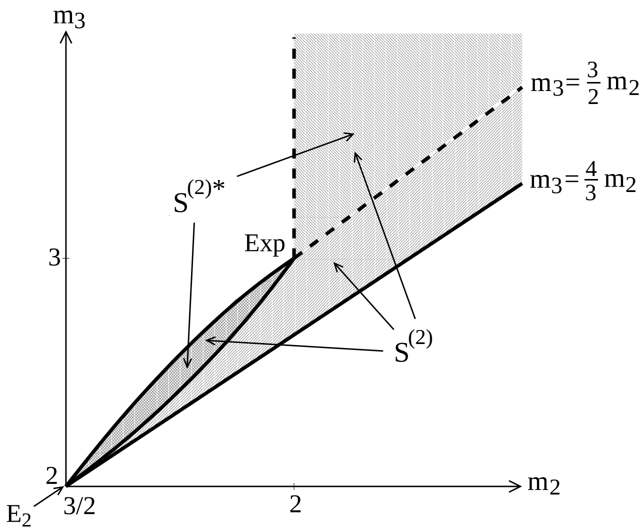


Figure 2.9: *Set $\mathcal{S}^{(2)}$ and set $\mathcal{S}^{(2)^*}$. Here, Exp denotes the exponential distribution, and E_2 denotes the Erlang-2 distribution.*

Characterizing PH distributions

All prior work on characterizing $\mathcal{S}^{(n)}$ has focused on characterizing $\mathcal{S}^{(2)^*}$, where $\mathcal{S}^{(2)^*}$ is the set of distributions that are well-represented by a two-phase Coxian$^+$ PH distribution. Observe $\mathcal{S}^{(2)^*} \subset \mathcal{S}^{(2)}$. Altiok [6] showed a sufficient condition for a distribution to be in $\mathcal{S}^{(2)^*}$. More recently, Telek and Heindl [190] proved the necessary and sufficient condition for a distribution to be in $\mathcal{S}^{(2)^*}$. While neither Altiok nor Telek and Heindl expressed these conditions in terms of normalized moments, the results can be expressed more simply with our normalized moments:

Theorem 2 *[144, 190] $G \in \mathcal{S}^{(2)^*}$ iff G satisfies exactly one of the following three conditions:*

$$\text{(i)}\ \frac{9m_2^G - 12 + 3\sqrt{2}(2 - m_2^G)^{\frac{3}{2}}}{m_2^G} \leq m_3^G \leq \frac{6(m_2^G - 1)}{m_2^G} \quad \textit{and} \quad \frac{3}{2} \leq m_2^G < 2,$$
$$\text{(ii)}\ m_3^G = 3 \quad \textit{and} \quad m_2^G = 2,\ \textit{or}$$
$$\text{(iii)}\ \frac{3}{2} m_2^G < m_3^G \quad \textit{and} \quad 2 < m_2^G.$$

Figure 2.9 shows an *exact* characterization of set $\mathcal{S}^{(2)}$ and set $\mathcal{S}^{(2)^*}$ via normalized moments. We provide an *exact* characterization of set $\mathcal{S}^{(2)}$ in [147]. In this chapter, we will characterize $\mathcal{S}^{(n)}$, for all integers $n \geq 2$.

2.4 Characterizing phase type distributions

Set $\mathcal{S}^{(n)}$ is characterized by Theorem 1: $\mathcal{S}^{(n)} \subset \mathcal{T}^{(n)} \subset \mathcal{S}^{(n+1)}$ *for all* $n \geq 2$. In this section we prove the first part of the theorem:

Lemma 1 *For all integers $n \geq 2$, $\mathcal{S}^{(n)} \subset \mathcal{T}^{(n)}$.*

The second part, $\mathcal{T}^{(n)} \subset \mathcal{S}^{(n+1)}$, follows immediately from the construction of the `Complete` solution (see Corollary 3 in Section 2.7).

We begin by defining the ratio of the normalized moments.

Definition 10 *The ratio of the normalized moments of a distribution F, r^F, is defined as $r^F = m_3^F/m_2^F$ and is also referred to as the r-value of F.*

A nice property of the r-value is that it is insensitive to the mass probability at zero:

Proposition 5 *Let Z be a mixture of two distributions, X and O, where where X is a nonnegative distribution with $\mu_1^X > 0$ and O is the distribution of the degenerate random variable $V \equiv 0$. Then, $r^Z = r^X$.*

Proof: Let $Z(\cdot)$, $X(\cdot)$, and $O(\cdot)$ be the (cumulative) distribution functions of Z, X, and O, respectively. Then, there exists $0 < p < 1$ such that $Z(\cdot) = pX(\cdot) + (1-p)O(\cdot)$. Therefore, by definition,

$$r^Z = \frac{(p\mu_3^X)(p\mu_1^X)}{(p\mu_2^X)^2} = \frac{(\mu_3^X)(\mu_1^X)}{(\mu_2^X)^2} = r^X.$$

■

Below, unless otherwise stated, we denote the (cumulative) distribution function of a distribution, X, by $X(\cdot)$. Below, we use the notation O repeatedly.

Definition 11 *Let O denote the distribution of the degenerate random variable $V \equiv 0$.*

Note that, using the normalized second moment and the r-value, $\mathcal{T}^{(n)}$ can be redefined as the set of distributions, F, that satisfy exactly one of the following two conditions:

$$\begin{array}{ll} \text{(i) } m_2^F > \dfrac{n+1}{n} & \text{and} \quad r^F \geq \dfrac{n+2}{n+1}, \text{ or} \\ \text{(ii) } m_2^F = \dfrac{n+1}{n} & \text{and} \quad r^F = \dfrac{n+2}{n+1}. \end{array}$$

To show $\mathcal{S}^{(n)} \subset \mathcal{T}^{(n)}$, consider an arbitrary distribution, $X \in \mathcal{S}^{(n)}$. By definition of $\mathcal{S}^{(n)}$, there exists an n-phase acyclic PH distribution, P, that well-represents X. Then $X \in \mathcal{T}^{(n)}$ iff $P \in \mathcal{T}^{(n)}$. Hence, it suffices to prove that all n-phase acyclic PH distributions are in $\mathcal{T}^{(n)}$. This can be shown by proving the two properties of the Erlang-n distribution: (i) the set of Erlang-n distributions has the least normalized second moment among all the n-phase (acyclic) PH distributions and (ii) the Erlang-n distribution has the least r-value among all the n-phase acyclic PH distributions. Note that the Erlang-n distribution, E_n, has

$$m_2^{E_n} = \frac{n+1}{n} \quad \text{and} \quad r^{E_n} = \frac{n+2}{n+1}.$$

Property (i) of the Erlang-n distribution immediately follows from the prior work by Aldous and Shepp [5] and O'Cinneide [142], who prove that the set of Erlang-n distributions is the unique class of n-phase PH distributions with the least second moment among all the n-phase PH distributions with a fixed first moment. Thus, all that remains is to prove property (ii).

Our approach is different from Aldous and Shepp [5] and O'Cinneide [142]. Aldous and Shepp prove the least variability of the Erlang-n distribution via quadratic variation (a property related to the second moment), and hence it is unlikely that their approach can be applied to prove property (ii), which relies on higher moments, in particular the r-value. O'Cinneide extends the work by Aldous and Shepp, considering a convex function, $f(\cdot)$, applied to a random variable having an n-phase PH distribution with a fixed mean. He proves, via the theory of majorization, that the expectation of $f(V)$ is minimized when the random variable V has an Erlang distribution. Unfortunately, the r-value of a distribution, G, is not an expectation of $f(V)$, where V is a random variable with distribution G, for a convex function $f(\cdot)$. Hence, the theory of majorization does not directly apply to the r-value.

Our proof makes use of the recursive structure of PH distributions and shows that an n-phase Erlang distribution has no greater r-value than any n-phase acyclic PH distribution. The key idea in our proof is that any acyclic PH distribution, P, can be seen as a mixture of the convolutions of exponential distributions, and one of the convolutions of exponential distributions has no greater r-value than P. This allows us to relate the minimal convolution to an Erlang distribution when all the rates of the exponential distributions are the same. The following lemma provides the key property of the r-value used in our proof.

Lemma 2 *Let $Z(\cdot) = \sum_{i=1}^{n} p_i X_i(\cdot)$, where $n \geq 2$ and X_i are nonnegative distributions with $\mu_1^{X_i} > 0$ for $1 \leq i \leq n$. Then, there exists $i \in \{1, ..., n\}$ such that $r^Z \geq r^{X_i}$.*

Proof: We prove the lemma by induction on n. Without loss of generality, we let $r^{X_1} \geq \cdots \geq r^{X_n}$.

Base case ($n = 2$): Let $v = \mu_1^{X_2}/\mu_1^{X_1}$ and $w = \mu_2^{X_2}/\mu_2^{X_1}$. Then,

$$
\begin{aligned}
r^Z - r^{X_2} &= \frac{p_1^2 r^{X_1} + p_1 p_2 r^{X_1} v + p_1 p_2 r^{X_2} \frac{w^2}{v} + p_2^2 r^{X_2} w^2}{(p_1 + p_2 w)^2} - r^{X_2} \\
&\geq \frac{\left(p_1^2 + p_1 p_2 v + p_1 p_2 \frac{w^2}{v} + p_2^2 w^2 - (p_1 + p_2 w)^2\right) r^{X_2}}{(p_1 + p_2 w)^2} \\
&= \frac{p_1 p_2 (w - v)^2 r^{X_2}}{v (p_1 + p_2 w)^2} \geq 0,
\end{aligned}
$$

where the first inequality follows from $r^{X_1} \geq r^{X_2}$.

Inductive case: Suppose that the lemma holds for $n \leq k$. When $n = k + 1$, Z can be seen as a mixture of two distributions, $Y(\cdot) = \frac{1}{1 - p_{k+1}} \sum_{i=1}^{k} p_i X_i(\cdot)$ and $X_{k+1}(\cdot)$. When $r^{X_{k+1}} \leq r^Z$, the lemma holds for $n = k + 1$. When $r^{X_{k+1}} > r^Z$, we have $r^Y \leq r^Z$ by the base case. By the inductive hypothesis, there exist $i \in \{1, ..., k\}$ such that $r^Y \geq r^{X_i}$. Thus, the lemma holds for $n = k + 1$. ■

We are now ready to prove that an n-phase Erlang distribution has no greater r-value than any n-phase PH distribution, which completes the proof of Lemma 1. A proof of the following lemma is postponed to Appendix A.

Lemma 3 *The Erlang distribution has the least r-value among all the acyclic PH distribution with a fixed number of phases, n, for all $n \geq 1$.*

2.5 EC distribution

The purpose of this section is twofold: to provide a detailed characterization of the EC distribution, and to discuss a narrowed-down subset of the EC distributions with only five free parameters (λ_Y is fixed) which we will use in our moment matching algorithms. Both results are summarized in Theorem 3.

To motivate the theorem in this section, suppose one is trying to match the first three moments of a given distribution, G, to a distribution, P, which is the convolution of exponential distributions (possibly with different rates) and a two-phase Coxian$^+$ PH distribution. If G has sufficiently high second and third moments, then a two-phase Coxian$^+$ PH distribution alone suffices and we need no exponential distributions (recall Theorem 2). If the variability of G is lower, however, we might try appending an exponential distribution to the two-phase Coxian$^+$ PH distribution. If that does not suffice, we might append two exponential distributions to the two-phase Coxian$^+$ PH distribution. Thus, if G has very low variability, we might be forced to use many phases to get the variability of P to be low enough. Therefore, to minimize the number of phases in P, it seems desirable to choose the rates of the exponential distributions so that the overall variability of P is minimized. One could express the appending of each exponential distribution as a "function" whose goal is to reduce the variability of P yet further.

Definition 12 *Let X be an arbitrary distribution.* **Function** *ϕ maps X to $\phi(X)$ such that $\phi(X) = Y * X$, the convolution of Y and X, where Y is an exponential distribution whose rate, λ_Y, is chosen so that the normalized second moment of $\phi(X)$ is minimized. Also, $\phi^i(X) = \phi(\phi^{i-1}(X))$ refers to the distribution obtained by applying function ϕ to $\phi^{i-1}(X)$ for integers $i \geq 1$, where $\phi^0(X) = X$.*

Observe that, when X is a k-phase PH distribution, $\phi^N(X)$ is a $(k+N)$-phase PH distribution.

In theory, function ϕ allows each successive exponential distribution which is appended to have a different rate. Surprisingly, however, the following theorem shows that if the exponential distribution Y being appended by function ϕ is chosen so as to minimize the normalized second moment of $\phi(X)$ (as specified by the definition), then the rate of each successive Y is always *the same* and is defined by the simple formula shown in the theorem below. The theorem also characterizes the normalized moments of $\phi^i(X)$.

Theorem 3 *Let $\phi^i(X) = Y_i * \phi^{i-1}(X)$, and let λ_{Y_i} be the rate of the exponential distribution Y_i for $i = 1, ..., N$. Then,*

$$\lambda_{Y_i} = \frac{1}{(m_2^X - 1)\mu_1^X} \tag{2.1}$$

for $i = 1, ..., N$. The normalized moments of $Z_N = \phi^N(X)$ are:

$$\begin{aligned} m_2^{Z_N} &= \frac{(m_2^X-1)(N+1)+1}{(m_2^X-1)N+1} \\ m_3^{Z_N} &= \frac{m_2^X m_3^X}{((m_2^X-1)(N+1)+1)\left((m_2^X-1)N+1\right)^2} \\ &\quad + \frac{(m_2^X-1)N\left(3m_2^X+(m_2^X-1)(m_2^X+2)(N+1)+(m_2^X-1)^2(N+1)^2\right)}{((m_2^X-1)(N+1)+1)\left((m_2^X-1)N+1\right)^2}. \end{aligned}$$

Proof: We first characterize $Z = \phi(X) = Y * X$, where X is an arbitrary distribution with a finite third moment and Y is an exponential distribution. The normalized second moment of Z is

$$m_2^Z = \frac{m_2^X + 2y + 2y^2}{(1+y)^2},$$

where $y = \mu_1^Y / \mu_1^X$. Observe that m_2^Z is minimized when $y = m_2^X - 1$, namely when

$$\mu_1^Y = (m_2^X - 1)\mu_1^X. \tag{2.2}$$

Observe that when μ_1^Y is set at this value, the normalized moments of Z satisfy:

$$m_2^Z = 2 - \frac{1}{m_2^X} \quad \text{and} \quad m_3^Z = \frac{1}{m_2^X(2m_2^X - 1)} m_3^X + \frac{3(m_2^X - 1)}{m_2^X}.$$

We next characterize $Z_i = \phi^i(X) = Y_i * \phi^{i-1}(X)$ for $2 \le i \le N$. By the above expression on m_2^Z and m_3^Z, the second part of the theorem on the normalized moments of Z_N follow from solving the following recurrence equations (where we use b_i to denote $m_2^{\phi^i(X)}$ and B_i to denote $m_3^{\phi^i(X)}$):

$$b_{i+1} = 2 - \frac{1}{b_i} \quad \text{and} \quad B_{i+1} = \frac{B_i}{b_i(2b_i - 1)} + \frac{3(b_i - 1)}{b_i}.$$

The solutions for these recurrence equations are

$$\begin{aligned} b_{i+1} &= \frac{(b_1 - 1)(i+1) + 1}{(b_1 - 1)i + 1} \\ B_{i+1} &= \frac{b_1 B_1 + (b_1 - 1)i\left(3b_1 + (b_1 - 1)(b_1 + 2)(i+1) + (b_1 - 1)^2(i+1)^2\right)}{\left((b_1 - 1)(i+1) + 1\right)\left((b_1 - 1)i + 1\right)^2} \end{aligned}$$

for all $i \ge 0$. These solutions can be verified by substitution. This completes the proof of the second part of the theorem.

The first part of the theorem on λ_{Y_i} is proved by induction. When $i = 1$, (2.1) follows from (2.2). Assume that (2.1) holds for $i = 1, ..., k$. Let $Z_k = \phi^k(X)$. By the second part of the theorem, which is proved above,

$$m_2^{Z_k} = \frac{(m_2^X - 1)(k+1) + 1}{(m_2^X - 1)k + 1}.$$

Thus, by (2.2),

$$\frac{1}{\lambda_{Y_{k+1}}} = \mu_1^{Y_{k+1}} = (m_2^{Z_k} - 1)\mu_1^{Z_k} = (m_2^X - 1)\mu_1^X.$$

∎

Corollary 1 *If X is in set $\left\{F \mid 2 < m_2^F\right\}$, then $Z = \phi^N(X)$ is in set $\left\{F \mid \frac{N+2}{N+1} < m_2^F < \frac{N+1}{N}\right\}$.*

Proof: By Theorem 3, m_2^Z is a continuous and monotonically increasing function of m_2^X. Thus, the infimum and the supremum of m_2^Z are given by evaluating m_2^Z at the infimum and the supremum, respectively, of m_2^X. When $m_2^X \to 2$, $m_2^Z \to \frac{N+2}{N+1}$. When $m_2^X \to \infty$, $m_2^Z \to \frac{N+1}{N}$. ∎

Corollary 1 suggests the number, N, of times that function ϕ must be applied to X to bring m_2^Z into a desired range, given the value of m_2^X.

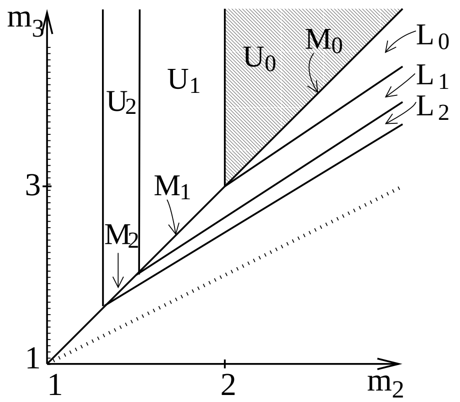


Figure 2.10: *A classification of distributions. The dotted lines delineate the set of all nonnegative distributions G ($m_3^G \geq m_2^G \geq 1$).*

Concluding remarks

Theorem 3 implies that the parameter λ_Y of the EC distribution can be fixed without excluding the distributions of lowest variability from the set of EC distributions. In the rest of the chapter, we constrain λ_Y as follows:

$$\lambda_Y = \frac{1}{(m_2^X - 1)\mu_1^X}, \tag{2.3}$$

and derive closed form representations of the remaining free parameters (n, p, λ_{X1}, λ_{X2}, p_X), where these free parameters will determine m_2^X and μ_1^X, which in turn gives λ_Y by (2.3). Obviously, at least three degrees of freedom are necessary to match three moments. As we will see, the additional degrees of freedom allow us to accept all input distributions in $\mathcal{PH}_3$ and to use a smaller number of phases.

2.6 Simple closed form solution

The `Simple` solution is the simplest among our three closed form solutions, and the `Complete` and `Positive` solutions will be built upon the `Simple` solution. Before, presenting the `Simple` solution, we first classify the input distributions. This classification is used, in particular, to determine the number of phases used in the `Simple` solution.

Preliminaries

Set $\mathcal{T}^{(n)}$, which is used to characterize set $\mathcal{S}^{(n)}$, gives us a sense of how many phases are necessary to well-represent a given distribution. It turns out that it is useful to divide set $\mathcal{T}^{(n)}$ into smaller subsets to describe the closed form solutions compactly. Roughly speaking, we divide the set $\mathcal{T}^{(n)} \setminus \mathcal{T}^{(n-1)}$ into three subsets, $\mathcal{U}_{n-1}$, $\mathcal{M}_{n-1}$, and $\mathcal{L}_{n-1}$ (see Figure 2.10). More formally,

Definition 13 *We define $\mathcal{U}_i$, $\mathcal{M}_i$, and $\mathcal{L}_i$ as follows:*

$$\mathcal{U}_0 = \left\{ F \middle| m_2^F > 2 \text{ and } m_3^F > 2m_2^F - 1 \right\},$$

$$\begin{aligned}
\mathcal{M}_0 &= \left\{F \middle| m_2^F > 2 \textit{ and } m_3^F = 2m_2^F - 1\right\}, \\
\mathcal{L}_0 &= \left\{F \middle| \frac{3}{2}m_2^F < m_3^F < 2m_2^F - 1\right\},
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{U}_i &= \left\{F \middle| \frac{i+2}{i+1} < m_2^F < \frac{i+1}{i} \textit{ and } m_3^F > 2m_2^F - 1\right\}, \\
\mathcal{M}_i &= \left\{F \middle| \frac{i+2}{i+1} < m_2^F < \frac{i+1}{i} \textit{ and } m_3^F = 2m_2^F - 1\right\}, \\
\mathcal{L}_i &= \left\{F \middle| \frac{i+3}{i+2}m_2^F < m_3^F < \frac{i+2}{i+1}m_2^F \textit{ and } m_3^F < 2m_2^F - 1\right\},
\end{aligned}$$

for nonnegative integers i. Also, let $\mathcal{U}^+ = \cup_{i=1}^{\infty}\mathcal{U}_i$, $\mathcal{M}^+ = \cup_{i=1}^{\infty}\mathcal{M}_i$, $\mathcal{L}^+ = \cup_{i=1}^{\infty}\mathcal{L}_i$, $\mathcal{U} = \mathcal{U}_0 \cup \mathcal{U}^+$, $\mathcal{M} = \mathcal{M}_0 \cup \mathcal{M}^+$, and $\mathcal{L} = \mathcal{L}_0 \cup \mathcal{L}^+$. Further, let

$$\begin{aligned}
\widehat{\mathcal{U}} &= \left\{F \middle| m_3^F > 2m_2^F - 1\right\} \supset \mathcal{U}, \\
\widehat{\mathcal{M}} &= \left\{F \middle| m_3^F = 2m_2^F - 1\right\} \supset \mathcal{M}, \\
\widehat{\mathcal{L}} &= \left\{F \middle| m_3^F < 2m_2^F - 1\right\} \supset \mathcal{L}.
\end{aligned}$$

Observe that $\widehat{\mathcal{U}}$ includes both $\mathcal{U}$ and borders between $\mathcal{U}_i$ and $\mathcal{U}_{i+1}$, for $i \geq 0$, that are not included in $\mathcal{U}$.

The sets $\widehat{\mathcal{U}}$, $\widehat{\mathcal{M}}$, and $\widehat{\mathcal{L}}$ provide a classification of distributions into three categories such that, for any distribution X, X and $\phi(X)$ lie in the same category.

Lemma 4 *Let $Z_N = \phi^N(X)$ for integers $N \geq 1$. If $X \in \widehat{\mathcal{U}}$ (respectively, $X \in \widehat{\mathcal{M}}$, $X \in \widehat{\mathcal{L}}$), then $Z_N \in \widehat{\mathcal{U}}$ (respectively, $Z_N \in \widehat{\mathcal{M}}$, $Z_N \in \widehat{\mathcal{L}}$).*

Proof: We prove the case when $N = 1$. The lemma then follows by induction. Let $Z = \phi(X)$. By Theorem 3, $m_2^X = 1/(2 - m_2^Z)$, and

$$\begin{aligned}
m_3^Z &> \text{ (respectively, } =, \text{ and } <) \quad \frac{2m_2^X - 1}{m_2^X(2m_2^X - 1)} + 3\frac{m_2^X - 1}{m_2^X} \\
&> \text{ (respectively, } =, \text{ and } <) \quad 2m_2^Z - 1,
\end{aligned}$$

where the last equality follows from $m_2^X = 1/(2 - m_2^Z)$. ∎

By Corollary 1 and Lemma 4, it follows that:

Corollary 2 *Let $Z_N = \phi^N(X)$ for $N \geq 0$. If $X \in \mathcal{U}_0$ (respectively, $X \in \mathcal{M}_0$), then $Z_N \in \mathcal{U}_N$ (respectively, $Z_N \in \mathcal{M}_N$).*

The corollary implies that for any $G \in \mathcal{U}_N \cup \mathcal{M}_N$, G can be well-represented by an $(N + 2)$-phase EC distribution with no mass probability at zero ($p = 1$), because, for any $X \in \mathcal{U}_0 \cup \mathcal{M}_0$, X can

$(n,\, p,\, \lambda_Y,\, \lambda_{X1},\, \lambda_{X2},\, p_X) = \texttt{Simple}(\mu_1^G,\, \mu_2^G,\, \mu_3^G)$

Input: the first three moments of a distribution G: μ_1^G, μ_2^G, and μ_3^G.

Output: parameters of the EC distribution, $(n,\, p,\, \lambda_Y,\, \lambda_{X1},\, \lambda_{X2},\, p_X)$

1. $m_2^G = \frac{\mu_2^G}{(\mu_1^G)^2}$; $\quad m_3^G = \frac{\mu_3^G}{\mu_1^G \mu_2^G}$.
2. $p = \begin{cases} \frac{1}{2m_2^G - m_3^G} & \text{if } m_3^G < 2m_2^G - 1, \\ 1 & \text{otherwise.} \end{cases}$
3. $\mu_1^W = \frac{\mu_1^G}{p}$; $\quad m_2^W = p m_2^G$; $\quad m_3^W = p m_3^G$.
4. $n = \left\lfloor \frac{m_2^W}{m_2^W - 1} + 1 \right\rfloor$.
5. $m_2^X = \frac{(n-3)m_2^W - (n-2)}{(n-2)m_2^W - (n-1)}$; $\quad \mu_1^X = \frac{\mu_1^W}{(n-2)m_2^X - (n-3)}$.
6. $\alpha = (n-2)(m_2^X - 1)\left(n(n-1)(m_2^X)^2 - n(2n-5)m_2^X + (n-1)(n-3)\right)$.
7. $\beta = \left((n-1)m_2^X - (n-2)\right)\left((n-2)m_2^X - (n-3)\right)^2$.
8. $m_3^X = \frac{\beta m_3^W - \alpha}{m_2^X}$.
9. $u = \frac{6 - 2m_3^X}{3m_2^X - 2m_3^X}$; $\quad v = \frac{12 - 6m_2^X}{m_2^X(3m_2^X - 2m_3^X)}$.
10. $\lambda_{X1} = \frac{u + \sqrt{u^2 - 4v}}{2\mu_1^X}$; $\lambda_{X2} = \frac{u - \sqrt{u^2 - 4v}}{2\mu_1^X}$; $p_X = \frac{\lambda_{X2}(\lambda_{X1}\mu_1^X - 1)}{\lambda_{X1}}$; $\lambda_Y = \frac{1}{(m_2^X - 1)\mu_1^X}$.

Figure 2.11: *An implementation of the* `Simple` *solution, defined for* $G \in \mathcal{PH}_3^- \equiv \mathcal{U} \cup \mathcal{M} \cup \mathcal{L}$.

be well-represented by a two-phase Coxian$^+$ PH distribution, and hence $Z_N = \phi^N(X)$ can be well-represented by an $(N+2)$-phase EC distribution. (Recall $\mathcal{U_N}, \mathcal{M_N}, \mathcal{L_N} \subset \mathcal{T}_{\mathcal{N}+\infty}$.) Below, it will also be shown that for any $G \in \mathcal{L}_N$, G can be well-represented by an $(N+2)$-phase EC distribution with positive mass probability at zero ($p < 1$).

By Corollary 2, it is relatively easy to provide a closed form solution for the parameters $(n,\, p,\, \lambda_{X1},\, \lambda_{X2},\, p_X)$ of an EC distribution, Z, so that a given distribution is well-represented by Z. Essentially, one just needs to find an appropriate N and solve $Z = \phi^N(X)$ for X in terms of normalized moments, which is immediate since N is given by Corollary 1 and the normalized moments of X can be obtained from Theorem 3.

The `Simple` solution

We are now ready to present the `Simple` solution. The `Simple` solution assumes that $G \in \mathcal{PH}_3^-$, where $\mathcal{PH}_3^- = \mathcal{U} \cup \mathcal{M} \cup \mathcal{L}$. Observe $\mathcal{PH}_3^-$ includes almost all distributions in $\mathcal{PH}_3$. Only the borders between the $\mathcal{U}_i$'s, $\mathcal{M}_i$'s, and $\mathcal{L}_i$'s are not included. Figure 2.11 shows an implementation of the `Simple` solution. The solution differs according to the classification of the input distribution G. When $G \in \mathcal{U}_0 \cup \mathcal{M}_0$, a two-phase Coxian$^+$ PH distribution suffices to match the first three moments. When $G \in \mathcal{U}^+ \cup \mathcal{M}^+$, G is well-represented by an EC distribution with $p = 1$. When $G \in \mathcal{L}$, G is well-represented by an EC distribution with $p < 1$.

(i) If $G \in \mathcal{U}_0 \cup \mathcal{M}_0$ (see Figure 2.12(i)), then a two-phase Coxian$^+$ PH distribution suffices to match the first three moments, i.e., $p = 1$ and $n = 2$ ($N = 0$). The parameters $(\lambda_{X1},\, \lambda_{X2},\, p_X)$ of

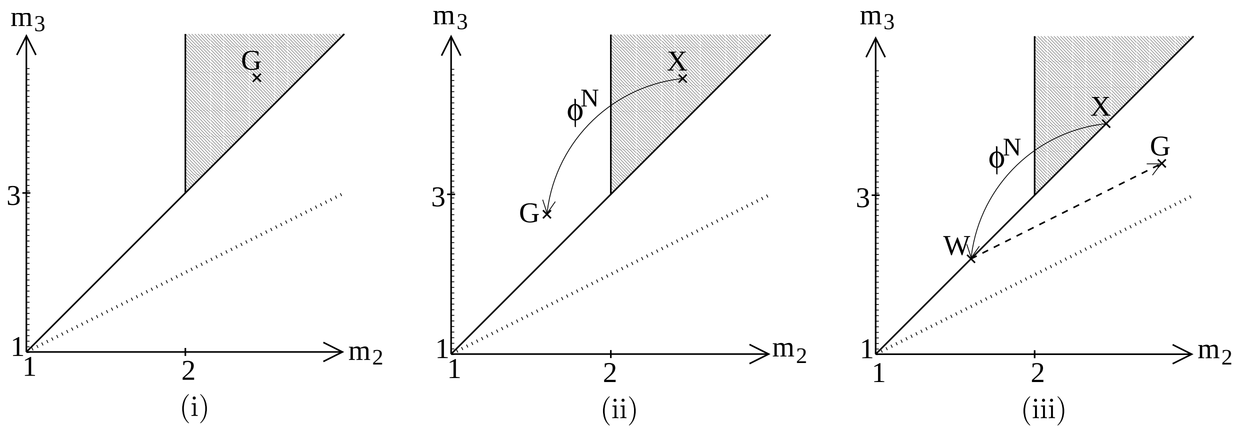


Figure 2.12: *Ideas in the* `Simple` *solution. Let G be the input distribution. (i) If $G \in \mathcal{U}_0 \cup \mathcal{M}_0$, G is well-represented by a two-phase Coxian$^+$ PH distribution X. (ii) If $G \in \mathcal{U}^+ \cup \mathcal{M}^+$, G is well-represented by $\phi^N(X)$, where X is a two-phase Coxian$^+$ PH distribution. (iii) If $G \in \mathcal{L}$, G is well-represented by Z, where Z has a distribution function $p\phi^N(X)(\cdot) + (1-p)O(\cdot)$.*

the two-phase Coxian$^+$ PH distribution are chosen as follows [144, 190]:

$$\lambda_{X1} = \frac{u + \sqrt{u^2 - 4v}}{2\mu_1^G}, \quad \lambda_{X2} = \frac{u - \sqrt{u^2 - 4v}}{2\mu_1^G}, \quad \text{and} \quad p_X = \frac{\lambda_{X2}(\lambda_{X1}\mu_1^G - 1)}{\lambda_{X1}}, \tag{2.4}$$

where

$$u = \frac{6 - 2m_3^G}{3m_2^G - 2m_3^G} \quad \text{and} \quad v = \frac{12 - 6m_2^G}{m_2^G(3m_2^G - 2m_3^G)}. \tag{2.5}$$

(ii) <u>If $G \in \mathcal{U}^+ \cup \mathcal{M}^+$</u> (see Figure 2.12(ii)), Corollary 1 specifies the number of phases needed:

$$n = \min\left\{k \middle| m_2^G > \frac{k}{k-1}\right\} = \left\lfloor \frac{m_2^G}{m_2^G - 1} + 1 \right\rfloor. \tag{2.6}$$

Let $N = n - 2$. Next, we find the two-phase Coxian$^+$ PH distribution $X \in \mathcal{U}_0 \cup \mathcal{M}_0$ such that G is well-represented by $Z = \phi^N(X)$. By Theorem 3, this can be achieved by setting

$$m_2^X = \frac{(n-3)m_2^G - (n-2)}{(n-2)m_2^G - (n-1)}, \quad m_3^X = \frac{\beta m_3^G - \alpha}{m_2^X}, \quad \text{and} \quad \mu_1^X = \frac{\mu_1^G}{(n-2)m_2^X - (n-3)},$$

where

$$\begin{aligned}
\alpha &= (n-2)(m_2^X - 1)\Big(n(n-1)(m_2^X)^2 - n(2n-5)m_2^X + (n-1)(n-3)\Big) \\
\beta &= \Big((n-1)m_2^X - (n-2)\Big)\Big((n-2)m_2^X - (n-3)\Big)^2.
\end{aligned}$$

Thus, we set $p = 1$, and the parameters $(\lambda_{X1}, \lambda_{X2}, p_X)$ of X are given by case (i), using m_2^X, m_3^X, and μ_1^X, specified above.

(iii) <u>If $G \in \mathcal{L}$</u> (see Figure 2.12(iii)), then let

$$p = \frac{1}{2m_2^G - m_3^G}, \quad m_2^W = pm_2^G, \quad m_3^W = pm_3^G, \quad \text{and} \quad \mu_1^W = \frac{\mu_1^G}{p}.$$

Observe that p satisfies $0 \leq p < 1$. Also, since W is in $\mathcal{M}$, W can be chosen as an EC distribution with no mass probability at zero. If $W \in \mathcal{M}_0$, the parameters of W are provided by case (i), using m_2^W, m_3^W, and μ_1^W, specified above. If $W \in \mathcal{M}^+$, the parameters of W are provided by case (ii), using m_2^W, m_3^W, and μ_1^W, specified above. G is then well-represented by distribution Z, where $Z(\cdot) = pW(\cdot) + (1-p)O(\cdot)$.

Analyzing the number of phases required

The number of phases used in the `Simple` solution is characterized by the following theorem.

Theorem 4 *The* `Simple` *solution uses at most* $\mathrm{OPT}(G) + 1$ *phases to well-represent a distribution* $G \in \mathcal{PH}_3^-$.

Proof: Since $S^{(n)} \subset \mathcal{T}^{(n)}$ (by Lemma 1), it suffices to prove that if a distribution $G \in \mathcal{T}^{(n)}$, then at most $n+1$ phases are needed. If $G \in \mathcal{T}^{(n)} \cap (\mathcal{U} \cup \mathcal{M})$, then $m_2^G > \frac{n+1}{n}$. Also, if $G \in \mathcal{T}^{(n)} \cap \mathcal{L}$, then

$$m_2^W = \frac{m_2^G}{2m_2^G - 1} > \frac{n+1}{n}.$$

Thus, by (2.6), the EC distribution provided by the `Simple` solution has at most $n+1$ phases. ■

2.7 Complete closed form solution

The `Complete` solution improves upon the `Simple` solution in the sense that it is defined for all the input distributions $G \in \mathcal{PH}_3$. Figure 2.13 shows an implementation of the `Complete` solution. Below, we elaborate on the `Complete` solution, and prove an upper bound on the number of phases used in the `Complete` solution.

The `Complete` solution

Consider an arbitrary distribution $G \in \mathcal{PH}_3$. Our approach consists of two steps, the first of which involves constructing a baseline EC distribution, and the second of which involves reducing the number of phases in this baseline solution. If $G \in \mathcal{PH}_3^-$, then the baseline solution used is simply given by the `Simple` solution. Also, if $G \notin \mathcal{PH}_3^-$ but $G \in \widehat{\mathcal{L}} \cup \widehat{\mathcal{M}}$, then it turns out that the `Simple` solution could be defined for this G, and this gives the baseline solution. If $G \notin \mathcal{PH}_3^-$ but $G \in \widehat{\mathcal{U}}$, then to obtain the baseline EC distribution we first find a distribution $W \in \mathcal{PH}_3^-$ such that $r^W = r^G$ and $m_2^W < m_2^G$ and then set p such that G is well-represented by distribution Z, where $Z(\cdot) = pW(\cdot) + (1-p)O(\cdot)$ (see Figure 2.14(a)). The parameters of the EC distribution that well-represents W are then obtained by the `Simple` solution.

To reduce the number of phases used in the baseline EC distribution, we exploit the subset of two-phase Coxian$^+$ PH distributions that are not used in the `Simple` solution. The `Simple` solution is based on the fact that a distribution X is well-represented by a two-phase Coxian$^+$ PH distribution when $X \in \mathcal{U}_0 \cup \mathcal{M}_0$. In fact, a wider range of distributions are well-represented by the set of two-phase Coxian$^+$ PH distributions. In particular, if X is in set $\mathcal{S} = \left\{\mathcal{F} \middle| \frac{\ni}{\in} \leq \Updownarrow_{\in}^{\mathcal{X}} \leq \in \text{ and } \Updownarrow_{\ni}^{\mathcal{X}} = \in\Updownarrow_{\in}^{\mathcal{X}} - \infty \right\}$,

$(n,\ p,\ \lambda_Y,\ \lambda_{X1},\ \lambda_{X2},\ p_X) = \texttt{Complete}(\mu_1^G,\ \mu_2^G,\ \mu_3^G)$

Input: the first three moments of a distribution G: μ_1^G, μ_2^G, and μ_3^G.

Output: parameters of the EC distribution, $(n,\ p,\ \lambda_Y,\ \lambda_{X1},\ \lambda_{X2},\ p_X)$

1. $m_2^G = \frac{\mu_2^G}{(\mu_1^G)^2}; \quad m_3^G = \frac{\mu_3^G}{\mu_1^G \mu_2^G}.$

2. $p = \begin{cases} \frac{(m_2^G)^2 + 2m_2^G - 1}{2(m_2^G)^2} & \text{if } m_3^G > 2m_2^G - 1, \text{ and } \frac{1}{m_2^G - 1} \text{ is an integer,} \\ \frac{1}{2m_2^G - m_3^G} & \text{if } m_3^G < 2m_2^G - 1, \\ 1 & \text{otherwise.} \end{cases}$

3. $\mu_1^W = \frac{\mu_1^G}{p}; \quad m_2^W = p m_2^G; \quad m_3^W = p m_3^G.$

4. $n = \begin{cases} \left\lceil \frac{m_2^W}{m_2^W - 1} \right\rceil - 1 & \text{if } m_3^W = 2m_2^W - 1, \text{ and } m_2^W \le 2, \\ \left\lfloor \frac{m_2^W}{m_2^W - 1} + 1 \right\rfloor & \text{otherwise.} \end{cases}$

(When $n = 1$, set $p_X = 0$ and $\lambda_{X1} = \frac{1}{\mu_1^W}$, and exit. Below, we assume $n \ge 2$.)

5. $m_2^X = \frac{(n-3)m_2^W - (n-2)}{(n-2)m_2^W - (n-1)}; \quad \mu_1^X = \frac{\mu_1^W}{(n-2)m_2^X - (n-3)}.$

6. $\alpha = (n-2)(m_2^X - 1)\left(n(n-1)(m_2^X)^2 - n(2n-5)m_2^X + (n-1)(n-3)\right).$

7. $\beta = \left((n-1)m_2^X - (n-2)\right)\left((n-2)m_2^X - (n-3)\right)^2.$

8. $m_3^X = \frac{\beta m_3^W - \alpha}{m_2^X}.$

9. $u = \begin{cases} 1 & \text{if } 3m_2^X = 2m_3^X \\ \frac{6 - 2m_3^X}{3m_2^X - 2m_3^X} & \text{otherwise} \end{cases}; \quad v = \begin{cases} 0 & \text{if } 3m_2^X = 2m_3^X \\ \frac{12 - 6m_2^X}{m_2^X(3m_2^X - 2m_3^X)} & \text{otherwise} \end{cases}.$

10. $\lambda_{X1} = \frac{u + \sqrt{u^2 - 4v}}{2\mu_1^X}; \ \lambda_{X2} = \frac{u - \sqrt{u^2 - 4v}}{2\mu_1^X}; \ p_X = \frac{\lambda_{X2}(\lambda_{X1}\mu_1^X - 1)}{\lambda_{X1}}; \ \lambda_Y = \frac{1}{(m_2^X - 1)\mu_1^X}.$

Figure 2.13: *An implementation of the* `Complete` *solution, defined for* $G \in \mathcal{PH}_3$.

then X is well-represented by a two-phase Coxian$^+$ PH distribution (see Figure 2.14(a)). By exploiting two-phase Coxian$^+$ PH distributions in $\mathcal{S} \cup \mathcal{U}_{\prime} \cup \mathcal{M}_{\prime}$, the `Complete` solution reduces the number of phases used. The above ideas lead to the following solution:

(i) If $G \in \widehat{\mathcal{U}} \cap \mathcal{PH}^-$, then the `Simple` solution provides the parameters $(n,\ p,\ \mu_{X1},\ \mu_{X2},\ p_X)$.

(ii) If $G \in \widehat{\mathcal{U}} \cap (\mathcal{PH}^-)^c$ (see Figure 2.14(a)), where $(\mathcal{PH}^-)^c$ denotes the complement of $\mathcal{PH}^-$, then let

$$n = \frac{2m_2^G - 1}{m_2^G - 1}, \tag{2.7}$$

and

$$m_2^W = \frac{1}{2}\left(\frac{n-1}{n-2} + \frac{n}{n-1}\right), \quad m_3^W = \frac{m_3^G}{m_2^G} m_2^W, \quad \text{and} \quad \mu_1^W = \frac{\mu_1^G}{p_W},$$

where $p_W = m_2^W / m_2^G$. G is then well-represented by Z, where $Z(\cdot) = p_W W(\cdot) + (1 - p_W) O(\cdot)$, where W is an EC distribution with normalized moments m_2^W and m_3^W and mean μ_1^W. The parameters $(n,\ \mu_{X1},\ \mu_{X2},\ p_X)$ of W are provided by the `Simple` solution. Also, set $p = p_W$, since W has no mass probability at zero.

(iii) If $G \in \widehat{\mathcal{M}} \cup \widehat{\mathcal{L}}$, then the `Simple` solution provides the parameters $(n,\ p,\ \mu_{X1},\ \mu_{X2},\ p_X)$, except

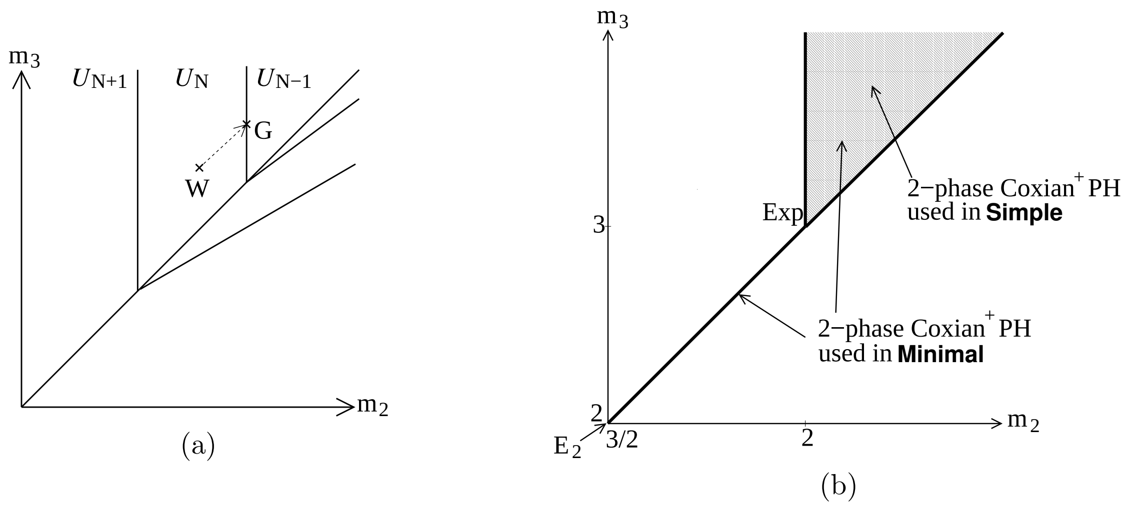


Figure 2.14: *Ideas in the* `Complete` *solution. (a) A distribution G not in $\mathcal{PH}_3^-$ is well-represented by an EC distribution W mixed with O. (b) The set of two-phase Coxian$^+$ PH distributions used is extended.*

that (2.6) is replaced by

$$n = \begin{cases} \left\lceil \frac{m_2^G}{m_2^G - 1} \right\rceil - 1 & \text{if } m_2^G \leq 2, \\ 2 & \text{otherwise.} \end{cases} \tag{2.8}$$

The next theorem guarantees that parameters obtained with the reduced n are still feasible.

Theorem 5 *If X is in set $\left\{F \mid \frac{3}{2} \leq m_2^F \leq 2 \text{ and } m_3^F = 2m_2^F - 1\right\}$, then $Z = \phi^N(X)$ is in set $\left\{F \mid \frac{N+1}{N} \leq m_2^F \leq \frac{N}{N-1} \text{ and } m_3^F = 2m_2^F - 1\right\}$.*

Proof: By Theorem 3, m_2^Z is a continuous and monotonically increasing function of m_2^X, Thus,

$$\frac{N+1}{N} \leq m_2^Z \leq \frac{N}{N-1}$$

follows by simply evaluating m_2^Z at the lower and upper bound of m_2^X. $m_3^Z = 2m_2^Z - 1$ follows from Lemma 4. ■

Analyzing the number of phases required

The number of phases used in the `Complete` solution is characterized by the following theorem.

Theorem 6 *The* `Complete` *solution uses at most $OPT(G) + 1$ phases to well-represent any distribution $G \in \mathcal{PH}_3$.*

Proof: If $G \in \mathcal{PH}_3^-$, the number of phases used in the `Complete` solution is at most that used in the `Simple` solution, i.e. $\leq$ OPT$(G) + 1$. Thus, it suffices to prove that if a distribution $G \in$

$(\mathcal{T}^{(n)} \setminus \mathcal{T}^{(\backslash-\infty)}) \cap (\mathcal{PH}_{\ni}^{-})^{\rfloor}$, then at most $n+1$ phases are needed (recall $\mathcal{S}^{(n)} \subset \mathcal{T}^{(n)}$ by Lemma 1). If $G \in (\mathcal{T}^{(n)} \setminus \mathcal{T}^{(n-1)}) \cap (\mathcal{PH}_{\ni}^{-})^{\rfloor} \cap \widehat{\mathcal{U}}$, then $m_2^G = n/(n-1)$. Thus, by (2.7), the number of phases used is $n+1$. If $G \in (\mathcal{T}^{(n)} \setminus \mathcal{T}^{(n-1)}) \cap (\mathcal{PH}_{\ni}^{-})^{\rfloor} \cap (\widehat{\mathcal{M}} \cup \widehat{\mathcal{L}})$, then the number of phases given by (2.8) is exactly n, except for input distribution G with $m_2^G \geq 2$ and $r^G = 3/2$ (i.e. exponential distribution possibly mixed with O) for which (2.8) gives 1. Thus, the number of phases used in the `Complete` solution for $G \in (\mathcal{PH}_{\ni}^{-})^{\rfloor} \cap (\widehat{\mathcal{M}} \cup \widehat{\mathcal{L}})$ is OPT(G). ■

Theorem 6 implies that any distribution in $\mathcal{S}^{(n)}$ is well-represented by an EC distribution of $n+1$ phases. In the proof, we prove a stronger property that any distribution in $\mathcal{T}^{(n)}$, which is a superset of $\mathcal{S}^{(n)}$, is well-represented by an EC distribution of $n+1$ phases, which implies the following corollary:

Corollary 3 *For all integers $n \geq 2$, $\mathcal{T}^{(n)} \subset \mathcal{S}^{(n+1)}$.*

2.8 Positive closed form solution

The `Positive` solution is built upon the `Complete` solution, but does not have mass probability at zero. The key idea in the design of the `Positive` solution is to match the input distribution either by a mixture of an EC distribution (with no mass probability at zero) and an exponential distribution, or by the convolution of an EC distribution (with positive mass probability at zero) and an exponential distribution. The use of these types of distributions makes intuitive sense, since they can approximate the EC distribution with mass probability at zero arbitrarily closely by letting the rate of the exponential distributions approach infinity. Therefore, in this section, we extend the definition of the EC distribution and use the extended EC distribution to well-represent the input distribution.

Definition 14 *An extended EC distribution has a distribution function either of the form $pX(\cdot) + (1-p)W(\cdot)$ or of the form $Z(\cdot) * X(\cdot)$, where X is an EC distribution with no mass probability at zero; Z and W are exponential distributions.*

See Figure 2.3 for the Markov chain whose absorption time defines an extended EC distribution. Note that the parameter n in an extended EC distribution denotes the number of phases in the EC portion of the extended EC distribution. Therefore, the total number of phases in the extended EC distribution is $n+1$.

The `Positive` solution

The `Positive` solution is defined for almost all the input distributions in $\mathcal{PH}_3$. Specifically, it is defined for all the distributions in

$$\mathcal{U} \cup \widehat{\mathcal{M}} \cup \left\{ F \middle| r^F \neq \frac{3}{2} \text{ and } m_3^F < 2m_2^F - 1 \right\}.$$

Figure 2.15 shows an implementation of the `Positive` solution. Below, we elaborate on the `Positive` solution, and prove an upper bound on the number of phases used in the `Positive` solution.

When the input distribution G is in $\mathcal{U} \cup \overline{\mathcal{M}}$, the EC distribution produced by the `Minimal` solution does not have a mass probability at zero. When G is in $\left\{ F \mid m_3^F < 2m_2^F - 1 \text{ and } r^F > \frac{3}{2} \right\}$, G can be

$(n,\, p,\, \lambda_Y,\, \lambda_{X1},\, \lambda_{X2},\, p_X,\, \lambda_W,\, \lambda_Z) = \texttt{Positive}(\mu_1^G,\, \mu_2^G,\, \mu_3^G)$

If $G \in \mathcal{U} \cup \left\{ F \mid m_3^F = 2m_2^F - 1 \right\} \cup \left\{ F \mid m_3^F < 2m_2^F - 1 \text{ and } m_2^F > 2 \text{ and } r^F > \frac{3}{2} \right\}$,
use `Complete`. Otherwise,

1. $m_2^G = \frac{\mu_2^G}{(\mu_1^G)^2}$; $\quad m_3^G = \frac{\mu_3^G}{\mu_1^G \mu_2^G}$; $\quad r^G = \frac{m_3^G}{m_2^G}$; $\quad k = \lfloor \frac{2m_2^G - m_3^G}{m_3^G - m_2^G} \rfloor$.

If $m_3^G \geq \frac{(k+1)m_2^G + (k+4)}{2(k+2)} m_2^G$,

2. $w = \frac{2 - m_2^G}{4\left(\frac{3}{2} - r^G\right)}$; $\quad p = \frac{(2-m_2^G)^2}{(2-m_2^G)^2 + 4(2m_2^G - 1 - m_3^G)}$; $\quad m_2^X = 2w$;
$m_3^X = 2m_2^X - 1$; $\quad \mu_1^X = \frac{\mu_1^G}{p + (1-p)w}$; $\quad \lambda_W = \frac{1}{w\mu_1^X}$; $\quad \lambda_Z = \infty$; $\quad$ go to 3.

If $r^G < \frac{(k+1)m_2^G + (k+4)}{2(k+2)}$ and $m_2^G = 2$,

2. $z = \frac{m_3^G - 2\frac{k+3}{k+2}}{3 - m_3^G}$; $\quad m_2^X = 2(1+z)$; $\quad m_3^X = \frac{k+3}{k+2} m_2^X$; $\quad \mu_1^X = \frac{\mu_1^G}{1+z}$.
$\lambda_Z = \frac{1}{z\mu_1^X}$; $\quad \lambda_W = \infty$; $\quad$ go to 3.

If $m_3^G < \frac{(k+1)m_2^G + (k+4)}{2(k+2)} m_2^G$ and $m_2^G \neq 2$,

2. $z = \dfrac{m_2^G\left((m_3^G - 3) - 2\frac{k+3}{k+2}(m_2^G - 2)\right) + m_2^G\sqrt{(m_3^G - 3)^2 + 8\frac{k+3}{k+2}(m_2^G - 2)(\frac{3}{2} - r^G)}}{2\frac{k+3}{k+2}(m_2^G - 2)^2}$;
$m_2^X = (1+z)\left(m_2^G(1+z) - 2\right)$; $\quad m_3^X = \frac{k+3}{k+2} m_2^X$; $\quad \mu_1^X = \frac{\mu_1^G}{1+z}$.
$\lambda_Z = \frac{1}{z\mu_1^X}$; $\quad \lambda_W = \infty$; $\quad$ go to 3.

3. $\mu_2^X = m_2^X (\mu_1^X)^2$; $\quad \mu_3^X = m_3^X \mu_1^X \mu_2^X$.
4. $(n,\, p,\, \lambda_Y,\, \lambda_{X1},\, \lambda_{X2},\, p_X) = \texttt{Complete}(\mu_1^X,\, \mu_2^X,\, \mu_3^X)$

Figure 2.15: *An implementation of the* `Positive` *solution, defined for a distribution* $G \in \mathcal{U} \cup \widehat{\mathcal{M}} \cup \left\{ F \middle| r^F \neq \frac{3}{2} \text{ and } m_3^F < 2m_2^F - 1 \right\}$.

well-represented by a two-phase Coxian$^+$ PH distribution, whose parameters are given by (2.4)-(2.5). Below, we focus on input distributions $G \in \left\{ F \middle| m_3^F < 2m_2^F - 1 \text{ and } r^F < \frac{3}{2} \right\}$.

We first consider the first approach of using a mixture of an EC distribution (with no mass probability at zero), X, and an exponential distribution, W, (i.e. $\lambda_Z = \infty$). Let

$$\widehat{\mathcal{L}}_N = \left\{ F \middle| \frac{N+3}{N+2} m_2^F < m_3^F \leq \frac{N+2}{N+1} m_2^F \text{ and } m_3^F < 2m_2^F - 1 \right\}.$$

Given a distribution $G \in \widehat{\mathcal{L}}_N$, we seek m_2^X, m_3^X, $0 < p < 1$, and $w > 0$ such that

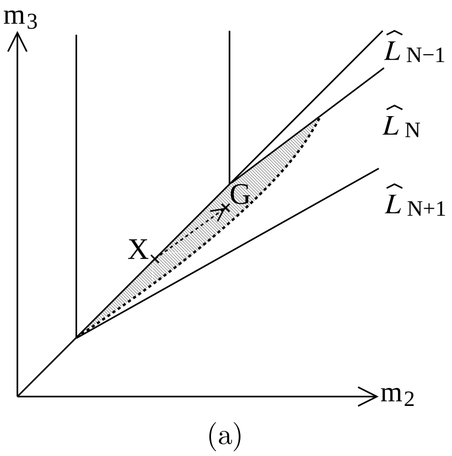
$$\frac{N+2}{N+1} \leq m_2^X < \frac{N+1}{N} \tag{2.9}$$

$$m_3^X = 2m_2^X - 1 \tag{2.10}$$

$$m_2^G = \frac{p m_2^X + 2(1-p)w^2}{(p + (1-p)w)^2} \tag{2.11}$$

$$m_3^G = \frac{p m_2^X m_3^X + 6(1-p)w^3}{(p + (1-p)w)(p m_2^X + 2(1-p)w^2)} \tag{2.12}$$

Note that (2.9)-(2.10) guarantee that we can find, via the `Complete` solution, an $(N+1)$-phase EC



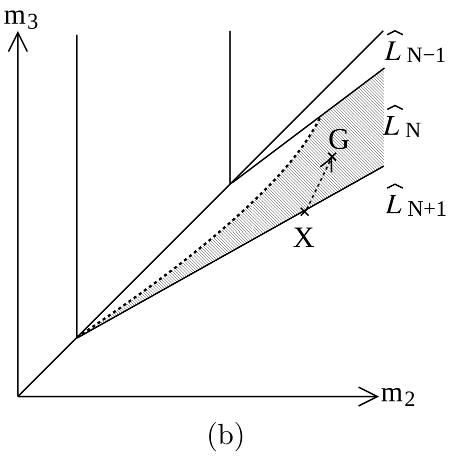


Figure 2.16: *Two regions in $\mathcal{L}_N$ that input G can fall into under the* `Positive` *solution: (a) G is well-represented by a mixture of an EC distribution X and an exponential distribution W; (b) G is well-represented by the convolution of an EC distribution X and an exponential distribution Z.*

distribution, X, such that X has no mass probability at zero and has normalized moments m_2^X and m_3^X. Let W be the exponential distribution with $\mu_1^W = w/\mu_1^X$. (2.11)-(2.12) guarantee that, by choosing μ_1^X appropriately, G is well-represented by a distribution Y, where

$$Y(\cdot) = pX(\cdot) + (1-p)W(\cdot).$$

The following lemma provides conditions on the input distribution for which the first approach is defined.

Lemma 5 *Suppose*

$$G \in \widehat{\mathcal{L}}_N \quad \textit{and} \quad \frac{(N+1)m_2^G + (N+4)}{2(N+2)} \leq r^G$$

for $N \geq 1$ (see Figure 2.16(a)). Let

$$\begin{aligned} w &= \frac{2 - m_2^G}{4\left(\frac{3}{2} - r^G\right)} \\ p &= \frac{(2-m_2^G)^2}{(2-m_2^G)^2 + 4(2m_2^G - 1 - m_3^G)} \\ m_2^X &= 2w \\ m_3^X &= 2m_2^X - 1. \end{aligned}$$

Then, $w > 0$, $0 < p < 1$, and conditions (2.9)-(2.12) are satisfied.

A proof of Lemma 5 is postponed to Appendix A.

The key idea behind Lemma 5 is to fix some of the parameters so that the set of equations becomes simpler and yet there exists a unique solution. The difficulty in finding closed form solutions is that we are given a system of nonlinear equations with high degree (2.10)-(2.12), and the solutions are not unique. By fixing some of the parameters, the system of equations can be reduced to have a unique solution. We find that w given by Lemma 5 has nice characteristics. First, m_2^X leads to a very simple expression: $m_2^X = 2w$. Second, with this expression of m_2^X, the expression involving r^G is significantly simplified:

$$r^G = \frac{2pr^X + 3(1-p)w}{2(p + (1-p)w)}.$$

Now, solving (2.10)-(2.12) for p and w is a relatively easy task, and p and w immediately gives m_2^X and m_3^X. Although Lemma 5 allows us to find a simple closed form solution, the set of input distributions defined for Lemma 5 is rather small. This necessitates the second approach of using the convolution of an EC distribution and an exponential distribution. Note that the second approach alone does not suffice, either. Applying the first approach to a small set of input distributions and applying the second approach to the rest of the input distribution, in fact, lead to simpler closed form solutions by both approaches.

Next, we consider the second approach of using the convolution of an EC distribution (with mass probability at zero) and an exponential distribution (i.e. $\lambda_W = \infty$). Given a distribution $G \in \widehat{\mathcal{L}}_N$ (we assume $r^G \neq \frac{3}{2}$), we seek m_2^X, m_3^X, and $z > 0$ such that

$$\begin{aligned}
m_2^X &\geq \frac{N+2}{N+1} && (2.13)\\
m_3^X &= \frac{N+3}{N+2} m_2^X && (2.14)\\
m_2^G &= \frac{m_2^X + 2z + 2z^2}{(1+z)^2} && (2.15)\\
m_3^G &= \frac{m_2^X m_3^X + 3m_2^X z + 6z^2 + 6z^3}{(m_2^X + 2z + 2z^2)(1+z)} && (2.16)
\end{aligned}$$

Note that (2.13)-(2.14) guarantees that we can find, via the `Complete` solution, an $(N+1)$-phase EC distribution, X, such that X has no mass probability at zero and has normalized moments m_2^X and m_3^X. Let Z be the exponential distribution whose first moment is $\mu_1^Z = z\mu_1^X$, where μ_1^X is the first moment of X. (2.15)-(2.16) guarantee that, by choosing μ_1^X appropriately, G is well-represented by the convolution of X and Z.

The following lemma provides conditions on the input distribution for which the second approach is defined.

Lemma 6 *Suppose*

$$G \in \widehat{\mathcal{L}}_N \quad \textit{and} \quad r^G < \frac{(N+1)m_2^G + (N+4)}{2(N+2)}$$

for $N \geq 1$ (see Figure 2.16(b)). If $m_2^G = 2$, we choose

$$\begin{aligned}
z &= \frac{m_3^G - 2\frac{N+3}{N+2}}{3 - m_3^G}\\
m_2^X &= 2(1+z)\\
m_3^X &= \frac{N+3}{N+2} m_2^X.
\end{aligned}$$

If $m_2^G \neq 2$, we choose

$$
\begin{aligned}
z &= \frac{m_2^G\left((m_3^G-3) - 2\frac{N+3}{N+2}(m_2^G-2)\right) + m_2^G\sqrt{(m_3^G-3)^2 + 8\frac{N+3}{N+2}(m_2^G-2)\left(\frac{3}{2}-\frac{m_3^G}{m_2^G}\right)}}{2\frac{N+3}{N+2}(m_2^G-2)^2} \\
m_2^X &= (1+z)\left(m_2^G(1+z)-2z\right) \\
m_3^X &= \frac{N+3}{N+2}m_2^X.
\end{aligned}
$$

Then, $z > 0$ and conditions (2.13)-(2.16) are satisfied.

A proof of Lemma 6 is postponed to Appendix A.

Analyzing the number of phases required

The number of phases used in the `Positive` solution is characterized by the following theorem.

Theorem 7 *The* `Positive` *solution uses at most* OPT$(G)+1$ *phases to well-represent any distribution* $G \in \mathcal{U} \cup \widehat{\mathcal{M}} \cup \left\{F \middle| r^F \neq \frac{3}{2} \text{ and } m_3^F < 2m_2^F - 1\right\}$.

Proof: Since $S^{(n)} \subset \mathcal{T}^{(n)}$ (by Lemma 1), it suffices to prove that if a distribution $G \in \mathcal{T}^{(n)}$, then at most $n+1$ phases are needed. When an input distribution $G \in \mathcal{U} \cup \widehat{\mathcal{M}}$, the `Positive` solution is the same as the `Complete` solution, and hence requires the same number of phases, which is at most OPT$(G)+1$. When $G \in \left\{F \middle| r^F > \frac{3}{2} \text{ and } m_3^F < 2m_2^F - 1\right\}$, the number of phase used in the `Positive` solution is $2 = \text{OPT}(G)$. When $G \in \left\{F \middle| r^F < \frac{3}{2} \text{ and } m_3^F < 2m_2^F - 1\right\}$, it is immediate, from the construction of the solution, that the `Positive` solution requires at most one more phase than the `Complete` solution. For this G, the `Complete` solution requires OPT(G) phases, and hence the `Positive` solution requires OPT$(G)+1$ phases. ■

2.9 Concluding remarks

In this chapter, moment matching algorithms are proposed. Also, to prove the minimality of the number of phases used in our moment matching algorithms, the set, $\mathcal{S}^{(n)}$, of distributions that are well-represented by an n-phase acyclic phase type (PH) distribution is characterized.

Our moment matching algorithms have broad applicability in computer science, engineering, operations research, etc., where the performance evaluation or optimization in stochastic environment is needed. For example, in this thesis, a moment matching algorithm is used to model a multiserver system as a (multidimensional) Markov chain, as well as in a key step of the analysis of the multidimensional Markov chain via dimensionality reduction. Furthermore, many optimization problems in stochastic environment can be formulated as Markov decision problems [159] by mapping general (input) probability distributions into PH distributions.

Our moment matching algorithms can also be used as a building block to construct a solution with additional properties. For example, the `Positive` solution can be used as a building block for yet another solution, `ZeroMatching`, that not only matches the first three moments of the input distribution but also matches the mass probability at zero. Consider a distribution G whose mass probability at zero is q. Then, G can be expressed as a mixture of O (the distribution of a random variable $V \equiv 0$) and a distribution F that does not have mass probability at zero. The `Positive` solution can be used to match the first three moments of F by an extended EC distribution, E. Now, a mixture of O and E, whose distribution function is $qO(\cdot) + (1-q)E(\cdot)$, matches the first three moments and mass probability at zero of G. Observe that the `ZeroMatching` solution uses at most $\mathrm{OPT}(G) + 1$ phases.

Beyond proving the minimality of the number of phases used in our moment matching algorithms, our characterization of set $\mathcal{S}^{(n)}$ via set $\mathcal{T}^{(n)}$ is found to be useful in developing our moment matching algorithms, since the characterization of $\mathcal{S}^{(n)}$ allows us to determine how close our PH distribution is to the minimal PH distribution, and provides intuition for coming up with improved algorithms. Another benefit of characterizing $\mathcal{S}^{(n)}$ is that some existing moment matching algorithms, such as Johnson and Taaffe's nonlinear programming approach [90], require knowing the number of phases, n, in the minimal PH distribution. The current approach involves simply iterating over all choices for n [90], whereas our characterization would immediately specify n.

In developing moment matching algorithms, we have introduced various theoretical concepts such as the normalized moments, r-value, sets $\mathcal{S}^{(n)}$ and $\mathcal{T}^{(n)}$, function ϕ, and the EC distribution, and have proved their properties. These new concepts and their properties are found to be quite useful in developing moment matching algorithms, and they have recently stimulated the research in this area.

For example, Bobbio, et al. [22] have recently used the normalized moments to provide *exact* characterizations of sets $\mathcal{S}^{(n)}$ and $\mathcal{S}^{(n)^*}$, where $\mathcal{S}^{(n)}$ is the set of distributions that can be well-represented by an n-phase acyclic PH distribution and $\mathcal{S}^{(n)^*}$ is the set of distributions that can be well-represented by an n-phase acyclic PH distribution *with no mass probability at zero*. Bobbio, et al. use their exact characterization of $\mathcal{S}^{(n)}$ to construct a *minimal* acyclic PH distribution that well-represents any distribution in $\mathcal{PH}_3$. The PH distributions to which an input distribution is mapped in [22], the Exp-Erlang distribution and the Erlang-Exp distribution, are similar to (but simpler than) the EC (Erlang-Coxian) distribution and the extended EC distribution introduced in this chapter. In fact, the EC distribution is reduced to the Exp-Erlang distribution by setting $p = 1$ and $\lambda_{X1} = \lambda_Y$ (see Figure 2.2), and the extended EC distribution is reduced to the Erlang-Exp distribution by setting $\lambda_W = \infty$, $p_X = 1$, and $\lambda_{X1} = \lambda_{X2} = \lambda_Y$ (see Figure 2.3). Since the work by Bobbio, et al. builds upon the results in this chapter, we summarize their main results in Appendix B.

In response to increased request, the closed form solutions developed in this chapter have been largely implemented, and the latest implementation of the solutions is made available at an online code repository:

http://www.cs.cmu.edu/∼osogami/code/.

Future directions

Our moment matching algorithms (or the ones recently proposed by Bobbio et al. [22]) by no mean provide perfect solution for mapping a general distribution into a PH distribution. Interesting future directions include extension to matching more moments and extension to a sequence of correlated PH distributions. Our moment matching algorithms match the first three moments of an input distribution, but it is desirable to match more moments. Also, a PH distribution can be used to model a sequence of independent random variables (such as a sequence of job sizes and a sequence of interarrival times); however, in some applications, it is desirable to capture the correlation in the sequence of random variables. The Markovian arrival process (see Section 3.2) generalizes the PH distribution, and it represents a sequence of correlated PH distributions. However, how we should map a sequence of correlated random variables into a Markovian arrival process is not well understood (see e.g. [72] for this direction of work).

An interesting future direction in characterizing PH distributions is to characterize the set of distributions that are well-represented by an n-phase *general* PH distribution, not limited to an acyclic ones. Such a characterization could be used to prove the minimality of the number of phases used in our closed form solutions in a stronger sense. Although our experience via numerical experiments suggests that allowing cycles in the PH distribution does not help to reduce the number of phases used to match the three moments, a rigorous characterization is not known to date. Note, however, that an acyclic PH distribution has a computational advantage over a cyclic one, since the generator matrix of the Markov chain whose absorption time defines an acyclic PH distribution is upper triangular. Therefore, in some applications, one might prefer an acyclic PH distribution with more phases to a cyclic PH distribution with less phases. Another interesting future direction is to characterize the set of distribution whose first k-moments can be matched by an n-phase (acyclic or general) PH distribution for $k > 3$. Such a characterization would be useful in designing moment matching algorithms that match the first $k > 3$ moments of an input distribution.

Chapter 3

Dimensionality reduction of Markov chains

How can we analyze Markov chains on a multidimensionally infinite state space?

Markov modeling is a common approach to analyzing the performance of computer systems, service facilities, and manufacturing systems. The performance (such as mean response time) of a system can be analyzed by modeling the system as a Markov chain and analyzing the stationary probabilities in the Markov chain. However, the performance analysis of a multiserver system with multiple classes of jobs has a common source of difficulty: the Markov chain that models the system behavior has a state space that grows infinitely in *multiple* dimensions. We start this chapter by providing two simple examples of such multidimensional Markov chains (Markov chains on a multidimensionally infinite state space).

First, consider two processors, each serving its own M/M/1 queue, where one of the processors (the "donor") can help the other processor (the "beneficiary") with its jobs, during times when the donor queue is empty. Specifically, the beneficiary jobs (respectively, donor jobs) arrive according to a Poisson process with rate λ_B (respectively, λ_D). The service demands of these jobs have exponential distributions with rate μ_B and μ_D, respectively. When there are no donor jobs and there are at least two beneficiary jobs, the donor server processes a beneficiary job, and hence the beneficiary jobs completes with rate $2\mu_B$. Figure 3.1(a) shows a Markov chain on a two-dimensionally infinite state space (2D Markov chain) that models the behavior of this system. In this Markov chain, one dimension tracks the number of donor jobs, and the other dimension tracks the number of beneficiary jobs. Since the behavior of beneficiary jobs depends on the number of donor jobs (specifically, whether there are 0 or ≥ 1 donor jobs) in the system, the 2D Markov chain cannot simply be decomposed into two 1D Markov chains (Markov chains on a one-dimensionally infinite state space).

Next, consider an M/M/2 queue with two priority classes, where high priority jobs have preemptive priority over low priority jobs. Specifically, the low (respectively, high) priority jobs arrive according to a Poisson process with rate λ_L (respectively, λ_H). The service demands of these jobs have exponential distributions with rate μ_L and μ_H, respectively. When there are n high priority jobs, only $\max\{2-n,0\}$ servers are available for low priority jobs. Figure 3.1(b) shows a 2D Markov chain that models the behavior of this system. In this Markov chain, one dimension tracks the number of high priority jobs, and the other dimension tracks the number of low priority jobs. Again, since the behavior of low priority jobs depends on the number of high priority jobs in the system,

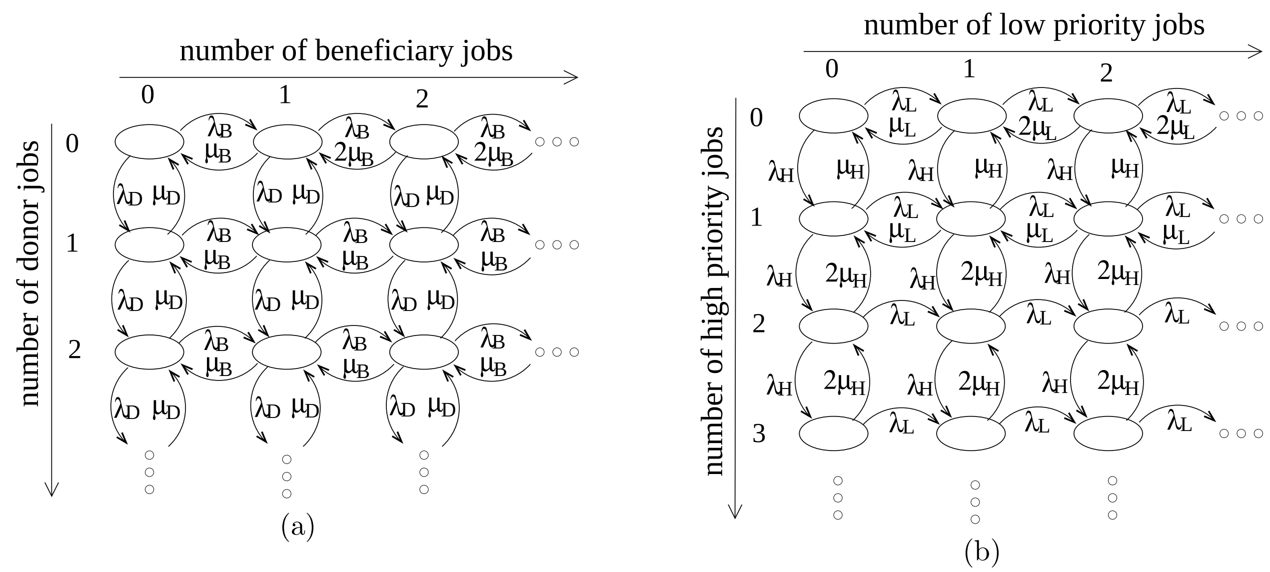


Figure 3.1: *Examples of multidimensional Markov chains that model multiserver systems: (a) a multiserver system with cycle stealing and (b) an M/M/2 queue with two preemptive priority classes.*

the 2D Markov chain cannot simply be decomposed into two 1D Markov chains. As we will see, when there are m priority classes, the performance analysis of the lowest priority class involves an mD Markov chain (Markov chain on an m-dimensionally infinite state space).

The goal of this chapter is to provide an analytical tool, which we refer to as dimensionality reduction (DR), that allows us to analyze these and more complex multidimensional Markov chains. DR reduces a multidimensional Markov chain to a 1D Markov chain, which closely approximates the multidimensional Markov chain and can be analyzed efficiently. We will define a broad class of (multidimensional) Markov chains, which we will refer to as RFB/GFB processes (recursive foreground-background processes or generalized foreground-background processes), that allow us to model many interesting multiserver systems with multiple classes of jobs, and then show an analysis of these Markov chains via DR. DR involves approximations, but we will show that the mean response time computed via DR is usually within a few percent of the simulated value. DR will enable one to analyze the performance of many multiserver systems by simply modeling them as RFB/GFB processes.

3.1 Overview

In this section, we introduce the ideas behind DR by analyzing a particular multiserver system: a queue with two servers and two priority classes. The description of DR in this section is quite intuitive, with many figures and few formulas. More details and mathematical expressions that are convenient for implementation of DR will be provided in later sections. We start with the simpler case of exponential service demand distributions, and provide a basic idea behind DR. The full range of ideas behind DR will be provided via an analysis for the case of (two phase) PH service demand distributions. We also briefly discuss the class of Markov chains to which DR can be applied to. Again, details will be provided in later sections.

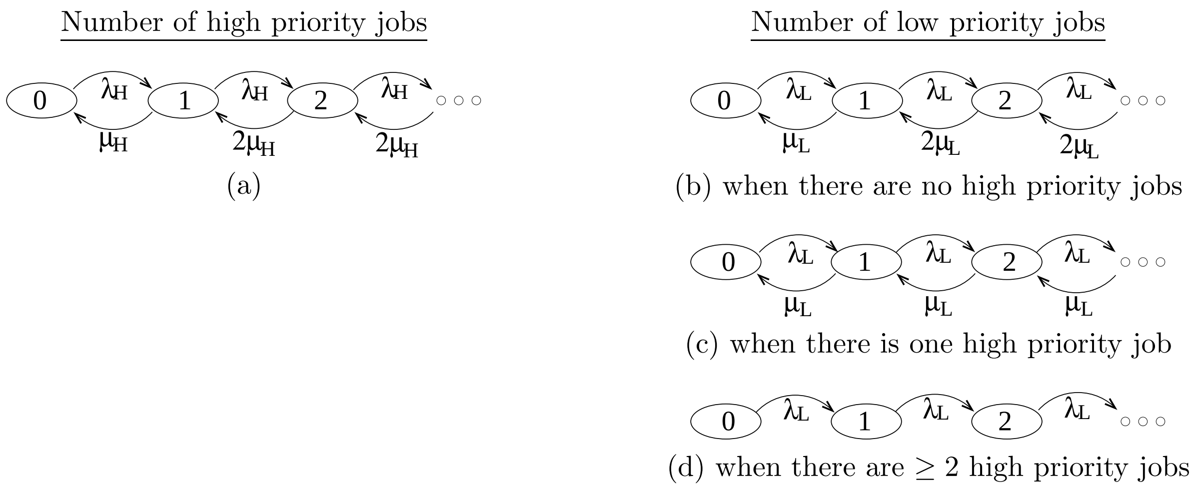


Figure 3.2: *Markov chains for an M/M/2 queue with two preemptive priority classes. (a) Markov chain for the number of high priority jobs. (b)-(d) Markov chains for the number of low priority jobs conditioned on the number of high priority jobs.*

3.1.1 Analysis of priority M/M/2 queue via DR

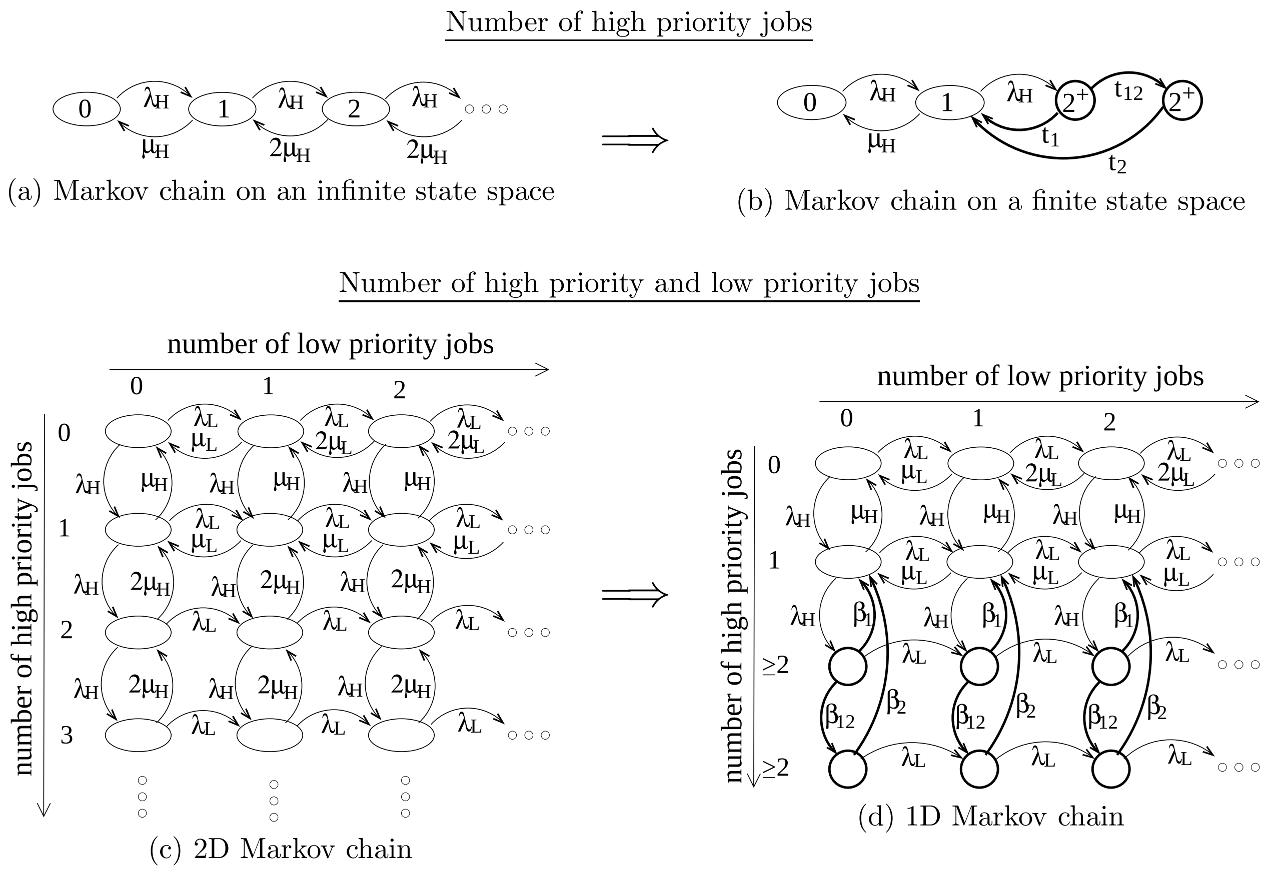
In this section, we introduce the basic idea behind DR by sketching an analysis of an M/M/2 queue with two priority classes via DR. Specifically, we reduce the 2D Markov chain in Figure 3.1(b) to a 1D Markov chain that closely approximates the 2D Markov chain, such that the 1D Markov chain can be analyzed efficiently via existing approaches.

First, observe that the performance of high priority jobs can be analyzed independently of the low priority jobs, since the behavior of the high priority jobs is not affected by the (number of) low priority jobs in the system. Specifically, Figure 3.2(a) shows the Markov chain that exactly models the number of high priority jobs in the system, and an analysis of this Markov chain allows us to calculate the mean response time of high priority jobs.

By contrast, the behavior of the low priority jobs is affected by the number of high priority jobs in the system. Specifically, when there are no high priority jobs in the system, two servers are available for the low priority jobs (see Figure 3.2(b) for the behavior of the low priority jobs in this case); when there is one high priority job in the system, one server is available for the low priority jobs (see Figure 3.2(c)); when there are more than one high priority jobs in the system, no server is available for the low priority jobs (see Figure 3.2(d)).

Observe that the 2D Markov chain in Figure 3.1(b) consists of the Markov chains in Figure 3.2. Specifically, each column in the 2D Markov chain corresponds to the Markov chain in Figure 3.2(a); the first row (respectively, the second row) in the 2D Markov chain corresponds to the Markov chain in Figure 3.2(b) (respectively, Figure 3.2(c)), and each of the other rows in the 2D Markov chain corresponds to the Markov chain in Figure 3.2(d).

Our goal is to reduce this 2D Markov chain into a 1D Markov chain that closely approximates the 2D Markov chain. Toward this end, observe that the behavior of the low priority jobs is determined only by whether there are 0, 1, or ≥ 2 high priority jobs. As long as there are ≥ 2 high priority jobs in the system, the precise number of high priority jobs does not affect the behavior of the low priority jobs. Also, observe that the infinite number of *rows* in the 2D Markov chain stems from the



(a) Markov chain on an infinite state space
(b) Markov chain on a finite state space
(c) 2D Markov chain
(d) 1D Markov chain

Figure 3.3: *Dimensionality reduction of 2D Markov chain.*

infinite number of states (levels) in the Markov chain for the high priority jobs (Figure 3.2(a)).

This motivates us to construct a Markov chain, for the high priority jobs, that differentiates only between 0, 1, or ≥ 2 high priority jobs (see Figure 3.3(b)). In Figure 3.3(b), the sojourn time in states with ≥ 2 high priority jobs is approximated by a two-phase Coxian$^+$ PH distribution, which we introduce in Section 2.2. Below, we refer to this sojourn time in states with ≥ 2 high priority as the "busy period" (of the high priority jobs), since all the servers are busy with serving high priority jobs during this period. Specifically, a busy period denotes the time from when two servers become busy until only one server is busy. We will use the moment matching algorithm developed in Chapter 2 to *well-represent* (see Definition 1) the duration of the busy period by a PH distribution.

Using Figure 3.3(b) as the Markov chain for the high priority jobs, we obtain a *1D* Markov chain that models the M/M/2 queue with two priority classes (see Figure 3.3(d)). Observe that the 1D Markov chain tracks the number of low priority jobs exactly, but differentiates only between 0, 1, and ≥ 2 high priority jobs. Note that if the duration of the busy period is exactly matched by the PH distribution, the behavior of the low priority jobs in the 1D Markov chain (Figure 3.3(d)) would be exactly the same as that in the 2D Markov chain (Figure 3.3(c)). We will see that the 1D Markov chain in Figure 3.3(d) can be analyzed efficiently via matrix analytic methods (see Section 3.2), and this allows us to compute the mean response time of low priority jobs.

3.1.2 Analysis of priority M/PH/2 queue via DR

Next, we introduce the full range of ideas behind DR by sketching an analysis of an M/PH/2 queue with two priority classes via DR. We will see that the approach that we followed in the case of an M/M/2 queue cannot simply be applied in the case of an M/PH/2 queue. In particular, we will see that a *collection* of PH distributions is needed to approximate the duration of the busy period. We assume that the service demand of high priority jobs has a two-phase Coxian$^+$ PH distribution shown in Figure 3.4(a), and the service demand of low priority jobs has an exponential distribution with rate μ_L. As before, the low (respectively, high) priority jobs arrive according to a Poisson process with rate λ_L (respectively, λ_H).

As in the case of an M/M/2 queue, the mean response time of high priority jobs can be analyzed independently of low priority jobs, since the behavior of the high priority jobs is not affected by the (number of) low priority jobs in the system. Specifically, Figure 3.4(b) shows the Markov chain that exactly models the number of high priority jobs in the system. Here, the states with ℓ high priority jobs are differentiated by the "phases" of the jobs in service (i.e., phases of the corresponding Coxian$^+$ PH distributions) for each ℓ. In the figure, the numbers shown in states (circles) denote the "phases" of the jobs in service (i.e., phases of the corresponding Coxian$^+$ PH distributions). That is, $(1,1)$ denotes that two jobs in service are both in phase 1; $(1,2)$ denotes that two jobs are in service, where one job is in phase 1 and the other is in phase 2; $(2,2)$ denotes that two jobs in service are both in phase 2.

As in the case of an M/M/2 queue, the behavior of the low priority jobs is determined by whether there are 0, 1, or ≥ 2 high priority jobs in the system. *By conditioning* on the number of high priority jobs in the system, the number of low priority jobs can be modeled by the same Markov chains as in the case of an M/M/2 queue (i.e., Figures 3.2(b)-(d)).

Figure 3.4(c) shows a portion of the 2D Markov chain, where there are ℓ low priority jobs, that models our M/PH/2 queue with two priority classes. Observe that each column in the 2D Markov chain corresponds to the Markov chain in Figure 3.4(b), and each row in the 2D Markov chain corresponds to one of the three Markov chains in Figures 3.2(b)-(d).

Again, our goal is to reduce the 2D Markov chain into a 1D Markov chain that closely approximates the 2D Markov chain. In the case of an M/M/2 queue, this dimensionality reduction is made possible by approximating the duration of the busy period (of the high priority jobs) by a *single* PH distribution (Figure 3.3(a) to Figure 3.3(b)). Unfortunately, the same approach does not work in the case of the M/PH/2 queue, since the duration of the busy period in the M/PH/2 queue depends on the "phase" of the job in service both at the beginning and end of the busy period (more precisely, immediately before the busy period and immediately after the busy period). In the case of the two-phase Coxian$^+$ PH distribution, there are four different types of busy periods depending on the phase of the job in service immediately before the busy period starts (phase 1 or 2) and the phase of the job in service immediately after the busy period ends (phase 1 or 2).

Figure 3.5 shows a Markov chain on a finite state space for the high priority jobs, where the four types of busy periods are replaced by four two-phase Coxian$^+$ PH distributions, respectively. The right half of the Markov chain represents the busy period, where there are ≥ 2 high priority jobs in the system. Specifically, the two circles on the right side labeled busy period of type (i,j) represents the busy period where the job in service at the beginning of the busy period is in phase i and the job in service at the end of the busy period is also in phase j for $1 \leq i,j \leq 2$.

Using Figure 3.5 as the Markov chain for the high priority jobs, we obtain a 1D Markov chain

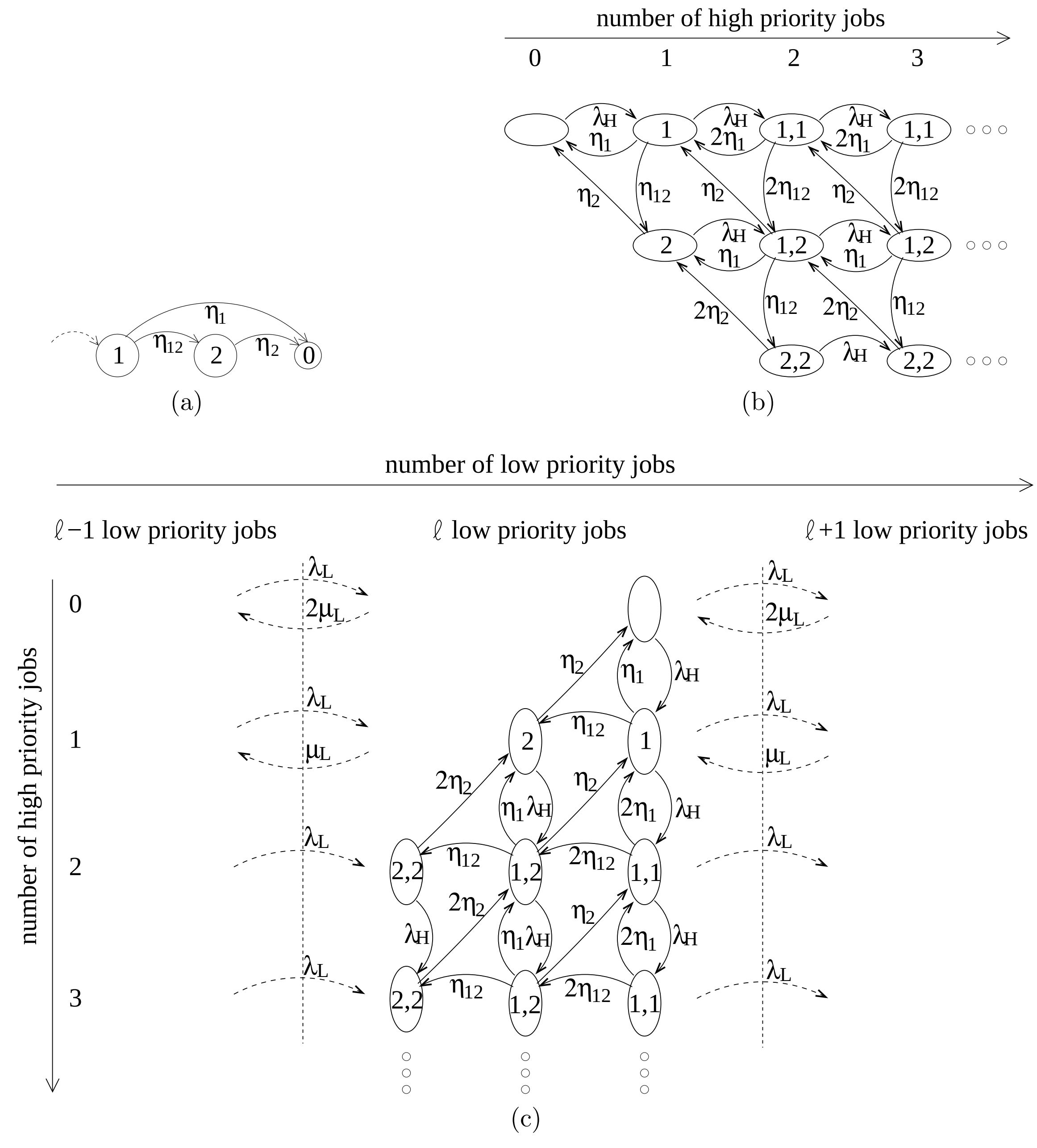


Figure 3.4: *Markov chains for an M/PH/2 queue with two preemptive priority classes. (a) A two-phase Coxian$^+$ PH distribution is shown as the absorption time in a Markov chain (state 0 denotes the absorbing state). (b) The Markov chain for the number of high priority jobs in an M/PH/2 queue when the service demand has the two-phase Coxian$^+$ PH distribution shown in (a); see Section 2.2. Here, the numbers shown in states (circles) denote the "phases" of the jobs in service (i.e., phases of the corresponding Coxian$^+$ PH distributions). (c) The 2D Markov chain for an M/PH/2 queue with two preemptive priority classes.*

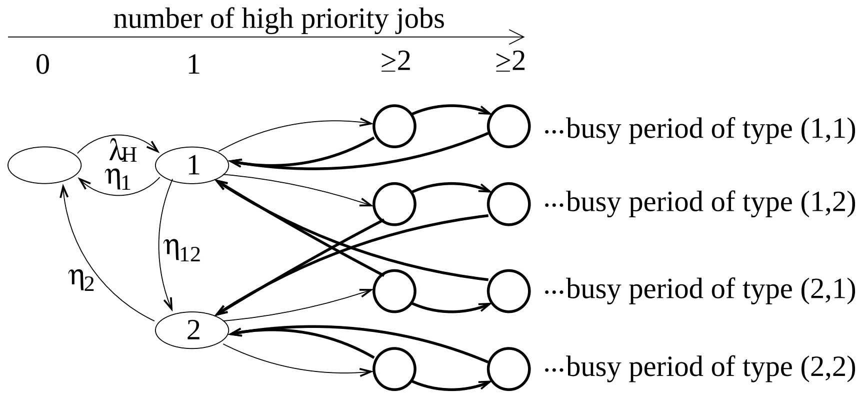


Figure 3.5: *Markov chain on a finite state space for the high priority jobs in an M/PH/2 queue with two preemptive priority classes. Labels on the transitions to/in the busy periods are omitted for clarity.*

that models the M/PH/2 queue with two priority classes (see Figure 3.6 for a portion of the 1D Markov chain). Observe that the 1D Markov chain tracks the number of low priority jobs exactly, but differentiates only between 0, 1, and ≥ 2 high priority jobs. When there is one high priority job in the system, the 1D Markov chain also tracks the phase of the high priority job in service.

Note that if the following conditions are satisfied, the behavior of the low priority jobs in the 1D Markov chain (Figure 3.6) would be exactly the same as that in the 2D Markov chain (Figure 3.4(c)).

- The duration of the conditional busy period (the busy period conditioned on the phase of the job in service at the beginning and end of the busy period) is exactly matched by the PH distribution.
- The probability of experiencing each type of busy period (the probability that the phase of the job in service at the end of the busy period is i given that the phase of the job in service at the beginning of the busy period is j for each i and j) is exactly matched.

We will choose the four PH distributions such that the durations of the conditional busy periods are *well-represented* (see Definition 1) by the PH distributions. Also, we will *exactly* match the probability of experiencing each type of busy period.

3.1.3 RFB and GFB processes

We now consider the class of (multidimensional) Markov chains to which DR can be applied. We will start with a simpler (limited) class of 2D Markov chains, which we refer to as the foreground-background (FB) process. The FB process includes the Markov chain that models an M/PH/2 queue with two priority classes (Figure 3.4(c)) as well as the Markov chains in Figure 3.1. We will then discuss two types of generalization of the FB process: the recursive FB process (RFB process) and the generalized FB process (GFB) process (see Figure 3.7). Both processes can be analyzed via DR. The RFB process includes Markov chains on higher ($m > 2$) dimensions, and the GFB process includes 2D Markov chains with fewer restrictions on their structure. In this section, we will provide only an intuitive idea of the structure of the FB, RFB, and GFB processes. Precise definitions will be given in Section 3.4.

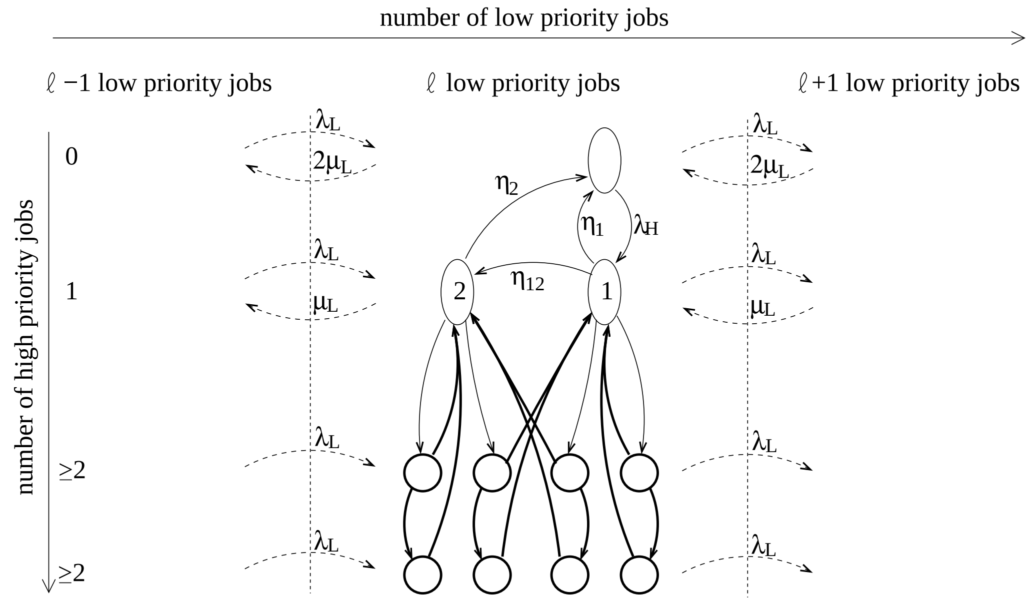


Figure 3.6: *1D Markov chain for an M/PH/2 queue with two preemptive priority classes. Labels on the transitions in the busy periods are omitted for clarity.*

Roughly speaking, an FB process consists of a foreground process and a background process, where the behavior of the foreground process can depend on the state (e.g., number of jobs) of the background process. For example, in the case of an M/M/2 queue with two priority classes, the Markov chain for the high priority jobs (Figure 3.2(a)) can be seen as the background process, and the Markov chain for the low priority jobs (Figure 3.2(b)-(d)) can be seen as the foreground process. In general, we assume that a foreground process is a QBD process as defined in Section 3.2, but the behavior of the foreground process can depend on the "level" (e.g., number of jobs) of the background process[1].

The RFB process generalizes the FB process to higher ($m > 2$) dimensions. Roughly speaking, an RFB process consists of m processes, where the behavior of the i-th process can depend on the state (e.g., number of jobs) of the $(i-1)$-th process for $2 \leq i \leq m$. We will see that DR can be applied recursively to the RFB process.

The GFB process allows a background process to also depend on a foreground process. Recall the 2D Markov chain (FB process) in Figure 3.3(c) and the 1D Markov chain in Figure 3.3(d) that we obtain via DR. Observe that the top two rows in the 2D Markov chain in Figure 3.3(c) do not have to have any special structure so that it can be reduced to a 1D Markov chain as in Figure 3.3(d). In the GFB process, we assume that there exists a level κ in the *background* process such that the background and foreground processes behave independently of each other while the *background* process is in levels $\geq \kappa$, but the background and foreground process can have complex interaction while the background process is in levels $< \kappa$.

[1]We also assume that there exists a level κ in the background process such that the behavior of the foreground process stays the same while the background process is in levels $\geq \kappa$. For example, in an M/M/2 queue with two priority classes, the behavior of the low priority jobs (foreground process) stays the same while there are ≥ 2 high priority jobs (background process is in levels ≥ 2).

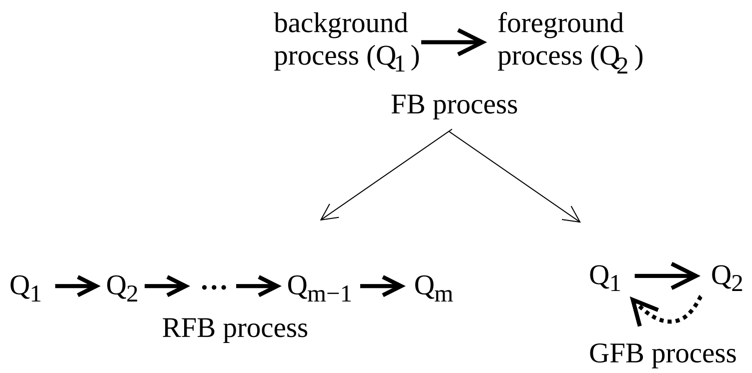


Figure 3.7: *FB, RFB, and GFB processes.*

3.1.4 Organization of this chapter

The rest of this chapter is organized as follows. Section 3.2 provides a brief tutorial on the QBD process and matrix analytic methods for analyzing the QBD process. For the purpose of understanding the rest of the chapter, one only needs to understand the definitions of the QBD process. In Section 3.3, we review the state of the art in the analysis of multidimensional Markov chains. In Section 3.4, we define the FB, RFB, and GFB processes and provide many examples of multiserver systems that can be modeled as these processes. In Section 3.5, we analyze the FB, RFB, and GFB processes via DR. In Section 3.6, we introduce approximations in DR which reduce the computational complexity. In Sections 3.5-3.6, we analyze the *stationary probabilities* is the FB, RFB, and GFB processes. Some technical details in DR are postponed to Section 3.7. In Section 3.8, we discuss how the stationary probabilities in the FB, RFB, and GFB processes can be translated into performance measures such as the mean and variance of response time and queue length. In Section 3.9, we evaluate the accuracy and running time of DR and its approximations.

3.2 Brief tutorial on matrix analytic methods

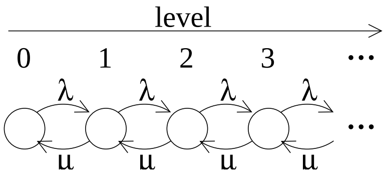
In this section, we provide a brief tutorial on the quasi-birth-and-death (QBD) process and matrix analytic methods for analyzing QBD processes. We will also provide an overview of the Markovian arrival process (MAP), first introduced by M. F. Neuts [135, 137]. The MAP is a generalization of the PH distribution, and it represents a correlated sequence of PH distributions. In particular, we will discuss how a MAP/PH/1 queue is modeled as a QBD process.

3.2.1 Quasi-birth-and-death process

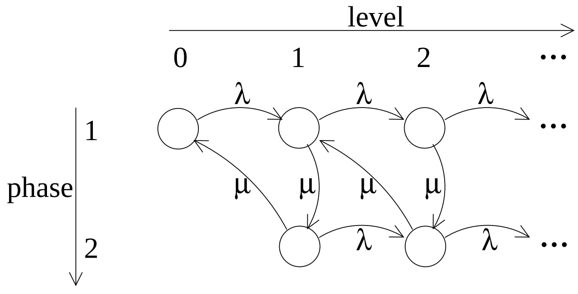
Examples of QBD processes

We start by providing canonical examples of QBD processes. Here, we provide both pictorial explanation and more formal explanation. Pictorial explanation gives intuitive understanding of the QBD process, and more formal explanation allows us to get used to the notation that we use later.

First, a birth-and-death process is an example of a QBD process. Figure 3.8(a) shows an example of a birth-and-death process. This birth-and-death process models the number of jobs in an M/M/1 queue, where jobs arrive according to a Poisson process with rate λ, and the service demand has



(a) birth-and-death process



(b) QBD process

Figure 3.8: *Examples of QBD processes.*

an exponential distribution with rate μ. In general, a birth-and-death process is a Markov chain on the states $\{0, 1, 2, 3, ...\}$, where transitions are allowed only to the neighboring states. That is, from state ℓ, there are transitions only to state $\ell - 1$ and state $\ell + 1$ for $\ell \geq 1$, and from state 0, there is a transition only to state 1. Thus, the generator matrix of a birth-and-death process is of the form:

$$\mathbf{Q} = \begin{pmatrix} -f_0 & f_0 & & \\ b_1 & -(b_1 + f_1) & f_1 & \\ & b_2 & -(b_2 + f_2) & \ddots \\ & & \ddots & \ddots \end{pmatrix}, \tag{3.1}$$

where f_ℓ denotes the (foreword) transition from state ℓ to state $\ell + 1$, and b_ℓ denotes the (backward) transition from state ℓ to state $\ell - 1$. In Figure 3.8(a), $f_\ell = \lambda$ and $b_\ell = \mu$ for all ℓ.

Figure 3.8(b) shows an example of a QBD process that is not a birth-and-death process. This QBD process models the number of jobs in a M/Er/1 queue, where jobs arrive according to a Poisson process with rate λ, and their service demand has an Erlang-2 distribution (as defined in Section 2.2), which has parameters

$$\overrightarrow{\tau} = (1, 0) \quad \text{and} \quad \mathbf{T} = \begin{pmatrix} -\mu & \mu \\ 0 & -\mu \end{pmatrix}.$$

That is, a job is in phase 1 when it arrives. After a job in phase 1 receives service for a random time having an exponential distribution with rate μ, the job transitions into phase 2. After a job in phase 2 receives service for a random time having an exponential distribution with rate μ, the job is completed.

Definition of QBD process

In general, a QBD process is a Markov chain on the state space $\{(i, \ell) | 1 \leq i \leq n_\ell, \ell \geq 0\}$, where the state space can be divided into levels, and level ℓ has n_ℓ states (phases) for each ℓ[2]. For example, in Figure 3.8(b), $n_0 = 1$ and $n_\ell = 2$ for $\ell \geq 1$. In a QBD process, transitions are allowed only to the

[2] In the literature, the standard notation is "(ℓ, i)," where the level is followed by the phase. In this thesis, we choose (i, ℓ), because a level (respectively, phase) corresponds to a column (respectively, row) in our figures of Markov chains. Notice that (row,column) is a standard matrix notation.

neighboring levels or within the same level. Thus, a QBD process has a generator matrix of the form:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{L}^{(0)} & \mathbf{F}^{(0)} & & & \\ \mathbf{B}^{(1)} & \mathbf{L}^{(1)} & \mathbf{F}^{(1)} & & \\ & \mathbf{B}^{(2)} & \mathbf{L}^{(2)} & \mathbf{F}^{(2)} & \\ & & \ddots & \ddots & \ddots \end{pmatrix}, \tag{3.2}$$

where submatrix $\mathbf{F}^{(\ell)}$ encodes (forward) transitions from level (column) ℓ to level $\ell+1$ for $\ell \geq 0$, submatrix $\mathbf{B}^{(\ell)}$ encodes (backward) transitions from level ℓ to level $\ell-1$ for $\ell \geq 1$, and submatrix $\mathbf{L}^{(\ell)}$ encodes (local) transitions within level ℓ for $\ell \geq 0$. Specifically, (i,j) element of $\mathbf{F}^{(\ell)}$ is the transition rate from state (i,ℓ), i.e. phase i of level ℓ, to state $(j,\ell+1)$ for all i,j; (i,j) element of $\mathbf{B}^{(\ell)}$ is the transition rate from state (i,ℓ) to state $(j,\ell-1)$ for all i,j; (i,j) element of $\mathbf{L}^{(\ell)}$ is the transition rate from state (i,ℓ) to state (j,ℓ) for $i \neq j$.

For example, in Figure 3.8(b),

$$\begin{aligned}
\mathbf{F}^{(0)} &= \begin{pmatrix} \lambda & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{F}^{(\ell)} = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \quad \text{for } \ell \geq 1, \\
\mathbf{B}^{(1)} &= \begin{pmatrix} 0 \\ \mu \end{pmatrix} \quad \text{and} \quad \mathbf{B}^{(\ell)} = \begin{pmatrix} 0 & 0 \\ \mu & 0 \end{pmatrix} \quad \text{for } \ell \geq 2, \\
\mathbf{L}^{(0)} &= -\lambda \quad \text{and} \quad \mathbf{L}^{(\ell)} = \begin{pmatrix} -(\mu+\lambda) & \mu \\ 0 & -(\mu+\lambda) \end{pmatrix} \quad \text{for } \ell \geq 1.
\end{aligned}$$

3.2.2 Markovian arrival process

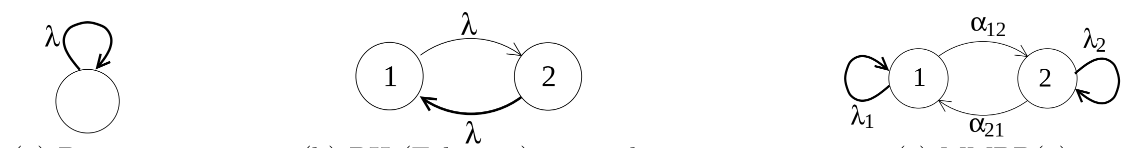
An application of the QBD process is modeling the number of jobs (queue length) in a MAP/PH/1/FCFS queue, where jobs arrive according to a Markovian arrival process (MAP), jobs are served in the order of their arrivals (FCFS), and their service demand has a PH distribution (as defined in Section 2.2). A MAP is used to model a job arrival process in Chapter 7. A PH distribution can also be used to model a job arrival process (PH renewal process), by specifying the interarrival time by the PH distribution. However, the interarrival times in a PH renewal process are i.i.d. random variables, while some applications require capturing the correlation in the interarrival times. The MAP generalizes the PH renewal process in the sense that the interarrival times in a MAP can be correlated.

For the purpose of understanding the later chapters, one only needs to know the definition of the MAP and a subclass of the MAP, the MMPP (Markov modulated Poisson process). In Appendix C, we will provide some basic properties of the MAP, so that a reader can further study the characteristics of the MAPs that we will use.

Examples of Markovian arrival processes

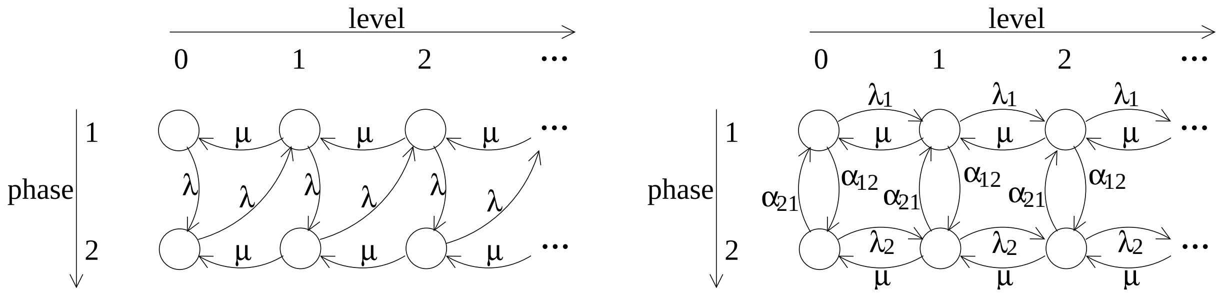
We start by providing canonical examples of MAPs. Again, we provide both pictorial explanation and more formal explanation. We will view a MAP as a point process, which is a random sequence of "events" such as the epochs of job arrivals.

First, a Poisson process is a MAP. In a Poisson process, the intervals between consecutive events are independent and identically distributed exponential random variables. Figure 3.9(a) illustrates a Poisson process as the epochs of transitions in a Markov chain. When there is a transition (from a



(a) Poisson (b) PH (Erlang-2) renewal (c) MMPP(2)

Figure 3.9: *Examples of Markovian arrival processes.*



(a) Er/M/1 (b) MMPP(2)/M/1

Figure 3.10: *QBD processes for MAP/M/1 queues.*

state to itself) in the Markov chain, there is an event in the Poisson process. Recall the birth-and-death process modeling an M/M/1 queue (Figure 3.8(a)), where jobs arrive according to a Poisson process with rate λ. At the moment of an "event" in the Poisson process, the number of jobs (or the level of the birth-and-death process) is incremented by one.

Second, a PH renewal process is a MAP. In a PH renewal process, the intervals between consecutive events are independent and identically distributed with a PH distribution. Figure 3.9(b) illustrates a PH renewal process, when the PH distribution is an Erlang-2 distribution, as the epochs of *some* transitions in a Markov chain. There is a transition (thick arrow) associated with an event. This transition corresponds the transition into the absorbing state in the Markov chain whose absorption time defines the Erlang-2 distribution (see Figure 2.6(b)). Thus, a PH renewal process has an event when the Markov chain (for the PH distribution) transitions into the absorbing state, and at this moment the Markov chain immediately transitions back to a initial state according to the initial probability vector. Figure 3.10(a) shows a QBD process modeling an Er/M/1 queue, where jobs arrive according to a PH renewal process shown in Figure 3.9(b), and their service demand has an exponential distribution with rate μ.

More formally, consider the PH$(\vec{\tau}, \mathbf{T})$ distribution (as defined in Section 2.2). Let $\vec{t} = -\mathbf{T}\vec{1}$. Matrix $\mathbf{D} = \mathbf{T} + \vec{t}\vec{\tau}$, defines the generator matrix of a Markov chain. For example, the above Erlang-2 distribution has parameters

$$\vec{\tau} = (1, 0) \quad \text{and} \quad \mathbf{T} = \begin{pmatrix} -\lambda & \lambda \\ 0 & -\lambda \end{pmatrix},$$

and hence

$$\vec{t}\vec{\tau} = \begin{pmatrix} 0 \\ \lambda \end{pmatrix} (1, 0) = \begin{pmatrix} 0 & 0 \\ \lambda & 0 \end{pmatrix}.$$

The nonzero (i,j) element of $\vec{t}\,\vec{\tau}$ is a transition associated with an event. This transition can be seen as a transition into the absorbing state followed by an instantaneous transition into a initial state following the initial probability vector $\vec{\tau}$. The (i,j) element of $\mathbf{T}$ is a transition that is not associated with an event for $i \neq j$.

Our third example, a Markov modulated Poisson process (MMPP), allows correlation between inter-event times. Figure 3.9(c) illustrates an MMPP of order 2, MMPP(2), as the epochs of some transitions in a Markov chain. There are transitions (thick arrows) associated with events. The transitions between the two states are not associated with events. While the Markov chain is in state 1, events occur with rate λ_1, and while the Markov chain is in state 2, events occur with rate λ_2. For example, an MMPP(2) can be seen as a job arrival process which alternates between high arrival period (state 1) and low arrival period (state 2). During the high arrival period, the arrival rate is λ_1, and during the low arrival period, the arrival rate is λ_2, where $\lambda_1 > \lambda_2$. Figure 3.10(b) shows a QBD process modeling an MMPP(2)/M/1 queue, where jobs arrive according to a MMPP(2) process shown in Figure 3.9(c), and their service demand has an exponential distribution with rate μ.

More formally, the Markov chain in Figure 3.9(c) has infinitesimal generator

$$\mathbf{D} = \mathbf{D}_0 + \mathbf{D}_1,$$

where

$$\mathbf{D}_0 = \begin{pmatrix} -(\alpha_{12}+\lambda_1) & \alpha_{12} \\ \alpha_{21} & -(\alpha_{21}+\lambda_2) \end{pmatrix} \quad \text{and} \quad \mathbf{D}_1 = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}.$$

The nonzero (i,j) element of $\mathbf{D}_1$ is a transition associated with an event in the MMPP(2), and the (i,j) element of $\mathbf{D}_0$ (here, $i \neq j$) is a transition that is not associated with an event.

Definition of MAP

In general, a MAP is a point process defined by the epochs of some transitions in a Markov chain.

Definition 15 *Consider a Markov chain with infinitesimal generator $\mathbf{D} = \mathbf{D}_0 + \mathbf{D}_1$, where all the off-diagonal elements of $\mathbf{D}_0$ and all the elements of $\mathbf{D}_1$ are nonnegative. The transitions associated with $\mathbf{D}_1$ are called type 1 transitions. A MAP with parameters $(\mathbf{D}_0, \mathbf{D}_1)$, MAP$(\mathbf{D}_0, \mathbf{D}_1)$, is a point process where an event occurs when a type 1 transition occurs in the Markov chain.*

Figure 3.11 illustrates a MAP of order 2 having parameters

$$\mathbf{D}_0 = \begin{pmatrix} -\sigma_1 & \alpha_{12} \\ \alpha_{21} & -\sigma_2 \end{pmatrix} \quad \text{and} \quad \mathbf{D}_1 = \begin{pmatrix} \lambda_{12} & \lambda_{21} \\ \lambda_{21} & \lambda_{22} \end{pmatrix},$$

where $\sigma_i = \sum_{j \neq i} \alpha_{ij} + \sum_j \lambda_{ij}$ for $i = 1, 2.$. To completely specify a MAP, the initial probability vector in the Markov chain needs to be specified. Throughout, we assume that the initial probability vector is the same as the stationary probability vector. That is, our MAPs are *stationary* MAPs.

A MAP$(\mathbf{D}_0, \mathbf{D}_1)$ is called a **Markov modulated Poisson process**, **MMPP**, if $\mathbf{D}_1$ is diagonal. That is, in the Markov chain that defines an MMPP, all the transitions that are associated with events do not change the state.

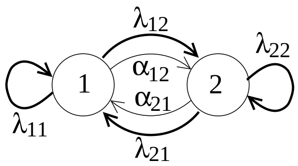


Figure 3.11: *A MAP(2). Transitions shown in thick arrows (transitions with λ_{ij}) are associated with events in the MAP(2). Transitions with α_{ij} only change the state.*

QBD process modeling MAP/PH/1/FCFS queue

The QBD process that models a MAP/PH/1 queue length has a compact and convenient representation when we use the notation introduced above and in Section 2.2,

Consider a simpler case of an M/PH/1 queue, where jobs arrive according to a Poisson process with rate λ, and the service demand has a PH($\vec{\tau}, \mathbf{T}$) distribution. The generator matrix for the queue length can be represented as

$$\widehat{\mathbf{Q}} = \begin{pmatrix} \widehat{\mathbf{L}}^{(0)} & \widehat{\mathbf{F}} & & & \\ \widehat{\mathbf{B}} & \widehat{\mathbf{L}} & \widehat{\mathbf{F}} & & \\ & \widehat{\mathbf{B}} & \widehat{\mathbf{L}} & \widehat{\mathbf{F}} & \\ & & \ddots & \ddots & \ddots \end{pmatrix},$$

where $\widehat{\mathbf{F}} = \lambda\widehat{\mathbf{I}}$, $\widehat{\mathbf{B}} = \vec{t}\,\vec{\tau}$, $\widehat{\mathbf{L}} = \mathbf{T} - \lambda\widehat{\mathbf{I}}$, and $\widehat{\mathbf{L}}^{(0)} = -\lambda\widehat{\mathbf{I}}$. Here, $\widehat{\mathbf{I}}$ is an identity matrix of order equal to $\mathbf{T}$, and $\vec{t} = -\mathbf{T}\vec{1}$.

Next, consider a MAP/M/1 queue, where jobs arrive according to a MAP($\mathbf{D}_0$,$\mathbf{D}_1$), and the service demand has an exponential distribution with rate μ. The generator matrix for the queue length can be represented as

$$\check{\mathbf{Q}} = \begin{pmatrix} \check{\mathbf{L}}^{(0)} & \check{\mathbf{F}} & & & \\ \check{\mathbf{B}} & \check{\mathbf{L}} & \check{\mathbf{F}} & & \\ & \check{\mathbf{B}} & \check{\mathbf{L}} & \check{\mathbf{F}} & \\ & & \ddots & \ddots & \ddots \end{pmatrix},$$

where $\check{\mathbf{F}} = \mathbf{D}_1$, $\check{\mathbf{B}} = \mu\check{\mathbf{I}}$, $\check{\mathbf{L}} = \mathbf{D}_0 - \mu\check{\mathbf{I}}$, and $\check{\mathbf{L}}^{(0)} = \mathbf{D}_0$. Here, $\check{\mathbf{I}}$ is an identity matrix of order equal to $\mathbf{D}_0$.

Finally, consider a MAP/PH/1 queue, where jobs arrive according to a MAP($\mathbf{D}_0$,$\mathbf{D}_1$), and the service demand has a PH($\tau, \mathbf{T}$) distribution. The generator matrix for the queue length can be represented as

$$\mathbf{Q} = \begin{pmatrix} \mathbf{L}^{(0)} & \mathbf{F} & & & \\ \mathbf{B} & \mathbf{L} & \mathbf{F} & & \\ & \mathbf{B} & \mathbf{L} & \mathbf{F} & \\ & & \ddots & \ddots & \ddots \end{pmatrix}$$

where $\mathbf{F} = \mathbf{D}_1 \otimes \widehat{\mathbf{I}}$, $\mathbf{B} = \check{\mathbf{I}} \otimes \vec{t}\,\vec{\tau}$, $\mathbf{L} = \check{\mathbf{I}} \otimes \mathbf{T} + \mathbf{D}_0 \otimes \widehat{\mathbf{I}}$, and $\mathbf{L}^{(0)} = \mathbf{D}_0 \otimes \widehat{\mathbf{I}}$, where $\otimes$ denotes the Kronecker product. Observe that the QBD process for a MAP/PH/1 queue can be expressed as a "superposition" of the QBD processes for an M/PH/1 queue and a MAP/M/1 queue: specifically, $\mathbf{F} = \check{\mathbf{F}} \otimes \widehat{\mathbf{I}}$, $\mathbf{B} = \check{\mathbf{I}} \otimes \widehat{\mathbf{B}}$, $\mathbf{L} = \check{\mathbf{I}} \otimes (\widehat{\mathbf{L}} - \widehat{\mathbf{L}}^{(0)}) + \check{\mathbf{L}}^{(0)} \otimes \widehat{\mathbf{I}}$, and $\mathbf{L}^{(0)} = \check{\mathbf{L}}^{(0)} \otimes \widehat{\mathbf{I}}$.

3.2.3 Matrix analytic methods

In this section, we analyze the stationary probabilities in the QBD process via matrix analytic methods. We will also discuss how the stationary probabilities can be translated into other performance metrics such as the mean response time. We will describe the analysis rather formally, so that a reader can apply the methods immediately. See [111, 136, 137] for more details and intuitions behind the analysis.

Stationary probabilities

We first discuss some key properties of the stationary probabilities in a QBD process. Consider a simpler case of the birth-and-death process having the generator matrix shown in (3.1). Let π_ℓ be the stationary probability that the birth-and-death process is in level ℓ for $\ell \geq 0$. Then, it is easy to see that $\pi_\ell = \rho_\ell \pi_{\ell-1}$, where $\rho_\ell = \frac{f_{\ell-1}}{b_\ell}$, for $\ell \geq 0$. It turns out that the stationary probabilities in a QBD process have a similar property. Let $\overrightarrow{\pi_\ell}$ be the stationary probability vector in a QBD process having generator matrix shown in (3.2). Here, the i-th element of vector $\overrightarrow{\pi_\ell}$ denotes the stationary probability that the QBD process is in phase i of level ℓ, i.e. state (i, l). It turns out that there exists a matrix $\mathbf{R}^{(\ell)}$ such that $\overrightarrow{\pi_\ell} = \overrightarrow{\pi_{\ell-1}}\mathbf{R}^{(\ell)}$ for each $\ell \geq 1$.

Specifically, the stationary probability vector in the QBD process is given recursively by

$$\overrightarrow{\pi_\ell} = \overrightarrow{\pi_{\ell-1}}\mathbf{R}^{(\ell)}, \tag{3.3}$$

where $\mathbf{R}^{(\ell)}$ is given recursively via:

$$\mathbf{F}^{(\ell-1)} + \mathbf{R}^{(\ell)}\mathbf{L}^{(\ell)} + \mathbf{R}^{(\ell)}\mathbf{R}^{(\ell+1)}\mathbf{B}^{(\ell+1)} = \mathbf{0}. \tag{3.4}$$

When the QBD process has an infinite number of levels, the state space of the QBD process needs to be truncated, so that $\mathbf{R}^{(\ell)}$ matrices can be calculated from a certain large enough integer $\ell = L$ to $\ell = 1$ recursively via (3.4). The truncation level L needs to be chosen carefully such that the stationary probability that the QBD process is above level L is negligible [28].

When the QBD process repeats after a certain level, $\hat{\ell}$, (i.e., $\mathbf{F}^{(\ell)} = \mathbf{F}$, $\mathbf{L}^{(\ell)} = \mathbf{L}$, and $\mathbf{B}^{(\ell)} = \mathbf{B}$ for all $\ell \geq \hat{\ell}$), $\mathbf{R}^{(\ell)}$ is the same for all $\ell > \hat{\ell}$, and $\mathbf{R} = \mathbf{R}^{(\ell)}$ (for $\ell > \hat{\ell}$) is given by the minimal nonnegative solution to the following matrix quadratic equation[3]:

$$\mathbf{F} + \mathbf{R}\mathbf{L} + (\mathbf{R})^2\mathbf{B} = \mathbf{0}. \tag{3.5}$$

Once $\mathbf{R}^{(\hat{\ell}+1)} = \mathbf{R}$ is obtained, $\mathbf{R}^{(\ell)}$ for $1 \leq \ell \leq \hat{\ell}$, is given recursively via (3.4). Various approaches for calculating $\mathbf{R}$ have been proposed, and in Figure 3.12 we show an algorithm for calculating $\mathbf{R}$, $\mathbf{R}^{(\ell)}$'s, and other relevant matrices.

Once $\mathbf{R}$ and $\mathbf{R}^{(\ell)}$'s are obtained, the stationary probability vector $\overrightarrow{\pi_\ell}$ can be calculated recursively from $\overrightarrow{\pi_0}$ via (3.3) for $\ell \geq 1$. Thus, all that remains is to calculate $\overrightarrow{\pi_0}$. Vector $\overrightarrow{\pi_0}$ is given by a positive solution of

$$\overrightarrow{\pi_0}\left(\mathbf{L}^{(0)} + \mathbf{R}^{(1)}\mathbf{B}^{(1)}\right) = \overrightarrow{0},$$

[3]Any other nonnegative solution $\mathbf{R}'$ of (3.5) satisfies $(\mathbf{R})_{ij} \leq (\mathbf{R}')_{ij}$ for all i and j.

Input: $\mathbf{F}$, $\mathbf{L}$, $\mathbf{B}$, ϵ
Output: $\mathbf{R}$, $\mathbf{G}$, $\mathbf{U}$

$\mathbf{H} = (-\mathbf{L})^{-1}\mathbf{F}$;
$\mathbf{K} = (-\mathbf{L})^{-1}\mathbf{B}$;
$\mathbf{G} = \mathbf{K}$;
$\mathbf{T} = \mathbf{H}$;
<u>repeat</u>
  $\mathbf{U} = \mathbf{HK} + \mathbf{KH}$;
  $\mathbf{M} = (\mathbf{H})^2$;
  $\mathbf{H} = (\mathbf{I} - \mathbf{U})^{-1}\mathbf{M}$;
  $\mathbf{M} = (\mathbf{K})^2$;
  $\mathbf{K} = (\mathbf{I} - \mathbf{U})^{-1}\mathbf{M}$;
  $\mathbf{G} = \mathbf{G} + \mathbf{TK}$;
  $\mathbf{T} = \mathbf{TH}$;
<u>until</u> $||\vec{1} - \mathbf{G}\vec{1}||_\infty \le \epsilon$
$\mathbf{U} = \mathbf{L} + \mathbf{FG}$;
$\mathbf{R} = \mathbf{F}(-\mathbf{U})^{-1}$;

(a) repeating part

Input: $\mathbf{R}$, $\mathbf{G}$, $\mathbf{U}$
Output: $\mathbf{R}^{(\ell)}$'s, $\mathbf{G}^{(\ell)}$'s, $\mathbf{U}^{(\ell)}$'s

$\mathbf{U}^{(\hat{\ell}+1)} = \mathbf{U}$
$\mathbf{G}^{(\hat{\ell}+1)} = \mathbf{G}$
$\mathbf{R}^{(\hat{\ell}+1)} = \mathbf{R}$
<u>for</u> $\ell = \hat{\ell}$ <u>to</u> 1
  $\mathbf{U}^{(\ell)} = \mathbf{L}^{(\ell)} + \mathbf{F}^{(\ell)}\mathbf{G}^{(\ell+1)}$
  $\mathbf{G}^{(\ell)} = (-\mathbf{U}^{(\ell)})^{-1}\mathbf{B}^{(\ell)}$
  $\mathbf{R}^{(\ell)} = \mathbf{F}^{(\ell-1)}(-\mathbf{U}^{(\ell)})^{-1}$
<u>end</u>

(b) nonrepeating part

Figure 3.12: *Algorithms for calculating* $\mathbf{R}$, $\mathbf{R}^{(\ell)}$*'s, and other relevant matrices [111]. The input matrices (*$\mathbf{R}$, $\mathbf{G}$, *and* $\mathbf{U}$*) for part (b) are given by the output of part (a).*

normalized by the equation

$$\vec{\pi_0}\sum_{\ell=0}^{\infty}\prod_{i=1}^{\ell}\mathbf{R}^{(i)}\vec{1} = 1 \quad \Longleftrightarrow \quad \vec{\pi_0}\left(\sum_{\ell=0}^{\hat{\ell}-1}\prod_{i=1}^{\ell}\mathbf{R}^{(i)} + \left(\prod_{i=1}^{\hat{\ell}}\mathbf{R}^{(i)}\right)(\mathbf{I}-\mathbf{R})^{-1}\right)\vec{1} = 1,$$

where $\vec{0}$ and $\vec{1}$ are vectors with an appropriate number of elements of 0 and 1, respectively.

Translating stationary probabilities into performance measures

The stationary probabilities obtained above can be used to compute other relevant performance measures. Specifically, we consider a QBD process modeling a MAP/PH/1 queue or other queueing systems where level ℓ of the QBD process consists of the states with ℓ jobs in the system. We first analyze the moments of the number of jobs in the system. The mean response time follows immediately from the mean number of jobs in the system. We will also discuss how to compute higher moments of response time.

First, the mean number of jobs in the system, N, is given by

$$\mathsf{E}[N] = \sum_{\ell=0}^{\infty}\ell\vec{\pi_\ell}\vec{1}$$

$$
\begin{aligned}
&= \sum_{\ell=0}^{\hat{\ell}-1} \ell \vec{\pi_\ell} \vec{1} + \sum_{i=0}^{\infty} (\hat{\ell} + i) \vec{\pi_{\hat{\ell}}} (\mathbf{R})^i \vec{1} \\
&= \sum_{\ell=0}^{\hat{\ell}-1} \ell \vec{\pi_\ell} \vec{1} + \hat{\ell} \vec{\pi_{\hat{\ell}}} (\mathbf{I} - \mathbf{R})^{-1} \vec{1} + \vec{\pi_{\hat{\ell}}} (\mathbf{I} - \mathbf{R})^{-2} \mathbf{R} \vec{1} \qquad (3.6)
\end{aligned}
$$

Similarly, the r-th moment of the number of jobs in the system can be computed via

$$
\mathsf{E}[N^r] = \sum_{\ell=0}^{\infty} \ell^r \vec{\pi_\ell} \vec{1}
$$

for $r \geq 2$.

The mean response time, $\mathsf{E}[T]$, is given immediately from $\mathsf{E}[N]$ via Little's law:

$$
\mathsf{E}[T] = \frac{\mathsf{E}[N]}{\lambda},
$$

where λ is the arrival rate.

Computation of higher moments of response time, T, depends on particular queueing systems. When jobs are completed in the order of their arrivals (first-in-first-out, FIFO), a generalization of Little's law (distributional Little's law) applies [63, 32]. For example, in a MAP/PH/1/FCFS queue, jobs are completed in the FIFO order. If, in addition, the arrival process is Poisson (and some technical assumptions are satisfied [19, 202], e.g. in an M/PH/1/FCFS queue), the r-th moment of response time, $\mathsf{E}[T^r]$, is given simply as a function of the first r moments of the number of jobs in the system:

$$
\mathsf{E}[T^r] = \frac{1}{\lambda^r} \mathsf{E}[N(N-1)\cdots(N-r+1)]
$$

for $r \geq 2$. For more general arrival processes, see [19, 202]. For some queueing systems including M/PH/k queues ($k \geq 2$), where jobs are served in FCFS order but not necessarily completed in FIFO order, the distributional Little's law does not apply. In these cases, higher moments of response time may be computed via an analysis of passage times in a QBD process. We will elaborate on this method in Section 3.8.

3.3 State of the art in the analysis of multidimensional Markov chains

As multidimensional Markov chains are prevalent in the analysis of multiserver systems, several approaches have been proposed for their analysis. A most popular approach is to simply truncate the state space of the Markov chain and analyze the resulting Markov chain, for example, via matrix analytic methods. By contrast, DR reduces a multidimensional Markov chain into a 1D Markov chain, but the state space is not simply truncated. Rather, the infinite portion of the state space is aggregated into a smaller space, capturing detailed behavior (such as the first three moments and correlations of the sojourn time distributions in the infinite portion) of the original multidimensional Markov chain. There are also approaches that can be applied to multidimensional Markov chains without state space truncation; however, these approaches are limited in the class of Markov chains to which they can be applied either due to essential restrictions or due to computational complexity. We will see that these approaches have limitations in the analysis of the RFB and GFB processes.

3.3.1 Approaches using matrix analytic methods

As we have seen in Section 3.2, matrix analytic methods allow an efficient evaluation of QBD processes with a finite number of phases (1D Markov chain), but they can also be applied to more general processes including M/G/1 and G/M/1 type processes [111, 136]. The M/G/1 (respectively, G/M/1) type process generalizes the QBD process, and it allows transitions from level ℓ to any levels $\geq \ell - 1$ (respectively, $\leq \ell + 1$) for each ℓ. The M/G/1 and G/M/1 type processes can be analyzed efficiently only when they have a finite number of phases (1D Markov chain). The tree process is a multidimensional Markov chain that can be analyzed efficiently via matrix analytic methods, but the class of tree processes is rather limited; specifically, the RFB and GFB processes do not appear to be modeled as a tree process.

A popular approach in applying matrix analytic methods to multidimensional Markov chains is to simply truncate the state space so that the resulting process becomes a 1D Markov chain (QBD process with a finite number of phases) [61, 93, 94, 112, 138, 160, 185, 186]. For example, Green [61] and Stanford and Grassmann [185, 186] analyze cycle stealing without switching cost (see Section 3.4) by truncating the state space. Likewise, Kao and Narayanan [94], and Ngo and Lee [138] analyze the preemptive priority queue with two classes (see Section 3.4) by truncating the state space. Kao and Narayanan [93], Kao and Wilson [95], and Leemans [112] analyze the nonpreemptive priority queue (see Section 3.4) by truncating the state space. Finally, Rao and Posner [160] analyze the 2D Markov chain for the coupled processor model[4] by truncating the state space. In general, the errors that are introduced by ignoring portions of the state space with an infinite number of states can be significant, especially at higher traffic intensities. Note that each of the above systems analyzed in [61, 93, 94, 112, 138, 185, 186] by truncating the state space can be modeled as RFB or GFB process (see Section 3.4), and hence can be analyzed via DR; the coupled processor model that is analyzed in [160] by truncating the state space does not appear to be modeled as RFB or GFB process.

Since simple truncation of the state space can lead to erroneous conclusions [108], recently various truncation/aggregation approaches have been proposed by Green [60], Heindl and Telek [73], and Heindl, Zhang, and Smirni [74]. All of these works propose approaches to decompose tandem queues by approximating the departure process from a queue, which can in general be modeled exactly as a MAP (as defined in Section 2.2) with an infinite number of phases, by a MAP with a finite number of phases. The truncation/aggregation approaches proposed in [60, 73, 74] are specialized for tandem queues or networks of queues, and it is not clear how they are applied to RFB/GFB processes; on the other hand, tandem queues do not appear to be modeled as RFB/GFB processes. Hence, DR is complementary to the truncation/aggregation approaches proposed in [60, 73, 74].

3.3.2 Other approaches

Other analytic solution methods for multidimensional Markov chains can be classified into two approaches: the direct approach and the approach via generating functions. The direct approach solves the equilibrium equations directly (without transforming them). This includes power series methods, product form methods, and compensation methods. On the other hand, the approach via generating

[4] The coupled processor model is related to cycle stealing without switching cost (Figure 3.1(a)). In this model two processors each serve their own class of jobs, and if either is idle it may help the other, increasing the rate of the other processor. This help incurs no switching time and has a benefit even if only a single job is present (i.e. two processors can work on the single job).

functions solves the functional equation for the generating function of the stationary probabilities. This includes uniformization methods and boundary value methods.

Power series methods express the stationary probability as a power series of a certain parameter such as system load [21, 75, 104]. Power series methods can, in theory, be applied to *any* Markov chain. However, the application of power series methods is often limited to simple Markov chains on low dimensional state spaces due to its computational complexity. For example, Kao and Wilson [95] analyze the nonpreemptive multiserver priority queue with two priority classes by power series methods. They report that power series methods experience more inaccuracy as compared to matrix analytic methods with state space truncation. Note that the nonpreemptive multiserver priority queue with two priority classes can be modeled as a GFB process (see Section 3.4), and hence can be analyzed via DR.

Product form methods express the stationary probability as a product of stationary probabilities for respective dimensions (see e.g. [16, 97, 195]). Product form methods have been developed mainly for the analysis of networks of queues, and the class of Markov chains that have product form solutions is limited. In general, the RFB and GFB processes do not appear to have product form solutions.

Compensation methods express the stationary probability as a sum of an infinite number of product forms [1, 2, 3]. Compensation methods apply to a large class of Markov chains on a two dimensional grid of the first (positive) quadrant. However, an essential limitation of compensation methods is that transitions to the "north," "north-east," and "east" are prohibited. Therefore, compensation methods do not apply to RFB and GFB processes in general. It is possible to extend compensation methods to higher dimensions, but the limitation becomes more severe in higher dimensions [196].

Approaches via generating functions solve the functional equation for the generating function of the stationary probabilities by uniformization [43, 100] or by reducing the functional equation to a boundary value problem [44, 42]. Although approaches via generating functions apply to a broad class of Markov chains on a two dimensional state space, they do not appear to be applicable to the case of higher dimensions. For example, the FB process can, in theory, be reduced to a boundary value problem if the foreground process repeats after a certain level. However, an RFB process with $m > 2$ processes does not appear to be solvable via generating functions. Also, approaches via generating functions often experience numerical instability. For example, Fayolle and Iasnogorodski [51] and Konheim, Meilijson and Melkman [103] reduce the 2D Markov chain for the coupled processor models to (Dirichlet or Riemann-Hilbert) boundary value problems, assuming that service demands have exponential distributions. Due to complexity in the expressions, they were not evaluated in either work. (DR does not appear to be applicable to the 2D Markov chain for the coupled processor model, either.)

In the case of an M/M/k queue with two priority classes (preemptive or nonpreemptive), exact solution methods have been proposed utilizing the spacial structure of this model [127]; for example, in the case of a preemptive priority queue, there is no service completion of lower priority jobs when there are $\geq k$ high priority jobs (there are no backward transitions in the foreground process while the background process is in levels $\geq k$; see Figure 3.1(b)). However, these approaches do not apply to the RFB and GFB processes in general. Very recently, these exact solution methods are extended to non-exponential service times by Sleptchenko [177] (not yet published) and Sleptchenko et. al. [178] (not yet published). The Markov chain is analyzed via a combination of generating functions and matrix analytic methods. In theory, their technique can be generalized to PH distributions, though they evaluate only hyperexponential distributions due to the increased complexity when using more general PH distributions.

3.4 FB, RFB, and GFB processes

In this section, we define the foreground-background (FB) process, the recursive FB (RFB) process, and the generalized FB (GFB) process, and provide examples of these processes. In the rest of this chapter, we denote a matrix by a bold face letter such as $\mathbf{X}$ and its (i,j) element by $(\mathbf{X})_{i,j}$, and we use $\vec{x}$ to denote a vector and $(\vec{x})_i$ to denote its i-th element. Also, we use matrix $\mathbf{Q}_Y$ to denote the generator matrix of a QBD process "characterized by parameter Y." Here, Y may be a single letter or number, denoting process Y or the Y-th QBD process, or Y may be a pair of numbers (i,j), denoting the i-th QBD process of type j. Unless otherwise stated, we express $\mathbf{Q}_Y$ using submatrices, $\mathbf{L}_Y^{(h)}$, $\mathbf{F}_Y^{(h)}$, and $\mathbf{B}_Y^{(h)}$, such that

$$\mathbf{Q}_Y = \begin{pmatrix} \mathbf{L}_Y^{(0)} & \mathbf{F}_Y^{(0)} & & & \\ \mathbf{B}_Y^{(1)} & \mathbf{L}_Y^{(1)} & \mathbf{F}_Y^{(1)} & & \\ & \mathbf{B}_Y^{(2)} & \mathbf{L}_Y^{(2)} & \mathbf{F}_Y^{(2)} & \\ & & \ddots & \ddots & \ddots \end{pmatrix},$$

where $\mathbf{L}_Y^{(h)}$ encodes (local) transitions within level h, $\mathbf{F}_Y^{(h)}$ encodes (forward) transitions from level h to level $h+1$, and $\mathbf{B}_Y^{(h)}$ encodes (backward) transitions from level h to level $h-1$, for each h (see Section 3.2).

3.4.1 Definition of FB process

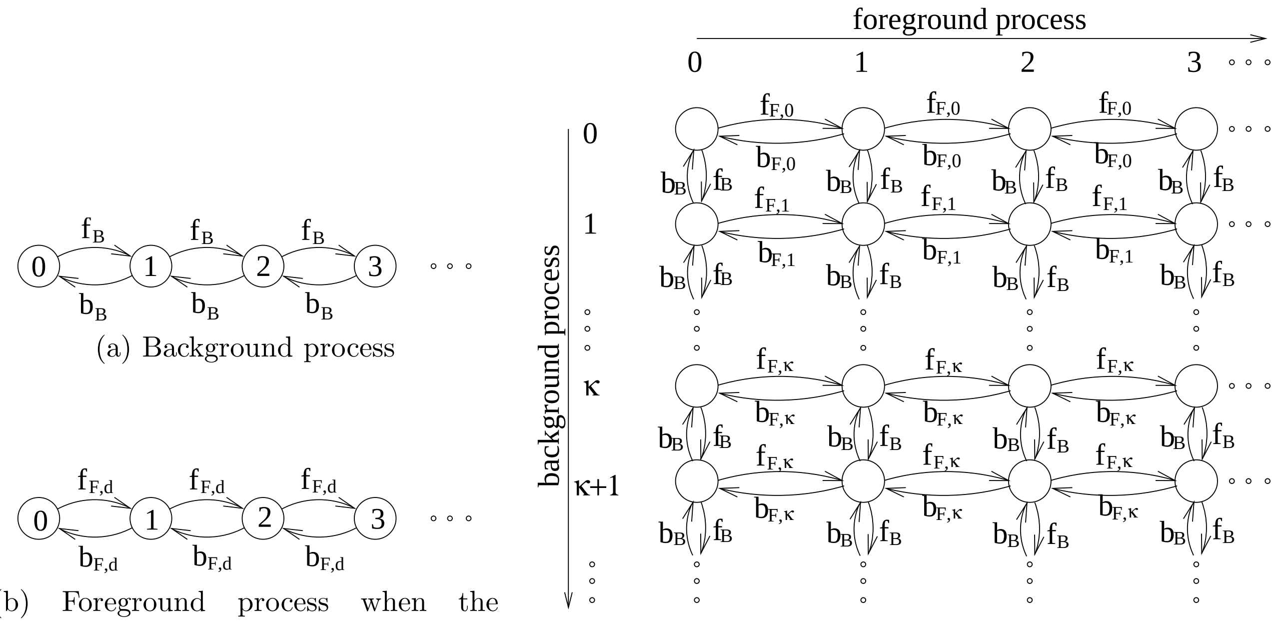
Intuitively, an FB process consists of a background (QBD) process and a foreground (QBD) process, where the behavior (infinitesimal generator) of the foreground process can depend on the level of the background process. Additionally, we require that there exists a level, κ, in the background process such that the infinitesimal generator of the foreground process does not change while the background process is in levels $\geq \kappa$.

Consider a simpler case where the foreground and background processes are homogeneous birth-and-death processes. The background process, B, has a fixed generator matrix, $\mathbf{Q}_B$ (see Figure 3.13(a)):

$$\mathbf{Q}_B = \begin{pmatrix} -f_B & f_B & & \\ b_B & -(b_B+f_B) & f_B & \\ & b_B & -(b_B+f_B) & \ddots \\ & & \ddots & \ddots \end{pmatrix}.$$

On the other hand, the generator matrix of the foreground process, F, depends on the level of the background process. That is, when process B is in level d (i.e., when the state of B is d), the process F evolves according to the generator matrix $\mathbf{Q}_{F,d}$ (see Figure 3.13(b)):

$$\mathbf{Q}_{F,d} = \begin{pmatrix} -f_{F,d} & f_{F,d} & & \\ b_{F,d} & -(b_{F,d}+f_{F,d}) & f_{F,d} & \\ & b_{F,d} & -(b_{F,d}+f_{F,d}) & \ddots \\ & & \ddots & \ddots \end{pmatrix}.$$



(a) Background process

(b) Foreground process when the background process is in level d

(c) FB process

Figure 3.13: *FB process consisting of a foreground birth-and-death process and a background birth-and-death process.*

The FB process assumes that there exists a level, κ, of the background process such that $\mathbf{Q}_{F,d} = \mathbf{Q}_{F,\kappa}$ for all $d > \kappa$. Figure 3.13(c) shows the FB process consisting of the foreground and background processes in Figures 3.13(a)-(b).

Recall the 2D Markov chains shown in Figure 3.1. These Markov chains are FB processes. In Figure 3.1(a), the background process tracks the number of donor jobs, and the foreground process tracks the number of beneficiary jobs. In Figure 3.1(b), the background process tracks the number of high priority jobs, and the foreground process tracks the number of low priority jobs. Also, recall the 2D Markov chain shown in Figure 3.4(c). This 2D Markov chain is an FB process, where the background process is a QBD process (and not a birth-and-death process). Again, the background process tracks the number of high priority jobs, and the foreground process tracks the number of low priority jobs.

In general, an FB process is defined by a vector of generator matrices $(\mathbf{Q}_B, (\mathbf{Q}_{F,0}, ..., \mathbf{Q}_{F,\kappa}))$. Here, $\mathbf{Q}_{F,d}$ denotes the generator matrix of the foreground process when the background process is in level d. We require that $\mathbf{Q}_B$ and $\mathbf{Q}_{F,d}$ satisfy the following characteristics:

- $\mathbf{Q}_B$ is a generator matrix for a QBD process; i.e., the background process is a QBD process.
- $\mathbf{Q}_{F,d} = \mathbf{Q}_{F,\kappa}$ for all $d > \kappa$; i.e., the infinitesimal generator of the foreground process stays the same while the background process is in levels $\geq \kappa$.
- $\mathbf{Q}_{F,d}$ is a generator matrix for a QBD process for $0 \leq d \leq \kappa$; i.e., given the level of the background process, the foreground process is a QBD process.
- The order of the submatrix $\mathbf{L}_{F,d}^{(\ell)}$ is the same for all d, for each ℓ; i.e., the state space of the foreground process is fixed.

3.4.2 Definition of RFB process

The RFB process generalizes the FB process to higher dimension. Intuitively, an RFB process consists of m (QBD) processes, where the infinitesimal generator of the i-th process can depend on the level of the $(i-1)$-th process for $2 \le i \le m$. Additionally, we require that there exists a level, κ_{i-1}, in the $(i-1)$-th process such that the generator matrix of the i-th process does not change while the $(i-1)$-th process is in levels $\ge \kappa_{i-1}$.

More formally, an RFB process consisting of m processes is characterized by parameters

$$(\mathbf{Q}_1, (\mathbf{Q}_{2,0}, ..., \mathbf{Q}_{2,\kappa_1}), (\mathbf{Q}_{3,0}, ..., \mathbf{Q}_{3,\kappa_2}), ..., (\mathbf{Q}_{m,0}, ..., \mathbf{Q}_{m,\kappa_{m-1}})).$$

Here, $\mathbf{Q}_{i,d}$ denotes the generator matrix of the i-th process when the $(i-1)$-th process is in level d. We require that $\mathbf{Q}_m$ satisfy the following characteristic:

- $\mathbf{Q}_1$ is a generator matrix for a QBD process; i.e., the first process is a QBD process.

Also, we require that $\mathbf{Q}_{i,d}$ satisfy the following characteristics for $2 \le i \le m$:

- $\mathbf{Q}_{i,d} = \mathbf{Q}_{i,\kappa_{i-1}}$ for all $d > \kappa_{i-1}$; i.e., the infinitesimal generator of the i-th process stays the same while the $(i-1)$-th process is in levels $\ge \kappa_{i-1}$.
- $\mathbf{Q}_{i,d}$ is a generator matrix for a QBD process for $0 \le d \le \kappa_{i-1}$; given the level of the $(i-1)$-th process, the i-th process is a QBD process.
- The order of the submatrix $\mathbf{L}_{i,d}^{(\ell)}$ is the same for all d, for each ℓ; i.e., the state space of the i-th process is fixed.

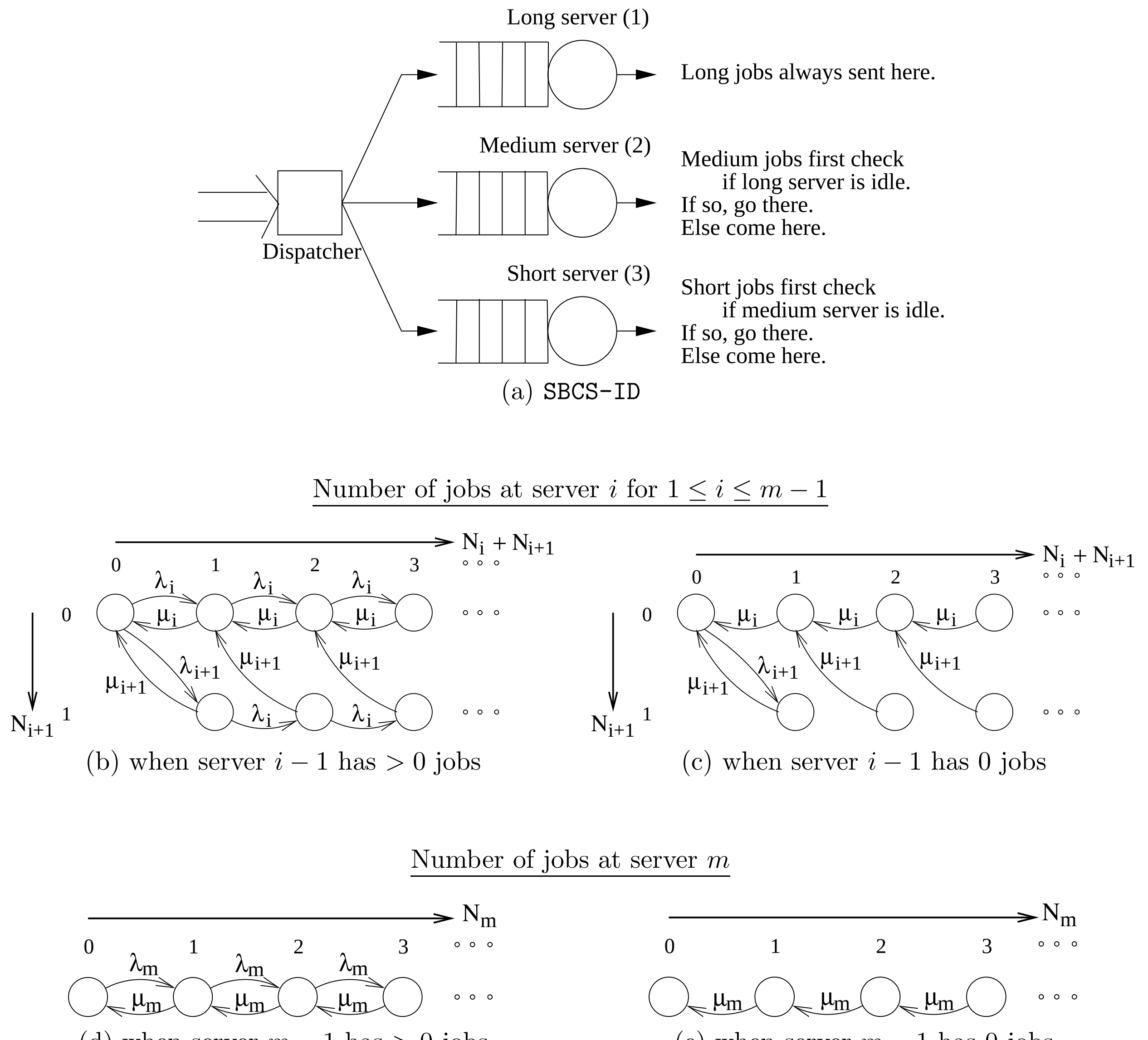
3.4.3 Examples of RFB processes

Below, we provide illustrative examples of interesting multiserver systems whose behavior is captured by RFB processes. In all the examples, we assume that arrival processes of jobs are Poisson processes, and service demand distributions are exponential distributions. However, these can be generalized to MAPs and PH distributions, respectively (and these systems can still be modeled as RFB processes).

Size-based task assignment with cycle stealing under immediate dispatching

We consider a task assignment policy in a server farm with a dispatcher (see Figure 3.14(a)), where jobs at each server are served in first-come-first-serve order. Consider m homogeneous servers and m classes of jobs. Class i jobs arrive at a dispatcher according to a Poisson process with rate λ_i and have an exponential service time distribution with rate μ_i ($\mu_i < \mu_j$ if $i < j$) for $1 \le i \le m$. We will show in Chapter 5 that the size-based task assignment with cycle stealing under immediate dispatching, `SBCS-ID`, provides lower mean response time than common assignment policies. Under `SBCS-ID`, class 1 jobs (largest jobs) are always dispatched to server 1. For $i \ge 2$, a class i job first checks to see if the $(i-1)$-th server is idle. If so, the class i job is dispatched to the $(i-1)$-th server; otherwise, it is dispatched to the i-th server. Observe that the arrival process (of class i jobs) at server i depends on whether server $i-1$ is idle or not. Below, we will model the system with `SBCS-ID` as an RFB process.

Figures 3.14(b)-(e) show the Markov chains for the number of jobs at server i, conditioned on the number of jobs at server $i-1$. Specifically, Figure 3.14(b) shows the Markov chain for the number



(a) SBCS-ID

(b) when server $i-1$ has > 0 jobs

(c) when server $i-1$ has 0 jobs

(d) when server $m-1$ has > 0 jobs

(e) when server $m-1$ has 0 jobs

Figure 3.14: *An example of an RFB process: Size-based task assignment with cycle stealing under immediate dispatching. (a) The* `SBCS-ID` *policy. (b)-(e) Markov chains for the number of jobs at each server, where N_j denotes the number of class j jobs at the server. The middle row shows the Markov chains for server $1 \le i \le m-1$, (b) when server $i-1$ has > 0 jobs ("server 0" is defined to have > 0 jobs always), and (c) when server $i-1$ has 0 jobs. The bottom row shows the Markov chains for server m, (d) when server $m-1$ has > 0 jobs, and (e) when server $m-1$ has 0 jobs.*

of jobs at server i when server $i-1$ has > 0 jobs, for $1 \le i \le m-1$ ("server 0" is defined to have > 0 jobs always). Since server $i-1$ has > 0 jobs, an arrival of a class i job is dispatched to server i. Figure 3.14(c) shows the Markov chain for the number of jobs at server i when server $i-1$ has 0 jobs, for $2 \le i \le m-1$. Since server $i-1$ has 0 jobs, an arrival of a class i job is dispatched to server $i-1$; hence there is no transition due to a class i job arrival. Note that there is only one Markov chain for server 1 (Figure 3.14(b)), since the behavior (arrival and service) at server 1 is independent of the number of jobs at the other servers. Figures 3.14(d)-(e) show the Markov chains for the number of jobs at server m when server $m-1$ has > 0 jobs and when server $m-1$ has 0 jobs, respectively. Since "class $m+1$" does not exist, Figures 3.14(d)-(e) have no transitions corresponding to λ_{m+1} and μ_{m+1}.

Now, it is easy to model the number of jobs in the system under SBCS as an RFB process. Here, the Markov chain at server i, Q_i, corresponds to the i-th process for $1 \le i \le m$. Since Q_1 is a finite-phase QBD process that does not depend on other processes, it can be seen as the first process. There are two forms of infinitesimal generators for Q_2, depending on the number of jobs at server 1, i.e. depending on the level of Q_1. Thus, Q_2 can be seen as the second process. Likewise, Q_i can be seen as the i-th process, since its infinitesimal generator is determined by the level of Q_{i-1} for $2 \le i \le m$. Note that the infinitesimal generator of Q_i is fixed while Q_{i-1} is in levels ≥ 1 for all i (i.e., $\kappa_1 = \cdots = \kappa_{m-1} = 1$).

Preemptive priority queue

Consider an M/M/k queue with m priority classes, where class i jobs have preemptive priority over class j jobs for $1 \le i < j \le m$ (i.e. class 1 has the highest priority). The behavior of class 1 jobs is not affected by the jobs of other classes. However, the behavior of class 2 jobs depends on the number of class 1 jobs in the system. Specifically, when there are n jobs of class 1, $\max\{k-n, 0\}$ servers are available for class 2 jobs. Likewise, the behavior of class i jobs depends on the total number of higher priority (classes $\le i-1$) jobs for $2 \le i \le m$ (when there are n higher priority jobs, $\max\{k-n, 0\}$ servers are available for class i jobs). We will study the performance of the preemptive priority queue in Chapter 4. Below, we will model the preemptive priority queue as an RFB process.

As we have seen in Section 3.1, when $m = 2$, the number of jobs in a priority M/M/k queue can be modeled as an FB process where the background process, B, captures the number of class 1 jobs (high priority jobs), and the foreground process, F, captures the number of class 2 jobs (low priority jobs). Here, the level ℓ of B corresponds to the state with ℓ high priority jobs. Note that the number of servers available for low priority jobs (and hence the behavior of F) is determined by the number of high priority jobs (equivalently, the level of B).

When $m > 2$, the number of jobs in a priority M/M/k queue can be modeled as an "extended" RFB process having m (QBD) processes. In the "extended" RFB process, the transitions in the i-th process depend on the level of a process, $\widehat{Q}_{i-1}$, that is equivalent to the $(i-1)$-th process, Q_{i-1} (see Figure 3.15(a)). Here, the only difference between $\widehat{Q}_{i-1}$ and Q_{i-1} is how levels are defined in these two processes. Specifically, the level of Q_{i-1} corresponds to the number of class $i-1$ jobs, while the level of $\widehat{Q}_{i-1}$ corresponds to the *total* number of higher priority (classes $\le i-1$) jobs.

Figures 3.15(b)-(c) show two equivalent QBD processes with different definitions of levels. Figure 3.15(b) is the 1D Markov chain that is reduced from a 2D Markov chain in the analysis of an M/M/2 queue with two priority classes (high priority and *middle* priority) via DR. (Recall the dimensionality reduction from Figure 3.3(c) to Figure 3.3(d); the low priority jobs are replaced by the

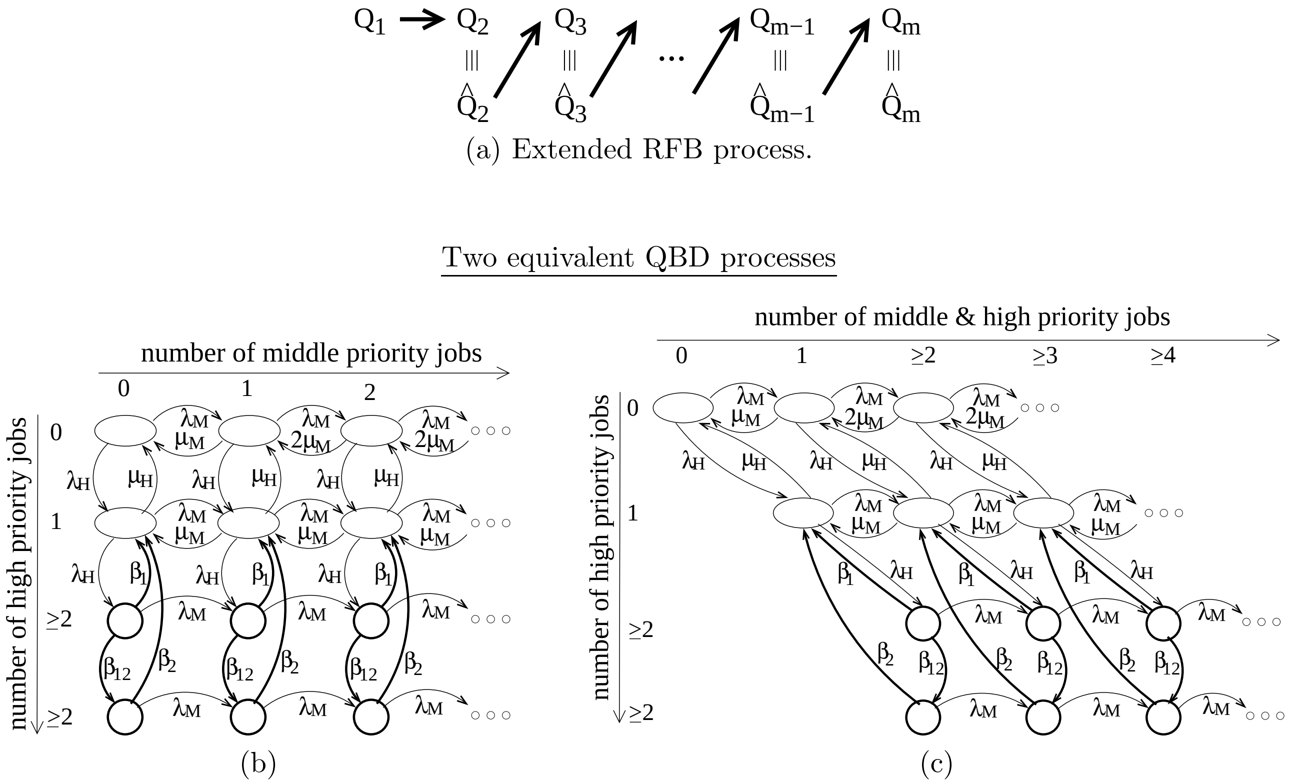


Figure 3.15: *Ideas in the RFB process. (a) Relationships among m processes in the extended RFB process. (b)-(c) Two equivalent QBD processes.*

middle priority jobs in Figure 3.15(b).) In the QBD process of Figure 3.15(b), level ℓ consists of the states with ℓ middle priority jobs. Observe that the QBD process in Figure 3.15(c) is exactly the same as the QBD process in Figure 3.15(b) except that they have different layout of states. In the QBD process of Figure 3.15(b), level 0 and level 1 consist of the states with 0 and 1 *total* jobs in the system, respectively. For $\ell > 1$, level ℓ consists of states with $\geq \ell$ *total* jobs in the system. Since the bottom two rows have states with ≥ 2 high priority jobs, the exact number of high priority jobs is not known (and hence the total number of jobs is not known, either). However, note that the QBD process in Figure 3.15(c) is exactly what we need in the analysis of *low* priority jobs in an M/M/2 queue with three priority classes (high, middle, and low). The behavior of the low priority jobs is determined by whether there are 0, 1, or ≥ 2 higher priority (high or middle priority) jobs in the system, and hence by the level of the QBD process in Figure 3.15(c).

In the general case of an M/M/k queue with m priority classes, it is intuitively easy to see that there is a QBD process $\widehat{Q}_{i-1}$ that is equivalent to Q_{i-1} such that the level of $\widehat{Q}_{i-1}$ corresponds to the total number of higher priority jobs in the system, since any event (job arrival or completion) changes the total number of jobs (and hence the level of $\widehat{Q}_{i-1}$) by 1. Since all the transitions are between neighboring levels, the process is a QBD process. For completeness, Figure 3.16 shows the Markov chains for the number of class i jobs conditioned on the number of higher priority (classes $\leq i-1$) jobs in the system.

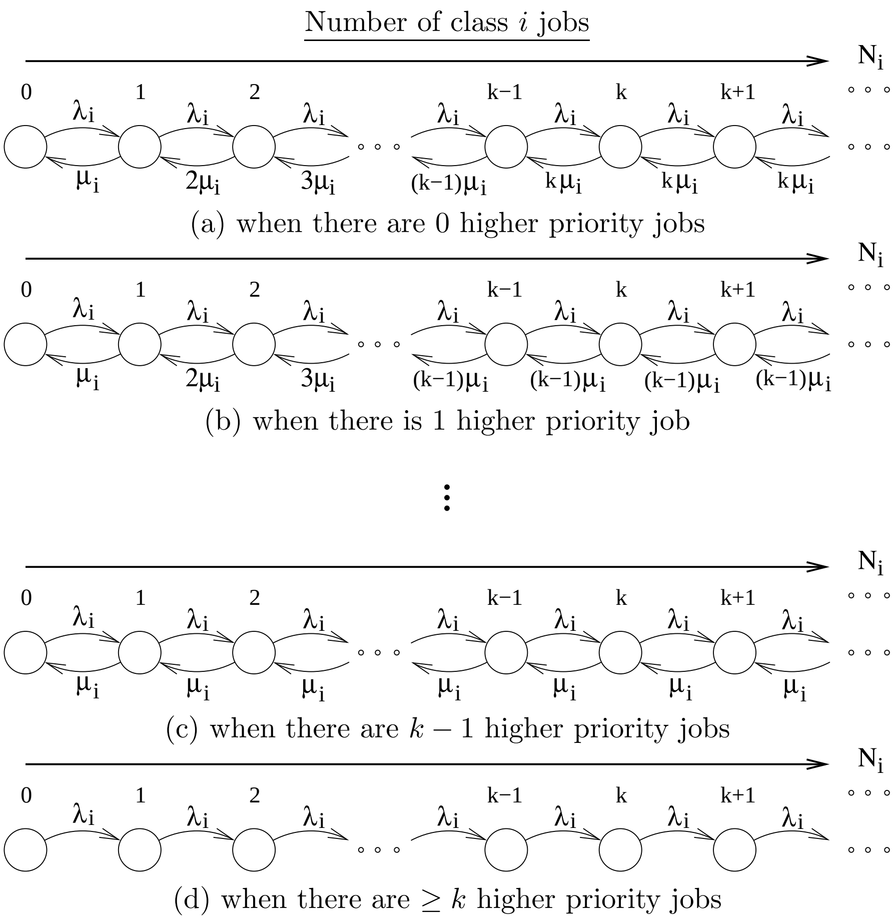


Figure 3.16: *An example of an RFB process: Preemptive priority queue. Figures show Markov chains for the number of jobs of class i (a) when there are* 0 *higher priority jobs, (b) when there are* 1 *higher priority jobs, (c) when there are* $k-1$ *higher priority jobs, and (d) when there are* $\geq k$ *higher priority jobs. Here,* N_i *denotes the number of class i jobs in the system.*

3.4.4 Definition of GFB process

The GFB process is a generalization of the FB process, and is a 2D Markov chain having less restrictions on its structure than the FB process. Specifically, we allow the background and foreground process to have complex interactions while the *background* process is in levels $< \kappa$ (and as a result, the GFB process behaves like a QBD process when the background process is in levels $< \kappa$), but the GFB process behaves like an FB process when the background process is in levels $\geq \kappa$ (see Figure 3.17). Recall that, in the FB process, the background process is a *fixed* QBD process, and only the behavior of the foreground process is affected by the level of the background process while the background process is in levels $< \kappa$.

Essential restrictions in the GFB process are that

- There exists a level, κ, in the *background* process such that each of the background and foreground processes is a fixed QBD process while the *background* process is in levels $\geq \kappa$.

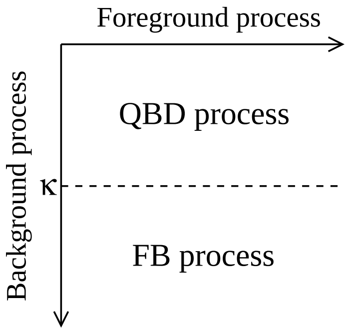


Figure 3.17: *The structure of the GFB process.*

- There is no transition from level $< \kappa$ to level $> \kappa$ (i.e., any transition from level $< \kappa$ to level $\geq \kappa$ is to level κ) in the *background* process.

Given a 2D Markov chain, if one can identify the background and foreground process that constitute the 2D Markov chain and that satisfy the above restrictions, the 2D Markov chain can be reduced to a 1D Markov chain via DR. However, the resulting 1D Markov chain may not be analyzed efficiently. Thus, additionally, we require that

- Any transition in the 2D Markov chain changes the level of the *foreground* process by at most one.

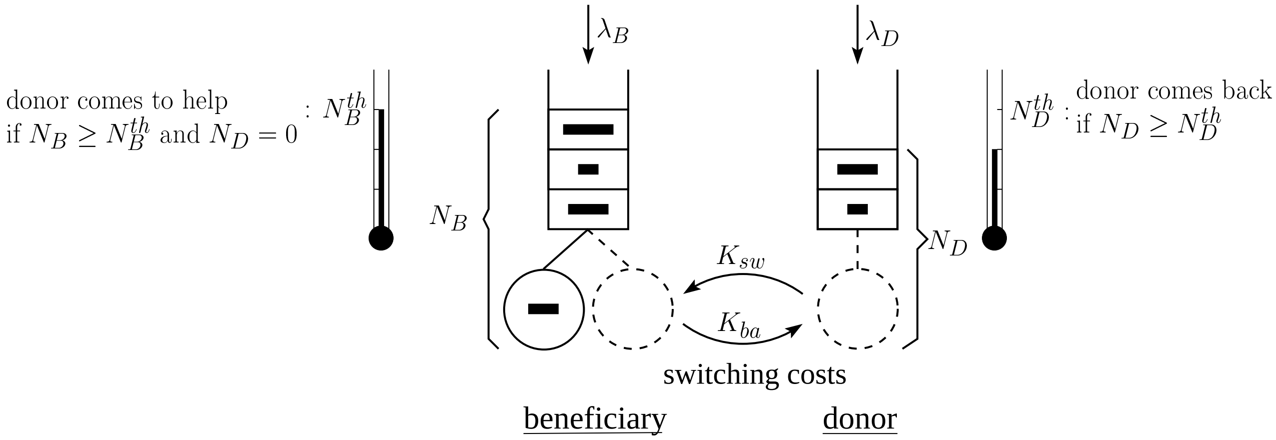
If a 2D Markov chain satisfy the above two requirements, the 2D Markov chain can be reduced to a 1D Markov chain via DR, and the 1D Markov chain is a QBD process. We call such a 2D Markov chain a GFB process.

3.4.5 Examples of GFB processes

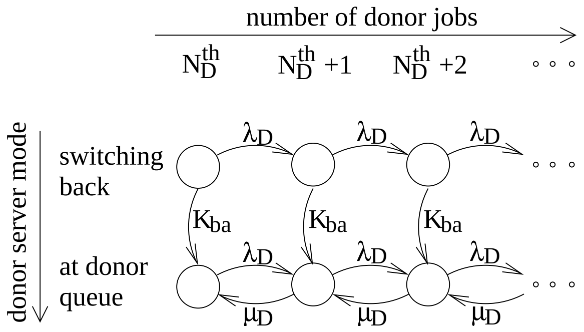
Below, we provide illustrative examples of interesting multiserver systems whose behavior is captured by GFB processes. In all the examples, we assume that arrival processes of jobs are Poisson processes, and service demand distributions are exponential distributions. However, these can be generalized to MAPs and PH distributions, respectively (and these systems can still be modeled as GFB processes).

Threshold-based policy for reducing switching costs in cycle stealing

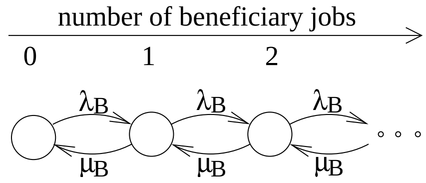
We consider two processors, each serving its own M/M/1 queue, where one of the processors (the donor) can help the other processor (the beneficiary) while the donor's queue is empty (see Figure 3.18(a)). There is a switching time, K_{sw}, required for the donor to start working on the beneficiary's jobs, as well as a switching back time, K_{ba}. We assume that K_{sw} and K_{ba} have exponential distributions (but this can be generalized to PH distributions). Due to non-zero switching times, the donor's switching may pay only when the beneficiary's queue length, N_B, is sufficiently long, and the donor's switching back may pay only when the donor's queue length, N_D, is sufficiently long. Thus, we consider the following threshold-based policy. If $N_B \geq N_B^{th}$ and $N_D = 0$, the donor starts switching to the beneficiary's queue. After K_{sw} time, the donor is available to work on the beneficiary's jobs. When N_D reaches N_D^{th}, the donor starts switching back to its own queue. After K_{ba} time, the donor resumes working on its own jobs. We will study the performance of the threshold-based policy



(a) Threshold-based policy for reducing switching costs in cycle stealing



(b) number of donor jobs when $N_D \geq N_D^{th}$



(c) number of beneficiary jobs when $N_D \geq N_D^{th}$

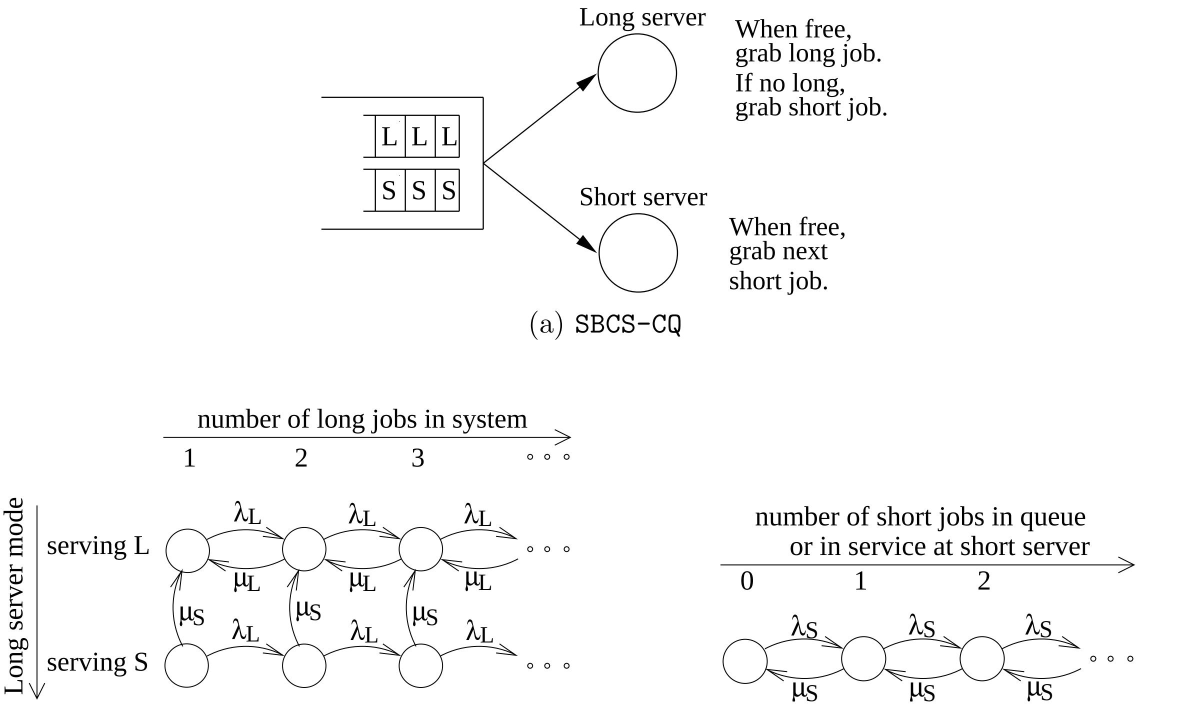
Figure 3.18: *An example of an GFB process: Threshold-based policy for reducing switching costs in cycle stealing. (a) Threshold-based policy for reducing switching costs in cycle stealing. (b) Markov chain for the number of donor jobs, N_D, (background process) when $N_D \geq N_D^{th}$. (c) Markov chain for the number of beneficiary jobs (foreground process) when $N_D \geq N_D^{th}$.*

for reducing switching costs in cycle stealing in Chapter 6. Below, we will model the system with this threshold-based policy as a GFB process.

The number of jobs in the above system can be modeled as a GFB process[5]. Here, the background process captures the number of the donor's jobs and the mode of the donor (whether it is at its own queue, is switching to the beneficiary's queue, is at the beneficiary's queue, or is switching back to its own queue), and the foreground process captures the number of the beneficiary's jobs. Specifically, level ℓ in the background process (respectively, foreground process) corresponds to the number of donor jobs (respectively, beneficiary jobs) in the system.

As long as $N_D \geq N_D^{th}$, the foreground and background processes can be modeled as QBD processes, respectively. Figure 3.18(b) shows the background process when $N_D \geq N_D^{th}$. Since $N_D \geq N_D^{th}$,

[5]In [151, 152], we analyze this system by reducing the 2D Markov chain into a 1D Markov chain but without modeling it as a GFB process. The approach in [151, 152] requires identifying and analyzing multiple types of busy periods for this particular system.



(a) SBCS-CQ

(b) number of long jobs when $N_L \geq 1$

(c) number of short jobs when $N_L \geq 1$

Figure 3.19: *An example of a GFB process: Size-based task assignment with cycle stealing under central queue. (a) The* SBCS-CQ *policy. (b) Markov chain for the number of long jobs in the queue or in service, N_L, (background process) when $N_L \geq 1$. (c) Markov chain for the number of short jobs in the queue or in service at the short server (foreground process) when $N_L \geq 1$.*

the donor server is either switching back to its own queue (top row) or serving donor jobs (bottom row). If the donor server is switching back, the donor jobs do not complete. After K_{ba} time, switching back is completed. While the donor server is serving donor jobs, the donor jobs are completed with rate μ_D. Figure 3.18(c) shows the foreground process given $N_D \geq N_D^{th}$. Since the donor server is not helping, the beneficiary jobs complete with rate μ_B. Observe that both Markov chains in Figures 3.18(b)-(c) are QBD processes. Also, any transition from level $< N_D^{th}$ to level $\geq N_D^{th}$ is (from level $N_D^{th} - 1$) to level N_D^{th} in the background process, since N_D is never incremented more than one at a time. Further, any transition changes the level of the foreground process by at most one, since any event (job arrival, job completion, and changes in the donor server mode) changes the number of beneficiary jobs by at most one. Thus, these foreground and background processes constitute a GFB process.

Size-based task assignment with cycle stealing under central queue

We consider size-based task assignment with cycle stealing under central queue (SBCS-CQ; see Figure 3.19(a)), where short jobs and long jobs arrive at the central queue according to Poisson processes with rate λ_S and λ_L, respectively. The service demand of the short jobs (respectively, long jobs) has an exponential distribution with rate μ_S (respectively, μ_L). We assume that jobs are nonpreemptive (i.e., once a job receives service, it runs to completion). To prevent short jobs from waiting behind long jobs, we designate one server as the short server, and the other as the long server. Whenever the short server becomes idle, it picks the first short job in the queue to run. Whenever the long server becomes idle, it picks the first long job in the queue. However, if there is no long job, the long server picks the first short job in the queue.

The number of jobs in the above system can be modeled as a GFB process[6]. Here, we define the background process so that it tracks the number of long jobs waiting in the queue or in service at the long server, N_L, as well as whether the long server is serving a long job, short job, or none. Specifically, level ℓ in the background process corresponds to N_L. Also, we define the foreground process so that level ℓ in the foreground process corresponds to the the number of short jobs waiting in the queue or in service at the short server, N_S. Note that whether there is a short job in service at the long server is captured in the background process and not in the foreground process.

As long as $N_L \geq 1$, the foreground and background processes can be modeled as QBD processes, respectively. Figure 3.19(b) shows the background process when $N_L \geq 1$. It tracks N_L as well as whether the long server is working on a long job (top row) or a short job (bottom row). If the long server is serving a short job, the short job is completed with rate μ_S, but no long jobs are completed. If the long server is serving long jobs, long jobs are completed with rate μ_L. Figure 3.19(c) shows the foreground process given $N_L \geq 1$. Since the long server is busy, it does not pick short jobs in the queue, but short jobs are completed with rate μ_S by the short server. Observe that both Markov chains in Figures 3.18(b)-(c) are QBD processes. Also, any transition from level 0 to level ≥ 1 is to level 1 in the background process, since N_L is never incremented more than one at a time. Further, any transition changes the level of the foreground process by at most one, since any event (job arrival, job completion, and changes in the long server mode) changes the number of short jobs by at most one. Thus, these foreground and background processes constitute a GFB process.

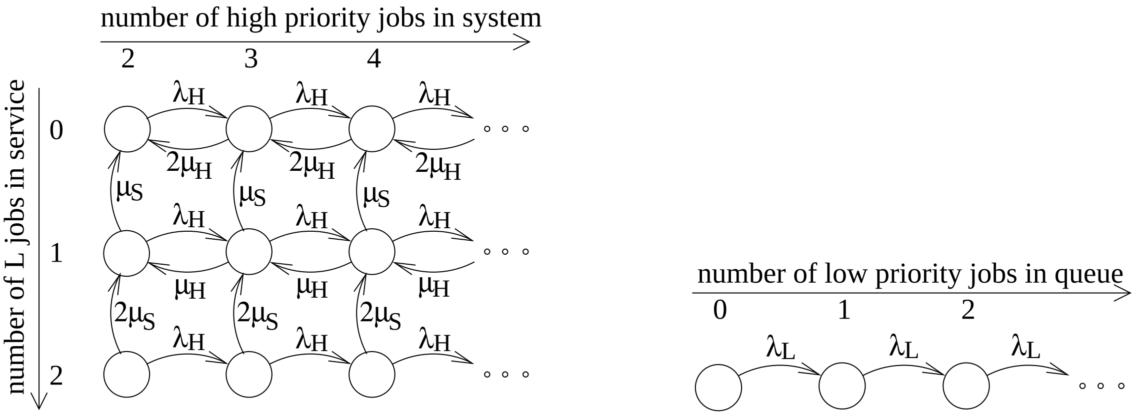
Nonpreemptive priority queue

Consider an M/M/k queue with two priority classes (high and low), where high priority jobs have *nonpreemptive* priority over low priority jobs. That is, a high priority job can move ahead of all the low priority jobs waiting in the queue, but low priority jobs in service are not interrupted by high priority jobs.

The number of jobs in this system can be modeled as a GFB process. Here, we define the background process so that it tracks the number of high priority jobs waiting in the queue or in service, N_H, as well as the number of low priority jobs in service. Specifically, level ℓ in the background process consists of states with $N_H = \ell$. Also, we define the foreground process so that it tracks the number of low priority jobs waiting in the queue, N_S. Specifically, level ℓ in the foreground process is the state with $N_S = \ell$. Note that the number of low priority jobs in service is captured in the background process and not in the foreground process.

As long as $N_H \geq k$ (there are enough high priority jobs to occupy all the servers), the foreground and background processes can be modeled as QBD processes, respectively. Figure 3.19 shows a part of the foreground and background processes when the number of servers $k = 2$. Figure 3.19(a) shows the background process when $N_H \geq 1$. It tracks N_H as well as the number of low priority jobs in service. When there are n low priority jobs in service, $2 - n$ servers are serving high priority jobs,

[6] In [67], we analyze the SBCS-CQ with an additional technique to reduce mean response time, renaming of servers (see Chapter 5). With renaming, the SBCS-CQ does not appear to be modeled as a GFB process, and in [67] we introduce an approximation in the analysis via DR.



(a) number of high priority jobs, N_H, when $N_H \geq 2$ (b) number of low priority jobs when $N_H \geq 2$

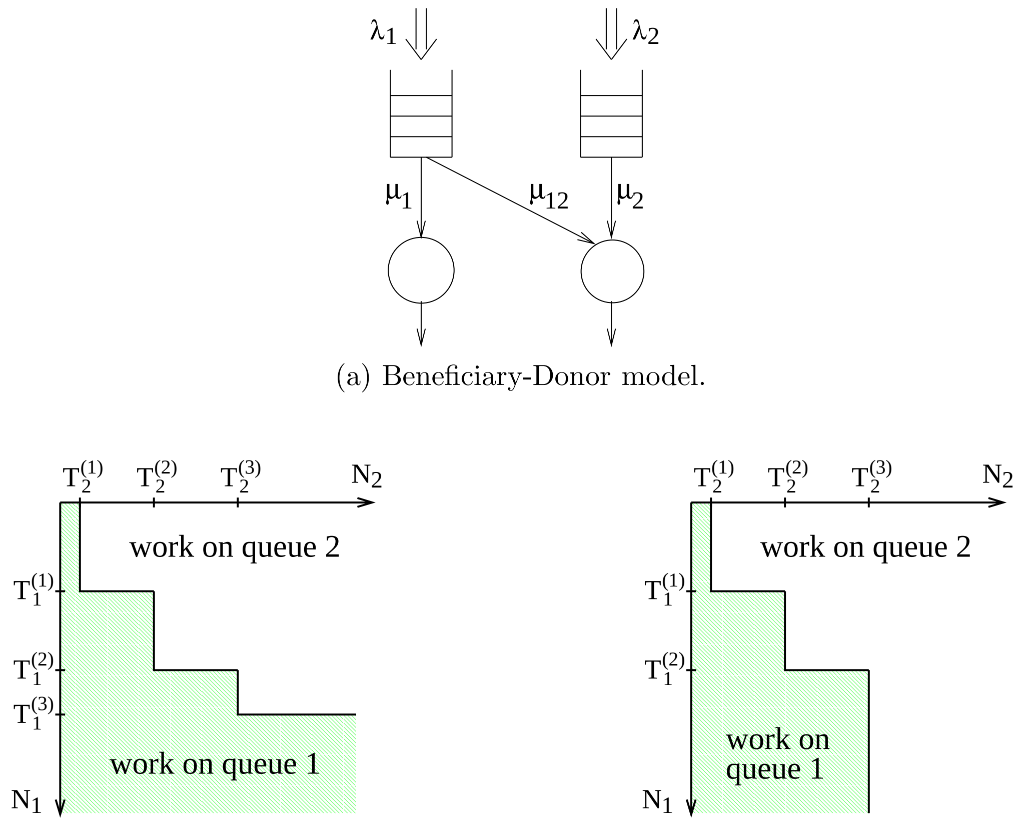
Figure 3.20: *An example of a GFB process: Nonpreemptive priority queue. (a) Markov chain for the number of high priority jobs, N_H, (background process) when $N_H \geq 2$. (b) Markov chain for the number of low priority jobs in the queue (foreground process) when $N_H \geq 2$.*

and high priority jobs are completed with rate $(2-n)\mu_H$ for $n = 0, 1, 2$. Figure 3.19(b) shows the foreground process given $N_H \geq 1$. Since all the servers are occupied, no low priority jobs waiting in the queue start receiving service, and the number of low priority jobs in the queue increases with rate λ_L. Observe that both Markov chains in Figures 3.18(a)-(b) are QBD processes. Also, any transition from level $< k$ to level $\geq k$ is (from level $k-1$) to level k in the background process, since N_H is never incremented more than one at a time. Further, any transition changes the level of the foreground process by at most one, since any event (job arrival, or job completion) changes the number of low priority jobs in the queue by at most one. Thus, these foreground and background processes constitute a GFB process.

Threshold-based policies in Beneficiary-Donor model

We consider a broad family of threshold-based (resource allocation) policies in the Beneficiary-Donor model, as shown in Figure 3.21(a). Here, jobs arrive at queue 1 and queue 2 according to Poisson processes with rate λ_1 and λ_2, respectively, and their service demands have exponential distributions. Server 1 processes only jobs in queue 1 with rate μ_1. Server 2 can process either jobs in queue 1 with rate μ_{12} or jobs in queue 2 with rate μ_2, and which jobs are processed by server 2 is determined by the queue lengths (the length of queue 1, N_1, and the length of queue 2, N_2) and threshold values. Figures 3.21(b)-(c) show examples of such threshold-based policies. We assume that there are a finite number of thresholds, and the rules are enforced preemptively. That is, there exists a threshold on queue 1, $\kappa \equiv T_1^{(i)}$, such that server 2 processes jobs in queue 1 whenever $N_1 \geq \kappa$ (type 1 threshold-based policy, as shown in Figure 3.21(b)), or there exists a threshold on queue 2, $\kappa \equiv T_2^{(i)}$, such that server 2 processes jobs in queue 2 whenever $N_2 \geq \kappa$ (type 2 threshold-based policy, as shown in Figure 3.21(c)). We will study the performance of these threshold-based policies in Chapter 7. Below, we model the Beneficiary-Donor model with the threshold-based policies as GFB processes.

The number of jobs in the system under the threshold-based policies (both type 1 and type 2) can be modeled as a GFB process. For the type 1 threshold-based policy, we define the background



(a) Beneficiary-Donor model.

(b) type 1 threshold-based policy

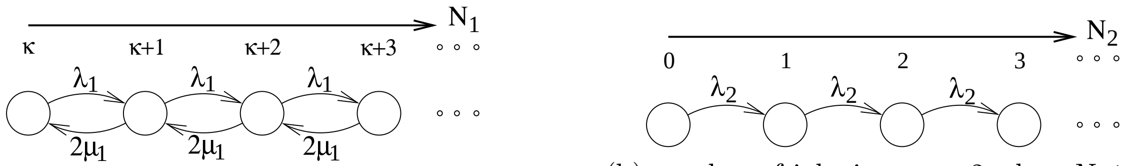
(c) type 2 threshold-based policy

Figure 3.21: *Threshold-based policies for the Beneficiary-Donor model. (a) The Beneficiary-Donor model. (b)-(c) show examples of threshold-based policies for the Beneficiary-Donor model: whether server 2 works on jobs from queue 1 or queue 2 is shown as a function of the length of queue 1,* N_1*, and the length of queue 2,* N_2*.*

process so that it tracks the number of jobs in queue 1, including the jobs in service, N_1; we define the foreground process so that it tracks the number of jobs in queue 2, including the job in service, N_2. Specifically, level ℓ in the background process (respectively, foreground process) corresponds to the state with $N_1 = \ell$ (respectively, $N_2 = \ell$). Likewise, for the type 2 threshold-based policy, we define the background process as tracking N_2 and the foreground process as tracking N_1.

Now, it is easy to see that these foreground and background processes constitute a GFB process. For the type 1 threshold-based policy, as long as $N_1 \geq \kappa$, the foreground and background processes can be modeled as birth-and-death processes, respectively. Figure 3.22(a) shows the background process when $N_1 \geq \kappa$, and Figure 3.22(b) shows the foreground process given $N_1 \geq \kappa$. Since $N_1 \geq \kappa$, server 2 processes jobs in queue 1, and jobs are completed with rate $2\mu_1$ from queue 1 (background process), while there is no job completion from queue 2 (foreground process). Also, any transition from level $< \kappa$ to level $\geq \kappa$ is (from level $\kappa - 1$) to level κ in the background process, since N_1 is never incremented more than one at a time. Further, any transition changes the level of the foreground process by at most one, since any event (job arrival, or job completion) changes N_2 by at most one. Thus, these foreground and background processes constitute a GFB process. Likewise, for the

Type 1 threshold-based policy



(a) number of jobs in queue 1, N_1, when $N_1 \geq \kappa$ (b) number of jobs in queue 2 when $N_1 \geq \kappa$

Type 2 threshold-based policy

(c) number of jobs in queue 2, N_2, when $N_2 \geq \kappa$ (d) number of jobs in queue 1 when $N_2 \geq \kappa$

Figure 3.22: *Examples of GFB processes: Threshold-based policies for the Beneficiary-Donor model. The top row illustrates the GFB process for the type 1 threshold-based policy. (a) Markov chain for the number of jobs in queue 1, N_1, (background process) when $N_1 \geq \kappa$. (b) Markov chain for the number of jobs in queue 2, N_2 (foreground process) when $N_1 \geq \kappa$. The bottom row illustrates the GFB process for the type 2 threshold-based policy. (c) Markov chain for N_2, (background process) when $N_2 \geq \kappa$. (d) Markov chain for N_1 (foreground process) when $N_2 \geq \kappa$.*

type 2 threshold-based policy, as long as $N_2 \geq \kappa$, the foreground and background processes can be modeled as birth-and-death processes, respectively, as shown in Figures 3.22(c)-(d). Again, since any event changes N_1 and N_2 by at most one, these foreground and background processes constitute a GFB process.

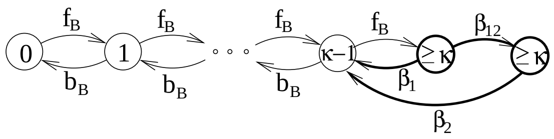
3.5 Dimensionality reduction

In this section, we analyze the stationary probabilities in the RFB and GFB processes via DR. The stationary probabilities can be used to derive other performance measures such as the mean response time, as discussed in Section 3.8. In Section 3.5.1, we start by analyzing a simple FB process whose background and foreground processes are homogeneous birth-and-death processes. In Section 3.5.2, we analyze the FB process, which constitutes the primary part of the analysis of the RFB and GFB processes. In Section 3.5.3, we analyze the RFB process by applying the analysis of the FB process recursively. In Section 3.5.4, we analyze the GFB process using the analysis in Section 3.5.2.

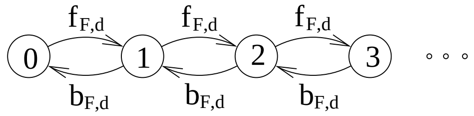
3.5.1 Analysis of the birth-and-death FB process

In this section, we analyze a simplest FB process shown in Figure 3.13. Our goal is to reduce the FB process (2D Markov chain) to a QBD process with a finite number of phases (1D Markov chain) which closely approximates the FB process.

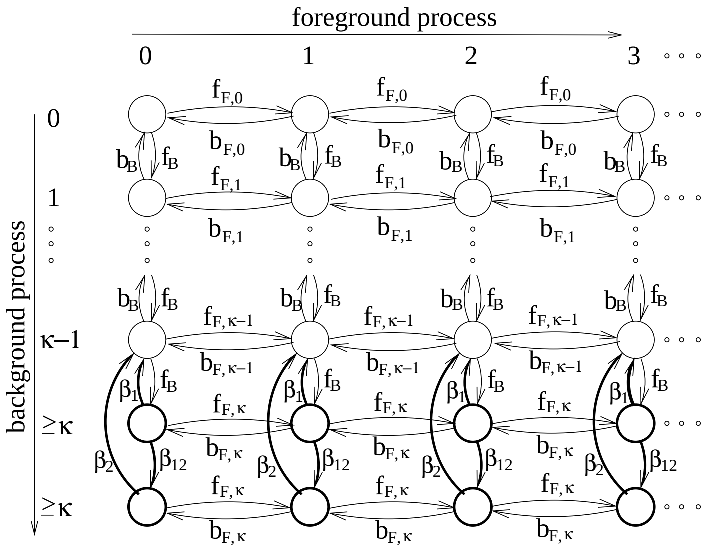
Observe that the infinite number of *phases* in the FB process stems from the infinite number



(a) Approximate background process



(b) Foreground process when the background process is in level d



(c) 1D FB process

Figure 3.23: *An analysis of the FB process in Figure 3.13 via DR. (a) The background process in Figure 3.13(a) is approximated by a Markov chain on a finite state space. (c) The FB process in Figure 3.13(c) is approximated by a 1D Markov chain.*

of states in the background process, B (Figure 3.13(a)). This motivates us to approximate B by a Markov chain with a finite number of states, $\widetilde{B}$ (Figure 3.23(a)). In process $\widetilde{B}$, all the states in levels $\geq \kappa$ of process B is aggregated into two states labeled $\geq \kappa$. That is, the sojourn time in levels $\geq \kappa$, T, is approximated by a two-phase PH distribution (as defined in Section 2.2) with parameters:

$$\overrightarrow{\tau} = (1,0) \quad \text{and} \quad \mathbf{T} = \begin{pmatrix} -(\beta_1+\beta_{12}) & \beta_{12} \\ 0 & -\beta_2 \end{pmatrix}.$$

We use the moment matching algorithm developed in Chapter 2 to choose the parameters of the PH distribution so that the first three moments of T are matched. The sojourn time T is exactly the same as the busy period in an M/M/1 queue where the arrival rate is f_B and the service rate is b_B. Thus, the first three moments of T are

$$\mathsf{E}[T] = \frac{1}{1-\rho}\frac{1}{b_B}, \quad \mathsf{E}\left[T^2\right] = \frac{2}{(1-\rho)^2}\frac{1}{b_B^2}, \quad \mathsf{E}\left[T^3\right] = \frac{6(1+\rho)}{(1-\rho)^5}\frac{1}{b_B^3},$$

where $\rho = \frac{f_B}{b_B}$. The following theorem implies that two phases are sufficient to match these three moments by a PH distribution.

Theorem 8 *Let T denote the duration of an M/GI/1 busy period where service demand G has an arbitrary distribution with finite third moment and where the job starting the busy period has size K whose distribution is in $\mathcal{T}^{(n)}$ (recall Definition 7 in Section 2.1). Then, the distribution of T is in $\mathcal{T}^{(n)}$.*

A proof of Theorem 8 is postponed to Appendix A. Theorem 8 is useful in determining the sufficient number of phases in a PH distribution to well-represent a busy period duration without explicitly computing the moments of the busy period.

By replacing the background process B by $\widetilde{B}$, the 2D Markov chain (Figure 3.13(c)) is reduced to a 1D Markov chain shown in Figure 3.23(c). More formally, the 1D Markov chain is a QBD process, and its generator matrix, $\mathbf{Q}$, is obtained via the generator matrix of $\widetilde{B}$, $\mathbf{Q}_{\widetilde{B}}$, as follows. First, observe that

$$\mathbf{Q}_{\widetilde{B}} = \begin{pmatrix} -f_B & f_B & & & \\ b_B & -(b_B+f_B) & \ddots & & \\ & \ddots & \ddots & f_B & \\ & & b_B & -(b_B+f_B) & f_B\overrightarrow{\tau} \\ & & & \overrightarrow{t} & \mathbf{T} \end{pmatrix},$$

where $\overrightarrow{t} = -\mathbf{T}\overrightarrow{1}$. Generator matrix $\mathbf{Q}$ is then given by

$$\mathbf{Q} = \begin{pmatrix} \mathbf{L}^{(0)} & \mathbf{F} & & \\ \mathbf{B} & \mathbf{L} & \mathbf{F} & \\ & \mathbf{B} & \mathbf{L} & \ddots \\ & & \ddots & \ddots \end{pmatrix},$$

where $\mathbf{F}$ and $\mathbf{B}$ are $(\kappa+2)\times(\kappa+2)$ diagonal matrices,

$$\mathbf{F} = \begin{pmatrix} f_{F,0} & & & & \\ & \ddots & & & \\ & & f_{F,\kappa-1} & & \\ & & & f_{F,\kappa} & \\ & & & & f_{F,\kappa} \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} b_{F,0} & & & & \\ & \ddots & & & \\ & & b_{F,\kappa-1} & & \\ & & & b_{F,\kappa} & \\ & & & & b_{F,\kappa} \end{pmatrix},$$

$\mathbf{L}^{(0)} = \mathbf{Q}_{\widetilde{B}} - \mathbf{F}$, and $\mathbf{L} = \mathbf{Q}_{\widetilde{B}} - \mathbf{F} - \mathbf{B}$.

In general, the stationary probabilities in the 1D Markov chain can immediately be translated into the stationary probabilities in the *foreground* process. On the other hand, the stationary probabilities in the background process needs to be analyzed independently, since its state space is aggregated in the 1D Markov chain. Observe, however, that the background process is easy to analyze, since it is simply a birth-and-death process without dependency on the foreground process.

3.5.2 Analysis of the FB process

In this section, we extend the analysis in Section 3.5.1 to the general FB process. As in Section 3.5.1, once the background process is approximated by a Markov chain on a finite state space, it is straightforward to reduce the FB process to a 1D Markov chain. Hence, in this section, we limit our focus on approximating the background process by a Markov chain on a finite state space; for completeness, we formally describe the construction of the 1D Markov chain using the approximated background process in Appendix 3.5.5. We will see that, in general, a *collection* of PH distributions is needed in the approximate background process, $\widetilde{B}$, (e.g. recall Section 3.1.2) to capture the dependency in the sequence of sojourn times in levels $\geq \kappa$ that B has. This is in contrast to the case when the background process is a birth-and-death process, which required only a *single* PH distribution in the approximate background process, since the sojourn times in levels $\geq \kappa$ are i.i.d. in this case. Below, we use $\mathbf{Q}_b$ to denote the generator matrix of B, and use $\mathbf{B}_b^{(\ell)}$, $\mathbf{L}_b^{(\ell)}$, and $\mathbf{F}_b^{(\ell)}$ to denote the submatrices of $\mathbf{Q}_b$, following the convention introduced in Section 3.4

We start by specifying the properties that $\widetilde{B}$ should have. Let $E_{i,j}$ be the event that the first state that we visit in level $\kappa-1$ is in phase j given that we transitioned from phase i in level $\kappa-1$ to any state in level κ. (Notice that in the analysis of the M/PH/2 queue with two priority classes in Section 3.1, each event $E_{i,j}$ corresponds to the event that the phase of the job in service immediately before a busy period starts is i and the phase of the job in service immediately after the busy period is j.) We require that $\widetilde{B}$ has the following key properties:

- The probability of event $E_{i,j}$ in $\widetilde{B}$ is the same as that in B for each i and j.
- The distribution of the sojourn time in levels $\geq \kappa$ *given event* $E_{i,j}$ in $\widetilde{B}$ well-represents that in B (i.e. the first three moments of the two distributions agree) for each i and j.

Observe that if the second property did not involve the approximation, then the foreground process that depends on background process B and the foreground process that depends on background process $\widetilde{B}$ would be stochastically equivalent. This is because the sequence of the sojourn times between changing levels (0, 1, ..., $\kappa-1$, and $\geq \kappa$) in B and that in $\widetilde{B}$ would be stochastically equivalent.

To construct $\widetilde{B}$, we first analyze the probability of event $E_{i,j}$ and (the moments of) the distribution of the sojourn time in levels $\geq \kappa$ given event $E_{i,j}$ (we denote this distribution as $D_{i,j}$) for each i and j. (Notice that in the analysis of the M/PH/2 queue with two priority classes in Section 3.1, $D_{i,j}$ corresponds to the distribution of the busy period duration given that the phase of the job in service immediately before a busy period starts is i and the phase of the job in service immediately after the busy period is j.) Let ξ be the number of phases in level $\kappa-1$ of B (and $\widetilde{B}$). Then, there are ξ^2 types of events $E_{i,j}$.

The probability of event $E_{i,j}$ is relatively easy to analyze. Let $\mathbf{P}$ be a matrix of order $\xi \times \xi$ whose (i,j) element is the probability of event $E_{i,j}$. Let $E_{i,h}^{(+)}$ be the event that B transitions to state (h,κ) given that the transition is from state $(i,\kappa-1)$ to any state in level κ. Also, let $E_{h,j}^{(-)}$ be the event that state $(j,\kappa-1)$ is the first state that B hits in level $\kappa-1$ given that B starts in state (h,κ). Then,

$$(\mathbf{P})_{i,j} = \sum_h \Pr\left(E_{i,h}^{(+)}\right) \Pr\left(E_{h,j}^{(-)}\right).$$

Thus, all that remains is to derive $\Pr\left(E_{i,h}^{(+)}\right)$ and $\Pr\left(E_{h,j}^{(-)}\right)$. The first quantity is derived via $\mathbf{F}_b^{(\kappa-1)}$ as follows:

$$\Pr\left(E_{i,h}^{(+)}\right) = \frac{(\mathbf{F}_b^{(\kappa-1)})_{i,h}}{\sum_{j=1}^{\xi}(\mathbf{F}_b^{(\kappa-1)})_{i,j}},$$

where $\frac{0}{0}$ is defined as 0. Let $\mathbf{G}^{(\kappa)}$ be a matrix whose (h,j) element is $\Pr\left(E_{h,j}^{(-)}\right)$. Then, matrix $\mathbf{G}^{(\kappa)}$ is obtained by the algorithm in Figure 3.12 (see Section 3.7 for an alternative algorithm and intuition). More formally, let Let $\mathbf{H}^{(\kappa-1)}$ be a matrix whose (i,h) element is $\Pr\left(E_{i,h}^{(+)}\right)$. Then, matrix $\mathbf{P}$ is given by

$$\mathbf{P} = \mathbf{H}^{(\kappa-1)}\mathbf{G}^{(\kappa)}.$$

Next, we analyze the moments of $D_{i,j}$. Let $\mathbf{M}_r$ be a matrix of order $\xi \times \xi$ whose (i,j) element is the r-th moment of $D_{i,j}$ for $r \geq 1$. Note that $D_{i,j}$ is a mixture of distributions of various conditional passage times from level κ to level $\kappa-1$. Specifically, let $T_{i,j}^{(\kappa)}$ be the time to go from state (i,κ) to state $(j,\kappa-1)$. Then, the distribution function of $D_{i,j}$ is given by

$$\sum_h \Pr\left(E_{i,h}^{(+)}\right)\Pr\left(E_{h,j}^{(-)}\right)\Pr\left(T_{h,j}^{(\kappa)}\le x \mid E_{h,j}^{(-)}\right).$$

Now, let $\mathbf{Z}_r^{(\kappa)}$ be a matrix whose (h,j) element is the r-th moment of $T_{h,j}^{(\kappa)}$ given event $E_{h,j}^{(\kappa)}$, i.e.

$$(\mathbf{Z}_r^{(\kappa)})_{h,j}=\int_0^\infty x^r d\Pr\left(T_{h,j}^{(\kappa)}\le x \mid E_{h,j}^{(-)}\right)$$

for $r=1,2,3$. Then,

$$(\mathbf{M}_r)_{i,j}=\sum_h \Pr\left(E_{i,h}^{(+)}\right)\Pr\left(E_{h,j}^{(-)}\right)(\mathbf{Z}_r^{(\kappa)})_{h,j}.$$

More formally, let $\mathbf{G}_r$ be a matrix whose (h,j) element is

$$(\mathbf{G}_r)_{h,j}=\Pr\left(E_{h,j}^{(-)}\right)(\mathbf{Z}_r^{(\kappa)})_{h,j}=(\mathbf{G})_{h,j}(\mathbf{Z}_r^{(\kappa)})_{h,j}.$$

Then,

$$\mathbf{M}=\mathbf{H}\mathbf{G}_r.$$

Thus, all that remains is to derive $\mathbf{Z}_r^{(\kappa)}$. We can calculate $\mathbf{Z}_r^{(\kappa)}$ by a trivial extension of Neuts' algorithm [134] for any r; we postpone this extension to Section 3.7.

We are now ready to construct $\widetilde{B}$. We use the moment matching algorithm developed in Chapter 2 to approximate $D_{i,j}$ by a PH distribution, $(\overrightarrow{\tau_{i,j}},\mathbf{T}_{i,j})$, matching the first three moments of $D_{i,j}$, $((\mathbf{M}_1)_{i,j},(\mathbf{M}_2)_{i,j},(\mathbf{M}_3)_{i,j})$, for all i and j. Let $\overrightarrow{t_{i,j}}=-\mathbf{T}_{i,j}\vec{1}$, where $\vec{1}$ is a column vector with all elements 1 and has order equal to the number of columns in $\mathbf{T}_{i,j}$.

Using $\mathbf{P}$ and $(\overrightarrow{\tau_{i,j}},\mathbf{T}_{i,j})$ defined above, the generator matrix of $\widetilde{B}$ is defined as

$$\mathbf{Q}_{\widetilde{B}}=\left(\begin{array}{cccc|cccc}
\mathbf{L}_b^{(0)} & \mathbf{F}_b^{(0)} & & & & & & \\
\mathbf{B}_b^{(1)} & \ddots & \ddots & & & & & \\
 & \ddots & \ddots & \mathbf{F}_b^{(\kappa-2)} & & & & \\
 & & \mathbf{B}_b^{(\kappa-1)} & \mathbf{L}_b^{(\kappa-1)} & (\vec{\lambda})_1\cdot\boldsymbol{\tau_1} & (\vec{\lambda})_2\cdot\boldsymbol{\tau_2} & \cdots & (\vec{\lambda})_\xi\cdot\boldsymbol{\tau_\xi} \\ \hline
 & & & \mathbf{t}_1 & \mathbf{T}_1 & & & \\
 & & & \mathbf{t}_2 & & \mathbf{T}_2 & & \\
 & & & \vdots & & & \ddots & \\
 & & & \mathbf{t}_\xi & & & & \mathbf{T}_\xi
\end{array}\right),$$

where $\vec{\lambda}=\mathbf{F}_b^{(\kappa-1)}\vec{1}$,

$$\mathbf{T}_i=\begin{pmatrix}\mathbf{T}_{i,1} & & \\ & \ddots & \\ & & \mathbf{T}_{i,\xi}\end{pmatrix},\quad \mathbf{t}_i=\begin{pmatrix}\overrightarrow{t_{i,1}} & & \\ & \ddots & \\ & & \overrightarrow{t_{i,\xi}}\end{pmatrix},\quad \text{and}$$

$$\boldsymbol{\tau_i}=\begin{pmatrix} & \mathbf{O}_{i-1,\xi} & \\ (\mathbf{P})_{i,1}\cdot\overrightarrow{\tau_{i,1}} & \cdots & (\mathbf{P})_{i,\xi}\cdot\overrightarrow{\tau_{i,\xi}} \\ & \mathbf{O}_{\xi-i,\xi} & \end{pmatrix}$$

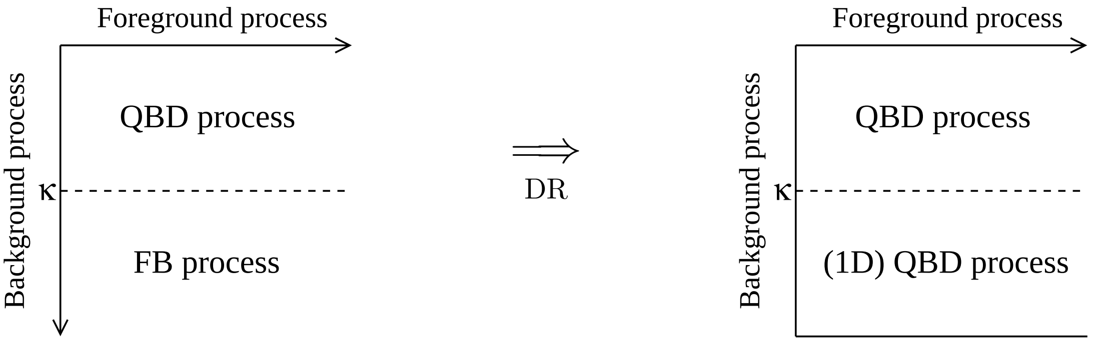


Figure 3.24: *An analysis of the GFB process via DR.*

for $i = 1, ..., \xi$. Here, $\vec{1}$ is a column vector with all elements 1 and has order equal to the number of columns in $\mathbf{F}_b^{(\kappa-1)}$, and $\mathbf{O}_{s,t}$ denotes a zero matrix of order $s \times t$. Recall that ξ is the number of phases in level $\kappa - 1$ of B (and $\widetilde{B}$).

3.5.3 Analysis of the RFB process

The stationary probabilities in the RFB process can be analyzed by applying the analysis in Section 3.5.2 recursively. Recall that, in the RFB process, the infinitesimal generator of the i-th process depend on the level of the $(i-1)$-th process for $2 \le i \le m$. In Section 3.5.2, we analyzed the case of $m = 2$ (the FB process).

We argue, by induction, that all the m processes that constitute the RFB process can be approximated by finite-phase QBD processes (1D Markov chains) via the approach in Section 3.5.2. Then, the stationary probabilities in the m processes can be obtained by analyzing the stationary probabilities in the 1D Markov chains. By our assumption, the first process is a finite-phase QBD process that does not depend on other processes, which proves the base case. Suppose that the i-th process is approximated by a QBD process with a finite number of *phases*, B_i. The QBD process B_i typically has an infinite number of *levels*. However, by the analysis in Section 3.5.2, B_i can be approximated by a QBD process with a finite number of *levels*, $\widetilde{B}_i$, such that B_i and $\widetilde{B}_i$ have stochastically similar effect on the $(i+1)$-th process, B_{i+1}. Now, using $\widetilde{B}_i$, process B_{i+1} can be approximated by a QBD process with a finite number of *phases*. This completes our argument.

3.5.4 Analysis of the GFB process

Most of the techniques necessary in the analysis of the GFB process are already provided in the analysis of the FB process. In this section, we describe how we can apply these techniques to the GFB process, where we use a particular GFB process shown in Figure 3.25 as an example. Intuitively, a GFB process can be reduced to a 1D Markov chain by reducing the FB portion (rows $\ge \kappa$) of the GFB process to a 1D Markov chain via DR as illustrated in Figure 3.24. The stationary probabilities in the 1D Markov chain will be immediately translated into those in the foreground process. We will need an additional technique in the analysis of the background process, since the background process in the GFB process depends on the foreground process, and hence cannot be analyzed independently as in the FB process.

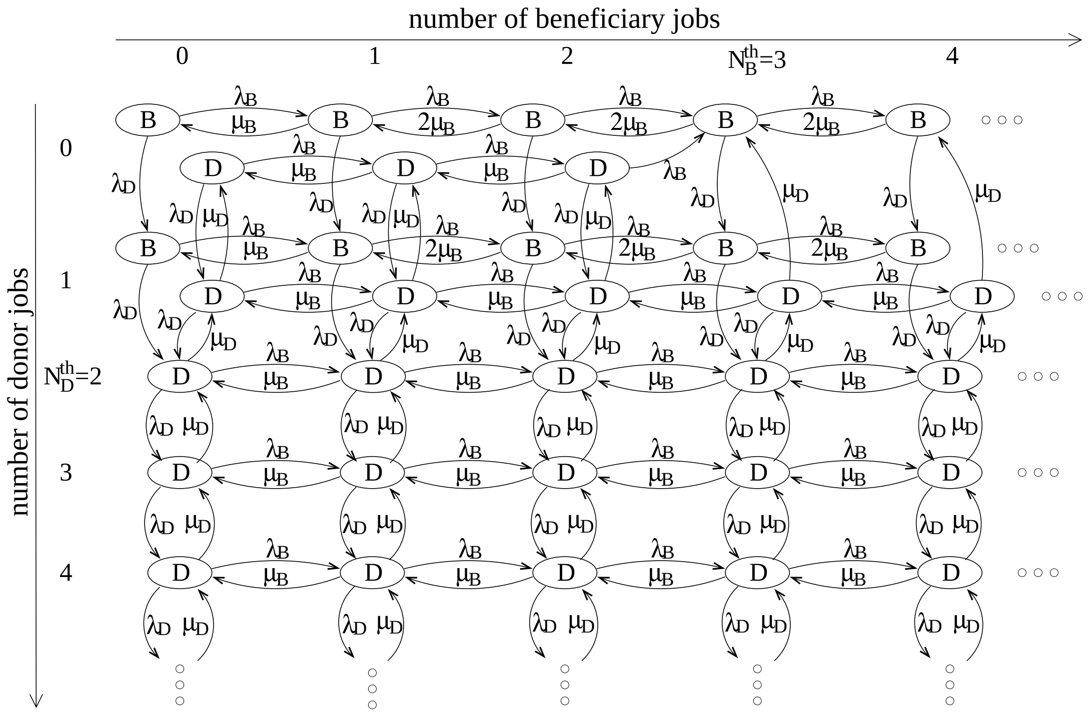


Figure 3.25: *A GFB process: Threshold-based policy for reducing switching costs in cycle stealing, where $N_D^{th} = 2$ and $N_B^{th} = 3$.*

The Markov chain shown in Figure 3.25 models the number of donor and beneficiary jobs, respectively, in the system under the threshold-based policy for reducing switching costs in cycle stealing (recall Figure 3.18(a)), where $N_D^{th} = 2$ and $N_B^{th} = 3$. To simplify the analysis, we assume that switching *times* are zero[7]. In Figure 3.25, the number of donor jobs (background process) is tracked in the vertical direction, and the number of beneficiary jobs (foreground process) is tracked in the horizontal direction. A state labeled D (respectively, B) denotes that the donor server is working or staying idle at the donor's queue (respectively, beneficiary's queue) in the state. Since the background process and the foreground process are independent birth-and-death processes when the number of donor jobs $N_D \geq \kappa \equiv N_D^{th}$, the Markov chain is a GFB process. However, since the background process and the foreground process have complex interaction while $N_D < \kappa$, it is *not* an FB process.

The GFB process can be reduced to a 1D Markov chain by replacing a portion of the background process, as is done in Section 3.5.2. Figure 3.26(a) shows a portion ($N_D \geq \kappa \equiv N_D^{th}$) of the background process. In general, a background process in a GFB process has a level κ such that the background process is a QBD process while it is in levels $\geq \kappa$. By using the techniques that we have introduced in Section 3.5.2, this portion of the background process can be approximated by a Markov chain on a finite state space as in Figure 3.26(b). By replacing levels $\geq \kappa \equiv N_D^{th}$ of the background process (Figure 3.26(a)) by the Markov chain on a finite state space (Figure 3.26(b)), the GFB process is reduced to a 1D Markov chain as shown in Figure 3.26(c).

The 1D Markov chain reduced from a GFB process is a QBD process and its stationary proba-

[7]A motivation for using threshold-based policy under zero switching *times* may be to reduce some "cost," such as money and risk, involved in the switching.

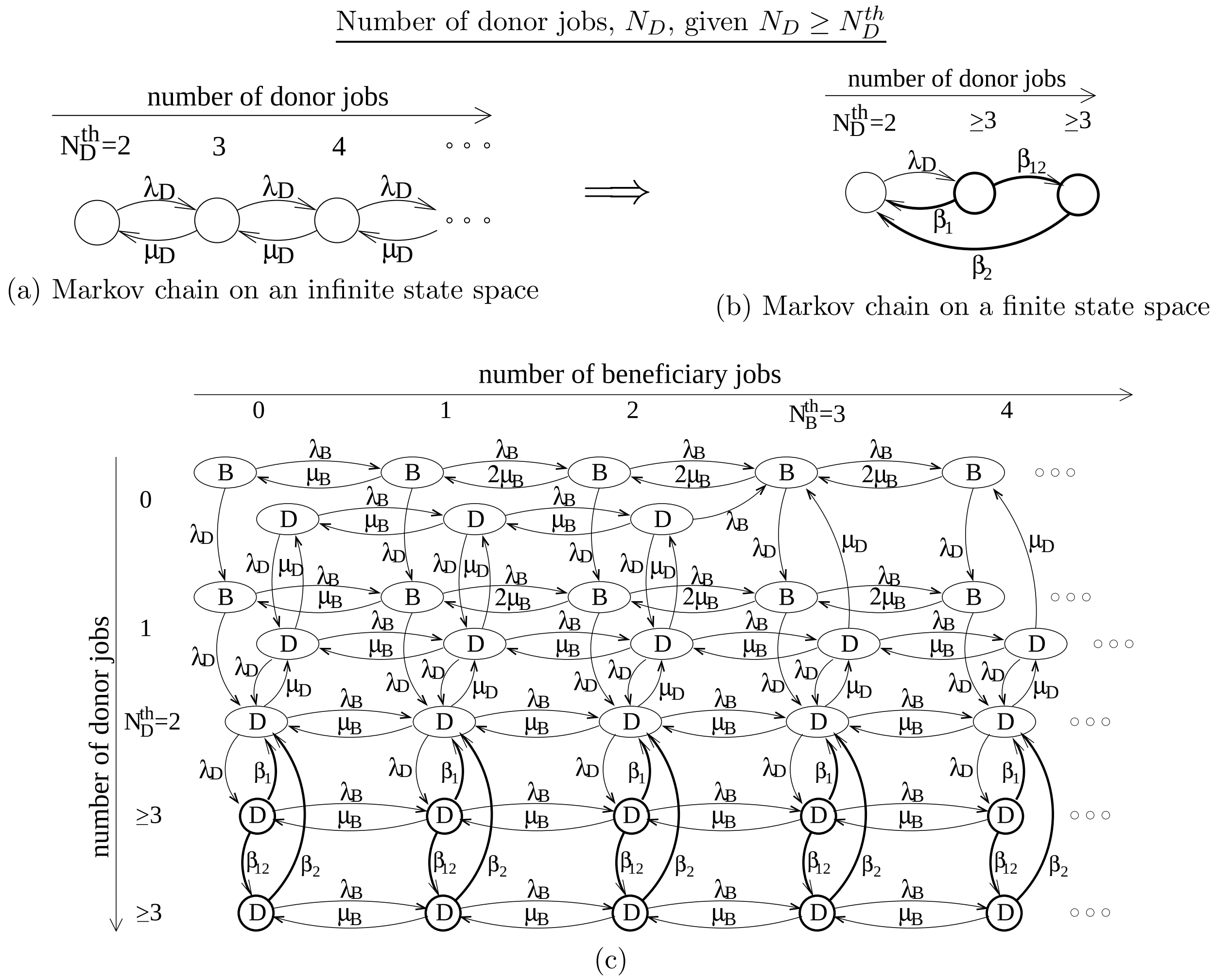


Figure 3.26: *An analysis of the GFB process in Figure 3.25 via DR. (a)-(b) Levels $\geq \kappa$ of the background process. (c) 1D Markov chain reduced from the GFB process.*

bilities can be analyzed via matrix analytic methods as in Section 3.2. As in the analysis of the FB process, the stationary probabilities in the 1D Markov chain can be immediately translated into the stationary probabilities in the foreground process, since the state space of the foreground process is not aggregated in the 1D Markov chain.

By contrast, levels $\geq \kappa + 1$ of the background process are aggregated in the 1D Markov chain; thus, the analysis of the stationary probabilities in the background process requires an additional technique. Since levels $\leq \kappa$ of the background process are exactly tracked in the 1D Markov chain, the stationary probabilities in this portion is given by the stationary probabilities in the 1D Markov chain. Note, in particular, that the probability that there are $N_D = \kappa \equiv N_D^{th}$ donor jobs, π_κ, is given by the stationary probability in the 1D Markov chain. Since the background process is a birth-and-death process when $N_D \geq \kappa \equiv N_D^{th}$ (Figure 3.26(a)), the probability that there are ℓ jobs, π_ℓ, is obtained recursively by

$$\pi_\ell = \rho \pi_{\ell-1},$$

where $\rho = \frac{\lambda_D}{\mu_D}$, for $\ell \geq \kappa + 1$.

In general, the stationary probabilities in levels $\leq \kappa$ of the background process are given by the stationary probabilities in the 1D Markov chain reduced from the GFB process. In particular, the stationary probability vector in level κ, $\overrightarrow{\pi_\kappa}$, is given by analyzing the 1D Markov chain. Then, the stationary probability vector in level $\ell > \kappa$, $\overrightarrow{\pi_\ell}$, is given recursively by

$$\overrightarrow{\pi_\ell} = \overrightarrow{\pi_{\ell-1}}\mathbf{R}^{(\ell)},$$

for $\ell \geq \kappa + 1$, where $\mathbf{R}^{(\ell)}$'s are the R matrices of the background process that can be calculated by the algorithm shown in Figure 3.12.

3.5.5 Constructing a 1D Markov chain using an approximate background process

In Section 3.5.2, we obtain the generator matrix of an approximate background process $\widetilde{B}$, $\mathbf{Q}_{\widetilde{B}}$. In this section, we derive the generator matrix of the 1D Markov chain reduced from the FB process by replacing the background process B by $\widetilde{B}$.

We first introduce notations for the generator matrices of the foreground process conditioned on the level of the background process. Let $\mathbf{Q}_d$ be the generator matrix of the foreground process, F, given that the background process is in level d. As the foreground process is a QBD process given the level of B, $\mathbf{Q}_d$ is of the form

$$\mathbf{Q}_d = \begin{pmatrix} \mathbf{L}_d^{(0)} & \mathbf{F}_d^{(0)} & & \\ \mathbf{B}_d^{(1)} & \mathbf{L}_d^{(1)} & \mathbf{F}_d^{(1)} & \\ & \mathbf{B}_d^{(2)} & \mathbf{L}_d^{(2)} & \ddots \\ & & \ddots & \ddots \end{pmatrix}$$

for $0 \leq d \leq \kappa$. Recall that the generator matrix of the foreground process when the background process is in levels $> \kappa$ is the same as $\mathbf{Q}_\kappa$.

The 1D Markov chain reduced from the FB process is also a QBD process, and its generator matrix, $\mathbf{Q}$, is given by

$$\mathbf{Q} = \begin{pmatrix} \mathbf{L}^{(0)} & \mathbf{F}^{(0)} & & \\ \mathbf{B}^{(1)} & \mathbf{L}^{(1)} & \mathbf{F}^{(1)} & \\ & \mathbf{B}^{(2)} & \mathbf{L}^{(2)} & \ddots \\ & & \ddots & \ddots \end{pmatrix},$$

where

$$\mathbf{F}^{(\ell)} = \begin{pmatrix} \mathbf{I}_0 \otimes \mathbf{F}^{(\ell)} & & & \\ & \ddots & & \\ & & \mathbf{I}_{\kappa-1} \otimes \mathbf{F}_{\kappa-1}^{(\ell)} & \\ & & & \mathbf{I}_\kappa \otimes \mathbf{F}_\kappa^{(\ell)} \end{pmatrix}$$

$$\mathbf{B}^{(\ell)} = \begin{pmatrix} \mathbf{I}_0 \otimes \mathbf{B}^{(\ell)} & & & \\ & \ddots & & \\ & & \mathbf{I}_{\kappa-1} \otimes \mathbf{B}_{\kappa-1}^{(\ell)} & \\ & & & \mathbf{I}_\kappa \otimes \mathbf{B}_\kappa^{(\ell)} \end{pmatrix}$$

$$\mathbf{L}^{(\ell)} = \begin{pmatrix} \mathbf{I}_0 \otimes \mathbf{L}^{(\ell)} & & & \\ & \ddots & & \\ & & \mathbf{I}_{\kappa-1} \otimes \mathbf{L}^{(\ell)}_{\kappa-1} & \\ & & & \mathbf{I}_{\kappa} \otimes \mathbf{L}^{(\ell)}_{\kappa} \end{pmatrix} + \mathbf{Q}_{\widetilde{B}} \otimes \mathbf{I'}_{\ell},$$

for each ℓ, where $\mathbf{I}_i$ is an identity matrix of order equal to the number of states in level ℓ of B for $1 \leq i \leq \kappa - 1$, $\mathbf{I}_\kappa$ is an identity matrix of order equal to the number of states corresponding to the collection of PH distributions in $\widetilde{B}$ (i.e., when the busy period, the sojourn time in levels $\geq \kappa$, is approximated by n PH distributions each with p phases, $\mathbf{I}_\kappa$ is an $np \times np$ identity matrix), and $\mathbf{I'}_\ell$ is the number of states in level ℓ of F for $\ell \geq 0$.

3.6 Approximations of dimensionality reduction

DR as introduced in Section 3.5.2 can still be computationally prohibitive, if it is used recursively as in Section 3.5.3. In this section, we introduce two new approximations in DR, namely DR-PI (Section 3.6.1) and DR-CI (Section 3.6.2), which can significantly reduce the computational complexity while keeping a reasonable accuracy. The key idea in our new approximations is that distributions $D_{i,j}$ (the sojourn time distribution in levels $\geq \kappa$ given event $E_{i,j}$) can be aggregated without losing too much information. In Section 3.6.3, we will discuss the computational complexity of DR, DR-PI, and DR-CI when they are used recursively.

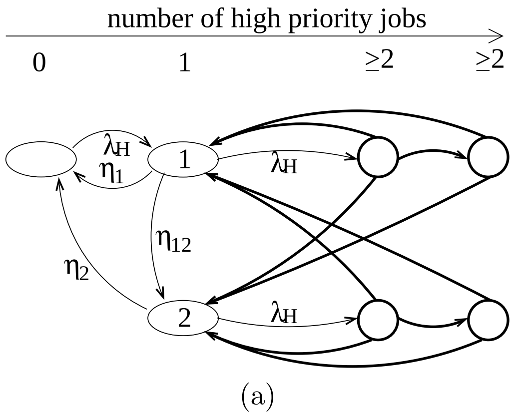
3.6.1 Dimensionality reduction with partial independence assumption

DR-PI (DR with partial independence assumption) ignores the dependency that the sojourn time in levels $\geq \kappa$ has on how the Markov chain *enters* levels $\geq \kappa$ (from level $\kappa - 1$). Specifically, we assume that $D_{i,j}$ is independent of i. Let $D'_j = D_{i,j}$ for all i for each j, and let $\widetilde{B}'$ denote the process that is the same as $\widetilde{B}$ except that $D_{i,j}$ is replaced by D'_j for all i for each j. Figure 3.27(a) shows the Markov chain for the high priority jobs (background process) that we obtain via DR-PI in the analysis of an M/PH/2 queue with two priority classes. (Recall the Markov chain that we obtain via DR, Figure 3.5 in Section 3.1, where the four types of busy periods are approximated by four PH distributions, respectively.) In DR-PI, the four types of busy periods are represented by *two* PH distributions, ignoring the dependency that the duration of the busy period has on the phase of the job in service at the beginning of the busy period.

In general, we require that process $\widetilde{B}'$ has the following two key properties:

- The probability of event $E_{i,j}$ in $\widetilde{B}'$ is the same as that in B (and that in $\widetilde{B}$ as well).
- We choose D'_j such that the *marginal* distribution of the sojourn time in levels $\geq \kappa$ in $\widetilde{B}'$ well-represents that in B (and hence in $\widetilde{B}$ as well).

Hence, $\widetilde{B}'$ and B would have stochastically the same total sojourn time in levels $\geq \kappa$ in the long run if the second property did not involve the approximation (i.e., if the marginal distribution is fitted exactly). However, $\widetilde{B}'$ and B have different autocorrelation in the sequence of the sojourn times in levels $\geq \kappa$.



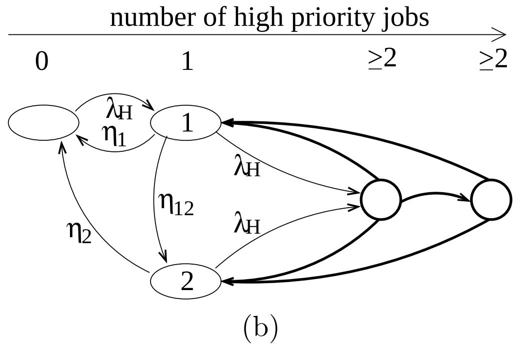


Figure 3.27: *Background processes on a finite state space, obtained via (a) DR-PI and (b) DR-CI. Labels on the transitions in the busy periods are omitted for clarity.*

More formally, the generator matrix of $\widetilde{B}'$, $\mathbf{Q}_{\widetilde{B}'}$, is determined as follows. Let $(\overrightarrow{M'_r})_j$ be the r-th moment of D'_j for $1 \le j \le \xi$ for each $r \ge 1$. We determine $(\overrightarrow{M'_r})_j$ so that $\widetilde{B}$ and $\widetilde{B}'$ have the same marginal r-th moment of the sojourn time in levels $\ge \kappa$:

$$(\overrightarrow{M'_r})_j = \left(\sum_{i=1}^{\xi} (\overrightarrow{\pi_{\kappa-1}})_i \cdot (\vec{\lambda})_i \cdot (\mathbf{P})_{i,j} \cdot (\mathbf{M}_r)_{i,j} \right) \Big/ \left(\sum_{i=1}^{\xi} (\overrightarrow{\pi_{\kappa-1}})_i \cdot (\vec{\lambda})_i \cdot (\mathbf{P})_{i,j} \right),$$

where $\overrightarrow{\pi_{\kappa-1}}$ denotes the stationary probability vector that $\widetilde{B}$ is in level $\kappa-1$, which can be calculated via matrix analytic methods as in Section 3.2. We approximate D'_j by a PH distribution, $(\overrightarrow{\tau'_j}, \mathbf{T'}_j)$, matching the first three moments of D'_j, $((\overrightarrow{M'_1})_j, (\overrightarrow{M'_2})_j, (\overrightarrow{M'_3})_j)$. Let $\overrightarrow{t'_j} = -\mathbf{T'}_j \cdot \vec{1}$ as before. Generator matrix $\mathbf{Q}_{\widetilde{B}'}$ is then defined by

$$\mathbf{Q}_{\widetilde{B}'} = \left(\begin{array}{cccc|c} \mathbf{L}_b^{(0)} & \mathbf{F}_b^{(0)} & & & \\ \mathbf{B}_b^{(1)} & \ddots & \ddots & & \\ & \ddots & \ddots & \mathbf{F}_b^{(\kappa-2)} & \\ & & \mathbf{B}_b^{(\kappa-1)} & \mathbf{L}_b^{(\kappa-1)} & \boldsymbol{\tau'} \\ \hline & & & \mathbf{t'} & \mathbf{T'} \end{array} \right),$$

where $\mathbf{L}_b^{(h)}$, $\mathbf{F}_b^{(h)}$, and $\mathbf{B}_b^{(h)}$ are submatrices of $\mathbf{Q}_b$ as defined in Section 3.5.2,

$$\mathbf{T'} = \begin{pmatrix} \mathbf{T'}_1 & & \\ & \ddots & \\ & & \mathbf{T'}_\xi \end{pmatrix}, \quad \mathbf{t'} = \begin{pmatrix} \overrightarrow{t'_1} & & \\ & \ddots & \\ & & \overrightarrow{t'_\xi} \end{pmatrix}, \quad \text{and}$$

$$\boldsymbol{\tau'} = \begin{pmatrix} (\vec{\lambda})_1 & & \\ & \ddots & \\ & & (\vec{\lambda})_\xi \end{pmatrix} \begin{pmatrix} (\mathbf{P})_{1,1} \cdot \overrightarrow{\tau'_1} & \cdots & (\mathbf{P})_{1,\xi} \cdot \overrightarrow{\tau'_\xi} \\ \vdots & & \vdots \\ (\mathbf{P})_{\xi,1} \cdot \overrightarrow{\tau'_1} & \cdots & (\mathbf{P})_{\xi,\xi} \cdot \overrightarrow{\tau'_\xi} \end{pmatrix}.$$

Observe that the number of PH distributions used to approximate the sojourn time distributions in levels $\geq \kappa$ is reduced from ξ^2, in $\widetilde{B}$, to ξ, in $\widetilde{B}'$. The next approximation, DR-CI, uses only one PH distribution.

3.6.2 Dimensionality reduction with complete independence assumption

DR-CI (DR with complete independence assumption) ignores not only the dependency that the length of the sojourn time in levels $\geq \kappa$ has on how the Markov chain enters levels $\geq \kappa$ but also the dependency on how it *exits* from levels $\geq \kappa$ (to level $\kappa - 1$). Specifically, we assume that $D_{i,j}$ is independent of i and j. Let $D'' = D_{i,j}$ for all i and j, and let $\widetilde{B}''$ denote the process that is the same as $\widetilde{B}$ except that $D_{i,j}$ is replaced by D'' for all i and j. Figure 3.27(b) shows the Markov chain for the high priority jobs (background process) that is obtained via DR-CI in the analysis of an M/PH/2 queue with two priority classes. In DR-CI, the four types of busy periods are represented by a *single* PH distribution, ignoring the dependency that the duration of the busy period has on the phase of the job in service at the beginning and end of the busy period.

We choose D'' such that $\widetilde{B}''$ has the above two key properties that $\widetilde{B}'$ has. The difference between $\widetilde{B}'$ and $\widetilde{B}''$ lies in the dependencies in the sequence of the sojourn times in levels $\geq \kappa$. Observe that the sequence of the sojourn times in levels $\geq \kappa$ is i.i.d. in $\widetilde{B}''$, while it has some dependencies in $\widetilde{B}'$.

More formally, the generator matrix of $\widetilde{B}''$, $Q_{\widetilde{B}''}$, is determined as follows. Let M''_r be the r-th moment of D'' for $r \geq 1$. We determine M''_r so that $\widetilde{B}''$ and B have the same marginal r-th moment of the sojourn time in levels $\geq \kappa$:

$$M''_r = \left(\sum_{j=1}^{\xi} \sum_{i=1}^{\xi} (\overrightarrow{\pi_{\kappa-1}})_i \cdot (\vec{\lambda})_i \cdot (\mathbf{P})_{i,j} \cdot (\mathbf{M}'_{\mathbf{r}})_{i,j} \right) \Big/ \left(\sum_{j=1}^{\xi} \sum_{i=1}^{\xi} (\overrightarrow{\pi_{\kappa-1}})_i \cdot (\vec{\lambda})_i \cdot (\mathbf{P})_{i,j} \right).$$

We approximate D'' by a PH distribution, $(\overrightarrow{\tau''}, \mathbf{T}'')$, matching the first three moments of D'', (M''_1, M''_2, M''_3). Let $\overrightarrow{t''} = -\mathbf{T}'' \cdot \vec{1}$ as before. Generator matrix $\mathbf{Q}_{\widetilde{B}''}$ is then defined by

$$\mathbf{Q}_{\widetilde{B}''} = \left(\begin{array}{cccc|c} \mathbf{L}_b^{(0)} & \mathbf{F}_b^{(0)} & & & \\ \mathbf{B}_b^{(1)} & \ddots & \ddots & & \\ & \ddots & \ddots & \mathbf{F}_b^{(\kappa-2)} & \\ & & \mathbf{B}_b^{(\kappa-1)} & \mathbf{L}_b^{(\kappa-1)} & \boldsymbol{\tau''} \\ \hline & & & \overrightarrow{t''} \cdot \overrightarrow{\chi} & \mathbf{T}'' \end{array} \right),$$

where $\boldsymbol{\tau''} = \text{transpose}(\vec{\lambda}) \cdot \overrightarrow{\tau''}$, and $\overrightarrow{\chi}$ is a row vector whose h-th element is

$$(\overrightarrow{\chi})_h = \left(\sum_{i=1}^{\xi} (\overrightarrow{\pi_{\kappa-1}})_i \cdot (\overrightarrow{\lambda})_i \cdot (\mathbf{P})_{i,h} \right) \Big/ \left(\sum_{i=1}^{\xi} \sum_{j=1}^{\xi} (\overrightarrow{\pi_{\kappa-1}})_i \cdot (\overrightarrow{\lambda})_i \cdot (\mathbf{P})_{i,j} \right).$$

3.6.3 Computational complexity of DR, DR-PI, and DR-CI

When DR, DR-PI, and DR-CI are applied to an RFB process, their running times differ primarily due to the differences in the orders of the submatrices of the generator matrix (the number of states in each level) of the 1D Markov chain reduced from the RFB process. Below, we compare how the orders of the submatrices of DR, DR-PI, and DR-CI grow as the number of processes, m, increases. For simplicity, assume that the infinitesimal generator of the i-th process is fixed while the $(i-1)$-th process is in levels $\geq \kappa$ for all i (i.e. $\kappa = \kappa_1 = \cdots = \kappa_{m-1}$), and that all the processes have the same number of states, ξ, in each level.

In DR, the order of the submatrices, $S_{DR(m)}$, grows double-exponentially with m (i.e. $O(2^{2^m})$); specifically, $S_{DR(m)}$ can be determined by the following recursive formula: $S_{DR(m)} = \kappa S_{DR(m-1)} + (S_{DR(m-1)})^2 N_{PH}$ for $m > 1$ and $S_{DR(1)} = \xi$, where N_{PH} is the number of phases used in a PH distribution that approximates a (conditional) sojourn time in levels $\geq \kappa$ (there are $(S_{DR(m-1)})^2$ types of sojourn times). In DR-PI, the order of the submatrices, $S_{PI(m)}$, grows exponentially with m: $S_{PI(m)} = (\kappa + N_{PH})^{m-1}\xi$. In DR-CI, the order of the submatrices, $S_{CI(m)}$, grows exponentially with m (but slower than $S_{PI(m)}$) when $\kappa > 1$: $S_{CI(m)} = \kappa^{m-1}(\xi + N_{PH}/(\kappa-1)) - N_{PH}/(\kappa-1)$, and it grows linearly with m when $\kappa = 1$: $S_{CI(m)} = (m-1)N_{PH} + \xi$.

3.7 Moments of inter-level passage times in QBD processes

Neuts' algorithm [134, 137] is an efficient algorithm that calculates the moments of various types of passage times in very general processes, i.e. M/G/1 type semi-Markov processes. Because of its generality, however, the description of the algorithm in [134, 137] is sophisticated. The purpose of this section is to make Neuts' algorithm more accessible to general readers by re-describing his algorithm restricted to the *first three moments* of inter-level passage times in *QBD processes*. We omit all proofs, which are provided in detail in [134], and we instead focus on intuition and interpretation. We include everything needed to apply Neuts' algorithm within our solution framework, so that general readers can apply our methodology.

Specifically, we consider a QBD process on the state space $\{(i,\ell) | 1 \leq i \leq n_\ell, \ell \geq 0\}$, which has generator matrix $\mathbf{Q}$:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{L}^{(0)} & \mathbf{F}^{(0)} & & \\ \mathbf{B}^{(1)} & \mathbf{L}^{(1)} & \mathbf{F}^{(1)} & \\ & \mathbf{B}^{(2)} & \mathbf{L}^{(2)} & \mathbf{F}^{(2)} \\ & & \ddots & \ddots \end{pmatrix}$$

as defined in Section 3.2. We assume that the QBD process repeats after level $\hat{\ell}$; i.e. $\mathbf{B}^{(\ell)} = \mathbf{B}$, $\mathbf{L}^{(\ell)} = \mathbf{L}$, and $\mathbf{F}^{(\ell)} = \mathbf{F}$ for all $\ell \geq \hat{\ell}$.

Our goal can be roughly stated as deriving the passage time required to get from state (i,ℓ) to level $\ell-1$ conditioned on the particular state first reached in level $\ell-1$. To state our goal more precisely, let $E_{i,j}^{(\ell)}$ be the event that state $(j,\ell-1)$ is the first state reached in level $\ell-1$ when starting in (i,ℓ), and let $T_{i,j}^{(\ell)}$ be the time to go from state (i,ℓ) to state $(j,\ell-1)$. Then, our goal is to derive the $n_\ell \times n_{\ell-1}$ matrix, $\mathbf{Z}_r^{(\ell)}$, where $(\mathbf{Z}_r^{(\ell)})_{i,j}$ is the r-th moment of $T_{i,j}^{(\ell)}$ given event $E_{i,j}^{(\ell)}$, for each ℓ and $r = 1, 2, 3$.

Observe that

$$\begin{aligned}(\mathbf{Z}_r^{(\ell)})_{i,j} &= \int_0^\infty x^r d\Pr\left(T_{i,j}^{(\ell)} \leq x \mid E_{i,j}^{(\ell)}\right) \\ &= \frac{\int_0^\infty x^r d\Pr\left(T_{i,j}^{(\ell)} \leq x \text{ AND } E_{i,j}^{(\ell)}\right)}{\Pr\left(E_{i,j}^{(\ell)}\right)}.\end{aligned}$$

Hence, it suffices to derive two quantities:

$$
\begin{aligned}
(\mathbf{G}^{(\ell)})_{i,j} &\equiv \Pr\left(E_{i,j}^{(\ell)}\right) \\
(\mathbf{G}_r^{(\ell)})_{i,j} &\equiv \int_0^\infty x^r d\Pr\left(T_{i,j}^{(\ell)} \le x \text{ AND } E_{i,j}^{(\ell)}\right)
\end{aligned}
$$

The rest of this section is organized as follows. In Section 3.7.1, we introduce some notations that we will need along the way. In Section 3.7.2, we derive $\mathbf{G}^{(\ell)}$ and $\mathbf{G}_r^{(\ell)}$ for each ℓ and $r = 1, 2, 3$. (Matrix $\mathbf{G}^{(\ell)}$ can also be obtained by the algorithm in Figure 3.12.) In Section 3.7.3, we mention some generalizations that Neuts' algorithm allows. In particular, we show an extension to the passage time from level ℓ to level $\ell - \psi$ for $1 \le \psi \le \ell$, which we will need in Section 3.8.

3.7.1 Preliminaries

We first define the transition probability matrix, $\mathbf{P}(x)$, such that its (s,t) element, $(\mathbf{P}(x))_{s,t}$, represents the probability that (i) the sojourn time at state s is $\le x$ AND (ii) the first transition out of state s is to state t. (Note that state s denotes a state (i,ℓ), where $s = i + \sum_{\ell'=0}^{\ell-1} n_{\ell'}$. State t is defined analogously.) By the following Lemma, which follows from the memoryless property of the exponential distribution, $(\mathbf{P}(x))_{s,t}$ is also the product of the probabilities of events (i) and (ii).

Lemma 7 *Let X_1 and X_2 be independent exponential random variables. Let $W = \min(X_1, X_2)$. Then*

$$\Pr(W < x \textit{ AND } X_1 < X_2) = \Pr(W < x) \cdot \Pr(X_1 < X_2)$$

Matrix $\mathbf{P}(x)$ has the same structural shape as $\mathbf{Q}$, although its entries are different, and thus it can be represented in terms of submatrices, analogous to those in $\mathbf{Q}$, indicating backward transitions, local transitions, and forward transitions, as follows:

$$
\mathbf{P}(x) = \begin{pmatrix}
\boldsymbol{\mathcal{L}}^{(0)}(x) & \boldsymbol{\mathcal{F}}^{(0)}(x) & & \\
\boldsymbol{\mathcal{B}}^{(1)}(x) & \boldsymbol{\mathcal{L}}^{(1)}(x) & \boldsymbol{\mathcal{F}}^{(1)}(x) & \\
 & \boldsymbol{\mathcal{B}}^{(2)}(x) & \boldsymbol{\mathcal{L}}^{(2)}(x) & \boldsymbol{\mathcal{F}}^{(2)}(x) \\
 & & \ddots & \ddots
\end{pmatrix}
$$

Specifically, the (i,j) element of $\boldsymbol{\mathcal{L}}^{(\ell)}(x)$ is the probability that the sojourn time in state (i,ℓ) is $\le x$ AND the first transition out of state (i,ℓ) is to state (j,ℓ). Likewise, the (i,j) element of $\boldsymbol{\mathcal{F}}^{(\ell)}(x)$ (respectively, $\boldsymbol{\mathcal{B}}^{(\ell)}(x)$) is the probability that the sojourn time in state (i,ℓ) is $\le x$ AND the first transition out of state (i,ℓ) is to state $(j,\ell+1)$ (respectively, to state $(j,\ell-1)$).

Next, we define the r-th moment of submatrices, $\boldsymbol{\mathcal{B}}^{(\ell)}(x)$, $\boldsymbol{\mathcal{L}}^{(\ell)}(x)$, $\boldsymbol{\mathcal{F}}^{(\ell)}(x)$, as follows:

$$
\boldsymbol{\mathcal{B}}_r^{(\ell)} \equiv \int_0^\infty x^r d\boldsymbol{\mathcal{B}}^{(\ell)}(x); \qquad \boldsymbol{\mathcal{L}}_r^{(\ell)} \equiv \int_0^\infty x^r d\boldsymbol{\mathcal{L}}^{(\ell)}(x); \qquad \boldsymbol{\mathcal{F}}_r^{(\ell)} \equiv \int_0^\infty x^r d\boldsymbol{\mathcal{F}}^{(\ell)}(x)
$$

for $r = 1, 2, 3$, and $\ell \ge 0$, where an integral of a matrix $\mathbf{M}$ is a matrix of the integrals of the elements in $\mathbf{M}$. For the repeating part, we define $\boldsymbol{\mathcal{B}}_r \equiv \boldsymbol{\mathcal{B}}_r^{(\hat{\ell})}$, $\boldsymbol{\mathcal{L}}_r \equiv \boldsymbol{\mathcal{L}}_r^{(\hat{\ell})}$, and $\boldsymbol{\mathcal{F}}_r \equiv \boldsymbol{\mathcal{F}}_r^{(\hat{\ell})}$, omitting the superscript.

We now define the limits as $x \to \infty$ of $\boldsymbol{\mathcal{B}}^{(\ell)}(x)$, $\boldsymbol{\mathcal{L}}^{(\ell)}(x)$, and $\boldsymbol{\mathcal{F}}^{(\ell)}(x)$ as follows:

$$\boldsymbol{\mathcal{B}}^{(\ell)} = \lim_{x\to\infty} \boldsymbol{\mathcal{B}}^{(\ell)}(x); \qquad \boldsymbol{\mathcal{L}}^{(\ell)} = \lim_{x\to\infty} \boldsymbol{\mathcal{L}}^{(\ell)}(x); \qquad \boldsymbol{\mathcal{F}}^{(\ell)} = \lim_{x\to\infty} \boldsymbol{\mathcal{F}}^{(\ell)}(x);$$

for $\ell \geq 0$. For the repeating part, we define $\boldsymbol{\mathcal{B}} \equiv \boldsymbol{\mathcal{B}}^{(\hat{\ell})}$, $\boldsymbol{\mathcal{L}} \equiv \boldsymbol{\mathcal{L}}^{(\hat{\ell})}$, and $\boldsymbol{\mathcal{F}} \equiv \boldsymbol{\mathcal{F}}^{(\hat{\ell})}$, omitting the superscript.

Example

Consider the QBD process shown in Figure 3.10(b), where

$$\mathbf{F}^{(\ell)} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \qquad \mathbf{L}^{(\ell)} = \begin{pmatrix} -\sigma_1 & \alpha_{12} \\ \alpha_{21} & -\sigma_2 \end{pmatrix} \qquad \mathbf{B}^{(\ell)} = \begin{pmatrix} \mu & 0 \\ 0 & \mu \end{pmatrix}$$

for $\ell \geq 1$, where $\sigma_1 = \lambda_1 + \mu + \alpha_{12}$ and $\sigma_2 = \lambda_2 + \mu + \alpha_{21}$. For this QBD process,

$$\begin{aligned}
\boldsymbol{\mathcal{F}}^{(\ell)}(x) &= \begin{pmatrix} 1-e^{-\sigma_1 x} & 0 \\ 0 & 1-e^{-\sigma_2 x} \end{pmatrix} \begin{pmatrix} \frac{\lambda_1}{\sigma_1} & 0 \\ 0 & \frac{\lambda_2}{\sigma_2} \end{pmatrix} & \boldsymbol{\mathcal{F}}^{(\ell)} &= \begin{pmatrix} \frac{\lambda_1}{\sigma_1} & 0 \\ 0 & \frac{\lambda_2}{\sigma_2} \end{pmatrix} \\
\boldsymbol{\mathcal{L}}^{(\ell)}(x) &= \begin{pmatrix} 1-e^{-\sigma_1 x} & 0 \\ 0 & 1-e^{-\sigma_2 x} \end{pmatrix} \begin{pmatrix} 0 & \frac{\alpha_{12}}{\sigma_1} \\ \frac{\alpha_{21}}{\sigma_2} & 0 \end{pmatrix} & \boldsymbol{\mathcal{L}}^{(\ell)} &= \begin{pmatrix} 0 & \frac{\alpha_{12}}{\sigma_1} \\ \frac{\alpha_{21}}{\sigma_2} & 0 \end{pmatrix} \\
\boldsymbol{\mathcal{B}}^{(\ell)}(x) &= \begin{pmatrix} 1-e^{-\sigma_1 x} & 0 \\ 0 & 1-e^{-\sigma_2 x} \end{pmatrix} \begin{pmatrix} \frac{\mu}{\sigma_1} & 0 \\ 0 & \frac{\mu}{\sigma_2} \end{pmatrix} & \boldsymbol{\mathcal{B}}^{(\ell)} &= \begin{pmatrix} \frac{\mu}{\sigma_1} & 0 \\ 0 & \frac{\mu}{\sigma_2} \end{pmatrix} \\
\boldsymbol{\mathcal{F}}_r^{(\ell)} &= \begin{pmatrix} \frac{r!}{(\sigma_1)^r} & 0 \\ 0 & \frac{r!}{(\sigma_2)^r} \end{pmatrix} \begin{pmatrix} \frac{\lambda_1}{\sigma_1} & 0 \\ 0 & \frac{\lambda_2}{\sigma_2} \end{pmatrix} \\
\boldsymbol{\mathcal{L}}_r^{(\ell)} &= \begin{pmatrix} \frac{r!}{(\sigma_1)^r} & 0 \\ 0 & \frac{r!}{(\sigma_2)^r} \end{pmatrix} \begin{pmatrix} 0 & \frac{\alpha_{12}}{\sigma_1} \\ \frac{\alpha_{21}}{\sigma_2} & 0 \end{pmatrix} \\
\boldsymbol{\mathcal{B}}_r^{(\ell)} &= \begin{pmatrix} \frac{r!}{(\sigma_1)^r} & 0 \\ 0 & \frac{r!}{(\sigma_2)^r} \end{pmatrix} \begin{pmatrix} \frac{\mu}{\sigma_1} & 0 \\ 0 & \frac{\mu}{\sigma_2} \end{pmatrix}
\end{aligned}$$

for $\ell \geq 1$.

3.7.2 Moments of passage time

We will use the matrices introduced above to derive $\mathbf{G}^{(\ell)}$ and $\mathbf{G}_r^{(\ell)}$ for $r = 1, 2, 3$. We first derive those for the repeating part, $\mathbf{G} \equiv \mathbf{G}^{(\hat{\ell})}$ and $\mathbf{G}_r \equiv \mathbf{G}_r^{(\hat{\ell})}$. Then, $\mathbf{G}$ and $\mathbf{G}_r$ are used to derive those for the nonrepeating part, $\mathbf{G}^{(\ell)}$ and $\mathbf{G}_r^{(\ell)}$ for $1 \leq \ell < \hat{\ell}$.

Moments of passage time in repeating part

Matrix $\mathbf{G}$ can be derived by the algorithm in Figure 3.12, but it is instructive to study an alternative derivation of $\mathbf{G}$. The idea is to simply iterate the following equation until it converges:

$$\mathbf{G} = \boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{L}}\mathbf{G} + \boldsymbol{\mathcal{F}}\mathbf{G}\mathbf{G} \tag{3.7}$$

The intuition behind Equation (3.7) should be clear: Recall that the (i, j) entry of $\mathbf{G}^{(\ell)}$ represents the probability that state $(j, \ell-1)$ is the first state reached in level $\ell-1$, given that we start in state (i, ℓ). Now the right hand side of Equation (3.7) represents this same probability by conditioning on whether the first move is a backward transition, a local transition, or a forward transition. Specifically, the (i, j) element of $\boldsymbol{\mathcal{B}}^{(\ell)}$ represents the probability that the first move leaving state (i, ℓ) is to state $(j, \ell-1)$. The (i, j) element of $\boldsymbol{\mathcal{L}}^{(\ell)}\mathbf{G}^{(\ell)}$ represents the probability that the first transition out of state (i, ℓ) is to state (v, ℓ) for some v multiplied by the probability that the the first state reached in level $\ell-1$ is $(j, \ell-1)$ when we start in state (v, ℓ), summed over all possible $v = 1, \ldots, n_\ell$. Finally, the (i, j) element of $\boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)}$ represents the product of three terms: (i) the probability that the first transition from (i, ℓ) is to some state $(v_1, \ell+1)$ in level $\ell+1$, (ii) the probability that from state $(v_1, \ell+1)$ the first state reached in level ℓ is some (v_2, ℓ), and (iii) the probability that from state (v_2, ℓ) the first state reached in level $\ell-1$ is state $(j, \ell-1)$, where the product is summed over all possible $v_1 = 1, \ldots, n_{\ell+1}$ and $v_2 = 1, \ldots, n_\ell$. Since we are in the repeating region, all the superscripts are equal to $\hat{\ell}$, and can be dropped by our notation.

Matrices $\mathbf{G}_r$ for $r = 1, 2, 3$ are derived in a similar manner, by using $\mathbf{G}$ which we have already derived. Matrix $\mathbf{G}_1$ is obtained by iterating:

$$\mathbf{G}_1 = \boldsymbol{\mathcal{B}}_1 + \boldsymbol{\mathcal{L}}_1\mathbf{G} + \boldsymbol{\mathcal{L}}\mathbf{G}_1 + \boldsymbol{\mathcal{F}}_1\mathbf{G}\mathbf{G} + \boldsymbol{\mathcal{F}}\mathbf{G}_1\mathbf{G} + \boldsymbol{\mathcal{F}}\mathbf{G}\mathbf{G}_1. \tag{3.8}$$

Similarly, matrix $\mathbf{G}_2$ is obtained by iterating:

$$\begin{aligned} \mathbf{G}_2 &= \boldsymbol{\mathcal{B}}_2 + \boldsymbol{\mathcal{L}}_2\mathbf{G} + 2\boldsymbol{\mathcal{L}}_1\mathbf{G}_1 + \boldsymbol{\mathcal{L}}\mathbf{G}_2 \\ &\quad + \boldsymbol{\mathcal{F}}_2\mathbf{G}\mathbf{G} + 2\boldsymbol{\mathcal{F}}_1(\mathbf{G}_1\mathbf{G} + \mathbf{G}\mathbf{G}_1) + \boldsymbol{\mathcal{F}}(\mathbf{G}_2\mathbf{G} + 2\mathbf{G}_1\mathbf{G}_1 + \mathbf{G}\mathbf{G}_2) \end{aligned} \tag{3.9}$$

and matrix $\mathbf{G}_3$ is obtained by iterating:

$$\begin{aligned} \mathbf{G}_3 &= \boldsymbol{\mathcal{B}}_3 + \boldsymbol{\mathcal{L}}_3\mathbf{G} + 3\boldsymbol{\mathcal{L}}_2\mathbf{G}_1 + 3\boldsymbol{\mathcal{L}}_1\mathbf{G}_2 + \boldsymbol{\mathcal{L}}\mathbf{G}_3 \\ &\quad + \boldsymbol{\mathcal{F}}_3\mathbf{G}\mathbf{G} + 3\boldsymbol{\mathcal{F}}_2(\mathbf{G}_1\mathbf{G} + \mathbf{G}\mathbf{G}_1) + 3\boldsymbol{\mathcal{F}}_1(\mathbf{G}_2\mathbf{G} + 2\mathbf{G}_1\mathbf{G}_1 + \mathbf{G}\mathbf{G}_2) \\ &\quad\ + \boldsymbol{\mathcal{F}}(\mathbf{G}_3\mathbf{G} + 3\mathbf{G}_2\mathbf{G}_1 + 3\mathbf{G}_1\mathbf{G}_2 + \mathbf{G}\mathbf{G}_3). \end{aligned} \tag{3.10}$$

We now give intuition behind expressions (3.8)-(3.10). The right hand side of (3.8) can be divided into three parts: *Part 0*: $\boldsymbol{\mathcal{B}}_1$, *Part 1*: $\boldsymbol{\mathcal{L}}_1\mathbf{G} + \boldsymbol{\mathcal{L}}\mathbf{G}_1$, and *Part 2*: $\boldsymbol{\mathcal{F}}_1\mathbf{G}\mathbf{G} + \boldsymbol{\mathcal{F}}\mathbf{G}_1\mathbf{G} + \boldsymbol{\mathcal{F}}\mathbf{G}\mathbf{G}_1$. The (i, j) element of *Part h* gives "the first moment of the distribution of $T_{i,j}^{(\ell)}$ given that the first transition out of state (i, ℓ) is to level $\ell + h - 1$ and event $E_{i,j}^{(\ell)}$" multiplied by "the probability that the first transition out of state (i, ℓ) is to level $\ell + h - 1$ and event $E_{i,j}^{(\ell)}$" for $h = 0, 1, 2$. *Part 1* consists of two terms. The first term, $\boldsymbol{\mathcal{L}}_1\mathbf{G}$, is the contribution of the time to the first transition, and the second term, $\boldsymbol{\mathcal{L}}\mathbf{G}_1$, is the contribution of the time it takes to reach $(j, \ell-1)$ after the first transition. Similarly, *Part 2* consists of three terms. The first term, $\boldsymbol{\mathcal{F}}_1\mathbf{G}\mathbf{G}$, is the contribution of the time to the first transition, the second term, $\boldsymbol{\mathcal{F}}\mathbf{G}_1\mathbf{G}$, is the contribution of the time it takes to come back from level $\ell + 1$ to level ℓ after the first transition, and the third term, $\boldsymbol{\mathcal{F}}\mathbf{G}\mathbf{G}_1$, is the contribution of the time it takes to go from level ℓ to level $\ell - 1$.

The right hand sides of (3.9) and (3.10) can similarly be divided into three parts: *Part 0* consists of terms containing $\boldsymbol{\mathcal{B}}$ or $\boldsymbol{\mathcal{B}}_r$; *Part 1* consists of terms containing $\boldsymbol{\mathcal{L}}$ or $\boldsymbol{\mathcal{L}}_r$; *Part 2* consists of terms containing $\boldsymbol{\mathcal{F}}$ or $\boldsymbol{\mathcal{F}}_r$. The three parts of (3.9) and (3.10) can be interpreted exactly the same way as the three parts of (3.8) except that "the first moment" in (3.8) must be replaced by "the second moment" and "the third moment" in (3.9) and (3.10), respectively. For example, the three terms in *Part 1* of (3.9) can be interpreted as follows. Let X be the time to the first transition and let Y be the time it takes from level ℓ to level $\ell-1$. Then, the second moment of the distribution of these two times is

$$E[(X+Y)^2] = E[(X)^2] + 2E[X]E[Y] + E[(Y)^2],$$

since X and Y are independent. Roughly speaking, $\boldsymbol{\mathcal{L}}_2\mathbf{G}$ corresponds to $E[(X)^2]$, $2\boldsymbol{\mathcal{L}}_1\mathbf{G}_1$ corresponds to $2E[X]E[Y]$, and $\boldsymbol{\mathcal{L}}\mathbf{G}_2$ corresponds to $E[(Y)^2]$. The other terms can be interpreted in the same way.

Extension to nonrepeating part

For $\ell < \hat{\ell}$, $\mathbf{G}^{(\ell)}$ and $\mathbf{G}_r^{(\ell)}$ are calculated recursively from $\ell = \hat{\ell}-1$ to $\ell = 1$ as follows:

$$\begin{aligned}
\mathbf{G}^{(\ell)} &= \boldsymbol{\mathcal{B}}^{(\ell)} + \boldsymbol{\mathcal{L}}^{(\ell)}\mathbf{G}^{(\ell)} + \boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)} \\
&= \left(\mathbf{I} - \boldsymbol{\mathcal{L}}^{(\ell)} - \boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\right)^{-1}\boldsymbol{\mathcal{B}}^{(\ell)} \\
\mathbf{G}_1^{(\ell)} &= \boldsymbol{\mathcal{B}}_1^{(\ell)} + \boldsymbol{\mathcal{L}}_1^{(\ell)}\mathbf{G}^{(\ell)} + \boldsymbol{\mathcal{L}}^{(\ell)}\mathbf{G}_1^{(\ell)} + \boldsymbol{\mathcal{F}}_1^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)} + \boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}_1^{(\ell+1)}\mathbf{G}^{(\ell)} + \boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}_1^{(\ell)} \\
&= \left(\mathbf{I} - \boldsymbol{\mathcal{L}}^{(\ell)} - \boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\right)^{-1}\left(\boldsymbol{\mathcal{B}}_1^{(\ell)} + \boldsymbol{\mathcal{L}}_1^{(\ell)}\mathbf{G}^{(\ell)} + \boldsymbol{\mathcal{F}}_1^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)} + \boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}_1^{(\ell+1)}\mathbf{G}^{(\ell)}\right) \\
\mathbf{G}_2^{(\ell)} &= \boldsymbol{\mathcal{B}}_2^{(\ell)} + \boldsymbol{\mathcal{L}}_2^{(\ell)}\mathbf{G}^{(\ell)} + 2\boldsymbol{\mathcal{L}}_1^{(\ell)}\mathbf{G}_1^{(\ell)} + \boldsymbol{\mathcal{L}}^{(\ell)}\mathbf{G}_2^{(\ell)} \\
&\quad + \boldsymbol{\mathcal{F}}_2^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)} + 2\boldsymbol{\mathcal{F}}_1^{(\ell)}(\mathbf{G}_1^{(\ell+1)}\mathbf{G}^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_1^{(\ell)}) \\
&\quad + \boldsymbol{\mathcal{F}}^{(\ell)}(\mathbf{G}_2^{(\ell+1)}\mathbf{G}^{(\ell)} + 2\mathbf{G}_1^{(\ell+1)}\mathbf{G}_1^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_2^{(\ell)}) \\
&= \left(\mathbf{I} - \boldsymbol{\mathcal{L}}^{(\ell)} - \boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\right)^{-1} \\
&\quad \Big(\boldsymbol{\mathcal{B}}_2^{(\ell)} + \boldsymbol{\mathcal{L}}_2^{(\ell)}\mathbf{G}^{(\ell)} + 2\boldsymbol{\mathcal{L}}_1^{(\ell)}\mathbf{G}_1^{(\ell)} \\
&\quad + \boldsymbol{\mathcal{F}}_2^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)} + 2\boldsymbol{\mathcal{F}}_1^{(\ell)}(\mathbf{G}_1^{(\ell+1)}\mathbf{G}^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_1^{(\ell)}) + \boldsymbol{\mathcal{F}}^{(\ell)}(\mathbf{G}_2^{(\ell+1)}\mathbf{G}^{(\ell)} + 2\mathbf{G}_1^{(\ell+1)}\mathbf{G}_1^{(\ell)})\Big) \\
\mathbf{G}_3^{(\ell)} &= \boldsymbol{\mathcal{B}}_3^{(\ell)} + \boldsymbol{\mathcal{L}}_3^{(\ell)}\mathbf{G}^{(\ell)} + 3\boldsymbol{\mathcal{L}}_2^{(\ell)}\mathbf{G}_1^{(\ell)} + 3\boldsymbol{\mathcal{L}}_1^{(\ell)}\mathbf{G}_2^{(\ell)} + \boldsymbol{\mathcal{L}}^{(\ell)}\mathbf{G}_3^{(\ell)} + \boldsymbol{\mathcal{F}}_3^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)} \\
&\quad + 3\boldsymbol{\mathcal{F}}_2^{(\ell)}(\mathbf{G}_1^{(\ell+1)}\mathbf{G}^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_1^{(\ell)}) + 3\boldsymbol{\mathcal{F}}_1^{(\ell)}(\mathbf{G}_2^{(\ell+1)}\mathbf{G}^{(\ell)} + 2\mathbf{G}_1^{(\ell+1)}\mathbf{G}_1^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_2^{(\ell)}) \\
&\quad + \boldsymbol{\mathcal{F}}^{(\ell)}(\mathbf{G}_3^{(\ell+1)}\mathbf{G}^{(\ell)} + 3\mathbf{G}_2^{(\ell+1)}\mathbf{G}_1^{(\ell)} + 3\mathbf{G}_1^{(\ell+1)}\mathbf{G}_2^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_3^{(\ell)}). \\
&= \left(\mathbf{I} - \boldsymbol{\mathcal{L}}^{(\ell)} - \boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\right)^{-1} \\
&\quad \Big(\boldsymbol{\mathcal{B}}_3^{(\ell)} + \boldsymbol{\mathcal{L}}_3^{(\ell)}\mathbf{G}^{(\ell)} + 3\boldsymbol{\mathcal{L}}_2^{(\ell)}\mathbf{G}_1^{(\ell)} + 3\boldsymbol{\mathcal{L}}_1^{(\ell)}\mathbf{G}_2^{(\ell)} + \boldsymbol{\mathcal{F}}_3^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)} \\
&\quad + 3\boldsymbol{\mathcal{F}}_2^{(\ell)}(\mathbf{G}_1^{(\ell+1)}\mathbf{G}^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_1^{(\ell)}) \\
&\quad + 3\boldsymbol{\mathcal{F}}_1^{(\ell)}(\mathbf{G}_2^{(\ell+1)}\mathbf{G}^{(\ell)} + 2\mathbf{G}_1^{(\ell+1)}\mathbf{G}_1^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_2^{(\ell)}) \\
&\quad + \boldsymbol{\mathcal{F}}^{(\ell)}(\mathbf{G}_3^{(\ell+1)}\mathbf{G}^{(\ell)} + 3\mathbf{G}_2^{(\ell+1)}\mathbf{G}_1^{(\ell)} + 3\mathbf{G}_1^{(\ell+1)}\mathbf{G}_2^{(\ell)})\Big).
\end{aligned}$$

Note that $\mathbf{G}^{(\hat{\ell})} = \mathbf{G}$ and $\mathbf{G}_r^{(\hat{\ell})} = \mathbf{G}_r$, for $r = 1, 2, 3$, are computed in the repeating part. The intuition for the above formulation is the same as in the repeating part.

3.7.3 Generalization

So far, we have derived the first passage time from level ℓ to level $\ell-1$. In this section, we extend this to the first passage time from level ℓ to level $\ell-\psi$ for $1 \le \psi \le \ell$. Let $T_{i,j}^{(\ell,\ell-\psi)}$ be the time to go from state (i,ℓ) to state $(j,\ell-\psi)$, and let $E_{i,j}^{(\ell,\ell-\psi)}$ be the event that state $(j,\ell-\psi)$ is the first state reached in level $\ell-\psi$ when starting in state (i,ℓ), for $1 \le \psi \le \ell$. Observe that $T_{i,j}^{(\ell,\ell-1)} = T_{i,j}^{(\ell)}$ and $E_{i,j}^{(\ell,\ell-1)} = E_{i,j}^{(\ell)}$. Then, our goal is to derive the $n_\ell \times n_{\ell-\psi}$ matrix, $\mathbf{Z}_r^{(\ell,\ell-\psi)}$, where $(\mathbf{Z}_r^{(\ell,\ell-\psi)})_{i,j}$ is the r-th moment of $T_{i,j}^{(\ell,\ell-\psi)}$ given event $E_{i,j}^{(\ell,\ell-\psi)}$, for each ℓ, $r=1,2,3$, and $1 \le \psi \le \ell$. Notice that $\mathbf{Z}_r^{(\ell,\ell-1)} = \mathbf{Z}_r^{(\ell)}$.

Observe that

$$\begin{aligned}
(\mathbf{Z}_r^{(\ell,\ell-\psi)})_{i,j} &= \int_0^\infty x^r d\Pr\left(T_{i,j}^{(\ell,\ell-\psi)} \le x \mid E_{i,j}^{(\ell,\ell-\psi)}\right) \\
&= \frac{\int_0^\infty x^r d\Pr\left(T_{i,j}^{(\ell,\ell-\psi)} \le x \text{ AND } E_{i,j}^{(\ell,\ell-\psi)}\right)}{\Pr\left(E_{i,j}^{(\ell,\ell-\psi)}\right)}.
\end{aligned}$$

Hence, it suffices to derive two quantities:

$$\begin{aligned}
(\mathbf{G}^{(\ell,\ell-\psi)})_{i,j} &\equiv \Pr\left(E_{i,j}^{(\ell,\ell-\psi)}\right) \\
(\mathbf{G}_r^{(\ell,\ell-\psi)})_{i,j} &\equiv \int_0^\infty x^r d\Pr\left(T_{i,j}^{(\ell,\ell-\psi)} \le x \text{ AND } E_{i,j}^{(\ell,\ell-\psi)}\right)
\end{aligned}$$

for $1 \le \psi \le \ell$.

Matrices $\mathbf{G}^{(\ell,\ell-\psi)}$ and $\mathbf{G}_r^{(\ell,\ell-\psi)}$ can be derived recursively from

$$\mathbf{G}^{(\ell,\ell-1)} = \mathbf{G}^{(\ell)} \quad \text{and} \quad \mathbf{G}_r^{(\ell,\ell-1)} = \mathbf{G}_r^{(\ell)}$$

via

$$\begin{aligned}
\mathbf{G}^{(\ell,\ell-\psi)} &= \mathbf{G}^{(\ell)}\mathbf{G}^{(\ell-1,\ell-\psi)} \\
\mathbf{G}_1^{(\ell,\ell-\psi)} &= \mathbf{G}^{(\ell)}\mathbf{G}_1^{(\ell-1,\ell-\psi)} + \mathbf{G}_1^{(\ell)}\mathbf{G}^{(\ell-1,\ell-\psi)} \\
\mathbf{G}_2^{(\ell,\ell-\psi)} &= \mathbf{G}^{(\ell)}\mathbf{G}_2^{(\ell-1,\ell-\psi)} + 2\mathbf{G}_1^{(\ell)}\mathbf{G}_1^{(\ell-1,\ell-\psi)} + \mathbf{G}_2^{(\ell)}\mathbf{G}^{(\ell-1,\ell-\psi)} \\
\mathbf{G}_3^{(\ell,\ell-\psi)} &= \mathbf{G}^{(\ell)}\mathbf{G}_3^{(\ell-1,\ell-\psi)} + 3\mathbf{G}_1^{(\ell)}\mathbf{G}_2^{(\ell-1,\ell-\psi)} + 3\mathbf{G}_2^{(\ell)}\mathbf{G}_1^{(\ell-1,\ell-\psi)} + \mathbf{G}_3^{(\ell)}\mathbf{G}^{(\ell-1,\ell-\psi)}.
\end{aligned}$$

Finally, we mention some other generalizations that Neuts' algorithm allows. (i) We restricted ourselves to the first three moments, but this approach can be generalized to any higher moments. (ii) We restricted to QBD processes, but this can be generalized to M/G/1 type semi-Markov processes. (iii) We restricted ourselves to the moments of the distribution of the duration of busy periods, but this can be generalized to the moments of the joint distribution of the duration of a busy period and the number of transitions during the busy period.

3.8 Computing various performance measures

So far, our focus has been on computing the stationary probabilities in the RFB/GFB process. In this section, we discuss how the stationary probabilities can be used to compute other performance

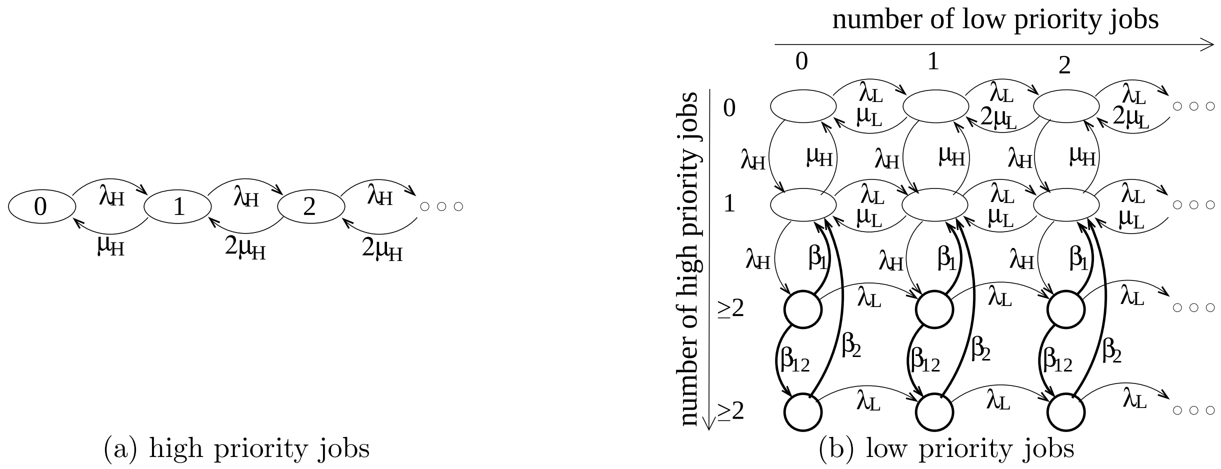


Figure 3.28: *Markov chains whose stationary probabilities are computed via DR in the analysis of an M/M/2 queue with two preemptive priority classes.*

measures. In Section 3.8.1, we consider the distribution of the number of jobs in the system and its moments. In Section 3.8.2, we consider the distribution of response time and its moments. Since the computation of these performance measures depends on particular multiserver systems and details of modeling, we will illustrate the approach via an example in the case of an M/M/2 queue with two preemptive priority classes (recall Figure 3.3), and briefly discuss how this can be applied to other multiserver systems.

Figure 3.28 shows the Markov chains whose stationary probabilities are computed via DR in the analysis of an M/M/2 queue with two preemptive priority classes. Figure 3.28(a) shows the background process, where state (level) ℓ corresponds to the state with ℓ high priority jobs for $\ell \geq 0$. Let $\pi_\ell^{(H)}$ be the stationary probability in state (level) ℓ of the background process. Observe that $\pi_\ell^{(H)}$ is the probability that there are ℓ high priority jobs in the system for $\ell \geq 0$. Figure 3.28(b) shows the 1D Markov chain reduced from the 2D Markov chain (FB process), where level ℓ consists of the states with ℓ low priority jobs for $\ell \geq 0$. Let $\vec{\pi}_\ell^{(L)}$ be the stationary probability vector in level ℓ of the 1D Markov chain shown in Figure 3.28(b). Observe that $\vec{\pi}_\ell^{(L)}\vec{1}$ is the probability that there are ℓ low priority jobs in the system for $\ell \geq 0$.

3.8.1 Distribution and moments of the number of jobs in the system

Deriving the distribution of the number of (high priority or low priority) jobs in the system and its moments is straightforward. Let N_H (respectively, N_L) be the number of high priority (respectively, low priority) jobs in the system. Then,

$$\Pr(N_H = n) = \pi_n^{(H)} \quad \text{and} \quad \Pr(N_L = n) = \vec{\pi}_n^{(L)}\vec{1}$$

for $n \geq 0$. Also, their r-th moments can be computed via

$$\mathsf{E}\left[(N_H)^r\right] = \sum_{\ell=0}^{\infty} \ell^r \pi_\ell^{(H)} \quad \text{and} \quad \mathsf{E}\left[(N_L)^r\right] = \sum_{\ell=0}^{\infty} \ell^r \vec{\pi}_\ell^{(L)}\vec{1}$$

for $r \geq 1$. As in (3.6), the infinite summation can be reduced to a finite summation, since the QBD processes in Figure 3.28 repeat after level $\hat{\ell} = 2$.

In general, the distribution of the number of jobs in the system that is modeled as an RFB/GFB process can be derived similarly, as long as the number of jobs is well defined for each state. For example, the number of jobs is well defined for each state in all the examples of RFB/GFB processes that we have provided in Section 3.4. Also, the distribution of the number of jobs in the queue (number of jobs that are waiting and not in service) and its moments can be computed similarly. However, note that the error in the higher moments are larger as we will see in Section 3.9, since we match only the first three moments of the "busy period" in DR.

3.8.2 Distribution and moments of response time

Let T_H (respectively, T_L) be the response time of the high priority jobs (respectively, low priority jobs). The mean response time follows immediately from $\mathsf{E}[N_H]$ and $\mathsf{E}[N_L]$ via Little's law:

$$\mathsf{E}[T_H] = \frac{\mathsf{E}[N_H]}{\lambda_H} \quad \text{and} \quad \mathsf{E}[T_L] = \frac{\mathsf{E}[N_L]}{\lambda_L}$$

where λ_H (respectively, λ_L) is the arrival rate of the high priority jobs (respectively, low priority jobs). However, as we discussed in Section 2.2, the distributional Little's law does not apply to per-class response time in an M/M/2 queue with two priority classes, since (high priority or low priority) jobs do not necessarily leave the system in the order of their arrivals. In fact, the higher moments of response time depend on details of which low priority job is preempted when a high priority job arrives and sees two low priority jobs in service: we assume that the low priority job that started to receive service *later* is preempted by the high priority job. Below, we illustrate a way to compute the distribution of per-class response time and its moments.

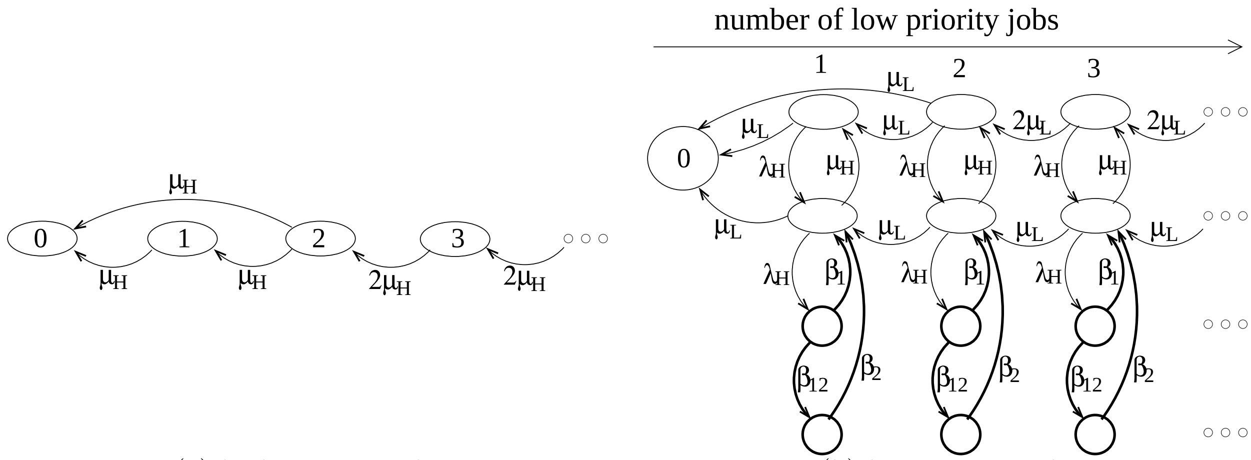
Response time of high priority jobs

We start with the simpler case of the high priority (class H) jobs. We derive the response time of the class H jobs by conditioning on the state that an arrival sees. Recall the Markov chain in Figure 3.28(a). By PASTA (Poisson arrivals see time averages), a class H arrival will see state ℓ with probability $\pi_\ell^{(H)}$ when it arrives. Let $\Pr(T_H \leq x \mid \ell)$ be the distribution function of the response time of the class H job that sees state ℓ when it arrives, and $\mathsf{E}[(T_H)^r \mid \ell]$ be its r-th moment. Then the distribution function of the response time of the class H jobs and its r-th moment are given by

$$\Pr(T_H \leq x) = \sum_{\ell=0}^{\infty} \Pr(T_H \leq x \mid \ell)\pi_\ell^{(H)} \quad \text{and} \quad \mathsf{E}[(T_H)^r] = \sum_{\ell=0}^{\infty} \mathsf{E}[(T_H)^r \mid \ell]\, \pi_\ell^{(H)}.$$

Thus, our goal is to derive $\Pr(T_H \leq x \mid \ell)$ and $\mathsf{E}[(T_H)^r \mid \ell]$ for each ℓ.

First, observe that a tagged (class H) arrival that sees state ℓ will cause the system state to change to $\ell + 1$ at that moment. We remove all the λ_H arcs from the Markov chain in Figure 3.28(a), so that there are no more arrivals. (Note that the behavior of the tagged arrival is not affected by the jobs that will arrive after the tagged arrival, since class H jobs are served in FCFS order.) This enables us to view the response time for the tagged arrival as the passage time of this modified Markov chain from state $\ell + 1$ to the state where the tagged arrival departs. The only complexity is in figuring out exactly in which state the tagged arrival departs.



(a) high priority jobs (b) low priority jobs

Figure 3.29: *Markov chains used to compute the response time in an M/M/2 queue with two preemptive priority classes.*

If the tagged arrival sees state $\ell = 0$, its response time is the same as its service time, which has exponential distribution with rate μ_H. If the tagged arrival sees state $\ell \geq 1$ (and the system state is changed to $\ell + 1$), the tagged arrival may depart when the modified Markov chain hits state 1 or state 0, depending on whether the tagged arrival is the last job to be completed or not. Note that by the memoryless property of exponential distributions, the tagged arrival is the last job to be completed with probability $\frac{1}{2}$. Thus, the response time of the tagged job is the passage time to go from state $\ell + 1$ to 0 with probability $\frac{1}{2}$, and the passage time to go from state $\ell + 1$ to 1 with probability $\frac{1}{2}$. Observe that the distribution of each of these passage times is simply a convolution of independent exponential distributions, and its moments can be derived easily.

Figure 3.29(a) summarizes the above argument. The response time of a job that sees state ℓ upon its arrival is the passage time from state $\ell + 1$ to state 0 in the Markov chain in Figure 3.29(a). Observe that Figure 3.29(a) is obtained from Figure 3.28(a) by removing all the transitions with λ_H's and splitting the transition from state 2 to state 1 (with rate $2\mu_H$) into two transitions, one to state 1 and the other to state 0 (each with rate μ_H).

Response time of low priority jobs

Next, we derive the response time of the low priority (class L) jobs, again by conditioning on the state that an arrival sees via PASTA. Recall the Markov chain in Figure 3.28(b). Let $\Pr(T_L \leq x \mid i, \ell)$ be the distribution function of the response time of the class L job that sees state (i, ℓ), i.e. phase i of level ℓ, when it arrives, and $\mathsf{E}\left[(T_L)^r \mid i, \ell\right]$ be its r-th moment. The distribution function of the response time of the class L jobs and its r-th moment are then given by

$$\Pr(T_L \leq x) = \sum_{\ell=0}^{\infty} \sum_i \Pr(T_L \leq x \mid i, \ell)(\vec{\pi}_\ell^{(H)})_i \tag{3.11}$$

$$\mathsf{E}\left[(T_L)^r\right] = \sum_{\ell=0}^{\infty} \sum_i \mathsf{E}\left[(T_L)^r \mid i, \ell\right](\vec{\pi}_\ell^{(H)})_i. \tag{3.12}$$

Thus, our goal is to compute $\Pr(T_L \leq x \mid i, \ell)$ and $\mathsf{E}\left[(T_L)^r \mid i, \ell\right]$ for each (i, ℓ).

Observe that a tagged (class L) arrival that sees state (i, ℓ) will cause the system state to change to $(i, \ell+1)$ at that moment. We remove all the λ_L arcs from the Markov chain in Figure 3.28(b), so that there are no more class L arrivals. (Note that the behavior of the tagged arrival is not affected by the class L jobs that will arrive after the tagged arrival, since class L jobs are served in FCFS order and we assume that the "youngest" class L job is preempted by a high priority job.) This enables us to view the response time for the tagged arrival as the first passage time of this modified Markov chain from state $(i, \ell+1)$ to the state where the tagged arrival departs. The only complexity is in figuring out exactly in which state the tagged arrival departs.

If the tagged arrival sees state $(i, \ell = 0)$ for some i, its response time is the passage time from state $(i, \ell = 1)$ to (any state in) level $\ell = 0$ in the modified Markov chain, since the tagged arrival is the only class L job in the modified system. If the tagged arrival sees a state in level $\ell \geq 1$ (and the system state is changed to level $\ell + 1$), the tagged arrival may depart when the modified Markov chain hits level 1 or level 0, depending on whether the tagged arrival is the last job to be completed or not. We will compute the response time by conditioning on the first state in level $\ell = 1$ that we hit in the modified Markov chain. Note that the first state in level $\ell = 1$ is either $(0, \ell = 1)$ or $(1, \ell = 1)$, since there are no backward transitions in the bottom two rows in the (modified) Markov chain.

If $(1, \ell = 1)$ is the first state in level $\ell = 1$, we must have gotten there from $(1, \ell = 2)$. This implies that the remaining class L job is the tagged arrival, since class L jobs are served in FCFS order and we assume that the "youngest" class L job is preempted by a high priority jobs. Thus, in this case the response time of the tagged arrival that sees (i, ℓ) upon its arrival is the passage time from $(i, \ell + 1)$ to $(1, \ell = 1)$ plus the passage time from $(1, \ell = 1)$ to level 0.

If $(0, \ell = 1)$ is the first state in level $\ell = 1$, we must have gotten there from state $(0, \ell = 2)$. This implies that the remaining class L job is equally likely to be the tagged arrival or not by the memoryless property of exponential distributions. Thus, in this case the response time of the tagged arrival that sees (i, ℓ) upon its arrival is the passage time from $(i, \ell+1)$ to $(0, \ell = 1)$ with probability $\frac{1}{2}$, and the passage time from $(i, \ell + 1)$ to $(0, \ell = 1)$ plus the passage time from $(0, \ell = 1)$ to level 0 with probability $\frac{1}{2}$.

Figure 3.29(b) summarizes the above argument. The response time of a job that sees state (i, ℓ) upon its arrival is the passage time from state $(i, \ell+1)$ to state 0 in the Markov chain in Figure 3.29(b). Observe that Figure 3.29(b) is obtained from Figure 3.28(b) by removing all the transitions with λ_L's and splitting the transition (with rate $2\mu_L$) from state $(0, \ell = 2)$ to state $(0, \ell = 1)$ into two transitions (each with rate μ_H), one to state $(0, \ell = 0)$ and the other to state 0.

The passage time from any state (i, ℓ) to state 0 in the Markov chain in Figure 3.29(b) is a PH distribution (as defined in Section 2.2), where state (i, ℓ) is the initial state and state 0 is the absorbing state. Note that the distribution function and moments of the PH distribution can be computed via Proposition 4. In fact, by (3.11), the distribution of the response time of the low priority jobs can be represented as a *single* PH distribution (a mixture of PH distributions is a PH distribution by Proposition 3), where the initial state is (i, ℓ) with probability $(\overrightarrow{\pi_{\ell+1}})_i$ for all i and $\ell \geq 0$, and state 0 is the absorbing state. To compute the distribution function of the single PH distribution and its moments via Proposition 4, the state space of the Markov chain in Figure 3.29(b) needs to be truncated. One can simply truncate it at some large level, or one can use DR to reduce the 1D Markov chain into a Markov chain on a finite state space as we reduce Figure 3.4(b) into Figure 3.5.

Alternatively, the moments of the passage time (from level ℓ to level 0) can be computed using the method provided in Section 3.7, which can be more computationally efficient than the approach via Proposition 4. Note that there are no forward transitions in the modified Markov chain, and the expressions in Section 3.7 are significantly simplified: specifically, matrices $\boldsymbol{\mathcal{F}}^{(\ell)}$ and $\boldsymbol{\mathcal{F}}_r^{(\ell)}$ are zero matrices in the modified Markov chain for all ℓ and r.

In general, the *mean* response time in the system that is modeled as an RFB/GFB process can be derived via Little's law, as long as the number of jobs is well defined for each state. However, the response time may or may not be represented as a passage time in a modified Markov chain, and its distribution and moments may or may not be analyzed via the approach introduced in this section. Also, the mean number of jobs in the queue (number of jobs that are waiting and not in service) can be derived via Little's law, but its distribution and moments may or may not be derived via an analysis of passage times. Also, note again that the error in the higher moments may be larger, since we match only the first three moments of the "busy period" in DR.

3.9 Validation

In this section, we numerically evaluate the accuracy and running time of DR as well as its approximations, DR-PI and DR-CI, through the analysis of the preemptive priority queue (introduced in Section 3.4) and the size-based task assignment with cycle stealing under immediate dispatching, `SBCS-ID`, (introduced in Section 3.4). Note that the number of *classes* corresponds to the number of processes constituting an RFB process in the analysis of both the preemptive priority queue and `SBCS-ID`. Throughout we assume that the arrival process is Poisson. We will see that

- DR has a very small error in predicting the first order metric such as mean delay and mean queue length. Specifically, the error in DR is within 3% for a range of parameters (loads, service demand distributions, and the number of classes, m) in the analysis of both the preemptive priority queue and `SBCS-ID`.

- The error in DR is slightly larger but is still small in predicting the second order metric such as the second moment of the queue length. Specifically, the error is within 7% for a range of parameters in the analysis of `SBCS-ID`.

- The error in DR-PI and DP-CI is slightly larger than DR but is still small for both the first and second order metrics. Specifically, the error is within 6% (respectively, 10%) in predicting the first (respectively, second) order metric for a range of parameters in the analysis of `SBCS-ID`.

- DR is computationally quite efficient when it is applied to 2D Markov chains, i.e. RFB processes with $m = 2$ (FB processes) and GFB processes. Specifically, in the analysis of the preemptive priority queue, the running time of DR is ≤ 0.1 seconds for up to $k = 10$ servers.

- The running time of DR can grow quickly as the number of processes, m, in an RFB process increases. Specifically, in the analysis of the preemptive priority queue with $k = 2$ servers, the running time of DR is within 30 seconds for up to $m = 11$ classes (processes); however, in the analysis of `SBCS-ID`, DR becomes computationally prohibitive with $m \geq 5$ classes (processes).

- DR-PI and DR-CI can reduce the running time of DR significantly. Specifically, in the analysis of `SBCS-ID`, the running time of DR-CI is within 30 seconds for up to $m = 20$ classes (processes).

In all our experiments below, we use the algorithm in Figure 3.12 to compute R and G matrices, which we need in the analysis via DR, DR-PI, and DR-CI, and we set the error bound at $\epsilon = 10^{-6}$. All the experiments are run on a 1 GHz Pentium III with 512 MB RAM, using Matlab 6 running on Linux.

3.9.1 Accuracy of dimensionality reduction

In this section, we evaluate the accuracy of DR, DR-PI, and DR-CI. In all our experiments below, the error in an analysis (DR, DR-PI, or DR-CI) is defined as a relative difference against simulated value:

$$\text{error} = 100 \times \frac{(\text{value by analysis}) - (\text{value by simulation})}{(\text{value by simulation})} \quad (\%).$$

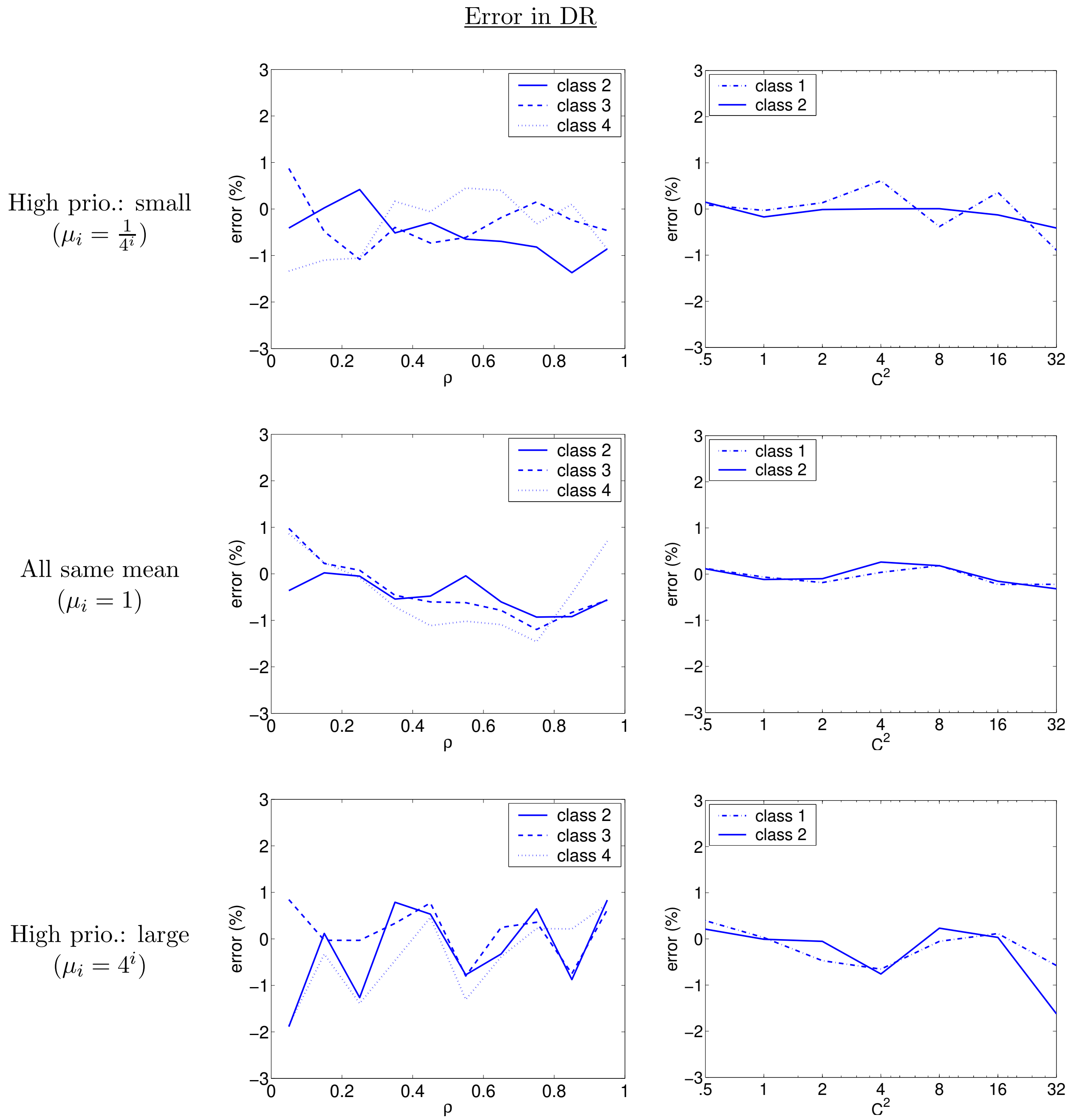
Simulation is kept running until the simulation error becomes less than 1% with probability 0.95 [85, 168, 193]. More precisely, our simulation keeps generating "sample values," where the i-th "sample value," v_i, is computed as a sample mean of the metric of interest in the i-th run with a large number (say ten million; this number can be different for different parameter settings) of events, until one of the following two conditions are satisfied[8]: (i) $i \geq 30$ and $1.96\sigma \leq 0.01\mu$, where μ and σ are the sample mean and the sample variance of the i "sample values," respectively; (ii) $2 \leq i < 30$ and $\sigma/(\sqrt{0.05i}) \leq 0.01\mu$. Then, the sample mean of the i "sample values" is output as the simulated value.

Unless otherwise stated, we approximate the "busy period" distribution by a two-phase PH distribution. In our parameter settings, two phases are sufficient to match the first three moments of the most of the busy periods. Even in a few cases where two phases are not sufficient, we are able to choose a two phase PH distribution whose first three moments are very close to those of the busy periods. Hence, using more phases to match the exact three moments does not appear to improve the accuracy appreciably.

We first evaluate the error in DR in predicting the mean delay in the preemptive priority queue. Here, delay of a job denotes the time the job spends waiting in the queue, i.e. delay does not include the service time. We choose to evaluate the error in mean delay rather than the error in mean response time (delay plus service time), since the error is larger in mean delay.

Figure 3.30 shows the error in DR in the analysis of the mean delay in the preemptive priority queue. Column (a) shows the error in the analysis of an M/M/2 queue with four priority classes, as a function of load, ρ. (Class 1 is not shown in the figure for clarity, but the analysis of class 1 does not involve DR, and its error is comparable to the error in classes 2-4.) The load of each class is one-quarter of the total load, ρ, i.e. each class has the same load. The service rate of class i jobs is set $\mu_i = \alpha^i$ for $\alpha = \frac{1}{4}, 1$, or 4. Namely, $\alpha < 1$ implies small high priority jobs, and $\alpha > 1$ implies large high priority jobs. The three rows in Figure 3.30 correspond to different values of α. Column (b) shows the error in the analysis of an M/PH/2 queue with two priority classes. Here, we assess the effect of C^2, the squared coefficient of variability of the job size distribution (both the high and low priority jobs), defined as the variance divided by the square of the mean. The load of each class is fixed at 0.3, and hence the total load is $\rho = 0.6$. We use a two phase Coxian$^+$ PH distribution

[8] The first condition is based on the assumption that the sample mean has a normal distribution when $i \geq 30$, and the second condition is based on the Chebyshev's inequality on the sample mean. In both cases, the true variance is assumed to equal to the sample variance.



(a) M/M/2 with 4 priority classes (b) M/PH/2 with 2 priority classes

Figure 3.30: *Accuracy of DR in the analysis of preemptive priority queues. (a) shows the error in the analysis of an M/M/2 with four priority classes as a function of load, ρ, where the load of class i is $\rho_i = \rho/4$ for each i. (b) shows the error in the analysis of an M/PH/2 with two priority classes as a function of C^2 of the service demands of both high and low priority jobs, where the total load is fixed at $\rho = 0.6$ and the load is balanced between the classes.*

(see Section 2.2) as the job size distribution[9], where the service rate of class i jobs is set $\mu_i = \alpha^i$ for $\alpha = \frac{1}{4}, 1$, or 4 as before.

Figure 3.30 shows that the error in DR is within 2% for all classes, for all ρ's, for all α's, and for all C^2's. On average, DR tends to underestimate the mean delay only by 0.2% of the simulation. Overall, the error in DR is quite small in predicting the first order metric (such as mean delay, mean response time, and mean queue length). We evaluate the error in DR when it is applied to other multiserver systems, and find that the error is within 3% in predicting the first order metric for a range of loads, job size distributions, and other parameters (see [67, 68, 70, 151, 152, 155]).

Next, we evaluate the accuracy of DR-PI and DR-CI as well as DR in predicting the first *two* moments of queue length distributions, through the analysis of `SBCS-ID`. We choose to evaluate the moments of queue length distributions, rather than delay or response time distributions, since the analysis of higher moments of delay and response time distributions are complicated, although possible as we show in Section 3.8. Throughout, we assume that class i jobs arrive according to a Poisson process with rate λ_i, and their service demand has an exponential distribution with rate μ_i for $1 \le i \le m$.

Figure 3.31 shows the error in DR, DR-PI, and DR-CI in predicting the first two moments of queue length distributions. Here, we assume that the load made up of each class is fixed at 0.8 (i.e. $\rho_i = \frac{\lambda_i}{\mu_i} = 0.8$), and μ_i is chosen such that class 1 jobs are the shortest and class m jobs are the longest, specifically $\mu_i = 2^i$. Column (a) shows the error in the first moment, and column (b) shows the error in the second moment. In the top row, the number of classes (and servers) is $m = 4$, and all of DR, DR-PI, and DR-CI are evaluated. In the middle row, the number of servers is $m = 6$, and here only DR-PI and DR-CI are evaluated, since evaluation via DR is computationally too expensive with $m = 6$ (see Section 3.9.2). In the bottom row, the number of servers is $m = 12$, and here only DR-CI is evaluated[10]. The horizontal axis shows the "server name," where server name i denotes the i-th server, and hence corresponds to the i-th process. Note that the scale of the vertical axis is different between (a) and (b).

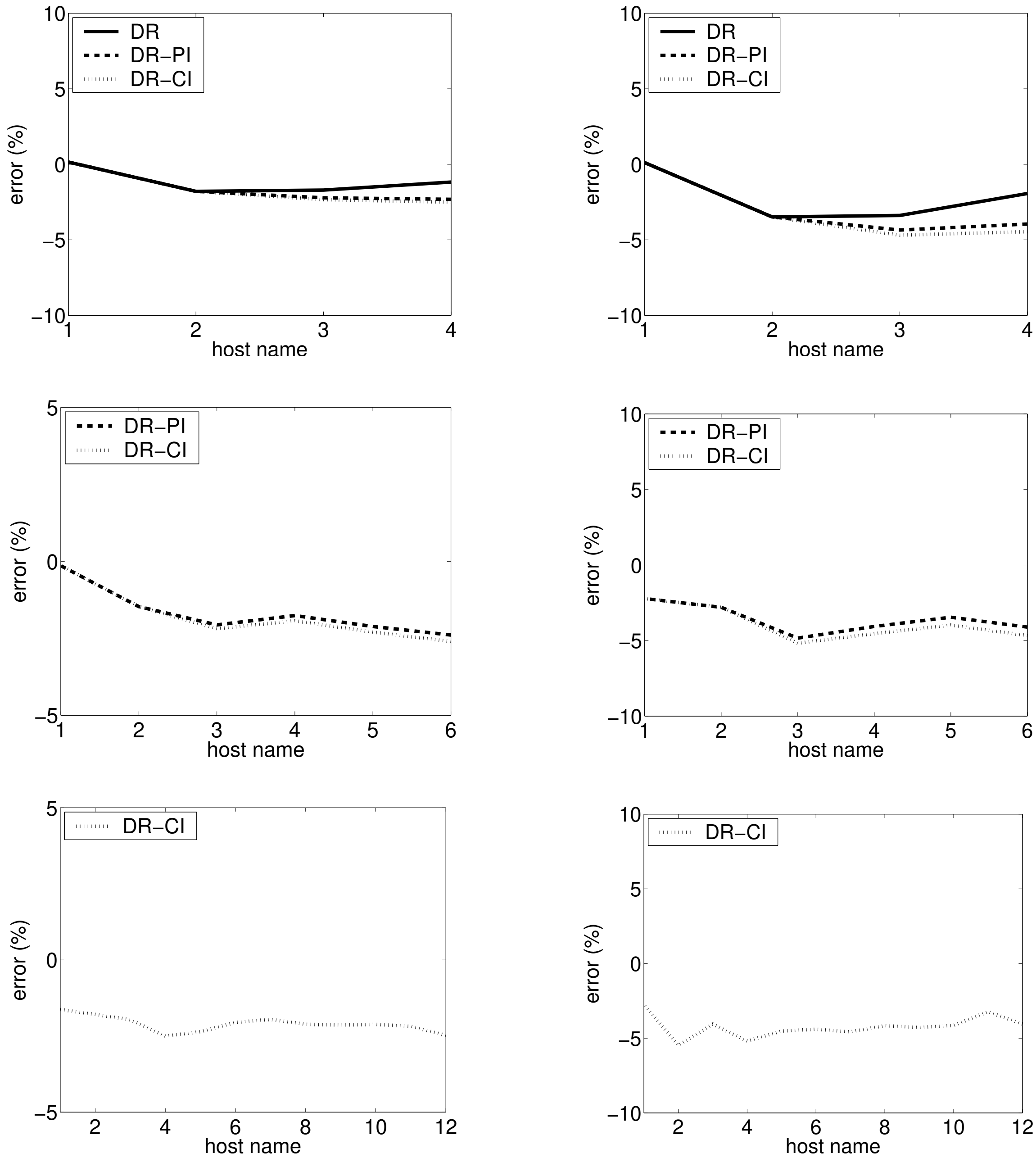
Figure 3.31 shows that the error in DR, DR-PI, and DR-CI is quite small in predicting the first moment, and it is always within 3%. The error in the second moment is slightly larger, but it is bounded by 6% for all of DR, DR-PI, and DR-CI. In both the first and second moments, the error does not appear to increase appreciably as the number of classes (processes) increases. Top two rows suggest that DR-PI tends to have only slightly larger error than DR, and DR-CI tends to have only slightly larger error than DR-PI. The small error in DR-PI and DR-CI suggests that the duration of the busy period of server i affects the queue length of server $i + 1$ primarily by the marginal distribution of the busy period duration, and the dependency in the sequence of busy period durations has a smaller effect.

Figure 3.32 shows how the system load affects the accuracy of RDR in predicting the first two moments of queue length distributions. Here, the error in DR, DR-PI, DR-CI is plotted at three different loads, $\rho_i \equiv \frac{\lambda_i}{\mu_i} = 0.6$, 0.8, or 0.9 for each i. Throughout, μ_i is fixed at $\mu_i = 2^i$. The figure

[9] To determine the parameters of the two phase Coxian$^+$ PH distribution, the third moment needs to be specified as well. Here, we choose the third moment, so that the normalized second and third moments, m_2 and m_3, satisfy $m_3 = 2m_2 - 1$. See Section 2.5 for an intuition for this choice; in particular, the exponential distribution and the Erlang distribution satisfy $m_3 = 2m_2 - 1$.

[10] Although DR-CI can be evaluated with $m > 12$ (see Figure 3.36), it is hard to have simulation converge with $m > 12$. Note that the fraction of arrivals of jobs with the largest size is very small: on average, there is only one arrival of a job with the largest size while there are 2^{m-1} arrivals of jobs with the smallest size. When $m = 12$, more than 1,000,000,000 events are needed for simulation to converge.

Error in DR, DR-PI, and DR-CI



(a) *First* moment of queue length

(b) *Second* moment of queue length

Figure 3.31: *Accuracy of DR, DR-PI, and DR-CI in predicting the first two moments of the queue length distributions, where* $\mu_i = 2^i$ *and* $\rho_i = 0.8$ *for each* i.

suggests that the error in DR, DR-PI, and DR-CI tends to become larger at higher load. Also, the error in DR-PI and DR-CI tends to become larger at a slightly faster rate (as the load increases) than the error in DR. For example, when $\rho_i = 0.6$ for $1 \leq i \leq 4$, the error in DR, DR-PI, DR-CI is within 1% (respectively, 2%) in predicting the first (respectively, second) moment of the queue length distribution. On the other hand, when $\rho_i = 0.9$ for $1 \leq i \leq 4$, the error in DR is within 3% (respectively, 5%) for the first (respectively, second) moment, while the error in RDR-PI and RDR-CI is within 4% (respectively, 7%) for the first (respectively, second) moment.

Figure 3.33 shows how the relative difference in the mean job size of each class affects the accuracy of DR, DR-PI, and DR-CI in predicting the first two moments of the queue length distribution. Here, the error in DR, DR-PI, DR-CI is plotted at four different job size configurations: $\mu_i = 4^i$, $\mu_i = 2^i$, $\mu_i = 2^{-i}$, or $\mu_i = 4^{-i}$. Throughout, $\rho_i = 0.8$ is fixed for $1 \leq i \leq 4$. Figure suggests that the error (in predicting both first and second moments) is greater when idle cycles of the server for longer jobs is stolen. We conjecture that this is primarily due to the fact that the busy period of server $i-1$, which is the sojourn time in levels ≥ 1 in the $(i-1)$-th process, is relatively larger as compared to the service time and interarrival time at server i if server $i-1$ is serving longer jobs, and hence the error in approximating/ignoring the distribution and dependency of the busy period has larger effect in predicting the queue length distribution at server i. In general the error in DR, DR-PI, and DR-CI tends to be an increasing function of α, when $\mu_i = \alpha^i$. Also, the error in DR-PI and DR-CI tends to become larger at a slightly faster rate (as α increases) than the error in DR. For example, when $\alpha = \frac{1}{4}$, the error in DR, DR-PI, and DR-CI is negligible in both the first and second moments of the queue length distribution. On the other hand, when $\alpha = 4$, the error in DR is within 3% (respectively, 7%) in predicting the first (respectively, second) moment, while the error in DR-PI and DR-CI is within 6% (respectively, 10%) in predicting the first (respectively, second) moment.

Finally, we study the effect of ignoring higher moments (specifically, the second and third moments) of the busy period distributions when they are approximated by PH distributions in DR, DR-PI, and DR-CI. Figure 3.34 shows the error in DR, DR-PI, and DR-CI when the busy period distributions are approximated by exponential distributions matching only the first moments. Other parameter settings are exactly the same as in Figure 3.31. Overall, ignoring the variability (and the third moment) of the busy periods can lead to as high an error as 10% in predicting the first moment of the queue length distribution and 20% in predicting the second moment.

3.9.2 Running time of dimensionality reduction

In this section, we evaluate the running time of DR, DR-PI, and DR-CI. We measure the running time as CPU time on a 1 GHz Pentium III with 512 MB RAM, using Matlab 6 running on Linux. In all our experiments, we approximate the "busy period" distribution by a two-phase PH distribution.

We first discuss the running time of DR when it is applied to analyze the preemptive priority queue, specifically an M/M/k queue with m priority classes. Figure 3.35 shows the running time (a) as a function of the number of servers, k, and (b) as a function of the number of classes, m. Here, we assume that all the m classes have the identical service demand distribution (the exponential distribution distribution with rate μ), and have the identical arrival rate, λ. We choose μ and λ such that the total load is 0.5, i.e. $\frac{m\lambda}{k\mu} = \frac{1}{2}$. Although not shown, the running time becomes longer when the load is higher or when the error bound ϵ is set smaller.

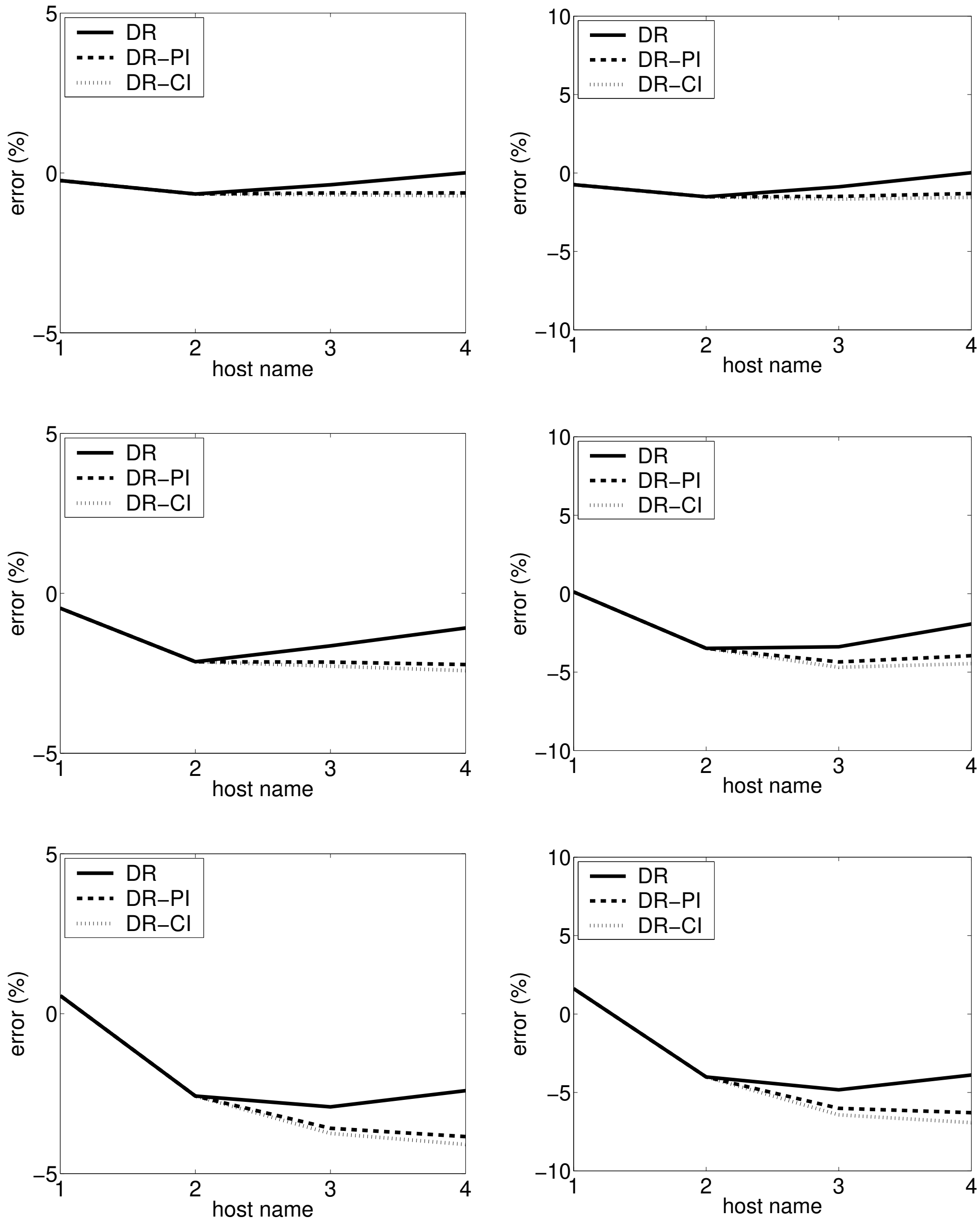
The solid line in Figure 3.35(a) illustrates the computational efficiency of the DR when it is applied to an FB process (2D Markov chain), showing the running time of DR for two priority

Effect of load in error

$\rho_i = 0.6$

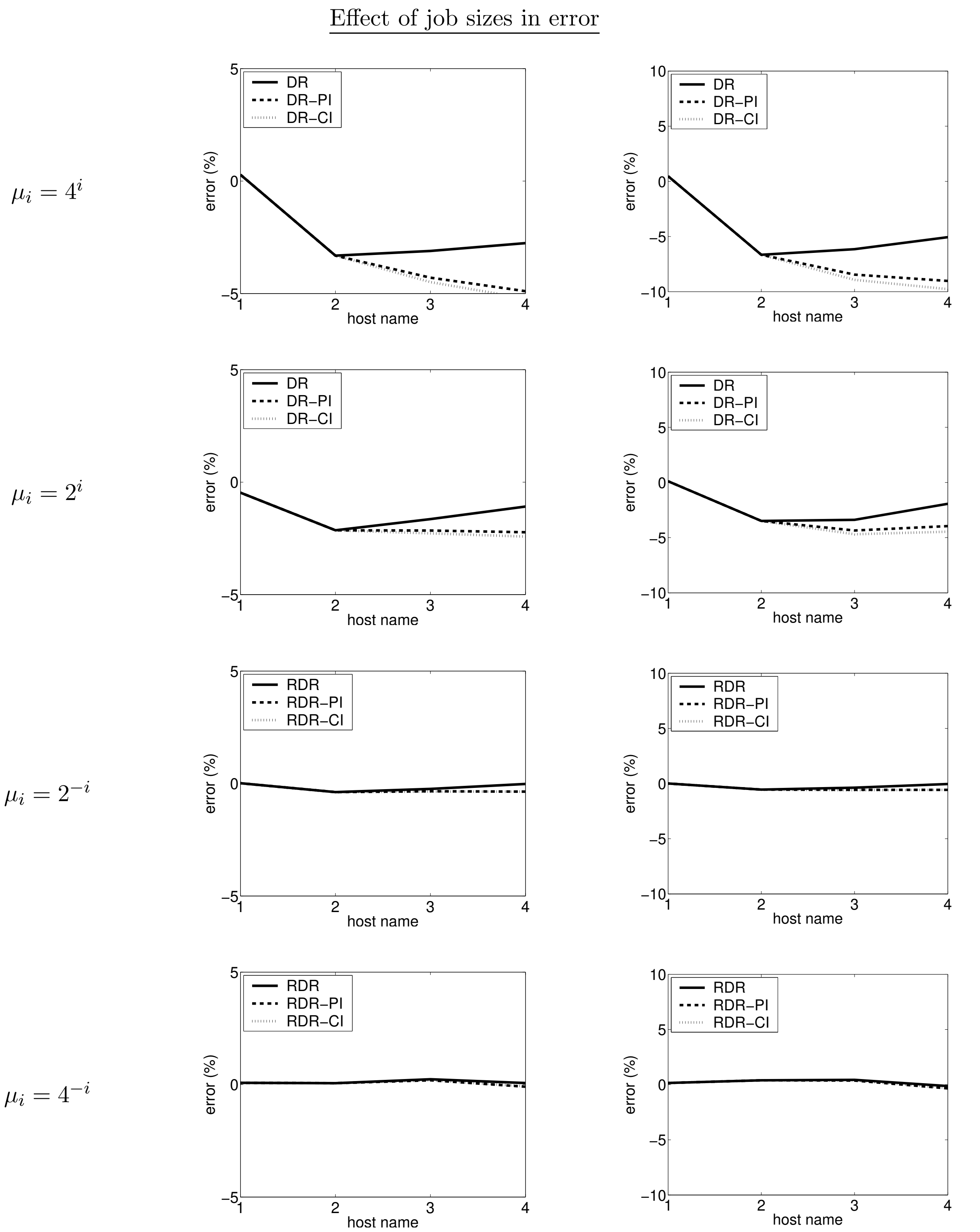
$\rho_i = 0.8$

$\rho_i = 0.9$



(a) *Second* moment of queue length (b) *First* moment of queue length

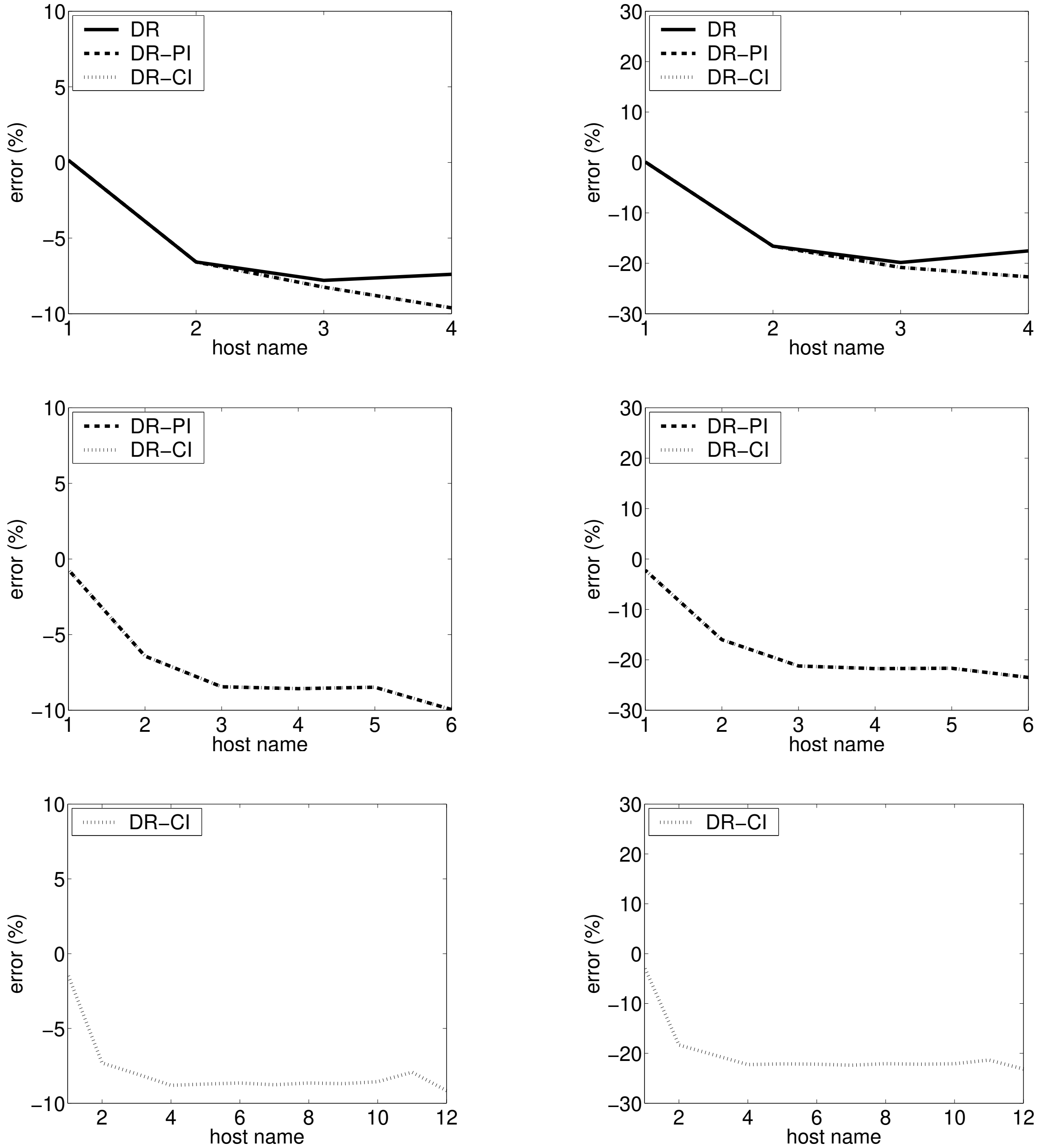
Figure 3.32: *Accuracy of DR, DR-PI, and DR-CI at different loads, where* $\mu_i = 2^i$.



(a) *First* moment of queue length (b) *Second* moment of queue length

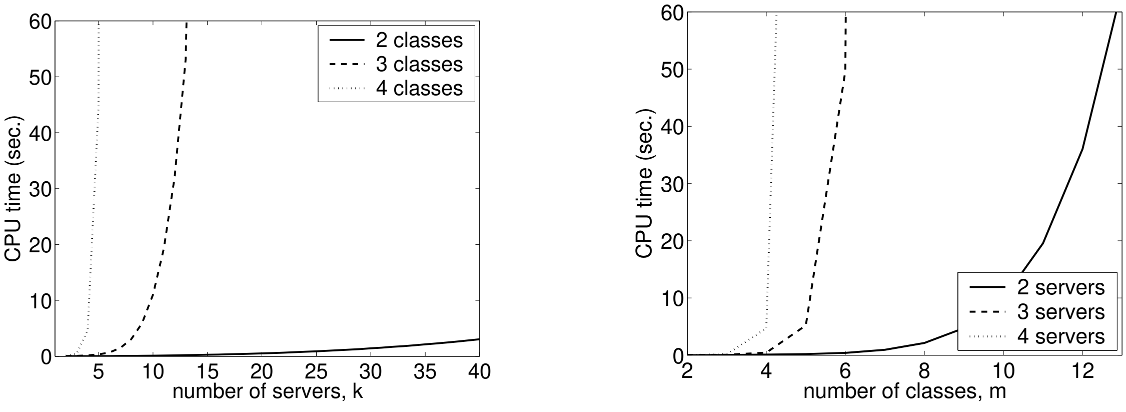
Figure 3.33: *Accuracy of DR, DR-PI, and DR-CI at different job size configurations, where* $\rho_i = 0.8$ *for each* i*.*

Error in matching only one moment



(a) *First* moment of queue length (b) *Second* moment of queue length

Figure 3.34: *Error in DR, DR-PI, and DR-CI when the busy period is approximated by an exponential distribution matching only the first moment.*



(a) (b)

Figure 3.35: *Running time of DR, when applied to an analysis of the preemptive priority queue with k servers and m classes (a) as a function of k and (b) as a function of m.*

classes ($m = 2$) as a function of the number of servers, k. Recall the DR analysis of the preemptive priority queue with two classes ($m = 2$), illustrated in Figure 3.3. Most of the running time is devoted to computing the stationary probabilities in the 1D Markov chain (Figure 3.3(d)) which is reduced from the FB process. When there are k servers, the 1D Markov chain has $k + 2$ states in each level (level $\geq k$), and the running time increases as the number of states in each level in the 1D Markov chain increases. As shown in the figure, the running time increases quite slowly with the number of servers, and in fact, the running time is within 30 seconds for up to $k = 89$ servers. Overall, DR is computationally efficient when it is applied to 2D Markov chains, i.e. RFB processes with $m = 2$ (FB processes) and GFB processes. For most of the analysis of the 2D Markov chains in Chapters 5-7, the running time of DR is within 0.1 seconds.

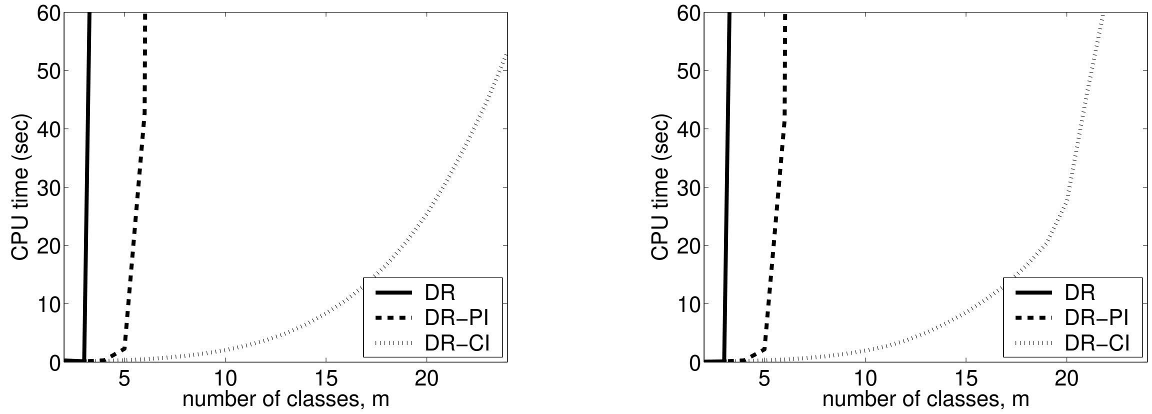
The solid line in Figure 3.35(b) shows the running time of DR as a function of the number of classes, m, when the number of servers is $k = 2$. This is the running time of DR when it is applied to mD Markov chains. Again, the running time increases gradually as m increases, and it is within 40 seconds for up to $m = 12$ servers (mD Markov chains). In fact, as we will see later, the running time is bounded by a polynomial in m.

However, the rest of Figure 3.35 suggests that the running time of DR increases quickly when *both* k and m increase. In general, the running time of DR is dominated by the time to compute the stationary probabilities in the 1D Markov chain that tracks the exact number of lowest priority jobs (class 1 jobs), i.e. the first (foreground) process. This 1D Markov chain has $\binom{m+k-3}{k-1}^2$ types of busy periods of higher priority jobs (jobs of classes 2 to m), and hence the 1D Markov chain has

$$\binom{m+k-2}{k-1} + 2\binom{m+k-3}{k-1}^2$$

states in each level[11]. That is, the number of states in each level in the 1D Markov chain is polynomial

[11]Observe that $\binom{m+k-2}{k-1}$ is the number of states with at most $k-1$ higher priority jobs (i.e., the number of ways to assign at most $k-1$ higher priority jobs to k homogeneous servers), and $\binom{m+k-3}{k-1}$ is the number of states with $k-1$ higher priority jobs.



(a) Stealing idle cycles of server for longer jobs (b) Stealing idle cycles of server for shorter jobs

Figure 3.36: *Running time of DR, DR-PI, and DR-CI, when applied to an analysis of* SBCS-ID.

in k if m is constant ($\Theta(k^m)$), and it is polynomial in m if k is constant ($\Theta(m^k)$); however, it is exponential in k and m if neither k nor m is constant.

Note that, in the analysis of the preemptive priority queue, the number of different types of busy periods of higher priority jobs in the m-th process (foreground process) does *not* depend on the number of phases in the PH distributions that are used in the analysis of higher priority jobs, i.e. in the analysis of the i-th (background) process for $1 \leq i \leq m-1$. However, this is not in general true in the analysis of the RFB process. Specifically, in the analysis of `SBCS-ID`, the number of phases in the PH distributions that are used in the analysis of the background processes affects the number of different types of busy periods in the foreground processes. As a result, the running time of DR increases more quickly as the number of classes, m, increases in the analysis of `SBCS-ID`. This is exactly a situation where an approximation of DR is needed.

Figure 3.36 shows the running time of DR, DR-PI, and DR-CI as a function of the number of classes, m, when they are applied to the analysis of `SBCS-ID`. In all the plots, we assume that the load made up of each class is fixed at 0.8 (i.e. $\rho_i \equiv \frac{\lambda_i}{\mu_i} = 0.8$), and μ_i is chosen such that class 1 jobs are the shortest and class m jobs are the longest ("stealing idle cycles of a server for longer jobs"; specifically, $\mu_i = 2^i$), or μ_i is chosen such that class 1 jobs are the longest and class m jobs are the shortest ("stealing idle cycles of a server for shorter jobs"; specifically, $\mu_i = 2^{-i}$). Although not shown, the running times of DR, DR-PI, and DR-CI tend to increase when the load is higher of when the error bound ϵ is set smaller.

In both cases, the evaluation of RDR becomes prohibitive when $m \geq 5$. The running time of DR-PI also quickly grows, and its evaluation becomes prohibitive when $m \geq 8$ in both cases. The running time of DR-CI grows far more slowly than DR and DR-PI. We are able to evaluate DR-CI for up to 21 servers in less than a minute. The running time of DR, DR-PI, and DR-CI can be compared to the number of states in each level (phases) in the 1D Markov chain that tracks the exact number of class 1 jobs (foreground process). In DR, the number of phases $S_{DR(m)}$, grows double exponentially; specifically, $S_{DR(m)}$ can be determined by the following recursive formula: $S_{DR(m)} = S_{DR(m-1)} + 2(S_{DR(m-1)})^2$ and $S_{DR(1)} = 1$. In DR-PI, the number of phases grows exponentially: $S_{PI(m)} = 3^{m-1}$. In DR-CI, the number of phases grows linearly: $S_{CI(m)} = 2m + 1$.

3.10 Concluding remarks

In this chapter, a new analytical approach, dimensionality reduction (DR), is proposed for analyzing multidimensional Markov chains (Markov chains on a multidimensionally infinite state space). The key idea in DR is in approximating an infinite portion of a Markov chain, which often corresponds to the "busy period" of a system, by a collection of PH distributions, which match the first three moments of the duration of the busy period conditioned on how the Markov chain enters and exits from the busy period. As DR involves approximations, we have validated its accuracy against simulation. Overall, we find that the error in DR is quite small: specifically, the error in DR is within 3% (often within 1%) for the first order metric (such as mean delay and mean queue length) and within 7% for the second order metric (such as the second moment of queue length) for a range of parameters. To reduce the computational complexity of DR, we have introduced two approximations of DR: DR-PI and DR-CI. Although these approximations slightly worsen the accuracy, they can significantly reduce the running time.

DR allows us to analyze the performance of a multiserver system whose behavior is modeled as a multidimensional Markov chain. Since it is a nontrivial task to determine whether DR can be applied to a particular multiserver system and how, we formally define a class of multidimensional Markov chains called RFB/GFB processes (recursive foreground-background processes or generalized foreground-background processes) that can be analyzed via DR. The definition of RFB/GFB processes makes it easier to determine whether a given multiserver system can be analyzed via DR, and our analysis of RFB/GFB processes enables one to analyze the multiserver system by simply modeling it as an RFB/GFB process. In fact, many multiserver systems with resource sharing or job prioritization can be modeled as RFB/GFB processes, and we will analyze the performance of some of these multiserver systems in Part II:

- multiserver systems with multiple priority classes (see Chapter 4),
- size-based task assignment with cycle stealing under immediate dispatching, `SBCS-ID`, for server farms (see Chapter 5),
- size-based task assignment with cycle stealing under central queue, `SBCS-CQ`, for server farms (see Chapter 5),
- threshold-based policies for reducing switching costs in cycle stealing (see Chapter 6), and
- various threshold-based policies for the Beneficiary-Donor model (see Chapter 7).

Beyond an analysis of multiserver systems, DR and ideas in DR have broad applicability in computer science, engineering, operations research, etc., where Markov chains on large state spaces are involved. The ideas in DR can be used to reduce the size of the state space from infinite to finite or from large to small. For example, many optimization problems in stochastic environment can be formulated as Markov decision problems [159], but determining the optimal solution (optimal policy) is often prohibitive due to a large state space. The ideas in DR may be used to reduce the size of the state space in Markov decision processes, so that the computational complexity in solving the Markov decision problems is reduced.

In response to increased request, DR as well as its approximations, DR-PI and DR-CI, as described in this chapter are largely implemented, and the latest implementation is made available at an online code repository:

http://www.cs.cmu.edu/~osogami/code/.

Future directions

Interesting future directions include establishing a theoretical guarantee (bound) on the accuracy of DR-based analyses (DR, DR-PI, and DR-CI) and increasing the accuracy or DR-based analyses. Although we validate the accuracy of DR-based analyses numerically, no theoretical guarantee is proved. Note that the accuracy of DR-based analyses depends on how the busy period is approximated by a collection of PH distributions. The moment matching algorithm developed in Chapter 2 enables matching the first three moments of the busy period for the first time, but matching more moments may increase the accuracy. DR can also be made as accurate as desired by setting the "aggregation" level κ higher (recall that DR approximates the set of states in levels $\geq \kappa$ by a collection of PH distributions). Increased accuracy of DR would be particularly needed in computing the higher moments or distribution function of various performance measures, as we have observed that the error tends to increase for higher moments computed via DR.

Another interesting future direction is an extension of DR for analyzing more complex multiserver systems. Although we apply DR in the analysis of the RFB/GFB processes in this chapter, the idea in DR has a broad applicability beyond the RFB/GFB processes. Below, we provide examples of possible extensions of DR.

For example, throughout this chapter, we assume that service demands have PH distributions, but in some cases, they can be extended to general distributions. For example, in the analysis of the system with threshold-based policy for reducing switching costs in cycle stealing, the donor's service demand and switching back time can be extended to general distributions [151, 152]. This extension relies on a (well known) analysis of the busy period in an M/GI/1 queue with a setup time. In fact, the idea in DR can be applied to the RFB/GFB processes with a quite broad class of background processes. The analysis of the RFB/GFB processes relies on the analysis of the "busy period" in a QBD process via Neuts' algorithm; however, Neuts' algorithm applies to a much broader class of Markov chains, namely the M/G/1 type semi-Markov process. Fully utilizing Neuts' algorithm, DR could be applied to the RFB/GFB process with much less restrictions on the background process.

Restrictions on the foreground process can also be relaxed. Our restrictions on the foreground process guarantee that the 1D Markov chains reduced from the RFB/GFB processes are QBD processes. However, the QBD process is not the only Markov chain that allows analytical tractability. For example, the M/G/1 and G/M/1 type processes can also be analyzed efficiently via matrix analytic methods [111]. Thus, the restrictions on the foreground process can be relaxed accordingly.

Further, the GFB process is limited to 2D Markov chains, but this can be extended to higher dimensions, as we extend the FB process to the RFB process. Finally, we remark that the multidimensional Markov chain that we analyze via DR need not have multidimensionally *infinite* state space. DR can reduce a Markov chain that have a *large* number of states in multiple dimensions into a Markov chain that have a small number of states in each dimension, since matrix analytic methods and Neuts' algorithm apply to QBD processes with finite number of levels as well.

Part II

Performance analysis of multiserver systems

Chapter 4

Configuring multiserver systems with multiple priority classes

How many servers are best?

In this chapter, we address a fundamental problem in designing multiserver systems: how many servers are best? Dimensionality reduction (DR) allows us to study this problem not only when jobs are served in the order of their arrivals (FCFS) but also when there are priorities among jobs. Results of our analysis illuminate principles on how the number of servers affects the performance of multiserver systems.

4.1 Introduction

An important problem in designing multiserver systems is how to set up system configurations, given a limited amount of resources and user requirements. For example, a system architect may want to decide whether he/she should buy a few fast (but expensive) servers or many slow (but cheap) servers to minimize mean response time, given a limited amount of budget. In this chapter, we study this problem in particular settings: *Is it preferable to use a single fast server of speed s, or k slow servers each of speed s/k to minimize mean response time? What is the optimal k?* Although our analysis via DR extends to more general settings, taking into account the actual costs of servers as a function of their speed, this would complicate the analysis and obscure the insights into the fundamental nature of multiserver systems. Our goal in addressing the above problem is to illuminate principles on how the number of servers affects the performance of multiserver systems, which are useful in multiserver configurations.

The question of "how many servers are best" in exactly our settings has been asked by a stream of research [120, 130, 131, 172, 175, 188], but all of this work is limited to the case where jobs are served in the order of their arrivals (first come first served, FCFS). However, real-world systems are often not simply FCFS, where all jobs have equal importance. Rather, there is often inherent scheduling in the system to allow high priority jobs to move ahead of low priority jobs. For example, the high priority jobs may carry more importance, e.g. representing users who pay more. Alternatively, high priority jobs may simply have small service demand, since giving priority to short jobs reduces mean response time overall. Either way, priorities are fundamental to many modern systems.

This motivates the question of what the optimal system configuration is (a few fast servers or many slow servers) in a setting where there are different priority classes. We specifically assume that the high priority jobs and the low priority jobs arrive according to independent Poisson processes, their service demands have phase type (PH) distributions (as defined in Section 2.2), and jobs within each class are served in FCFS order. With these assumptions, the system behavior can be modeled as a foreground-background (FB) process, as discussed in Section 3.4, and it allows us to analyze this system via DR, for the first time.

In particular, DR allows us to address the following questions:

1. Under what conditions are multiple slow servers preferable to a single fast server? Is the optimal number of servers sensitive to changes in the relative loads of the priority classes, changes in the variability of the service time distributions, etc.?

2. Does the answer to "how many servers are best" differ for the different priority classes? E.g., does the lower priority class prefer more or fewer servers than that preferred by the higher priority class?

3. How does the optimal number of servers in a *dual* priority system differ from the case when all jobs have been aggregated into a *single* priority class?

4. If one chooses a non-optimal number of servers, how does that affect the overall mean response time and the per-class mean response time?

We also compare the analysis via DR with existing approximations in the literature, and find that the existing approximations can lead to qualitatively misleading conclusions regarding the optimal number of servers. Since existing approximations have poor accuracy and DR becomes computationally expensive as the number of servers and priority classes increases, we propose a new approximation based on DR, which we refer to as DR-A (DR with class aggregation). DR-A allows us to efficiently analyze the performance of multiserver systems with *many* priority classes. We will study the error in the existing approximations and DR-A extensively, and find that the error in DR-A is within 5% for a range of parameters, which is much smaller as compared to the existing approximations.

The rest of this chapter is organized as follows. In Section 4.2, we discuss prior work dealing with our question of "how many servers are best?" In Section 4.3, we answer question 1 above in the case of an M/PH/k/FCFS queue with a single priority class. In Section 4.4, we address all the four questions above for the case of an M/PH/k dual-priority queue. In Section 4.5, we propose DR-A and study the accuracy of DR-A and existing approximations.

4.2 State of the art in the optimal number of servers

The question of how many servers are best has a long history, but all the work is limited to FCFS (single priority class). Further, the question has not been fully resolved even for the case of FCFS. Note that the study of this question has been limited to FCFS primarily because an analysis of multiple priority systems involves multidimensional Markov chains, and exact (or nearly exact) analysis is known only for exponential job size distributions (see Section 3.3). In describing the results below, we will assume that the performance metric of interest is mean response time rather than mean delay (the time a job waits in the queue before receiving service), since in general an infinite number of servers minimizes mean delay [34, 37] (also, Scheller-Wolf [172] shows that the tail of the stationary delay distribution is significantly reduced by increasing the number of servers under highly variable service demand distributions).

As early as 1958, Morse observed that for an M/M/k/FCFS system, the optimal number of servers is one ($k = 1$) [131]. In 1970, Stidham formalized and generalized the observation by Morse[188]; Stidham showed that a single server is optimal in a G/Er/k/FCFS system, namely when the service demand has an Erlang distribution, including an exponential distribution, or is deterministic, for a general arrival process. Stidham's result is extended by Brumelle to the system where the service demand is at most as variable as the exponential random variable; i.e., a single server is optimal in a G/G/k/FCFS system where the service demand distribution has the squared coefficient variation at most one ($C^2 \leq 1$) [31]. For general service demand distributions, only results for limiting cases are known. Specifically, the work by Reiman and Simon [161] can be used to show that a single server ($k = 1$) is optimal in the light traffic limit (when servers are almost always idle) [120], and the work by Iglehart and Whitt [79] implies that a multiserver system (GI/GI/k/FCFS) and its single server counterpart (GI/GI/1/FCFS) coincides in the heavy traffic limit (when servers are almost always busy).

All the above work proves that a single server ($k = 1$) is the best for special cases. However, Stidham also discusses that the optimality of a single server does not appear to hold for highly variable job size distributions [188]. Very recently, Molinero-Fernandez et. al. [130] consider the question of how many servers are best in an M/HT/k single priority system, where HT denotes a *heavy-tailed* service distribution. To answer this question, they approximate a heavy-tailed distribution with a bimodal distribution, and observe that in general multiple servers are better than a single server.

In summary, although the question of "how many servers are best?" has been addressed in a number of papers in the context of FCFS, results are limited to special cases such as light and heavy traffic limits and low variable and heavy tailed distributions. This motivates us to further study this question under a wide range of loads and job size variabilities in the context of FCFS before studying priority systems.

4.3 How many servers are best in FCFS system?

In this section, we study the question of "how many servers are best?" in an M/PH/k/FCFS queue (single priority class) with respect to minimizing mean response time. Matrix analytic methods allow us to evaluate the mean response time in a wide rage of loads and job size variability in this setting (see Section 3.2), and this complements the prior work with respect to "how many servers are best" in FCFS systems.

Figure 4.1(a) shows the optimal number of servers as a function of the load and variability of the job size, where the job size has a two-phase PH distribution. All of our results are expressed as a function of the variability of the job size distribution (specifically, its squared coefficient of variation, C^2)[1], and the server load, ρ. While other factors, e.g. the exact form of the job size distribution, might affect our results, we posit that load and variability be the most relevant factors. As is proved by Stidham [188], a single server minimizes mean response time when $C^2 = 1$ (exponential distribution) or when $C^2 = 0.5$ (Erlang-2 distribution). Observe, however, that under high job size

[1] To determine the parameters of the two phase PH distribution, the third moment needs to be specified as well. Here, we choose the third moment, so that the normalized second and third moments, m_2 and m_3, satisfy $m_3 = 2m_2 - 1$. See Section 2.5 for an intuition for this choice; in particular, the exponential distribution and the Erlang distribution satisfy $m_3 = 2m_2 - 1$.

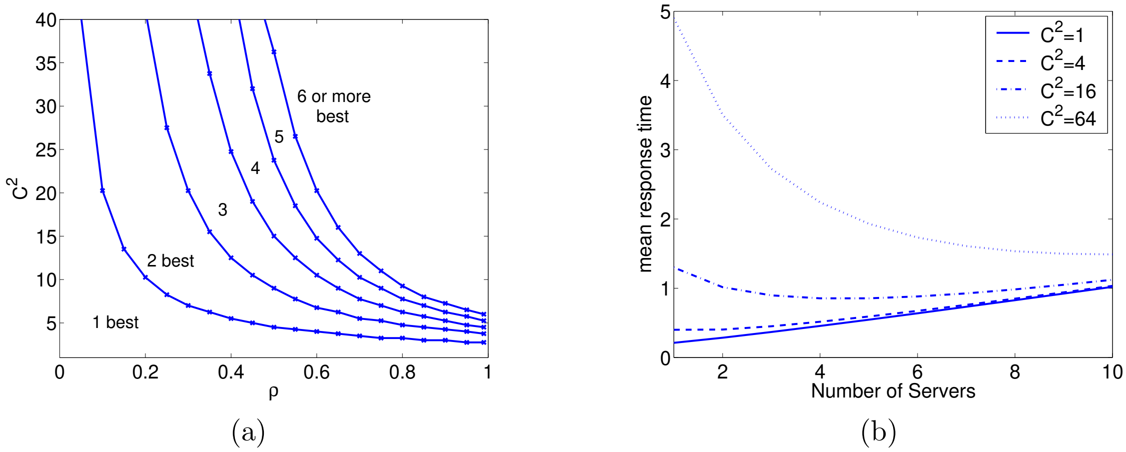


Figure 4.1: *How many servers are best in a single server (FCFS) system? (a) The optimal number of servers as a function of the load, ρ, and the squared coefficient of variation of the job sizes, C^2. (b) Mean response time as a function of the number of servers at various job size variabilities ($C^2 = 1, 4, 16, 64$).*

variability and/or high load, the optimal number of servers is more than 1; *we prefer multiple slow servers to a single fast server*. For example, at load $\rho = 0.4$ and $C^2 = 20$, we see that three servers are best. Computations are only done for up to six servers — the level curves shown will continue into the upper right portion of the plot if a larger number of servers is considered.

Figure 4.1(b) shows that for any particular job size variability, $C^2 > 1$, having a larger number of slower servers may reduce the mean response time up to a point, after which further increasing the number of servers increases the mean response time. To understand why, note that by increasing the number of servers (while maintaining fixed total capacity), we are allowing short jobs to avoid queueing behind long jobs — specifically, an arriving short job is more likely to find a server free. Thus, *increasing the number of servers mitigates the impact of job size variability*, hence improving performance. If the number of servers is too high however, servers are more likely to be idle, under-utilizing the system resources.

4.4 How many servers are best in priority system?

In this section, we study the question of "how many servers are best?" in an M/PH/k queue with two priority classes with respect to minimizing mean response time. The M/PH/k queue with two priority classes can be modeled as a foreground-background (FB) process, as discussed in Section 3.4, and the analysis of the FB process via DR (see Chapter 3) provides mean response time of high priority jobs and low priority jobs, respectively.

Throughout our evaluations, we will consider a range of load typically shown on the horizontal axis, and a range of variability in the high priority job sizes, typically shown on the vertical axis. When we vary the load, we vary only the arrival rates and keep the job size distributions. In all the results shown, the size of the high priority jobs has a two phase Coxian$^+$ PH distribution (as defined in Section 2.2). The mean job size for the high priority jobs is held fixed at $\mathsf{E}[X_H] = 1$, and its squared coefficient of variation is varied ($C^2 \geq 1$)[2]. On the other hand, throughout, we assume that the size of the low priority job has an exponential distribution ($C^2 = 1$), since the effect of its variability is less interesting to study, as it affects only the performance of low priority jobs. We consider three different mean sizes, $\mathsf{E}[X_L]$, for the low priority jobs: (i) $\mathsf{E}[X_L] = 1$, (ii) $\mathsf{E}[X_L] = 10$, and (iii) $\mathsf{E}[X_L] = 1/10$. Note that the mean service time changes depending on how many servers are in the system (1 fast server or k slow servers) so that the systems are comparable. The values specified above for $\mathsf{E}[X_H]$ and $\mathsf{E}[X_L]$ are the values for the maximum number of servers used in each plot and the mean sizes for each of the other number of servers is scaled appropriately. Finally, note that when we compare a dual priority system with its single priority counterpart (high priority jobs and low priority jobs are aggregated into a single class and served in FCFS order), the job size of the single aggregate class has a PH distribution that is a mixture of the two-phase PH distribution (for the high priority job size) and the exponential distribution (for the low priority job size).

Figures 4.2-4.4 illustrate the results of our analysis for the three cases: (i) $\mathsf{E}[X_H] = 1$ and $\mathsf{E}[X_L] = 1$, (ii) $\mathsf{E}[X_H] = 1$ and $\mathsf{E}[X_L] = 10$, and (iii) $\mathsf{E}[X_H] = 1$ and $\mathsf{E}[X_L] = 1/10$, respectively. For each figure, column (a) shows the case where the load made up by high and low priority jobs is equal ($\rho_H = \rho_L$), and column (b) shows the case where $\rho_H < \rho_L$. We also discuss, but omit showing, the case where $\rho_H > \rho_L$. For each figure we consider both the case of dual priority classes and the case of a single aggregate class.

Equal mean sizes

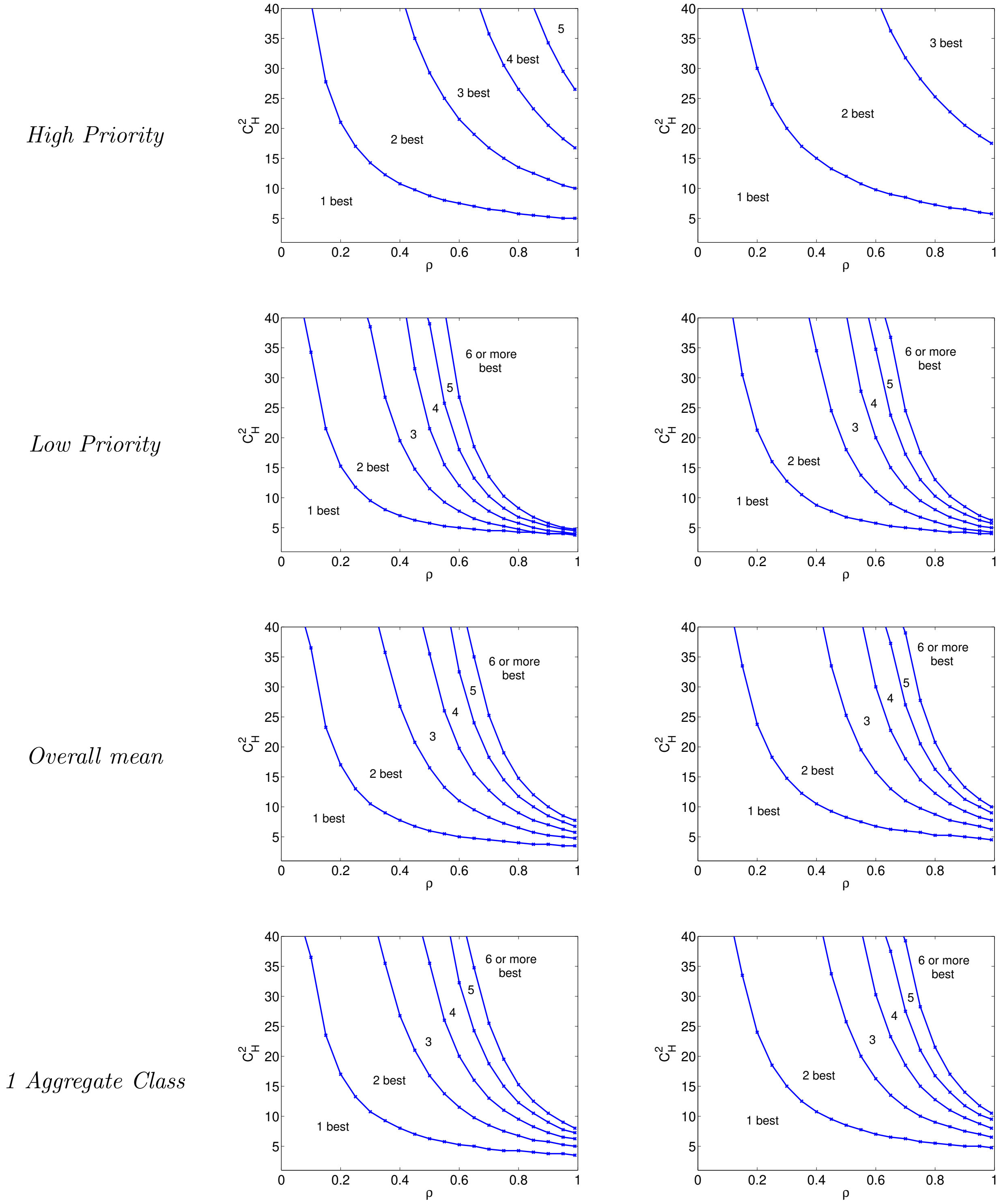
In the topmost plot of Figure 4.2(a), we see that the high priority jobs often prefer multiple servers. This is to be expected based on our results in Section 4.3, since the high priority jobs are served in FCFS order and are not affected by the low priority jobs.

Surprisingly, however, the second plot in column (a) shows that the number of servers preferred by low priority jobs is much greater than that preferred by high priority jobs. Low priority jobs prefer more servers because low priority jobs are preempted by high priority jobs and thus their mean response time improves with more servers, which allows them to escape from the dominance of high priority jobs. That is, *multiple servers reduce the impact of prioritization on the mean response time of low priority jobs.*

The preferred number of servers with respect to the overall mean response time (the average of all jobs, including both low and high priority jobs) is shown in the third plot in column (a), where we see that the number of servers preferred by the overall mean, as expected, is a hybrid of that preferred by low and high priority jobs. Note though that this hybrid is more weighted toward the preference of low priority jobs because adding extra servers only hurts high priority jobs a small amount, whereas adding extra servers helps low priority jobs enormously. Interestingly, the number of servers preferred with respect to the overall mean is nearly identical to that shown for a single aggregate class of high and low priority jobs, shown in the bottom most plot in column (a). To understand why, observe that all jobs in this case have the same mean, and thus prioritizing in favor of some of them over others does not affect the mean response time greatly. Even though the classes have different variabilities, that is a smaller-order effect. This will not remain true in general.

[2]To determine the parameters of the two phase PH distribution, the third moment needs to be specified as well. Here, we choose the third moment, so that the normalized second and third moments, m_2 and m_3, satisfy $m_3 = 2m_2 - 1$. See Section 2.5 for an intuition for this choice; in particular, the exponential distribution and the Erlang distribution satisfy $m_3 = 2m_2 - 1$.

(i) $\mathsf{E}[X_H] = 1$ and $\mathsf{E}[X_L] = 1$



(a) $\rho_H = \rho_L$ (b) $2\rho_H = \rho_L$

Figure 4.2: *How many servers are best when the two priority classes have the same mean job size?*

Figure 4.2(b) shows that when the high priority jobs make up a smaller fraction of the load, the same trends are evident, but the specific numbers are quite different. For example, the topmost plot in column (b) shows that the number of servers preferred by high priority jobs is much fewer, since there are fewer high priority jobs in the systems, and they benefit from a fewer but faster servers (as many slow servers lead to low utilization). Less obvious is the fact that the number of servers preferred by low priority jobs in column (b) is also fewer than that in column (a). This follows from the same reasoning: the low priority jobs are strongly affected by prioritization (interruptions by high priority jobs), and with fewer high priority jobs, there are fewer interruptions and thus fewer servers are needed to avoid queueing behind high priority jobs.

Since both the high and low priority jobs in column (b) prefer fewer servers than in column (a), it makes sense that their overall mean also indicates that fewer servers are desired (the third plot of column (b)). This third plot also matches the bottom most plot in column (b) consisting of a single aggregate class, as in column (a).

Not shown in Figure 4.2 is the case where high priority jobs comprise more of the load. In this case, both classes prefer more servers and, therefore, the average of the two classes also prefers more servers. The reason for this is the converse of the above situation: since the load made up by the high priority jobs is high, large high priority jobs are likely to block other high priority jobs in a single server system. Further, the low priority jobs are preempted more frequently by high priority jobs and therefore also want more servers to alleviate the effect. Again the single aggregate class is very similar to the two priority class with respect to the optimal number of servers.

High priority class has smaller mean

In Figure 4.3, we continue to hold the mean high priority job size at 1 and increase the low priority job size to 10. Note that giving high priority jobs (smaller jobs) preference lower the overall mean response time, in this case.

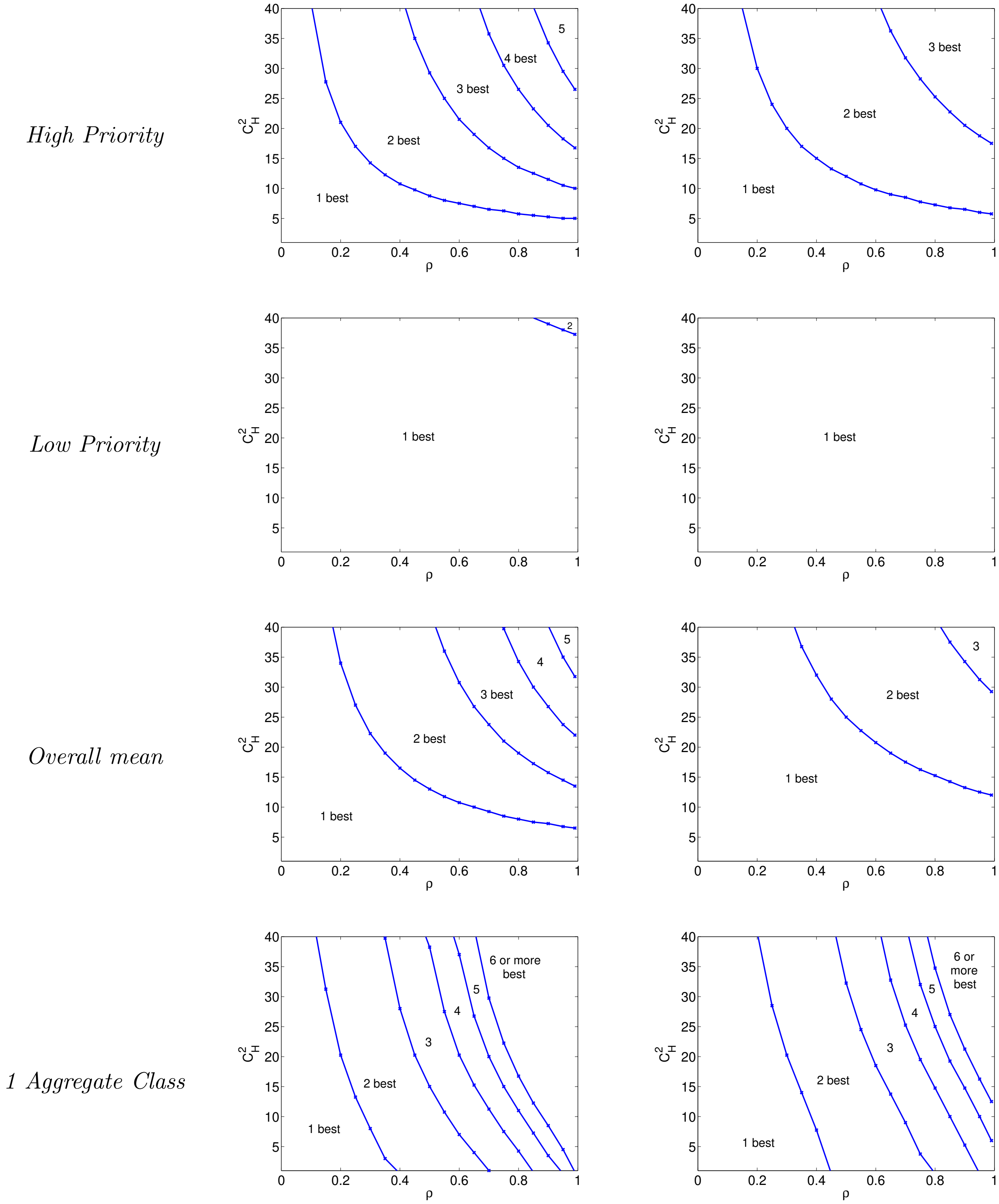
Notice that the preferred number of servers for the high priority jobs is identical to that in Figure 4.2 because the high priority job size distribution is unchanged. However, the number of servers preferred by low priority jobs is now very different: they almost always prefer only one server. This is explained by observing that the effect of multiple servers in reducing the impact of prioritization on low priority jobs is small when the high priority jobs are smaller, and that *a few fast servers has an advantage over many slow servers with respect to utilization.*

The overall preferred number of servers, averaged over the two priority classes, is again a hybrid of the preferences of the two classes, but this time is biased toward the preferences of the high priority jobs because they are in the majority, implying a preference for fewer servers than the corresponding graph in Figure 4.2. Recall that adding servers is a way to help small jobs avoid queuing behind larger jobs. Since we are in the case where small jobs have priority already, we do not need the effect of multiple servers. Thus, in this case, *priority classes can be viewed as a substitute for adding more servers.*

Comparing the overall preferred number of servers for the case of dual priorities with that preferred under a single aggregate class, we see that there is a significant difference in preferences. The single aggregate class prefers many more servers. This is in contrast to the case where the high priority jobs and the low priority jobs have the same mean size (Figure 4.2). This again is a consequence of the fact that in this case prioritization is a substitute for increasing the number of servers.

Figure 4.3(b) illustrates the same graphs for the case where the high priority jobs comprise less

(ii) $\mathsf{E}[X_H] = 1$ and $\mathsf{E}[X_L] = 10$



(a) $\rho_H = \rho_L$ (b) $2\rho_H = \rho_L$

Figure 4.3: *How many servers are best when the high priority jobs have a smaller mean job size?*

of the total load. The trends are the same as in column (a); however the preferred number of servers is significantly smaller in all figures. This follows from the same argument as that given for Figure 4.2(b). In the case where high priority jobs make up a greater proportion of the total load (not shown), the number of servers preferred is, as before, always higher than in column (a).

High priority class has larger mean

In Figure 4.4(a), we once again hold the mean high priority job size fixed at 1 and now assume the low priority job sizes have a mean size of 1/10. That is, we are giving priority to large jobs, which worsen the overall mean response time.

Once again, in the topmost plot of column (a), we see that the preferred number of servers for high priority jobs is unaffected, since the high priority mean job size distribution has not changed. The low priority jobs, shown in the second plot of column (a), have vastly different preferences from the prior case. Here the low priority jobs prefer a large number of servers. Because the low priority jobs are much smaller than the high priority jobs, they want more servers in order to avoid being blocked behind the (large) high priority jobs.

The preferred number of servers for the overall mean response time in the dual-priority system, shown in the third plot of column (a), is again a hybrid of the preferences of the low and high priority jobs, but this time is strongly biased toward the low priority jobs because there are more of them. Notice therefore, that the number of servers preferred is much greater in this case. Comparing this with the single class aggregate, we see that the single class prefers slightly fewer servers than the dual class overall mean. Since larger jobs have higher priority in this case, the multiple servers provide a huge benefit to (small) low priority jobs.

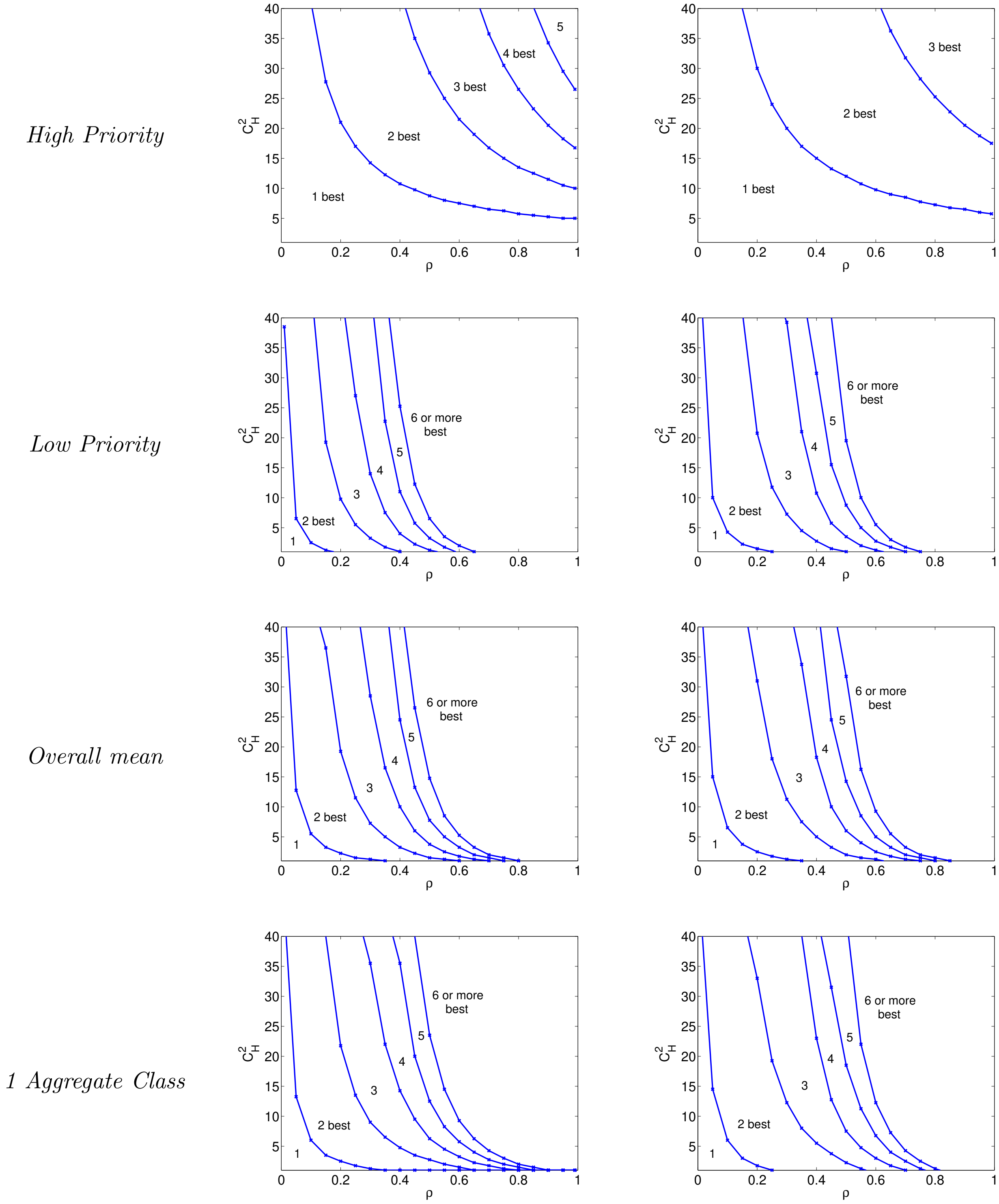
Figure 4.4(b) illustrates the same graphs for the case where the high priority jobs comprise less of the total load. The trends are the same as in column (a); however the preferred number of servers is significantly smaller in all figures. This follows from the same argument as that given for Figure 4.2(b). In the case (not shown) where high priority jobs make up a greater proportion of the total load, more servers are preferable.

Response time as a function of the number of servers

Figure 4.5 shows the mean response time as a function of the number of servers. In column (a), the high priority jobs have a smaller mean size, and in column (b), the low priority jobs have a smaller mean size. The differences in the number of servers preferred by each class, which we have discussed in the previous figures, can be read off of Figure 4.5 by observing that each of the plots in the figure have a "U-shape" and the bottom of the "U" indicates the optimal number of servers.

The key points made in Figure 4.5 are that: (i) the mean response time of both priority classes is sensitive to the number of servers and (ii) increasing the number of servers may reduce mean response time up to a point, but making the number of servers too large increases mean response time (the curve forms a "U-shape"). This figure also reinforces the prior messages that *the greater the variability of the high priority jobs, the greater the number of servers needed to mitigate this variability*, and how prioritization is performed has a large impact on the optimal number of servers (the optimal number of servers is much larger in column (b) than in column (a)).

(iii) $\mathsf{E}[X_H] = 1$ and $\mathsf{E}[X_L] = 1/10$



(a) $\rho_H = \rho_L$ (b) $2\rho_H = \rho_L$

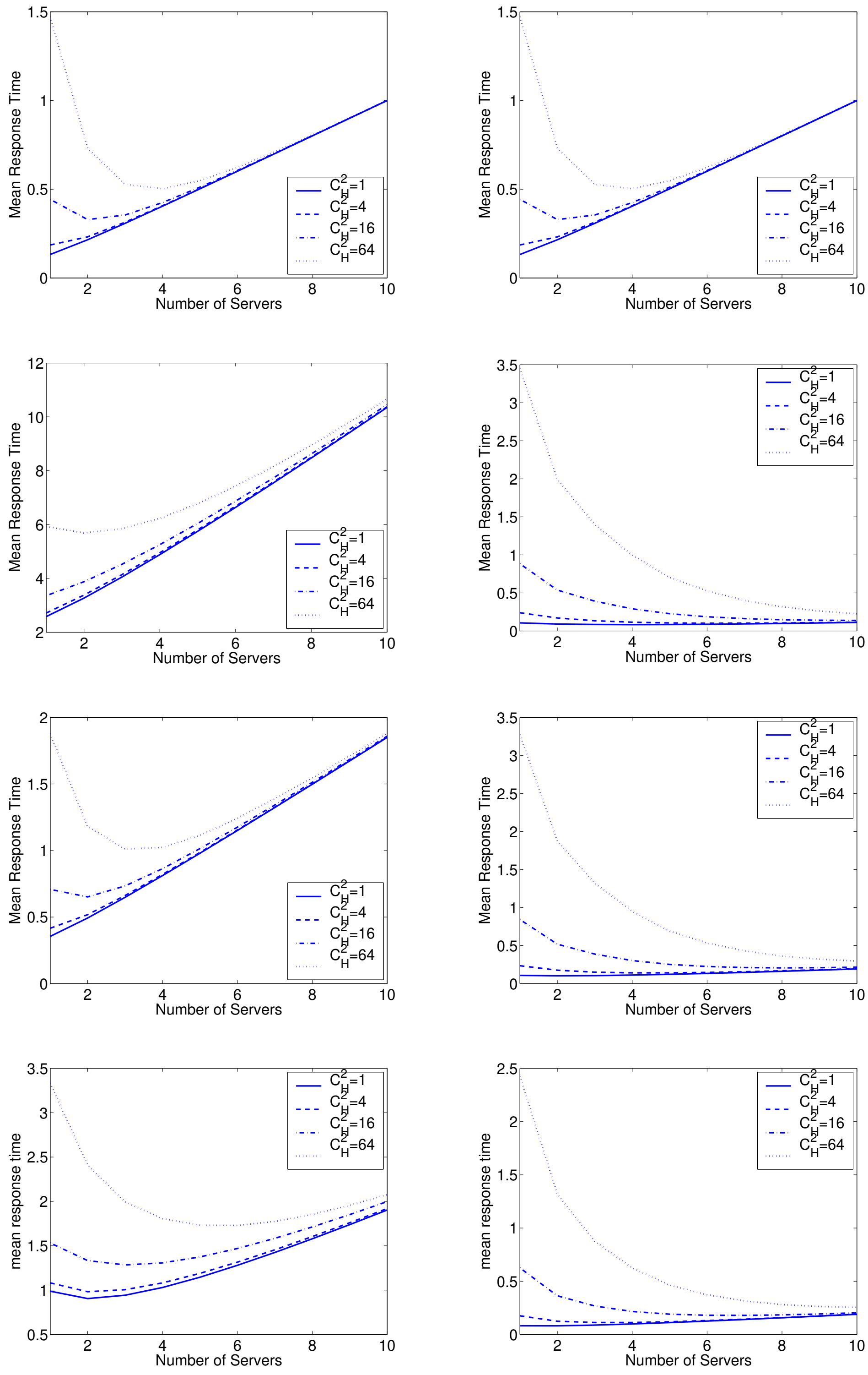
Figure 4.4: *How many servers are best when the high priority class has a larger mean job size?*

High Priority

Low Priority

Overall mean

1 Aggregate Class



(a) $\mathsf{E}[X_H] = 1$ and $\mathsf{E}[X_L] = 10$ (for $k = 10$)

(b) $\mathsf{E}[X_H] = 1$ and $\mathsf{E}[X_L] = \frac{1}{10}$ (for $k = 10$)

Figure 4.5: *Mean response time as a function of the number of servers. Here, $\rho_H = \rho_L = 0.3$ is fixed.*

Summary

To summarize the above observations, the optimal number of servers depends on the loads and job size variabilities, and whether a few fast servers are preferable to many slow servers can be understood by combining the following set of rules of thumb.

1. A few fast servers have an advantage over many slow servers with respect to high utilization. When the load is lower, the difference in the utilization becomes greater, and as a result the advantage of a few fast servers becomes greater (for both the high and low priority jobs).

2. Many slow servers have an effect of reducing the impact of *job size variability* on mean response time, since they allow small jobs to avoid queueing behind large jobs. When the variability of job sizes is higher and/or when the load is higher, this effect of many slow servers becomes greater, and as a result the advantage of many slow servers becomes greater (for both the high and low priority jobs).

3. Many slow servers have an effect of reducing the impact of prioritization on low priority jobs. When the mean and/or variability of the higher priority job size are larger and/or when the load of the higher priority job is higher, this effect of many slow servers becomes greater, and as a result the advantage of many slow servers becomes greater (for low priority jobs).

The third rule has an interesting implication on the overall mean response time of a dual priority system, since prioritizing small jobs has the same effect as having multiple servers in the sense that both allow small jobs to avoid queueing behind large jobs. Thus, when small jobs have higher priority, a smaller number of (fast) servers is more preferable. Also, since prioritizing large jobs has the counter effect, a larger number of (slow) servers is preferable when large jobs have higher priority.

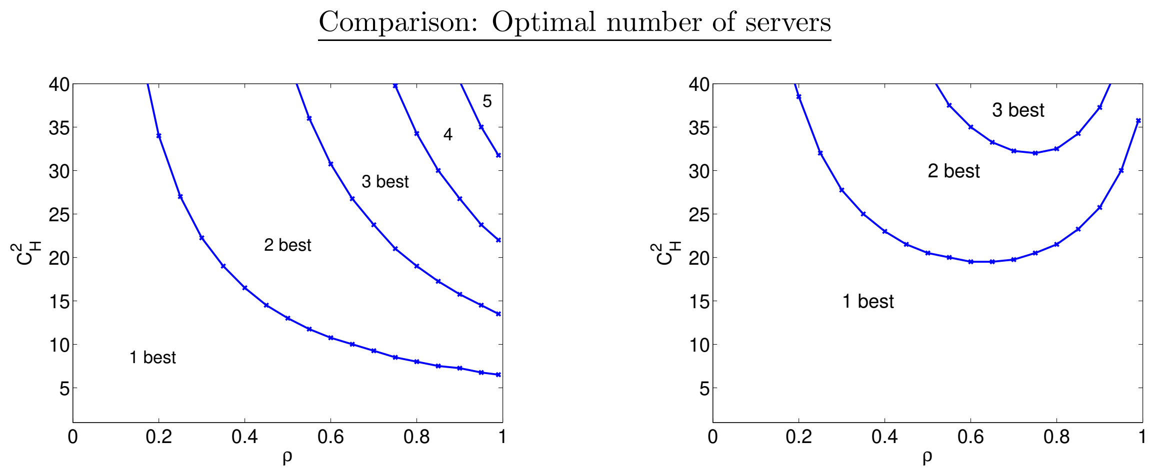
The above rules immediately provide answers to the first three questions that we have posed in the introduction. Also, our discussion regarding Figure 4.5 provides an answer to the fourth question. In particular, the mean response time (for both overall and per-class) can be dramatically different for different number of servers, and in all studied cases, there exists an "optimal" number of servers where using fewer or more servers results in worse performance under highly variable service distributions.

4.5 New approximations for many priority classes

4.5.1 Motivation

It is interesting to compare the results of our analysis with the approximation proposed by Buzen and Bondi (the BB approximation) [23], which is the only prior work to investigate the multiserver system with multiple priority classes under general job size distributions. The BB approximation uses the following intuitive framework to approximate the mean delay in a priority system, where $\mathsf{E}\left[D^{\mathrm{M/GI}/k/\mathrm{prio}}\right]$ is the overall expected delay (response time minus service time) under the M/GI/k queue with priority classes.

$$\mathsf{E}\left[D^{\mathrm{M/GI}/k/\mathrm{prio}}\right] \approx \mathsf{E}\left[D^{\mathrm{M/GI/1/prio}}\right] \left(\frac{\mathsf{E}\left[D^{\mathrm{M/GI}/k/\mathrm{FCFS}}\right]}{\mathsf{E}\left[D^{\mathrm{M/GI/1/FCFS}}\right]} \right) \tag{4.1}$$



(a) DR (b) BB

Figure 4.6: *Comparison of (a) DR with (b) BB with respect to the question of "how many servers are best?" Case shown assumes the high priority jobs have a smaller mean job size (*$\mathsf{E}[X_H] = 1$ *and* $\mathsf{E}[X_L] = 10$*), and the load made up by high priority jobs equals that comprised by low priority jobs.*

Figure 4.6 shows the optimal number of servers given by our analysis as compared with that predicted by the BB approximation. Here, the size of the high priority jobs has a two phase PH distribution with mean 10, and the low priority jobs have an exponential job size distribution with mean 1, as before. To compute the mean delay via BB, the mean delay in an M/GI/k/FCFS, $\mathsf{E}\left[D^{\text{M/GI/}k\text{/FCFS}}\right]$, is computed exactly via matrix analytic methods as before, as job sizes have a PH distribution; the mean delay in an M/GI/1/prio, $\mathsf{E}\left[D^{\text{M/GI/1/prio}}\right]$, is calculated via a known closed form expression [101].

Figure 4.6 shows that the BB approximation gives qualitatively misleading conclusion regarding the optimal number of servers. As load increases, so should the optimal number of servers; an effect not true under the BB approximation.

Since the only existing approximation, BB, leads to misleading conclusions, and computational complexity of DR becomes high when the number of classes increases, as we have discussed in Section 3.9, we propose a new approximation based on DR, which we refer to as DR-A (dimensionality reduction with class aggregation). DR-A allows us to efficiently compute the mean response time (per class and overall) of an M/PH/k queue with *many* priority classes. We will see that the error in DR-A in predicting mean delay is within 5% against simulation for a range of parameters, which is much smaller than the error in BB. We will evaluate the error in DR-A in predicting mean delay rather than mean response time, since the error in mean delay is larger.

4.5.2 Approximations for multiserver systems with multiple priority classes

We first introduce approximations for the mean delay in an M/GI/k queue with $m \geq 2$ priority classes (class i jobs have preemptive priority over jobs of classes $i+1$ to m for $1 \leq i \leq m-1$).

The abovementioned BB approximation was originally derived for exponential job size distributions [33] and was later applied to general job size distributions [23]. The BB approximation is the only (reasonable) approximation, for general job size distributions and for an arbitrary number of priority classes, that exists in the literature. Mitrani and King have proposed an approximation that works for exponential job size distributions and for an arbitrary number of priority classes [129], and this approach is also used by Nishida [140]. We refer to this approximation as MK-N. The MK-N approximation is defined and used only for exponential job size distributions, since MK-N requires an analysis of a multiserver queue with two priority classes, and an exact or nearly exact analysis has been known only for exponential job size distributions. Our analysis of an M/PH/k queue with multiple priority classes via DR allows us to extend MK-N to PH job size distributions. We refer to the new approximation as DR-A. Below, we provide more details of the three approximations.

The Buzen-Bondi (BB) approximation

The BB approximation is based on an intuitive observation that the "improvement" of priority scheduling over FCFS scheduling under multiple servers is similar to that for the case of a single server:

$$\frac{\mathsf{E}\left[D^{\mathrm{M/GI}/k/\mathrm{prio}}\right]}{\mathsf{E}\left[D^{\mathrm{M/GI}/k/\mathrm{FCFS}}\right]} \approx \frac{\mathsf{E}\left[D^{\mathrm{M/GI/1/prio}}\right]}{\mathsf{E}\left[D^{\mathrm{M/GI/1/FCFS}}\right]} \equiv \text{ scaling factor.} \tag{4.2}$$

Here $\mathsf{E}\left[D^{\mathrm{M/GI}/k/\mathrm{prio}}\right]$ is the overall mean delay under priority scheduling with k servers of speed $1/k$, and $\mathsf{E}\left[D^{\mathrm{M/GI}/k/\mathrm{FCFS}}\right]$ is defined similarly for FCFS. This relation is exact when job sizes are exponential with the same mean for all classes.

Specifically, BB analyzes the mean delay of class i jobs, $\mathsf{E}[D_i]$, by analyzing both the mean delay over classes 1 to i, $\mathsf{E}[D]$, and the mean delay of the higher priority classes (classes 1 to $i-1$), $\mathsf{E}[D_H]$, and then using the following relation:

$$\mathsf{E}[D] = \frac{\lambda_H}{\lambda_H + \lambda_i}\mathsf{E}[D_H] + \frac{\lambda_i}{\lambda_H + \lambda_i}\mathsf{E}[D_i],$$

where λ_H and λ_i are the arrival rates of the higher priority classes and class i jobs, respectively. To compute $\mathsf{E}[D]$ (respectively, $\mathsf{E}[D_H]$), BB aggregates classes 1 to i (respectively, classes 1 to $i-1$) into a single class and analyzes the corresponding M/GI/k/FCFS queue (e.g. via known approximations [27, 141]), and then calibrates this result using the scaling factor in (4.2), as shown in (4.1).

The Mitrani-King-Nishida (MK-N) approximation

The MK-N approximation analyzes the mean delay of class i jobs in an M/M/k queue with $m \geq 2$ priority classes by aggregating all the higher priority classes (classes 1 to $i-1$). The job size distribution of the aggregated class is then approximated with an exponential distribution by matching the first moment of the distribution.

In MK-N, the job size distribution of the aggregated higher priority classes is approximated by an exponential distribution, since the exact or nearly exact analysis of a multiserver system with two priority classes are known only for exponential distributions. However, DR that we have developed in Chapter 2 allows us to analyze the multiserver system with two priority classes for the case where job sizes have *PH* distributions, and the moment matching algorithm that we have developed in

Chapter 2 allows us to approximate the job size distribution of the aggregated higher priority classes by a PH distribution matching the first *three* moments. This motivates us to extend the MK-N approximation to PH distributions: DR-A.

New approximation: DR-A

The DR-A approximation analyzes the mean delay of class i jobs in an M/PH/k queue with $m \geq 2$ priority classes by aggregating all the higher priority classes, as in the MK-N approximation. By contrast to MK-N, the job size distribution of the aggregated class is approximated with a *PH* distribution by matching the first *three moments* of the distribution. Recently, an extension of MK-N to PH job size distributions (hyperexponential distributions, in particular) is also proposed, independently, by Sleptchenko et. al. [194] (we propose DR-A in [204]).)

4.5.3 Comparing DR-A with BB and MK-N

We now evaluate the accuracy of DR-A, BB, and MK-N. In all figures, we assume $k = 2$ servers and $m = 4$ priority classes. We consider both the case where each priority class has an exponential job size distribution ($C^2 = 1$; top half of Figure 4.7) and the case of two-phase PH job size distributions with $C^2 = 8$ (bottom half of Figure 4.7)[3]. Each class may have a different mean job size, and these are chosen to vary over a large range, determined by parameter α. Specifically, the mean job size of class i is set $\mathsf{E}[X_i] = \alpha^{4-i}$, where $\frac{1}{4} \leq \alpha \leq 4$. Thus, $\alpha < 1$ implies small high priority jobs. We equalize the load between the classes, i.e. $\rho_i = \rho/4$, where ρ_i is the load of class i. With these settings, the error in mean delay is evaluated for each class of jobs, where the error of an approximation is defined as follows:

$$\text{error} = 100 \times \frac{(\text{mean delay by approximation}) - (\text{mean delay by simulation})}{(\text{mean delay by simulation})} \quad (\%).$$

Thus, positive error means overestimation and negative error means underestimation of the approximation. Simulation is kept running until the simulation error becomes less than 1% with probability 0.95 (see Section 3.9 for more technical details of our simulation).

In evaluating the BB and MK-N approximations, we use accurate methods known to compute their components. For example, BB relies on knowing the mean delay for the M/PH/k/FCFS queue. We compute this delay precisely for the PH job size distribution using matrix analytic methods. MK-N relies on being able to analyze the case of two priority classes (since m classes are reduced to two). We analyze the two priority class case in MK-N using DR. We first discuss the accuracy of MK-N and DR-A, and then discuss the accuracy of BB.

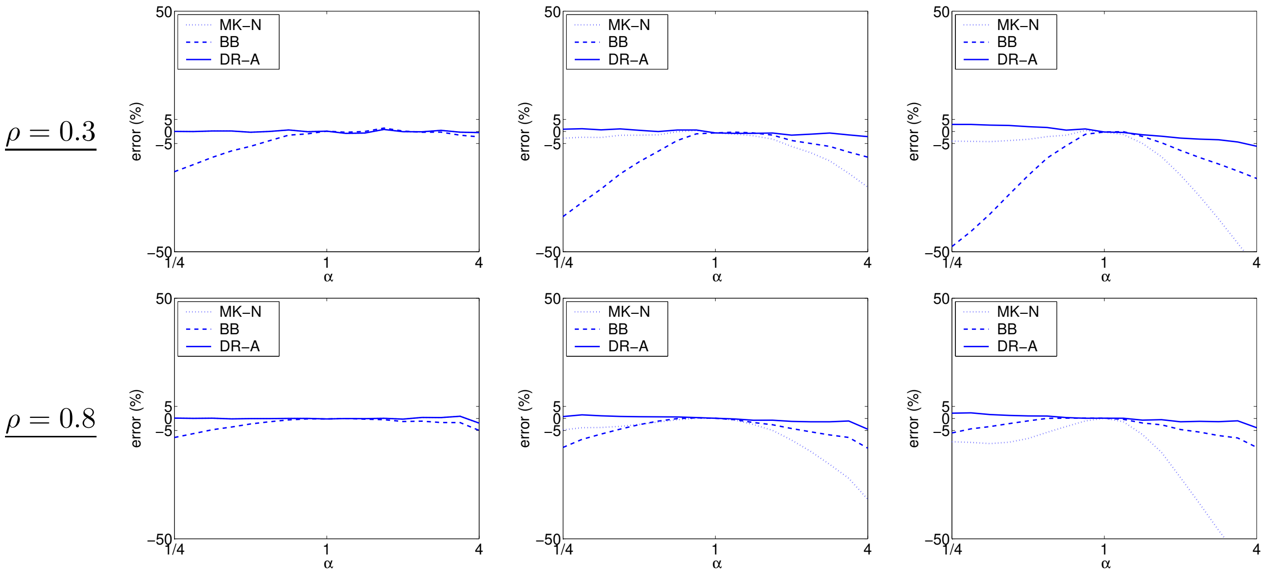
The MK-N and DR-A approximations

There are two sources of errors in MK-N and DR-A. The first source of error, which we call *priority error*, results from ignoring the priorities among the higher priority classes. The second source, which we call *distribution error*, results from the fact that the aggregated job size distribution matches only one moment in MK-N and three moments in DR-A.

Consider first the priority error. In the case of a single server, there is no priority error. This is because low priority jobs are excluded from service during busy periods of high priority classes, and

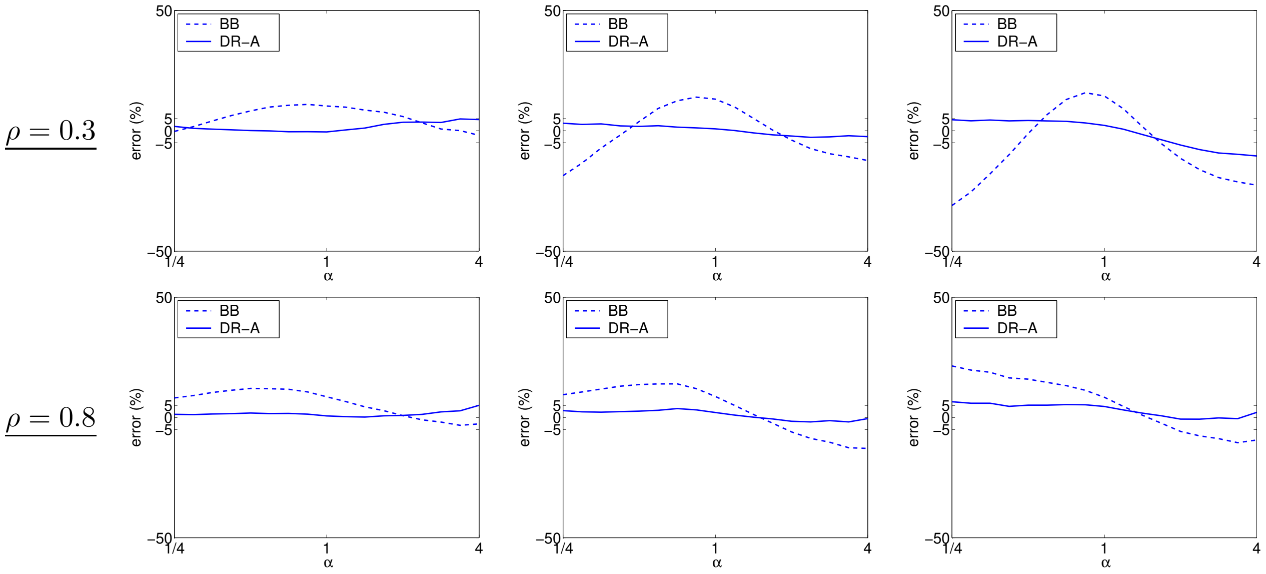
[3]The third moment of the two-phase PH distribution is chosen as before.

Comparison of approximations for M/M/2 with 4 priority classes ($C^2 = 1$)



(a) Class 2 (b) Class 3 (c) Class 4

Comparison of approximations for M/PH/2 with 4 priority classes ($C^2 = 8$)



(a) Class 2 (b) Class 3 (c) Class 4

Figure 4.7: *Comparison of DR-A, MK-N, and BB approximations for M/PH/2 with 4 priority classes with* $\mathsf{E}[X_4] = 1$, $\mathsf{E}[X_3] = \alpha$, $\mathsf{E}[X_2] = \alpha^2$, *and* $\mathsf{E}[X_1] = \alpha^3$, *as a function of* α. *Here, the squared coefficient of variability of the job size distributions are* $C^2 = 1$ *(top two rows) or* $C^2 = 8$ *(bottom two rows) for all classes. Load is balanced among the classes. Note that MK-N does not appear for* $C^2 = 8$, *because the error is beyond the scale of the graphs for most values of* α.

its duration is the same regardless of whether or not the higher priority classes are prioritized. In the case of multiple servers, however, priority error exists, since low priority jobs are more likely to be present at the end of a busy period.

Next consider the distribution error. The distribution error exists even in the case of a single server unless the first two moments of the aggregated job size distribution are matched, as the mean delay of the lowest priority jobs, $\mathsf{E}[D_L]$, depends on the first two moments of the job size distributions:

$$\mathsf{E}[D_L] = \frac{\rho_H}{1-\rho_H}\mathsf{E}[X_L] + \frac{\lambda \mathsf{E}[X^2]}{2(1-\rho_H)(1-\rho)}, \tag{4.3}$$

where ρ_H is the load of high priority jobs, ρ is the overall load, $\mathsf{E}[X_L]$ is the mean size of the lowest priority jobs, and $\mathsf{E}[X^2]$ is the second moment of the overall job size distribution. Thus, MK-N has the distribution error even in the case of a single server, but DR-A does not in this case. In the case of multiple servers, matching even the first two or three moments is not sufficient in general. This can be gleaned from recent results illustrating the sensitivity of the stability conditions for the M/GI/k systems to the exact tail behavior of the service distribution [172, 203].

Figure 4.7 illustrates the quantitative difference in the priority and distribution errors in MK-N and DR-A. Overall, the error in MK-N can sometimes exceed 50%, while the error in DR-A never exceeds 5% for exponential job size distributions and never exceeds 10% for PH job size distributions. Note that MK-N does not show up in the bottom half of Figure 4.7 (PH job sizes) because the error is too great (usually $> 50\%$) — approximating a mixture of PH distributions with $C^2 = 8$ by an exponential distribution clearly does not work. Below, we discuss the error per class in MK-N and DR-A. (Class 1 is not discussed, since its analysis is exact via matrix analytic methods.)

For class 2 jobs (Figure 4.7, column 1), MK-N and DR-A are identical to DR and have little error when $C^2 = 1$. This is because we use DR in evaluating MK-N with two priority classes. When $C^2 = 8$, DR-A is still equivalent to DR for an M/PH/k with two priority classes, and the error for DR is always within 5% for this case.

For class 3 (column 2), error in both MK-N and DR-A becomes apparent. Both approximations aggregate the two higher priority classes, and when $C^2 = 1$, the aggregated job size distribution is a two-phase hyperexponential distribution. The hyperexponential distribution is approximated by an exponential distribution in the MK-N approximation, while the exact two-phase hyperexponential distribution is used in DR-A. Therefore, the MK-N approximation has both priority error and distribution error, while DR-A has only priority error when $C^2 = 1$. We observe that the error in DR-A (priority error only) is orders of magnitude smaller than the error in MK-N (both priority and distribution error).

The effect of α on the error of MK-N in class 3 is significant. When $\alpha = 1$ and $C^2 = 1$, MK-N (and DR-A) is exact, since the aggregated distribution is exponential. As α gets smaller or larger than 1, MK-N underestimates the mean delay. This makes intuitive sense because MK-N ignores the variability among the high priority classes, which leads to a lower mean delay estimate for the lower priority jobs. We can also observe that the error in MK-N is smaller when the high priority classes are small ($\alpha < 1$) as compared to the case when $\alpha > 1$. This is because when α is large, approximating the aggregated higher priority classes by an exponential distribution results in ignoring huge jobs in class 1, and ignoring huge jobs leads to a great underestimation in the mean delay of the low priority class.

The effect of α on the error of DR-A in class 3 is much smaller than the effect on MK-N. When $\alpha = 1$ and $C^2 = 1$, DR-A is exact as is MK-N. When $\alpha = 1$ and $C^2 = 8$, DR-A does not have distribution error but does have priority error. However, as for class 2, priority error is very small in this case as well. For all ρ's and C^2's, DR-A tends to slightly overestimates the mean delay for $\alpha < 1$, and it tends to slightly underestimates for $\alpha > 1$. This is explained by reasoning about the busy periods made up of the high priority classes, during which both servers are serving jobs of class 1 or 2. When smaller jobs have priority, more larger jobs are left at the end of a busy period. Since larger jobs are less likely to be balanced between two servers, one of the servers will likely become free sooner, which allows the lower priority class to get service sooner. When larger jobs have priority, both servers are likely to become free at a similar time as low priority (smaller) jobs can fill in the gaps. This causes the lower priority jobs to wait longer before they get service at either server. When $\alpha < 1$, smaller jobs have priority, but DR-A ignores the priority and the busy period estimated by DR-A is longer, which in turn yields overestimation of the mean delay. The underestimation for $\alpha > 1$ can be explained analogously.

Continuing with class 3, we see that the load does not significantly affect the class 3 error, but at higher load the error in MK-N is slightly larger and the error in DR-A is slightly smaller. This implies that the priority error is smaller and the distribution error is larger at higher load. The priority error is due to the error in estimating the busy period of the higher priority jobs, and at high load the busy period is long and the relative difference between priority scheduling and FCFS with respect to the length of the busy period is smaller. Larger distribution error at higher loads can be explained through an analogy to the case of a single server. Recall equation (4.3), the mean delay of the lower priority class in an M/PH/1 queue with two priority classes. The distribution error in this case corresponds to the error in the second moment of the job size, $\mathsf{E}\left[X^2\right]$, and the term with the error, $\frac{\lambda \mathsf{E}\left[X^2\right]}{2(1-\rho_H)(1-\rho)}$, increases as the load, ρ, approaches 1.

For class 4 (column 3), we observe that the error has a similar trend as in class 3. The only difference is that the aggregation in DR-A is no longer exact for class 4, even when $C^2 = 1$; however, we still see that DR-A always has a small error. We conclude that the priority error is small regardless of the load and the relative sizes among the classes, but ignoring the variability among the job sizes of higher priority classes (aggregated) can lead to significant error.

The BB approximation

The BB approximation is based on the assumption that the "improvement" of priority scheduling over FCFS is similar between a single server system and a multiserver system, specifically (4.2). However, our study in Section 4.4 implies that the improvement can be quite different, since *prioritizing small jobs has the same effect as having multiple servers.* Figure 4.7 illustrates the error in BB quantitatively.

In rows 1-2 of Figure 4.7, the job sizes have exponential distributions. In this case, for all classes, BB is exact when all the classes have the same mean size ($\alpha = 1$), but it underestimates the mean delay when the higher priority classes are small ($\alpha < 1$) or large ($\alpha > 1$). This can be explained as follows. BB is based on the observation that the "improvement" of priority scheduling over FCFS under multiple servers is similar to the case of a single server (recall equation (4.2)). When all the classes have the same mean, priority scheduling provides the same mean delay as FCFS, regardless of the number of servers; hence (4.2) is exact, and so is BB. When high priority classes are small, priority scheduling improves the mean delay over FCFS because it allows small jobs to avoid waiting behind large jobs. However, the improvement under multiple servers is smaller than in the case of a single server, because having multiple servers also allows small jobs to avoid waiting behind large jobs under FCFS. Since BB assumes that the improvement is the same, it underestimates the mean delay. When class 1 is larger than class 2, prioritizing class 1 worsens the mean delay. However, the penalty due to prioritization under a single server is smaller than under multiple servers, since with a single server FCFS also forces large jobs to block small jobs, but multiple servers often allow small jobs to avoid waiting behind large jobs under FCFS. Again, since BB assumes that the improvement is the same, it underestimates the mean delay.

We can also observe that the error in BB is smaller under high load than low load ($\rho = 0.3$, row 1). This is because under high load single server systems and multiserver systems behave similarly. Overall, the error in BB can be as high as 50% under low load ($\rho = 0.3$) and 20% under high load ($\rho = 0.8$).

For non-exponential job size distributions, BB is no longer exact even when all classes have the same job size distributions, as shown in Figure 4.7 rows 3-4. For $C^2 = 8$, preemptive priority scheduling and FCFS yield different mean delay. For example, for job size distributions with decreasing failure rate, low priority jobs with longer remaining time are more likely to be preempted by higher priority jobs, which can improve the overall mean delay. However, this improvement differs between single server systems and multiserver systems. BB therefore can yield error in predicting the mean delay.

4.6 Concluding remarks

In this chapter, a fundamental nature of multiserver systems is studied, in particular with respect to how the number of servers affects the performance of multiserver systems. DR allows us to study this problem not only when jobs are served in the order of their arrivals (FCFS) but also when there are priorities among jobs, whose analysis involves multidimensional Markov chains. Note that an advantage of DR is that the job size can be modeled as a PH distribution, and this allows us to study the impact of job size variability on the performance.

Our analysis illuminates principles on the performance (mean response time, in particular) of multiserver systems as compared to single server systems, which can be characterized primarily by three rules of thumb.

(i) A single server system has an advantage over multiserver systems with respect to utilization, and this relative advantage becomes greater at higher load.

(ii) A multiserver system has an advantage over single server systems with respect to reducing the impact of *job size variability* on the mean response time, and this advantage becomes greater at higher load and at larger job size variability.

When jobs are served in FCFS order, the mean response time of single server and multiserver systems can be qualitatively characterized primarily by the above two rules. In particular, the rules characterize how the optimal number of servers is affected by the load and job size variability. However, the above two rules are not sufficient to characterize the mean response time when there are priorities among jobs, since

(iii) A multiserver system has an advantage over a single server system with respect to reducing the impact of prioritization on low priority jobs, and this advantage becomes greater when the mean and/or variability of the higher priority job size are larger and/or when the load of the higher priority job is higher.

We find that these three rules quantitatively characterize the mean response time under single and multiserver systems. In particular, these rules well characterizes how the optimal number of servers (per class or overall) is affected by the load and the mean and variability of the job size of each class.

In fact, the third rule has an important implication in designing scheduling policies in multiserver systems. Namely, *prioritizing small jobs to improve mean response time is not as effective in multiserver systems as in single server systems.*

We have also proposed an approximate analysis, DR-A, of the (per class or overall) mean response time in an M/PH/k queue with *many* ($m > 2$) priority classes. DR-A is based on DR, but as opposed to the two approximations that we have introduced in Section 3.6 (DR-PI and DR-CI), its running time does not grow as the number of priority classes, m, increases. The accuracy of DR-A as well as two existing approximations (BB and MK-N) is evaluated against simulation, and the results are discussed extensively. We find that the error in DR-A is within 5% for a range of loads and job size variabilities, while the error in BB and MK-N can be as high as 50%. Since BB is based on the assumption that the effect of priority is similar between a single server system and a multiserver system, the error in BB can also be explained by the above observation that prioritizing small jobs to improve mean response time is not as effective in multiserver systems as in single server systems.

In this chapter, we primarily limit our discussion to "how many servers are best?" However, the analysis of M/PH/k queue with m priority classes via DR has a broad applicability in capacity planning of multiserver systems with multiple priority classes. For example, in [204], we study the impact of system tasks on the performance of user tasks in the context of dependability systems, where systems tasks have higher priority than user tasks for the purpose of fault recovery, fault isolation, fault masking, intrusion detection, virus checking, etc.

The results in this chapter can be used to infer how the performance is affected by changing the number of servers or by prioritizing some of the jobs, in more complex multiserver systems. For example, in the rest of the thesis, we consider multiserver systems consisting of multiple queues and multiple servers, where each server behaves differently from another. We typically assume that there are no priorities *within* each queue, and there is only a single server *of each type*. By contrast, the model that we have studied in this chapter has multiple homogeneous servers and a single queue where some jobs have priority over other jobs. The results in this chapter can be used to infer how the mean response time is affected in the systems studied in the rest of the thesis when there are priorities within each queue and there are multiple servers of each type.

Chapter 5

Improving traditional task assignment policies

Which job should be assigned to which server?

In this chapter, we consider a fundamental problem of task assignment in a server farm: which job should be assigned to which server? We propose the size-based task assignment policies with cycle stealing, `SBCS-ID` and `SBCS-CQ`, and analyze their performance via DR, which we have developed in Chapter 3. Our analysis shows that `SBCS-IC` and `SBCS-CQ` can provide an unbounded benefit over traditional task assignment policies with respect to mean response time.

5.1 Task assignment in server farms

Distributed servers have become increasingly common because they allow for increased computing power while being cost-effective and easily scalable. In a distributed server system, jobs (tasks) arrive and must each be dispatched to exactly one of several host machines for processing. The rule for assigning jobs to host machines is known as the *task assignment policy.* The choice of the task assignment policy has a significant effect on the performance perceived by users. Designing a distributed server system thus often comes down to choosing the "best" possible task assignment policy for the given model and user requirements. While devising new task assignment policies is easy, analyzing even the simplest policies can prove to be difficult. Many of the long standing open questions in queueing theory involve the performance analysis of task assignment policies.

In this chapter, we consider the *particular model* of a distributed server system in which hosts are homogeneous and the execution of jobs is *nonpreemptive* (run-to-completion), i.e. the execution of a job cannot be interrupted and subsequently resumed. We consider both the case where jobs are *immediately dispatched* upon arrival to one of the host machines for processing (the immediate dispatching model), and the case where jobs are held in a *central queue* until requested by a host machine (the central queue model). The immediate dispatching of jobs is sometimes crucial for scalability and efficiency, as it is important that the router not become a bottleneck. On the other hand, the central queue model allows us to delay the decision of where to assign jobs, and this can lead to better utilization of resources and lower response time of jobs.

The immediate dispatching and central queue models with the nonpreemptive assumption are motivated by servers at supercomputing centers, where jobs are typically run to completion. The schedulers at supercomputing centers typically only support run-to-completion within a server machine for the following reasons. First, the memory requirements of jobs tend to be huge, making it expensive to swap out a job's memory for timesharing [52]. Also, many operating systems that enable timesharing for single-processor jobs do not facilitate preemption among several processors in a coordinated fashion [157]. Our models are also consistent with validated stochastic models used to study a high volume web sites [82, 184], studies of scalable systems for computers within an organization [166], and telecommunication systems with heterogeneous servers [24].

5.2 State of the art in task assignment policies

Below, we provide an overview of the literature on task assignment policies in both the immediate dispatching model and the central queue model, limiting our discussion to *nonpreemptive* task assignment policies.

By far the most common task assignment policies used in the immediate dispatching model are `Random` and `Round-Robin`. The `Random` policy assigns an incoming job to each server with probability $1/k$, where k is the number of servers, while the `Round-Robin` policy assigns jobs to servers in a cyclic order. `Random` and `Round-Robin` are simple, but they neither maximize utilization of the hosts, nor minimize mean response time.

In the immediate dispatching model, the `Shortest-Queue` policy has been shown to be optimal under various constraints and objective functions under the conditions that the service demand has an exponential distribution and job sizes are not known a priori [49, 106, 124, 192, 209]. Under the `Shortest-Queue` policy, incoming jobs are immediately dispatched to the host with the fewest number of jobs.

In the central queue model, the `M/G/k/FCFS` policy has been proven to minimize mean response time and maximize utilization under the condition that the service demand has an exponential distribution and job sizes are not known a priori [210]. The `M/G/k/FCFS` policy holds all jobs at the central queue, and when a host becomes free, it receives a job from the central queue in the order of their arrivals (first come first served, FCFS). The `M/G/k/FCFS` policy is provably identical to the `Least-Work-Remaining` policy in the immediate dispatching model [64]. The `Least-Work-Remaining` policy assumes that the jobs sizes are known a priori, and it sends each job to the host with the least total remaining work.

While policies like `Shortest-Queue` and `M/G/k/FCFS` perform well when job sizes have an *exponential* distribution (or distribution with increasing failure rate), they perform *poorly* when the job size distribution has higher variability [106, 201]. In such cases, it has been shown analytically and empirically that the `Dedicated` policy far outperforms these other policies with respect to minimizing mean response time [65, 174]. In the `Dedicated` policy, some hosts are designated as the "short hosts" and others as the "long hosts." Short jobs are always sent to the short hosts and long jobs to the long hosts. Observe that the `Dedicated` policy is defined for both the immediate dispatching model and the central queue model, and the two models are identical under the `Dedicated` policy. The `Dedicated` policy is popular in practice (e.g. Cornell Theory Center) where different host machines have different duration limitations: 0–1/2 hour, 1/2 – 2 hours, 2 – 4 hours, etc., and users must specify an estimated required service demand for each job. The `Dedicated` policy performs well when job sizes have high variability because it isolates shorts jobs from the long jobs, as waiting behind the long jobs is costly. The `Dedicated` policy is also popular in supermarkets and banks, where an "express" queue is created for "short" jobs.

The `Dedicated` policy requires to know job sizes upon their arrivals; however, even when the job size is not known, a policy similar to `Dedicated`, known as the `TAGS` policy (task assignment by guessing size) works almost as well when job sizes have high variability. Like `Dedicated`, the `TAGS` policy significantly outperforms other policies that do not segregate jobs by size [64].

5.3 Task assignment with cycle stealing

Motivation for Cycle Stealing

While the `Dedicated` policy may be preferable to the `M/G/k/FCFS` and `Shortest-Queue` policies for highly variable job sizes, it is clearly *not* optimal. One problem is that `Dedicated` leads to situations where the servers are not fully utilized: five consecutive short jobs may arrive, with no long job, resulting in an idle long host. This is especially likely in common computer workloads, where there are many short jobs and just a few very long jobs, resulting in longer idle periods between the arrivals of long jobs.

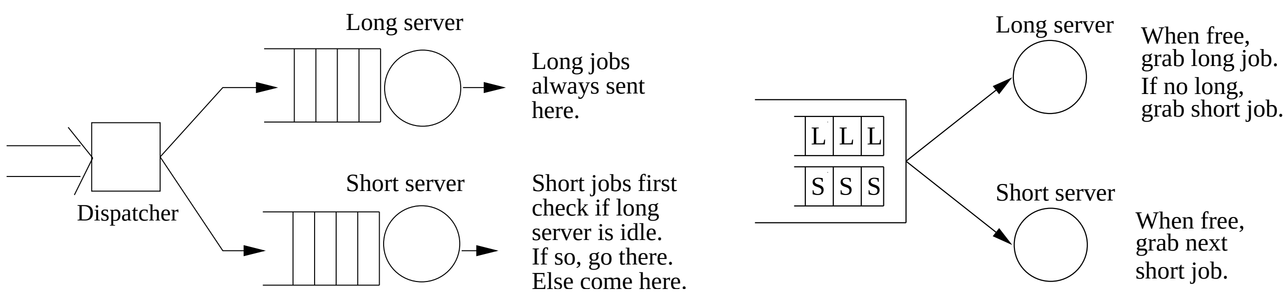
Ideally we would like a policy which combines the variance reducing benefit of the `Dedicated` policy with the high utilization property of `M/G/k/FCFS` and `Shortest-Queue`. Hereby, we would segregate jobs by size to provide isolation for short jobs, but when the long job host is free, we would *steal* the long host's idle cycles to serve excess short jobs. This would both decrease the mean response time of the short jobs, and enlarge the stability region of the overall system. Specifically, for systems where the short host is much more heavily loaded, granting the short jobs limited access to the long partition may be the difference between an overloaded system and a well behaved one. We grant the short jobs access to the long host only when the long host is *free*, so we don't *starve* the long jobs, causing them undue delay. Nonetheless, because jobs are not preemptive, there will still be a penalty to a long job which arrives to find a short job serving at the long host.

Specifically, we propose the following two size-based task assignment policies with cycle stealing.

Size-based task assignment with cycle stealing under immediate dispatching (`SBCS-ID`): The `SBCS-ID` policy (see Figure 5.1(a)) is defined for the immediate dispatching model. There is a designated short job host and a designated long job host. An arriving long job is always dispatched to the long job host. An arriving short job first checks to see if the long job host is idle. If it is idle, the short job is dispatched to the long job host. If the long job host is not idle (either working on a long job or a short job), then the arriving short job is dispatched to the short job host. Jobs at a host are serviced in FCFS order.

The `SBCS-ID` algorithm is an improvement over `Dedicated` for the short jobs. However, only those short jobs arriving *after* the long host has entered an idle period can steal long cycles; if a short job arrives just before the long host enters an idle period, the short job is not eligible for running during the idle partition. This is the motivation behind our next cycle stealing algorithm.

Size-based task assignment with cycle stealing under central queue (`SBCS-CQ`): The `SBCS-CQ` policy (see Figure 5.1(b)) is defined for the central queue model. Whenever the short job host becomes idle, it picks the first short job in the queue to run. Whenever the long job host becomes idle, it picks the first long job in the queue. However, if there is no long job, the long host picks the first short job in the queue. In `SBCS-CQ`, we introduce a minor technique that can slightly improve mean response time. Whereas in `SBCS-ID` the short and long hosts are designated



(a) SBCS-ID (b) SBCS-CQ

Figure 5.1: *Size-based task assignment with cycle stealing.*

in advance, in SBCS-CQ we allow renaming of hosts; i.e. if the long host is working on a short job, and the short host is idle, then the long host is renamed the short host and vice versa. Thus in SBCS-ID, there could be one short in front of one long job in the system with an idle (short) server, while this could not happen under SBCS-CQ.

Modeling and analysis

We assume that the service demand of the short jobs, X_S, has a phase type (PH) distribution (see Section 2.2), and the service demand of the long jobs, X_L, has another PH distribution. Thus, we do *not* require that the exact service demand of jobs be known in advance. Rather, we assume that, based on some information such as the owner of a job and the CPU requirement specified by a user, we can classify jobs into "short" jobs that are likely to be short and "long" jobs that are likely to be long. We will consider three scenarios: the expected size of "short" jobs is shorter than that of "long" jobs, "short" jobs and "long" jobs have an identical job size distribution, and a pathological case where the expected size of "short" jobs is longer than that of "long" jobs.

Throughout, we assume that the short jobs (respectively, long jobs) arrive according to a Poisson process with rate λ_S (respectively, λ_L), although the analysis via dimensionality reduction (DR) allows an extension to Markovian arrival processes (see Section 2.2), as we have discussed in Chapter 3. We define the load of the short jobs as $\rho_S = \lambda_S \mathsf{E}[X_S]$ and the load of the long jobs as $\rho_L = \lambda_L \mathsf{E}[X_L]$.

With the above assumptions, the system with SBCS-ID can be modeled as an FB process, and the system with SBCS-CQ (without renaming) can be modeled as a GFB process, as discussed in Section 3.4, and the analysis of these processes via DR provides mean response time of the short jobs and the long jobs, respectively, under SBCS-ID and SBCS-CQ. The system with SBCS-CQ with renaming does not appear to be modeled as a GFB process, and in [67], we analyze its performance via DR with a certain independence assumption (some random variables with dependency are assumed to be independent); however, this approximation is validated against simulation, which shows that the error is within 2% in almost all cases, and within 8% in all cases (large error occurs at high load) [67]. All the details of the analysis of the SBCS-ID and SBCS-CQ, used to generate the results in Section 5.4, are provided in [67, 68]. In particular, we derive a closed form expression for the mean response time of the long jobs under SBCS-ID via a virtual waiting time analysis in [68], as shown in the following theorem.

Theorem 9 *Let T_L represent the response time of the long jobs under* SBCS-ID, *and let $\widetilde{T}_L(s)$ denote* *it's Laplace transform. Then,*

$$\begin{aligned}
\widetilde{T}_L(s) &= \frac{s+\lambda_S-\widetilde{X}_S(s)\lambda_S}{s-\lambda_L+\widetilde{X}_L(s)\lambda_L}\cdot\pi_0\cdot\widetilde{X}_L(s)\\
\mathsf{E}[T_L] &= \frac{\rho_L}{1-\rho_L}\frac{\mathsf{E}[X_L^2]}{2\mathsf{E}[X_L]}+\frac{\rho_S}{1+\rho_S}\frac{\mathsf{E}[X_S^2]}{2\mathsf{E}[X_S]}+\mathsf{E}[X_L]\\
\mathsf{E}\left[T_L^2\right] &= \frac{\rho_L}{1-\rho_L}\frac{\mathsf{E}[X_L^3]}{3\mathsf{E}[X_L]}+\frac{\rho_S}{1+\rho_S}\frac{\mathsf{E}[X_S^3]}{3\mathsf{E}[X_S]}+\frac{\rho_L}{1-\rho_L}\frac{\mathsf{E}[X_L^2]}{\mathsf{E}[X_L]}(\mathsf{E}[T_L]-\mathsf{E}[X_L])\\
&\quad+2\mathsf{E}[X_L]\,\mathsf{E}[T_L]+\mathsf{E}\left[X_L^2\right]-2\mathsf{E}[X_L]^2,
\end{aligned}$$

where

$$\pi_0=\frac{1-\rho_L}{1+\rho_S}.$$

Here, π_0 represents the fraction of time that the long server is idle.

We postpone the proof of this theorem to Appendix A.

5.4 Performance of task assignment with cycle stealing

In this section we will study the results of our analysis of `SBCS-ID` and `SBCS-CQ`. Recall that the motivation behind cycle stealing is to improve the performance of the short jobs without inflicting too much penalty on the long jobs. However, some penalty to the long jobs is inevitable, since our model is nonpreemptive. In order to evaluate these benefits and penalties we compare with the `Dedicated` policy.

5.4.1 Summary of results

Our analysis of `SBCS-ID` and `SBCS-CQ` via DR will lead to the following conclusions:

- Both `SBCS-ID` and `SBCS-CQ` can provide an unbounded benefit, over `Dedicated`, to the short jobs while the long jobs are not penalized much by having their idle cycles stolen. The unbounded benefit is due to the increased stability region: `SBCS-ID` and `SBCS-CQ` can provide finite mean response time while `Dedicated` leads to infinite mean response time.

- `SBCS-CQ` is a superior policy to `SBCS-ID` from the perspective of *both* the short jobs and the long jobs.

- When the "short" jobs have shorter expected size than the "long" jobs, `SBCS-ID` and `SBCS-CQ` typically improves upon `Dedicated` with respect to the overall mean response time even at low load, ρ_S (with an unbounded benefit at high load). However, when the "short" jobs have longer expected size than the "long" jobs (pathological case), cycle stealing can worsen the overall mean response time at low load, ρ_S (although they can still provide an unbounded benefit at high load).

- The service demand variability of the long jobs has a surprisingly small effect on the mean response time of the short jobs under `SBCS-ID` but has a much larger effect under `SBCS-CQ`. Observe that when the long jobs have more variable sizes, long server provides more irregular help to the short jobs, which we expect worsens the response time of the short jobs.

We first characterize the stability regions under `Dedicated`, `SBCS-ID`, and `SBCS-CQ` in Section 5.4.2. In Section 5.4.3, we study the mean response time under these policies.

5.4.2 Stability

Under the `Dedicated` policy, the queues are stable iff $\rho_L < 1$ and $\rho_S < 1$. The stability region becomes wider under `SBCS-ID` and `SBCS-CQ`, as is characterized in the next theorem.

Theorem 10 *Under* `SBCS-ID`*, the queues are stable iff*

$$\rho_L < 1 \quad and \quad \rho_S < \frac{1-\rho_L+\sqrt{(1-\rho_L)^2+4}}{2}.$$

Under `SBCS-CQ`*, the (central) queue is stable iff* $\rho_L < 1$ *and* $\rho_S < 2-\rho_L$.

Proof: We first consider the stability conditions for `SBCS-ID`. Let $\widehat{\rho}_L$ (respectively, $\widehat{\rho}_S$) denote the load at the long server (respectively, short server). The queues are stable iff $\widehat{\rho}_L < 1$ and $\widehat{\rho}_S < 1$.

We first deduce $\widehat{\rho}_L$. The PASTA (Poisson arrivals see time averages) principle implies that the fraction of the short jobs that are dispatched to the long server is $1-\widehat{\rho}_L$, assuming $\widehat{\rho}_L < 1$. Thus,

$$\widehat{\rho}_L = \rho_S(1-\widehat{\rho}_L)+\rho_L \iff \widehat{\rho}_L = \frac{\rho_S+\rho_L}{1+\rho_S}.$$

We therefore have the constraint that

$$\widehat{\rho}_L = \frac{\rho_S+\rho_L}{1+\rho_S} < 1 \iff \rho_L < 1.$$

Next we deduce $\widehat{\rho}_S$. The PASTA principle implies that the fraction of the short jobs that are dispatched to the short server is $\widehat{\rho}_L$, assuming $\widehat{\rho}_L < 1$. Thus,

$$\widehat{\rho}_S = \rho_S \cdot \widehat{\rho}_L.$$

We therefore have the constraint that

$$\rho_S \cdot \frac{\rho_S+\rho_L}{1+\rho_S} < 1 \qquad \iff \qquad \rho_S < \frac{1-\rho_L+\sqrt{(1-\rho_L)^2+4}}{2}.$$

The stability conditions for `SBCS-CQ` follow immediately via the law of large numbers. The long jobs (the number of the long jobs in the central queue) are stable iff $\rho_L < 1$, and the short jobs are stable iff $\rho_S < 2-\rho_L$. ■

The restriction on ρ_S for stability under each of the three task assignment policies is shown in Figure 5.2 as a function of ρ_L ($\rho_L < 1$ is necessary for all the three policies). Observe the advantage of cycle stealing in extending the stability region. When ρ_L is near zero, ρ_S can be as high as about 1.61 (golden ratio) under `SBCS-ID` and close to 2 under `SBCS-CQ`.

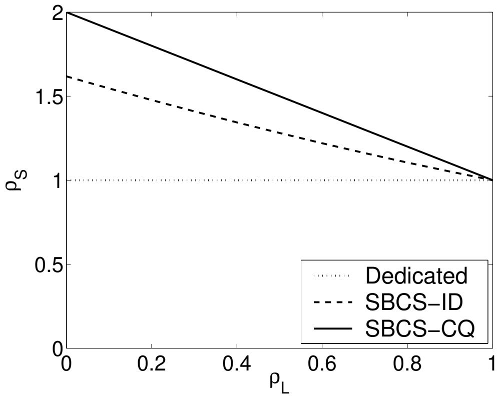


Figure 5.2: *Stability region for* `Dedicated`*,* `SBCS-ID`*, and* `SBCS-CQ`.

5.4.3 Mean response time

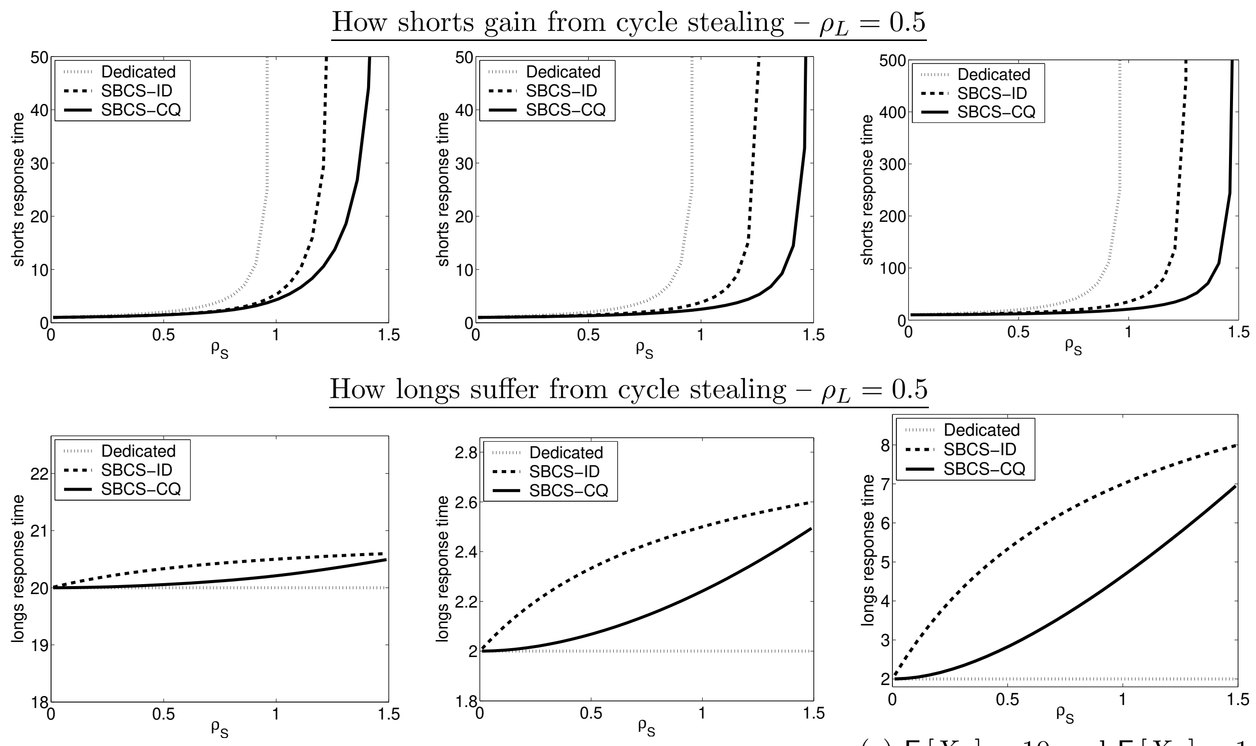
Figure 5.3 illustrates the benefit to the short jobs and the penalty to the long jobs under `SBCS-ID` and `SBCS-CQ` for three job size configurations: (a) "short" jobs are (10 times) shorter in expectation, (b) "short" jobs and "long" jobs are indistinguishable, and (c) "short" jobs are (10 times) longer in expectation. Here, we assume that both the short jobs and the long jobs have exponential distributions. The mean response time of the short jobs (respectively, long jobs) is plotted as a function of ρ_S in the top row (respectively, bottom row). In this figure, we fix $\rho_L = 0.5$, and consider the full range of ρ_S. Recall from Section 5.4.2 that `Dedicated` leads to instability when $\rho_S \geq 1$. However for `SBCS-CQ`, ρ_S is allowed as high as $2-\rho_L$, and for `SBCS-ID`, ρ_S is allowed as high as some intermediate value shown in Figure 5.2, which is not as high as $2-\rho_L$ and yet is higher than `Dedicated`.

Figure 5.3(a) (top row) shows that the short jobs benefit tremendously from cycle stealing. For $\rho_S > 0.8$, the mean improvement of cycle stealing algorithms over `Dedicated` is over an order of magnitude. As $\rho_S \to 1$, the mean response time under `Dedicated` goes to infinity, whereas it is 5.3 under `SBCS-ID` and 4.3 under `SBCS-CQ`. This shows the huge benefit that short jobs obtain by being able to steal idle cycles from the long host.

Figure 5.3(a) (top row) shows that the improvement of `SBCS-CQ` over `SBCS-ID` is also vast. As $\rho_S \to 1.28$, the mean response time under `SBCS-ID` goes to infinity whereas it is approximately 15.5 under `SBCS-CQ`. This follows as under `SBCS-ID` only *new* short arrivals can benefit from idle cycles, whereas under `SBCS-CQ` waiting short jobs may benefit.

Figure 5.3(a) (bottom row) shows that the penalty imposed on the long jobs by cycle stealing is relatively small. The penalty increases with ρ_S, but even when $\rho_S = 1$, the penalty to the long jobs is only 2.5% under `SBCS-CQ` and 3.0% under `SBCS-ID`. This is in contrast to the unbounded improvement for the short jobs.

Figure 5.3(b) shows the case when the short jobs and the long jobs are indistinguishable, and Figure 5.3(c) shows the pathological case where the "short" jobs have longer expected size than the "long" jobs. We see that trends are similar to column (a), with only the absolute magnitude of the numbers growing. Specifically, when the "short" jobs are longer or when the "long" jobs are shorter, both the benefit to the "short" jobs and the penalty to the "long" jobs become greater. Greater penalty to the "long" jobs is to be expected since a "long" job may get stuck behind a "short" job whose size is in fact ten time longer in expectation. Greater benefit to the "short" jobs may be



(a) $\mathsf{E}[X_S] = 1$ and $\mathsf{E}[X_L] = 10$ (b) $\mathsf{E}[X_S] = 1$ and $\mathsf{E}[X_L] = 1$ (c) $\mathsf{E}[X_S] = 10$ and $\mathsf{E}[X_L] = 1$ (pathological case)

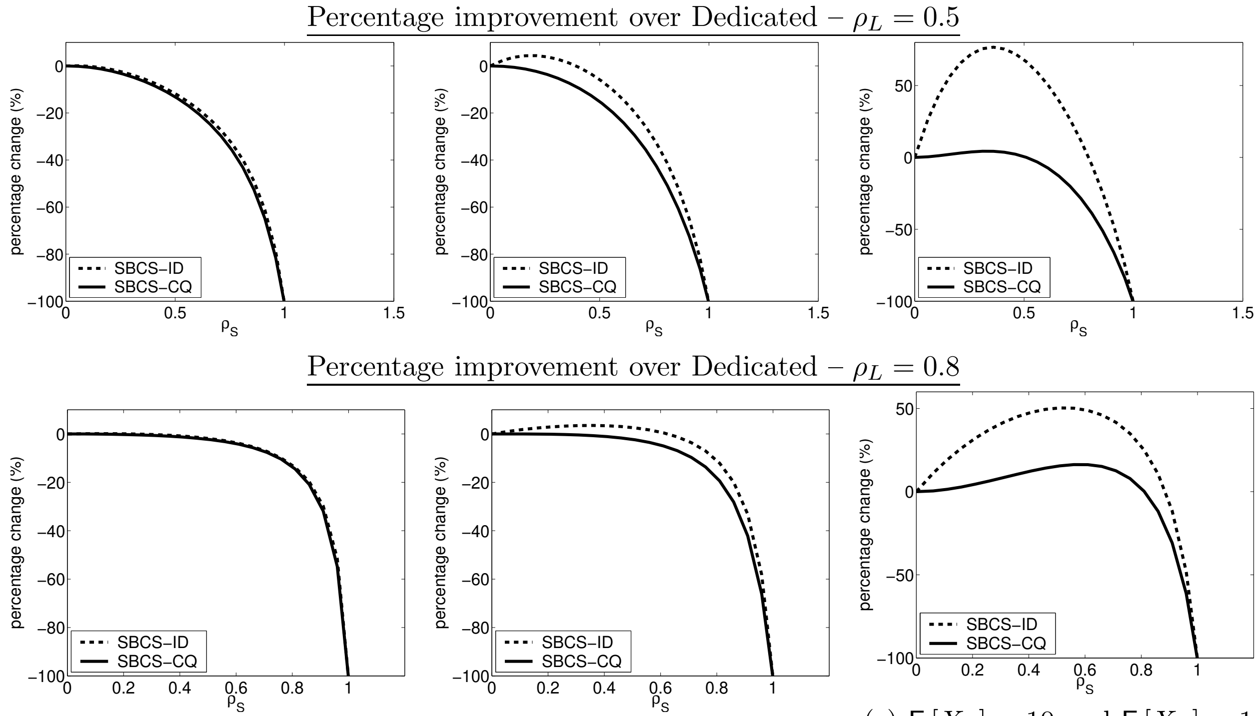
Figure 5.3: *Mean response time of short jobs and long jobs under* `Dedicated`, `SBCS-ID`, *and* `SBCS-ID`. *Job sizes have exponential distributions.*

explained as a counter effect to the greater penalty to the "long" jobs. Specifically, when the "long" jobs are shorter, the busy period of the "long" server is shorter, which allows the "long" server to help the "short" jobs more frequently (although the duration of the help is also shorter); at the beginning of the busy period, the "long" server may be serving the "short" jobs, and this implies that "short" jobs may utilize more *non*idle cycles of the "long" server.

One interesting observation is that the penalty to the long jobs appears lower under `SBCS-CQ` than under `SBCS-ID`. At first this seems quite contrary, since under `SBCS-CQ` more cycles are given to the short jobs; thus it is reasonable to expect the long jobs should suffer more. The reason this is not true is that under `SBCS-CQ` the servers are re-namable. Thus a long job arriving to find both servers serving short jobs need only wait for *the first* of the two servers to free up under `SBCS-CQ`.

Figure 5.4 illustrates the percentage improvement of the `SBCS-ID` and `SBCS-CQ` over the `Dedicated` with respect to the *overall* mean response time. The percentage change (in overall mean response time) against the `Dedicated` is plotted as a function of ρ_S for `SBCS-ID` and `SBCS-CQ`. In the top row (respectively, bottom row), $\rho_L = 0.5$ (respectively, $\rho_L = 0.8$) is fixed. Namely, in the top row, the parameter settings are exactly the same as in Figure 5.3, while ρ_L is set higher in the bottom row. Again, three columns correspond to three job size configurations.

Figure 5.4(a) shows that `SBCS-ID` and `SBCS-CQ` improves upon `Dedicated` with respect to the overall mean response time throughout the range of load, ρ_S, for both low ρ_L (top row) and high ρ_L (bottom row). As ρ_S becomes higher, the improvement over `Dedicate` becomes greater. At



(a) $\mathsf{E}[X_S] = 1$ and $\mathsf{E}[X_L] = 10$ (b) $\mathsf{E}[X_S] = 1$ and $\mathsf{E}[X_L] = 1$ (c) $\mathsf{E}[X_S] = 10$ and $\mathsf{E}[X_L] = 1$ (pathological case)

Figure 5.4: *Percentage change in the overall mean response time of* `SBCS-ID` *and* `SBCS-CQ` *against* `Dedicated`*. Job sizes have exponential distributions.*

$\rho_S = 1$, `Dedicated` becomes unstable, and the percentage change in the overall mean response time is -100% for both `SBCS-ID` and `SBCS-CQ`, which still provide finite overall mean response time at $\rho_S = 1$. (In fact, although not clear from the figure, the overall mean response time under `SBCS-ID` is slightly worse (within 0.1%) than that under `Dedicated` at very low load ($\rho_S \leq 0.2$). By contrast, the overall mean response time under `SBCS-CQ` is always lower than that under `Dedicated` with these parameter settings.)

Figure 5.4(b) shows that `SBCS-CQ` improves upon `Dedicated` with respect to the overall mean response time for all ρ_S, for both low ρ_L (top row) and high ρ_L (bottom row), even if the short jobs and long jobs are indistinguishable. Again, the improvement over `Dedicate` becomes greater as ρ_S becomes higher. On the other hand, the overall mean response time under `SBCS-ID` is slightly worse (within 5%) than that under `Dedicated` at low load (specifically, $\rho_S < 0.4$ for $\rho_L = 0.5$ (top row) and $\rho_S < 0.65$ for $\rho_L = 0.8$ (bottom row)). This makes intuitive sense, since `SBCS-ID` sends more jobs to the server with higher load if $\rho_S < \rho_L$. Overall, as long as $\rho_S \geq \rho_L$, both `SBCS-ID` and `SBCS-CQ` provides significant improvement over `Dedicated`, and this improvement becomes greater at higher ρ_S and becomes unbounded at $\rho_S \geq 1$.

Figure 5.4(c) shows that when the "short" jobs are longer than the "long" jobs (in the pathological case), the overall mean response time under both `SBCS-ID` and `SBCS-CQ` can be higher than that under `Dedicated`, but for high enough ρ_S, `SBCS-ID` and `SBCS-CQ` can still provide unbounded benefit over `Dedicated`. In the figure, `SBCS-CQ` improves upon `Dedicated` if $\rho_S > \rho_L$ for both low ρ_L (top row) and high ρ_L (bottom row). (The advantage of `SBCS-CQ` does not necessarily occur exactly at $\rho_S > \rho_L$.

For example, when $\rho_L \leq 0.3$, SBCS-CQ improves upon Dedicated for all ρ_S). On the other hand, SBCS-ID improves upon Dedicated only at high ρ_S, but again the improvement of both SBCS-ID and SBCS-CQ becomes greater at higher ρ_S and becomes unbounded at $\rho_S \geq 1$.

Figure 5.4 shows that at higher ρ_L (bottom row), both the benefit to the short jobs and the penalty to the long jobs are reduced, and overall the improvement (or disimprovement) of SBCS-ID and SBCS-CQ over Dedicated becomes smaller. This is to be expected, as there are fewer idle cycles to steal at higher ρ_L.

Figure 5.5 considers the effect of increasing the service demand variability of the long jobs. The parameter settings are the same as in Figure 5.3 and the top row of Figure 5.4, except that two curves are shown for each of Dedicated, SBCS-ID, and SBCS-CQ, corresponding to the case where the long jobs have a (two phase) PH distribution with $C_L^2 = 8$ and the case where the long jobs have an exponential distribution[1].

Theorem 9 implies that when C_L^2 is increased from 1 to 8, the mean response time of the long jobs under SBCS-ID and under Dedicated increases by the same amount, and this can also be verified in the middle row of Figure 5.5. The figure also suggests that the the mean response time of the long jobs under SBCS-CQ increases by approximately the same amount as Dedicated. Closer examination shows that the difference in the increase in the mean response time of the long jobs between SBCS-CQ and Dedicated is only at most 0.4% in (a), at most 0.03% in (b), and at most 2.8% in (c), and the difference appears to be larger when the error in the analysis of SBCS-CQ against simulation is larger. Since the mean response time of the long jobs is higher with higher C_L^2, the percentage penalty of the long jobs is therefore considerably lessened when C_L^2 is increased.

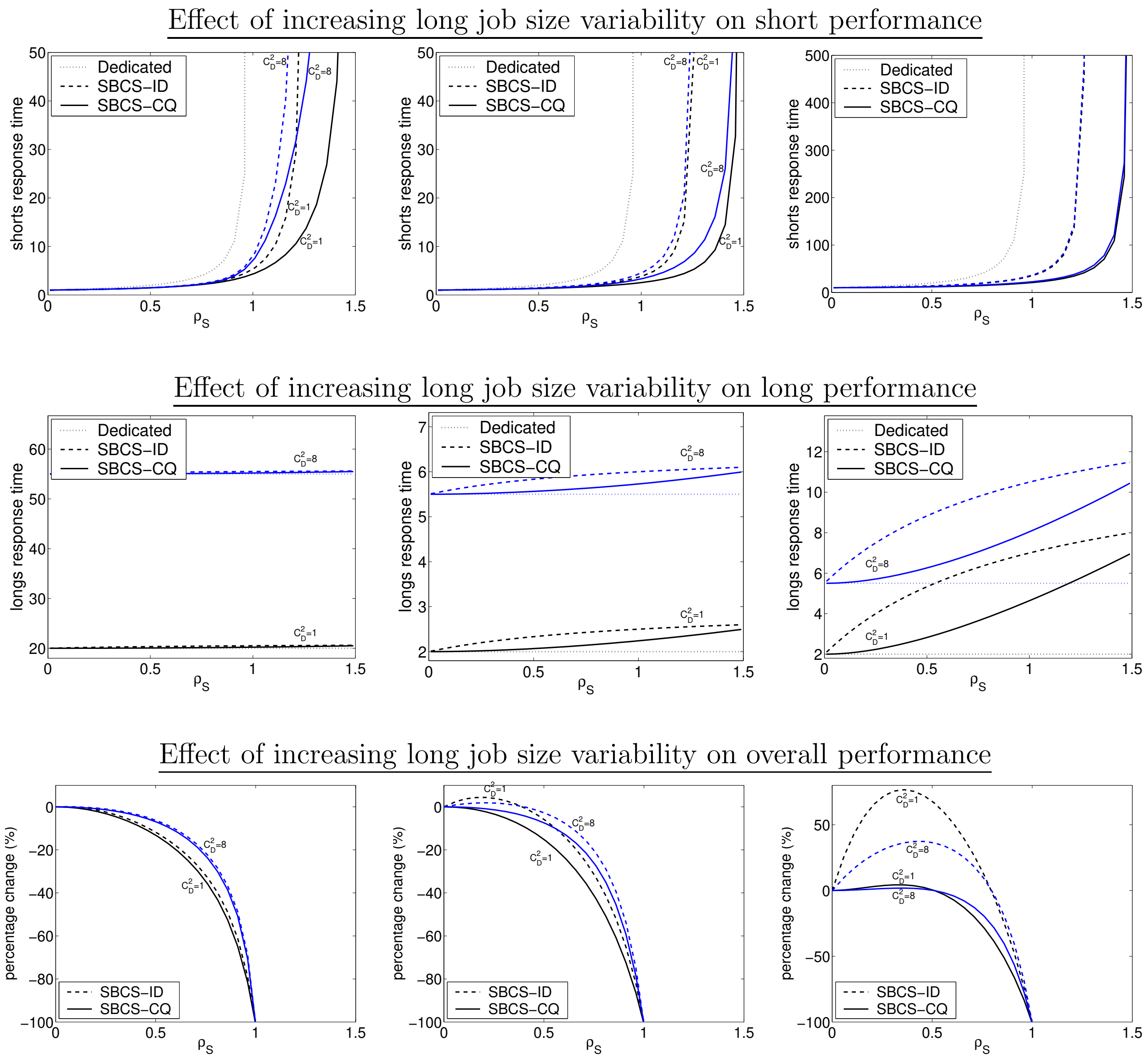
The top row of Figure 5.5 shows that increasing long job size variability has surprisingly little impact on the mean response time of the short jobs under SBCS-ID when $\rho_S < 1$, although there is a noticeable impact when $\rho_S > 1$ and the "short" jobs are shorter than the "long" jobs (column (a)). We would have expected the opposite: that shorts would prefer less variable long job sizes because that means less variable long busy periods and more regular help. In fact, this expectation better applies to SBCS-CQ: the mean response time of the short jobs under SBCS-CQ increases more significantly by increasing long job size variability. This makes intuitive sense, since the short jobs rely on the help from the long server more under SBCS-CQ than under SBCS-ID.

To summarize, we have seen that short jobs are tremendously helped by cycle stealing, and that SBCS-CQ offers greater improvements to short jobs than SBCS-ID. We have also seen that, provided that "short" jobs are no longer than "long" jobs, the penalty of cycle stealing on long jobs is negligible. This penalty is greater under SBCS-ID than under SBCS-CQ. Thus, SBCS-CQ is always superior to SBCS-ID, and both are typically far better than Dedicated.

5.5 Concluding remarks

In this chapter, size-based task assignment policies with cycle stealing, SBCS-ID and SBCS-CQ, are proposed, and their mean response time is analyzed via dimensionality reduction (DR), as the analysis of SBCS-ID and SBCS-CQ involves multidimensional Markov chains. Under SBCS-ID and SBCS-CQ, short jobs are processed by one server (short server) and long jobs are processed by another server

[1] To determine the parameters of the two phase PH distribution, the third moment needs to be specified as well. Here, we choose the third moment, so that the normalized second and third moments, m_2 and m_3, satisfy $m_3 = 2m_2 - 1$. See Section 2.5 for an intuition for this choice; in particular, the exponential distribution and the Erlang distribution satisfy $m_3 = 2m_2 - 1$.



(a) $\mathsf{E}[X_S] = 1$ and $\mathsf{E}[X_L] = 10$ (b) $\mathsf{E}[X_S] = 1$ and $\mathsf{E}[X_L] = 1$ (c) $\mathsf{E}[X_S] = 10$ and $\mathsf{E}[X_L] = 1$ (pathological case)

Figure 5.5: Effect of variability in long job size on the mean response time.

(long server), but when there are no long jobs, the long server also processes short jobs, i.e. short jobs can steal idle cycles of the long server. DR allows us to quantify the benefit and penalty of cycle stealing, and provides insights into good task assignment policies.

Our analysis shows that the short jobs can benefit immensely from cycle stealing, while long jobs experience only a small penalty. Also, we find that holding jobs at a central queue (`SBCS-CQ`) provides more benefit of cycle stealing to the short jobs with a smaller penalty to the long jobs than immediately dispatching jobs upon their arrivals (`SBCS-ID`). Thus, overall, `SBCS-CQ` is a superior policy to `SBCS-ID`. In fact, `SBCS-ID` often provides an unbounded benefit over the `Dedicated` policy (without cycle stealing), and `SBCS-CQ` often provides an unbounded benefit over `SBCS-ID`. The unbounded benefit stems from the increased stability region under `SBCS-ID` and `SBCS-CQ`. We provided the necessary and sufficient conditions for the stability under these policies.

We also consider the case where the "short" jobs are indistinguishable from the "long" jobs, and the pathological case where the "short" jobs are longer than the "long" jobs. Our analysis shows that `SBCS-ID` and `SBCS-CQ` can still provide an unbounded benefit over `Dedicated` when the load of short jobs is high, but the benefit diminishes as the load becomes lower. In fact, when the load of short jobs are low, cycle stealing (especially, `SBCS-ID`) can worsen the overall mean response time. However, since the performance improvement is most needed at high load, `SBCS-ID` and `SBCS-CQ` may still have a practical use even in the pathological case.

An advantage of the analysis via DR is that the job sizes can be modeled as a PH distribution. This allows us to study the impact of job size variability on performance. In particular, we study the impact of the variability of the long job sizes on the mean response time of the short jobs. We find that the impact is surprisingly small under `SBCS-ID`, but larger under `SBCS-CQ`. Another advantage of the analysis via DR is that it allows us to evaluate the mean response time accurately for a wide range of loads. This is in contrast to heavy traffic approximations, which becomes more accurate at higher loads. We have seen that the benefit and penalty of cycle stealing are quite different at low load and high load. Specifically, when the load of the short jobs is low, cycle stealing can worsen the mean response time. It would be interesting to see whether heavy traffic approximations can capture this.

In this chapter, we limit our discussion to a system with two homogeneous servers, but the analysis via DR can be extended to heterogeneous servers and multiple servers in various ways. For example, in [68], we study `SBCS-ID` with multiple short servers that independently receives short jobs and steal idle cycles of the long server. We find that the benefit of cycle stealing reduces as the number of short servers increases, but the most reduction comes from the first additional short server. In [68], we also extend Theorem 9 and Theorem 10 to the case of multiple short servers. Also, in Chapter 3, we show that `SBCS-ID` with m classes of jobs and m servers can be modeled as an RFB process, and validated the analysis via DR against simulation. These analysis will be useful in practice to decide, for example, how many servers are needed and how much capacity is needed for each server to guarantee certain mean response time, when `SBCS-ID` is used in real systems.

Chapter 6

Reducing switching costs in cycle stealing

When does cycle stealing pay, and how much?

In this chapter, we study fundamental problems regarding the benefit and penalty of cycle stealing, where the "donor" server can help the "beneficiary" server when there are no donor's jobs. Dimensionality reduction (DR) allows us to analyze the mean response time of donor's jobs and beneficiary's jobs under a wide range of parameter settings. Results of our analysis illuminate principles on the general benefits of cycle stealing and the design of cycle stealing policies.

6.1 Motivation

Since the invention of networks of workstations, systems designers have touted the benefits of allowing a user to take advantage of machines other than her own, at times when those machines are idle. This notion is often referred to as *cycle stealing*. Cycle stealing allows such a user, Betty, with multiple jobs, to offload one of her jobs to the machine of a different user, Dan, if Dan's machine is idle, giving Betty two machines to process her jobs. When Dan's workload resumes, Betty must return to using only her own machine. We refer to Betty as the *beneficiary*, to her machine as the beneficiary machine/server, and to her jobs as beneficiary jobs. Likewise, we refer to Dan as the *donor*, to his machine as the donor machine/server, and to his jobs as donor jobs.

Although cycle stealing provides obvious benefits to the beneficiary, these benefits come at some cost to the donor. For example, the beneficiary job may have to be checkpointed and the donor's working set memory reloaded before the donor server can resume, delaying the resumption of processing of donor jobs. Furthermore, there is also some wasted time for the beneficiary who may experience various remote execution costs, and must wait for her working set memory to be loaded into the donor machine before she can use the donor machine. In our model we refer to these additional costs associated with cycle stealing as *switching costs* (or *switching times*).

A primary goal of this chapter is to understand the benefit of cycle stealing for the beneficiary and the penalty to the donor, as a function of switching times. A secondary goal is to derive parameter settings for cycle stealing. In particular, given non-zero switching times, cycle stealing may pay only if the beneficiary's queue is "sufficiently" long. We seek to understand the optimal threshold on the beneficiary queue when switching to help, and the optimal threshold on the donor queue when

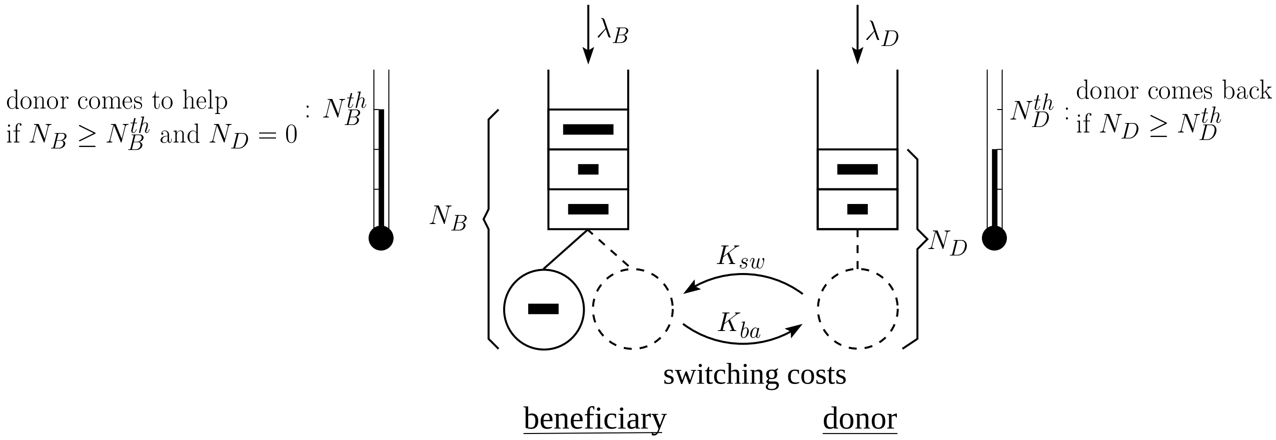


Figure 6.1: *A threshold-based policy for reducing switching costs in cycle stealing.*

switching back. More broadly, we seek general insights into which system parameters have the most significant impact on the effectiveness of cycle stealing.

6.2 Threshold based policy for reducing switching costs in cycle stealing

Specifically, we assume there are two queues, the *beneficiary* queue and the *donor* queue, with independent arrival processes and service time distributions operating as M/PH/1/FCFS queues (see Figure 6.1). Jobs arrive at rate λ_B (respectively, λ_D) at the beneficiary (respectively, donor) queue and have service demand X_B with a phase type (PH) distribution G_B (respectively, X_D with a PH distribution G_D). The load made up by beneficiary (respectively, donor) jobs is denoted by ρ_B (respectively, ρ_D) where $\rho_B = \lambda_B \mathsf{E}[X_B]$ and $\rho_D = \lambda_D \mathsf{E}[X_D]$.

To reduce the switching costs associated with cycle stealing, we consider the following threshold-based policy. If there are no donor jobs ($N_D = 0$), *and* if the number of the beneficiary jobs, N_B, is at least N_B^{th}, the donor transitions into the switching state, for a random amount of time, K_{sw}, having a PH distribution. After K_{sw} time, the donor server is available to work on the beneficiary queue and the beneficiary queue becomes an M/G_B/2/FCFS queue. When the number of donor jobs, N_D, reaches N_D^{th} (either during K_{sw}, or during the time the donor is helping the beneficiary), the donor transitions into a switching back state, for a random amount of time, K_{ba}, having a PH distribution. After the completion of the switch back, the donor server resumes working on its own jobs until the donor queue is empty. The donor server cannot work on any job while the donor is in the switching or switching back state.

A few details are in order. First, in the above model, the donor server continues to cooperate with the beneficiary even if there is no beneficiary work left for it to do — the donor server can switch back only when a donor job arrives. Second, we assume that if the donor server is working on a beneficiary job and a donor job arrives, that beneficiary job is returned to the beneficiary queue and will be resumed by the beneficiary server as soon as that server is available. The work done on the job is not lost (i.e. preemptive resume). Our performance metric throughout is mean response time (overall and for each class of jobs), where the response time of a job is the time from when the job arrives until it completes service. We assume the first three moments of the service times are finite, and queues are stable.

Our model deals with switching times in a general way, making the results both applicable to a shared-memory multiprocessor architecture and to a network of workstations (NOW). Our switching times, K_{sw} and K_{ba}, can be viewed as the time for switching between job types in a shared-memory multiprocessor architecture. In a NOW, there is an additional overhead incurred from migrating jobs from one server to another, where jobs which have not started running require negligible overhead, whereas migrating a "running" job (in progress) requires high overhead, since all of its state must be migrated with it. Our model captures this notion in that the switching back time, K_{ba}, can be viewed as the time incurred for preempting an already running job and returning it to the beneficiary.

Analysis

With the above assumptions, the system under the threshold-based policy for reducing switching costs in cycle stealing can be modeled as a GFB (generalized foreground-background) process, as discussed in Section 3.4. An analysis of the GFB process via dimensionality reduction (DR) provides mean response time of the beneficiary jobs and the donor jobs, respectively. In fact, the donor job size, X_B, and the switching back time, K_{ba}, can be extended to general distributions, as we show in [152]. All the details of the analysis to generate the results in Section 6.4 are provided in [152]. In particular, the mean response time of donor jobs has an alternative derivation via an M/GI/1 queue with generalized vacations [57], as shown in the following theorem:

Theorem 11 *The mean response time of donor jobs, $\mathsf{E}[T_D]$, is given by*

$$\mathsf{E}[T_D] \;=\; \mathsf{E}[X_D] + \frac{\lambda_D \mathsf{E}\left[X_D^2\right]}{2(1-\rho_D)} + \frac{\left(N_D^{th}(N_D^{th}-1) + 2N_D^{th}\lambda_D \mathsf{E}[K_{ba}] + \lambda_D^2 \mathsf{E}\left[K_{ba}^2\right]\right)p}{2\lambda_D\left((N_D^{th}+\lambda_D\mathsf{E}[K_{ba}])p + (1-p)\right)},$$

where

$$p = \frac{P_{N_D^{th}-1}}{P_{N_D^{th}-1} + P_0} \tag{6.1}$$

where P_0 is the probability that the donor server is at the donor queue and *the number of donor jobs is zero; $P_{N_D^{th}-1}$ is the probability that the donor server is either at the beneficiary queue or in the process of switching to the beneficiary queue* and *the number of donor jobs is $N_D^{th}-1$.*

Note that P_0 and $P_{N_D^{th}-1}$ can be calculated via the limiting probabilities in the 1D Markov chain reduced from the GFB process in the analysis of the mean response time of the beneficiary jobs via DR.

6.3 State of the art in cycle stealing

There has been no previous analytical work on the performance of cycle stealing with switching times and thresholds. The analysis of cycle stealing *without* switching times and thresholds under exponential job size distributions has been studied by Green [61] and Stanford and Grassmann [185, 186]. Since the analysis of cycle stealing involves 2D Markov chains even without switching times and thresholds, in [61, 185, 186], the 2D Markov chains are analyzed by truncating the state space (and the resulting 1D Markov chains are analyzed by matrix analytic methods (Section 3.2)).

Despite the lack of analytical study, cycle stealing has been implemented and used in various systems, including networks of workstations (NOWs). Below, we review cycle stealing mechanisms in NOWs, classifying them into three types, depending on the behavior when an idle workstation is reclaimed by its owner. Other computer systems with cycle stealing are also discussed briefly.

In the first type of cycle stealing mechanism, the jobs that have been running on an idle server are killed when the idle server is reclaimed by its owner. Butler [139] is a representative implementation of this type of cycle stealing. This cycle stealing model is also studied by Bhatt et. al. [20] and Rosenberg [163, 164, 165], who consider a theoretical problem of optimal cycle stealing strategy. This type of cycle stealing is the most conservative in that the cost (time) required for the owner to reclaim his idle machine is smallest. However, in this model, all the work that has been done on the idle workstation is lost on reclaiming.

In the second type of cycle stealing, the job that have been running on an idle workstation is thus migrated to other idle workstations, while keeping the state of the job. The cycle stealing model that we study in this chapter is close to this type. Condor [115, 116], Sprite [48], and Benevolent Bandit [54] are representative implementation of this type of cycle stealing[1]. A disadvantage of the first two types of cycle stealing mechanisms is that they cannot efficiently make use of short idle periods, and this is a motivation for the next mechanism.

In the third mechanism, the jobs that have been running on an idle workstation are preempted but keep staying on the workstation when the idle workstation is reclaimed by its owner. This mechanism for networks of workstations is implemented as Stealth Distributed Scheduler [109] and Linger-Longer [170]. Awerbuch et. al. [13] consider a problem of optimal policies in this mechanism.

Cycle stealing is also popular in Internet computing, which allows one to steal idle cycles of personal computers at home. Internet computing is used for the purpose of searching extraterrestrial intelligence, protein folding research, cancer research, AIDS research, finding large prime numbers, etc. Systems that support Internet computing include Javelin [41]. Other concepts related to load balancing or cycle stealing include dynamic resource allocation, which dynamically assign resources among multiple applications at data centers [9, 39, 38], and content distribution networks [107] such as Akamai. Adaptive parallelism is also related to cycle stealing; it dynamically reallocates processors to parallel applications, depending on the availability of the processors. Representative implementations of adaptive parallelism includes Piranha [36] and ATLAS [15]. More generally, the concept of using multiple systems as a single resource is studied in the context of grid computing [14, 40, 56]. System implementations towards facilitating grid computing includes AppLeS [18], Legion [62], and MOL [59].

6.4 Results

6.4.1 Summary of results

Our analysis yields many interesting results concerning cycle stealing. While cycle stealing obviously benefits the beneficiary jobs and hurts the donor jobs, we find that when $\rho_B \geq 1$, cycle stealing is profitable overall even under significant switching times, as it may ensure stability of the beneficiary queue. For $\rho_B < 1$, we define load-regions under which cycle stealing pays. We find that in general

[1]Other systems that support process migration include Charlotte [11], Accent [211], Mach [128], Emerald [91], Tue [179], and Locus [191].

the switching time is only prohibitive when it is large compared with $\mathsf{E}[X_D]$. Under zero switching times, cycle stealing always pays.

Two counterintuitive results are that when $\rho_B < 1$, the mean response time of the beneficiary jobs is surprisingly insensitive to the switching time, and also insensitive to the variability of the donor job size distribution. Even when the variability of the donor job sizes is very high, and donor help thus is very bursty, the beneficiary jobs still enjoy significant benefits.

Our analysis also allows us to investigate characteristics of the beneficiary and donor side thresholds, N_B^{th} and N_D^{th}, both with respect to their impact on stability and their impact on mean response time. With respect to beneficiary stability, we find that N_B^{th} has no effect, while increasing N_D^{th} increases the stability region. Donor stability is not affected by either threshold. With respect to overall mean response time, we find that mean response time is far more sensitive to changes in N_D^{th} than to changes in N_B^{th}. We find the optimal value of N_B^{th} tends to be well above 1. The reason is that increasing N_B^{th} does not appreciably diminish beneficiary gain, but it does alleviate donor pain. We find that the optimal setting of N_B^{th} is an increasing function of ρ_B, ρ_D, and switching times. By contrast, we find that the optimal value of N_D^{th} is often close to 1, provided $\rho_B < 1$. Increasing N_D^{th} significantly hurts the donor, although it may provide significant help to the beneficiary if ρ_B is high. We find that the optimal N_D^{th} is *not* a monotonic function of ρ_D, but is an increasing function of ρ_B and switching times.

6.4.2 Stability

In this section we derive stability conditions for cycle stealing with switching times and thresholds. We find for example that the stability condition for the donor jobs remains $\rho_D < 1$, regardless of whether the donor jobs experience switching times. By contrast, the stability region for the beneficiary jobs can be significantly below $2 - \rho_D$, specifically because the switching time erodes the beneficiary stability region. Also, interestingly, the value of N_B^{th} does not affect the stability region of either the donor or beneficiary jobs. By contrast, increasing N_D^{th} increases the stability region of the beneficiary jobs; however it has no effect of the stability region of the donor jobs.

Theorem 12 *The donor queue is stable iff $\rho_D < 1$.*

Proof: It makes intuitive sense that the donor queue is stable iff $\rho_D < 1$ regardless of the switching times and threshold values, since the donor would never switch if the donor queue was persistently backlogged. Below, we prove sufficiency more formally. The necessity of $\rho_D < 1$ is trivial.

Assume $\rho_D < 1$. Let B_D denote the length of a busy period at the donor queue. We first consider the case of $N_B^{th} = 0$. A busy period at the donor queue is started by a switching time K_{ba} and N_D^{th} donor jobs. As $\rho_D < 1$, the mean length of a busy period is

$$\mathsf{E}[B_D] = \frac{N_D^{th}\mathsf{E}[X_D] + \mathsf{E}[K_{ba}]}{1 - \rho_D} < \infty.$$

In this case B_D clearly has a proper limiting distribution, and hence the donor is stable. Next we consider the case of $N_B^{th} > 0$. In this case $\mathsf{E}[B_D]$ is smaller than in the case of $N_B^{th} = 0$ because there will be donor busy periods in which the donor server has not left the donor queue, implying (i) there is no switching back time, and (ii) the busy period is started by only one donor job. ∎

Before we derive the stability condition on ρ_B, we prove a lemma allowing us to assume $N_B^{th} = 0$.

Lemma 8 *If the beneficiary queue is stable at $N_B^{th} = 0$, then it is stable at $N_B^{th} = n$, $\forall\ 0 \le n < \infty$.*

Proof: Intuitively, the stability of the beneficiary queue is independent of N_B^{th}, since N_B^{th} would be irrelevant when the beneficiary queue was continuously backlogged.

More formally, let $L_B^{(n)}(t)$ denote the number of beneficiary jobs at time t given $N_B^{th} = n \ge 1$. Consider a new process $\widehat{L}_B^{(n)}(t)$ in which the number of jobs at time $t = 0$ is n, instead of 0 as in the original process, and no service is given by either server to a beneficiary job if there are $\le n$ jobs present at the beneficiary queue. Note that $\widehat{L}_B^{(n)}(t)$ stochastically dominates $L_B^{(n)}(t)$. Also, along any sample path, $\widehat{L}_B^{(n)}(t)$ will be equal to $n + L_B^{(0)}(t)$. Therefore, if $L_B^{(0)}(t)$ is proper, so too is $\widehat{L}_B^{(n)}(t)$ and hence $L_B^{(n)}(t)$. ■

We now prove the stability condition on ρ_B.

Theorem 13 *The beneficiary queue is stable iff*

$$\rho_B < 1 + \frac{\max(1-\rho_D, 0)\sum_{i=0}^{N_D^{th}} (N_D^{th} - i)\frac{(-\lambda_D)^i}{i!}\widetilde{K}_{sw}^{(i)}(\lambda_D)}{N_D^{th} + \lambda_D \mathsf{E}[K_{ba}]}, \tag{6.2}$$

where $\widetilde{K}_{sw}(s)$ is the Laplace transform of K_{sw} and $\widetilde{K}_{sw}^{(i)}(s)$ is its i-th derivative. In particular, when K_{sw} is exponentially distributed, the condition is expressed by the closed form:

$$\rho_B < 1 + \frac{\max(1-\rho_D, 0)\left\{N_D^{th} - \frac{q}{1-q}\left(1 - q^{N_D^{th}}\right)\right\}}{N_D^{th} + \lambda_D \mathsf{E}[K_{ba}]}, \tag{6.3}$$

where

$$q = \frac{\lambda_D \mathsf{E}[K_{sw}]}{1 + \lambda_D \mathsf{E}[K_{sw}]}.$$

Proof: We first prove sufficiency. By Lemma 8, we can assume $N_B^{th} = 0$. Let F denote the fraction of time that the donor server helps the beneficiary. Then the strong law of large numbers can be used to show that the necessary and sufficient condition for stability of the beneficiary jobs is $\rho_B < 1 + F$. If $\rho_D \ge 1$, $F = 0$. Thus we assume $\rho_D < 1$ and derive F using renewal reward theory.

Let a renewal occur every time the donor queue becomes empty (see Figure 6.2). Recall $N_B^{th} = 0$. By renewal-reward theory, the fraction of time donor helps beneficiary is $F = \frac{\mathsf{E}[R]}{\mathsf{E}[Y]}$, where R denotes the total time that donor helps beneficiary (reward) during the renewal cycle, and Y denotes the length of the renewal cycle. Observe that there may be any number of donor arrivals ranging from 0 to N_D^{th} during K_{sw} and we switch back only after the N_D^{th} arrival.

Let S denote the sum of the service times of N_D^{th} donor jobs and B_{S+ba} denote the length of the busy period started by these jobs of total size S plus K_{ba}. Then, as $\rho_D < 1$,

$$\mathsf{E}[Y] = N_D^{th}\mathsf{E}[I_D] + \mathsf{E}[B_{S+ba}] = \frac{N_D^{th}}{\lambda_D} + \frac{N_D^{th}\mathsf{E}[X_D] + \mathsf{E}[K_{ba}]}{1-\rho_D},$$

where I_D is the interarrival time for donor jobs.

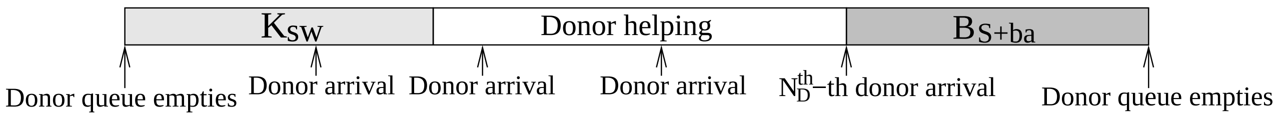


Figure 6.2: *A renewal cycle of the donor queue under the threshold-based policy for reducing switching costs in cycle stealing.]*

To compute $\mathsf{E}[R]$, we condition on the number of donor jobs arriving during K_{sw}. If there are i arrivals, then the mean time spent helping is the time until there are $(N_D^{th} - i)$ more donor arrivals, $(N_D^{th} - i)\mathsf{E}[I_D]$. Let p_i denote the probability that there are i donor jobs arriving during K_{sw}. Then,

$$p_i = \frac{(-\lambda_D)^i}{i!}\widetilde{K}_{sw}^{(i)}(\lambda_D).$$

Using p_i, $\mathsf{E}[R]$ is now derived as follows:

$$\mathsf{E}[R] \;=\; \sum_{i=0}^{N_D^{th}-1} (N_D^{th} - i)\frac{1}{\lambda_D}p_i.$$

Thus, $\rho_B < 1 + F$ is equivalent to (6.2). When K_{sw} is exponentially distributed, $p_i = q^i(1-q)$, where $q = \lambda_D/(\lambda_D + \mu_{sw})$. Therefore,

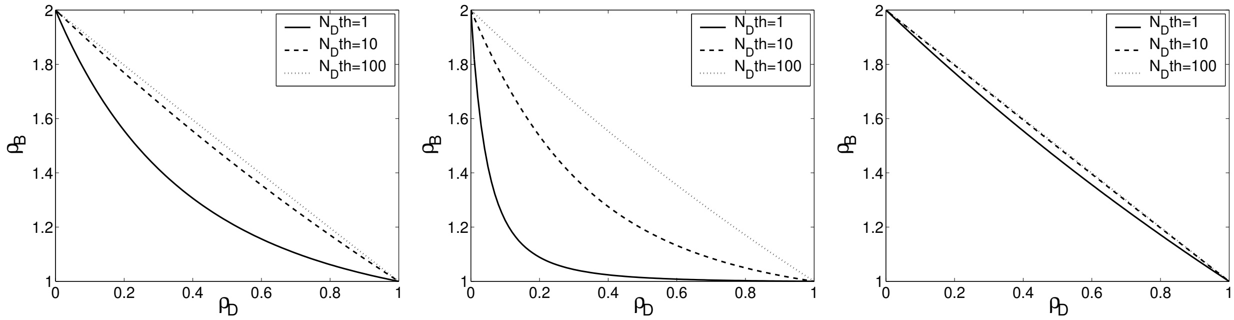
$$\mathsf{E}[R] \;=\; \frac{1}{\lambda_D}\left\{N_D^{th} - \frac{q}{1-q}(1 - q^{N_D^{th}})\right\}.$$

Thus, $\rho_B < 1 + F$ is equivalent to (6.3).

Above, we have proved the necessary and sufficient condition for $N_B^{th} = 0$. By Lemma 8, this is also the sufficient condition for $N_B^{th} > 0$. Now, we prove necessity for $N_B^{th} > 0$. When $N_B^{th} > 0$, the donor server does not necessarily help the beneficiary even when it is available for help. Therefore, there are two types of renewal periods. In the first type of renewal period, the donor server helps the beneficiary, i.e. $R > 0$. In this case, $\mathsf{E}[Y]$ is the same as for $N_B^{th} = 0$, and $\mathsf{E}[R]$ for $N_B^{th} > 0$ is at most $\mathsf{E}[R]$ for $N_B^{th} = 0$. In the second type of renewal period, the donor server does not help the beneficiary, i.e. $R = 0$. (In this case $\mathsf{E}[Y]$ can be smaller than that for $N_B^{th} = 0$.) The fraction of time, F, that the donor server helps the beneficiary can be expressed as $F = F_1 + F_2$, where F_1 is the fraction of time that the donor server helps the beneficiary and the renewal period is type 1, and F_2 is the fraction of time that the donor server helps the beneficiary and the renewal period is type 2. In fact, $F_2 = 0$. Therefore, $F = F_1 \le \mathsf{E}[R]/\mathsf{E}[Y]$. This proves the necessity for $N_B^{th} > 0$. ■

Note that the right hand side of (6.2) is an increasing function of N_D^{th}; in terms of stability, the larger N_D^{th}, the more stable the beneficiary queue. In particular, when $N_D^{th} = 0$, (6.2) is $\rho_B < 1$; as $N_D^{th} \to \infty$, (6.2) becomes $\rho_B < 2 - \rho_D$.

Figure 6.3 shows the stability condition for beneficiary jobs as a function of ρ_D when K_{sw} is exponentially distributed. In case (i), we set $\mathsf{E}[X_D] = 1$ and $\mathsf{E}[K_{sw}] = \mathsf{E}[K_{ba}] = 1$. The region below each line satisfies the stability condition. As N_D^{th} increases as high as 100, the effect of



(i) $\mathsf{E}[X_D] = 1, \mathsf{E}[K] = 1$ (ii) $\mathsf{E}[X_D] = 1, \mathsf{E}[K] = 10$ (iii) $\mathsf{E}[X_D] = 10, \mathsf{E}[K] = 1$

Figure 6.3: *Stability region for the threshold-based policy for reducing switching costs in cycle stealing. Switching times are denoted by* $K \equiv K_{sw} = K_{ba}$.

switching overhead becomes negligible, and the stability condition approaches $\rho_B < 2 - \rho_D$. In case (ii), we set $\mathsf{E}[X_D] = 1$ and $\mathsf{E}[K_{sw}] = \mathsf{E}[K_{ba}] = 10$. The switching time is large, and there is little benefit at moderate or high ρ_D in terms of stability, unless N_D^{th} is large. However, there is still large benefit at low ρ_D. In case (iii), we set $\mathsf{E}[X_D] = 10$ and $\mathsf{E}[K_{sw}] = \mathsf{E}[K_{ba}] = 1$. The stability region is larger; when $N_D^{th} = 1$, the stability region is almost the same as that of $N_D^{th} = 10$ in case (i). This is intuitive: when $\mathsf{E}[X_D] = 10$ and $N_D^{th} = 1$, the expected amount of donor work when the donor switches back is the same as that when $\mathsf{E}[X_D] = 1$ and $N_D^{th} = 10$.

6.4.3 Mean response time

In this section, we study the mean response time of the beneficiary jobs and the donor jobs, analyzed via DR. Throughout we will use the term "gain" to denote the improvement (drop) in mean response time experienced by beneficiary jobs under cycle stealing, as compared with dedicated servers (without cycle stealing), and the term "pain" to refer to the increase in mean response time experienced by donor jobs under cycle stealing as compared with dedicated servers:

$$\text{gain} = \frac{\mathsf{E}[T_B]^{\texttt{Dedicated}}}{\mathsf{E}[T_B]^{\texttt{CS}}} \quad \text{and} \quad \text{pain} = \frac{\mathsf{E}[T_D]^{\texttt{CS}}}{\mathsf{E}[T_D]^{\texttt{Dedicated}}},$$

where $\mathsf{E}[T_B]^{\texttt{Dedicated}}$ refers to the mean response time of beneficiary jobs under dedicated servers and $\mathsf{E}[T_B]^{\texttt{CS}}$ refers to the mean response time of beneficiary jobs under cycle stealing. $\mathsf{E}[T_D]^{\texttt{Dedicated}}$ and $\mathsf{E}[T_D]^{\texttt{CS}}$ are defined similarly. Observe that both pain and gain have been defined to range from 1 to ∞, where infinite gain corresponds to the situation where the mean response time under dedicated is infinite while it is finite under cycle stealing. In Figures 6.4-6.7, we fix the threshold values as $N_B^{th} = N_D^{th} = 1$ and study the effect of other parameters, and in Figures 6.8-6.9 we study the effect of the threshold values.

Benefits of cycle stealing: wide range ρ_B

Cycle stealing is always a win when $\rho_B \geq 1$*, but does not pay when* $\rho_B \leq 0.5$.

Figure 6.4 shows the mean response time for beneficiary jobs (rows 1 and 3) and donor jobs (rows 2 and 4) as a function of ρ_B, where ρ_D is low-to-medium ($\rho_D = 0.5$; top half) and medium-to-high

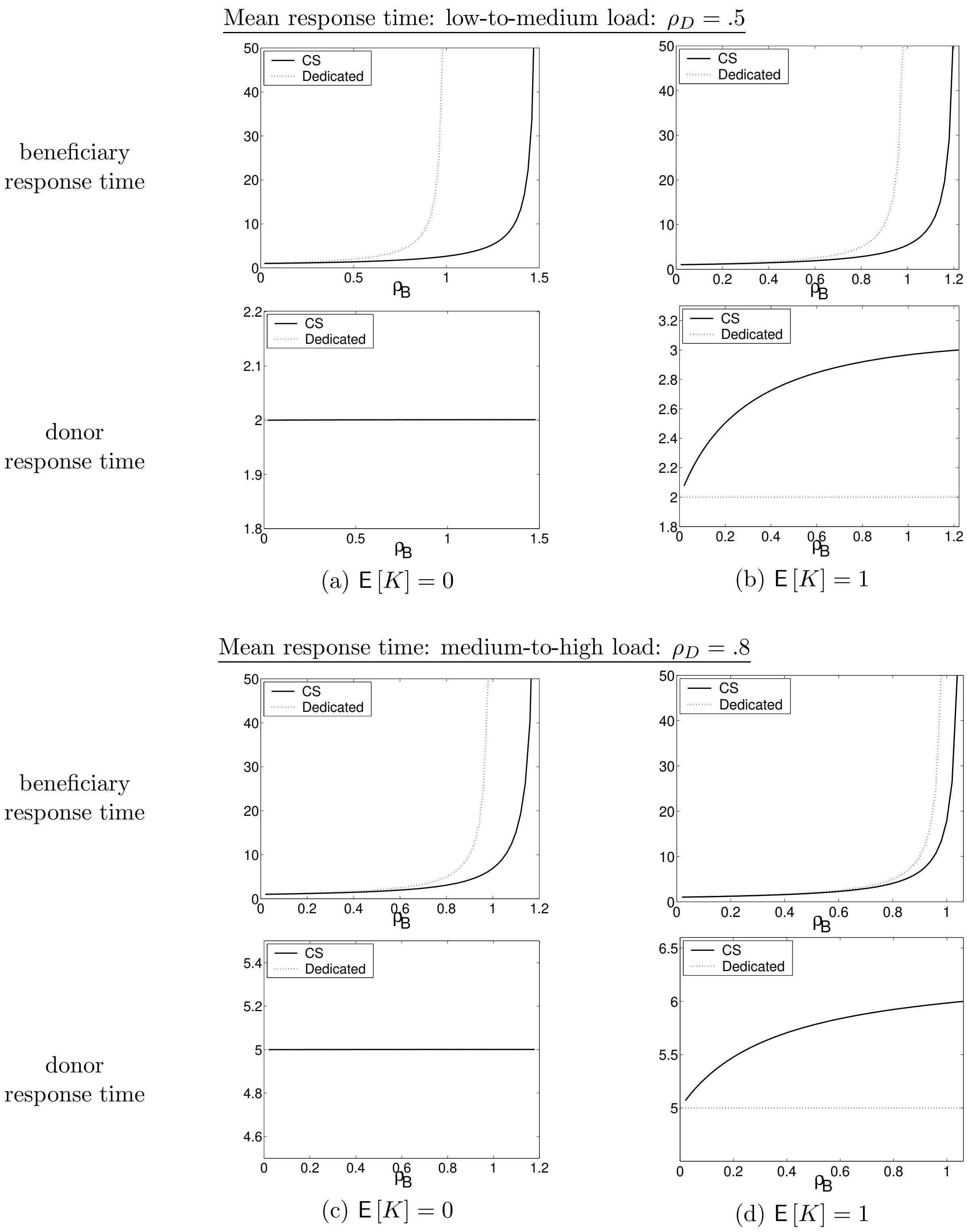


Figure 6.4: *The mean response time for beneficiary jobs and donor jobs as a function of ρ_B under cycle stealing and dedicated servers. In all figures X_B and X_D are exponential with mean 1; switching times ($K \equiv K_{sw} = K_{ba}$) are exponential with mean 0 or 1 as labeled.*

($\rho_D = 0.8$; bottom half). When $\rho_B \geq 1$ (and $\rho_D < 1$), cycle stealing can provide infinite gain to beneficiary jobs over dedicated servers, with comparatively little pain for the donor jobs. This is because the *stability region* for the beneficiary jobs under cycle stealing is much greater than under dedicated servers. While factors such as increased switching times and increased ρ_D do increase the mean response time of the beneficiary jobs, the gain is still infinite, and these factors are less important. We also see that the mean response time of donor jobs is bounded by the mean response time for an M/GI/1 queue with setup time K_{ba}. When $\rho_B \leq 0.5$, there is so little gain to the beneficiary jobs that cycle stealing with non-zero switching overhead does not pay. We therefore primarily focus the rest of the results section on the remaining case: $0.5 < \rho_B < 1$.

Benefit of cycle stealing: $.5 < \rho_B < 1.0$

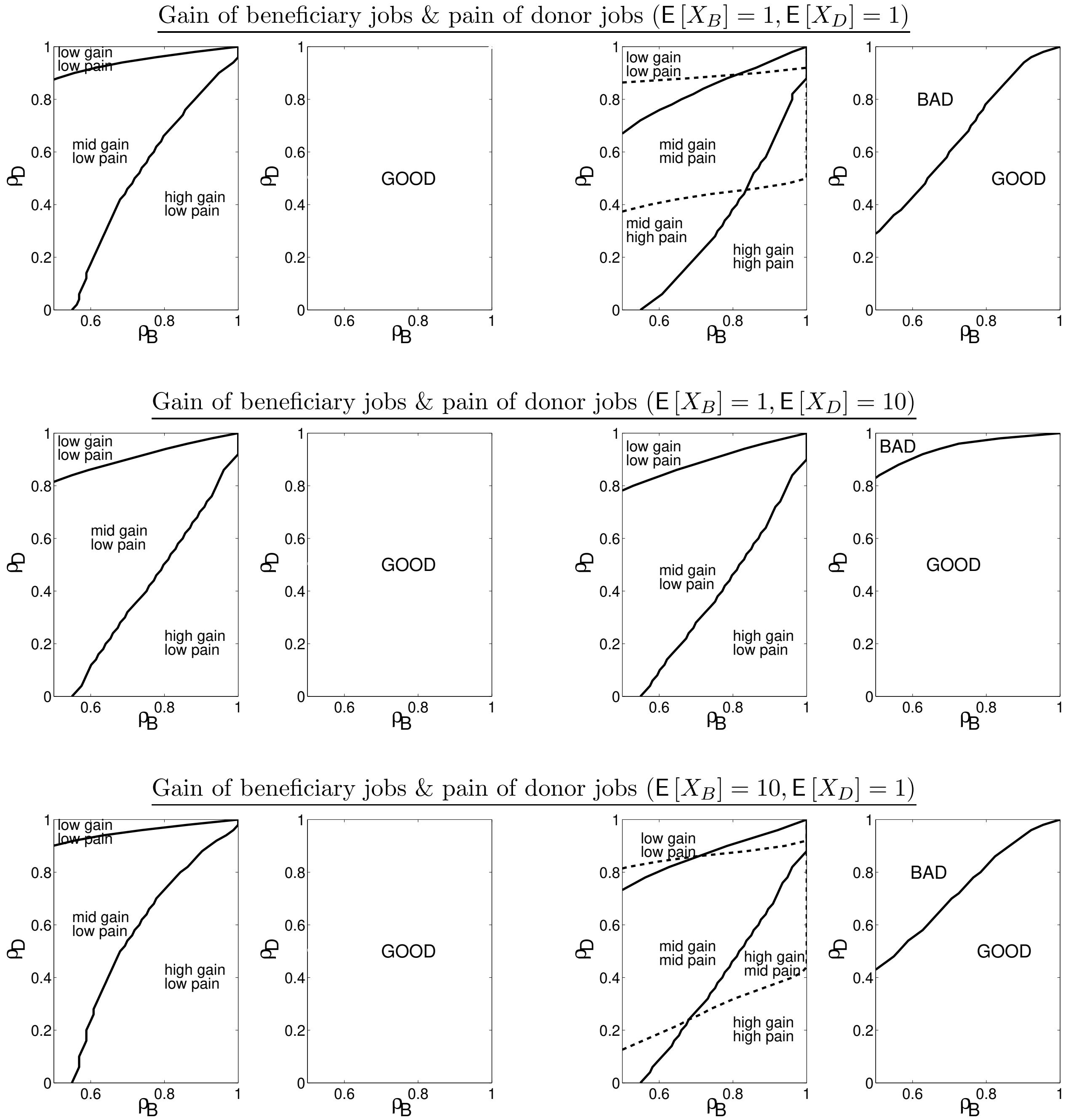
> *For $.5 < \rho_B < 1$, cycle stealing has regions of high gain and low pain and also regions where the reverse is true. These regions depend on job sizes, switching times, and loads.*

In Figure 6.5, we categorize performance into these gain/pain regions and also look at the *overall* mean response time (averaged over both beneficiary and donor jobs) to determine whether cycle stealing is "good" or "bad" overall. In general under higher ρ_B and lower ρ_D, cycle stealing is "good" overall, because the gain of the beneficiary jobs is so high in this region. We will find that when the switching times are short, cycle stealing leads to high gain and low pain. However long switching times can reverse this effect. More important than the absolute switching times are the switching times relative to the mean *donor* job size. We will find that the mean response time of the donor jobs is sensitive to the switching times, while surprisingly the mean response time of the beneficiary jobs is far less sensitive.

Figure 6.5 shows the gain of beneficiary jobs and the pain of donor jobs, where $.5 < \rho_B < 1$. The odd-numbered columns of Figure 6.5 divide the (ρ_B,ρ_D) space into regions of beneficiary gain and donor pain (low, mid, and high). We define *low gain* as a gain of 1.1 or less; *mid gain* as a gain of between 1.1 and 1.5; and *high gain* as a gain of over 1.5. Pain regions are defined similarly. While odd-numbered columns of Figure 6.5 consider the beneficiary and donor mean response time individually, the even-numbered columns of Figure 6.5 look at the overall mean response time and ask whether cycle stealing is "good" or "bad" with respect to the overall mean response time. The effect of mean job sizes is considered in Figure 6.5, where job sizes are exponential with mean 1 or 10.

Consider first row 1 in Figure 6.5. Under zero switching time (a)-(b), all regions are low pain regions (in fact zero pain), and higher ρ_B yields higher gain for the beneficiary jobs. Non-zero switching times (c)-(d) create only slightly reduced gain for the beneficiary jobs, but they create pain for the donor jobs. When ρ_D is very low, the pain appears high, but this is primarily due to the fact that "pain" is relative to the mean response time under dedicated servers, which is clearly low for small ρ_D. Although not shown, we have also investigated longer switching times, and these lead to the same trend of slightly less gain for beneficiary jobs and significantly more pain for donor jobs.

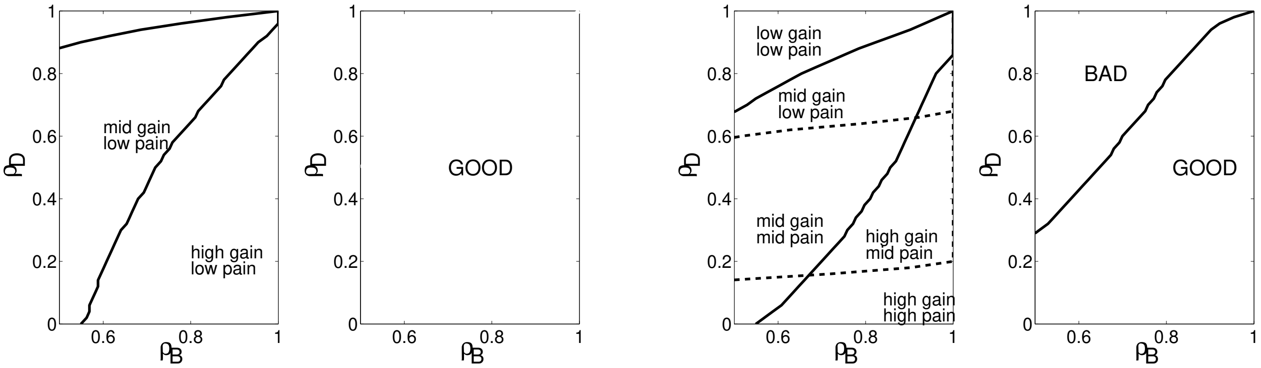
We now consider the effect of changes in job sizes. Row 2 of Figure 6.5 differs from row 1 only in X_D, which now has mean 10. The effect of this change is dramatic: now a switching time of 1 has almost no effect on either beneficiary jobs or donor jobs. This makes sense since the setup time experienced by donor jobs is now relatively small compared to their size. Row 3 in Figure 6.5 differs from row 1 only in X_B, which now has mean 10. Comparing these rows, we see the increase in $\mathsf{E}[X_B]$ has a surprisingly small effect on both beneficiary jobs and donor jobs, as compared with increasing $\mathsf{E}[X_D]$. This is because the donor still experiences the setup time, which has the same mean size



(a) $\mathsf{E}[K] = 0$ (b) $\mathsf{E}[K] = 0$ (c) $\mathsf{E}[K] = 1$ (d) $\mathsf{E}[K] = 1$

Figure 6.5: *The gain of beneficiary jobs and pain of donor jobs ((a) and (c)) and the effect of cycle stealing on the overall mean response time relative to dedicated servers ((b) and (d)). In (a) and (c), solid lines delineate high/mid/low gain regions, and dashed lines delineate high/mid/low pain regions. X_B and X_D have exponential distributions, and their means are as labeled. Switching times are exponential with mean 0 or 1 as labeled, where $K \equiv K_{sw} = K_{ba}$.*

Gain of beneficiary jobs & pain of donor jobs ($\mathsf{E}[X_B] = \mathsf{E}[X_D] = 1, C_D^2 = 8$)



(a) $\mathsf{E}[K] = 0$ (b) $\mathsf{E}[K] = 0$ (c) $\mathsf{E}[K] = 1$ (d) $\mathsf{E}[K] = 1$

Figure 6.6: *The gain of beneficiary jobs and pain of donor jobs ((a) and (c)), and the effect of cycle stealing on the overall mean response time relative to dedicated servers ((b) and (d)). In (a) and (c), solid lines delineate high/mid/low gain regions, and dashed lines delineate high/mid/low pain regions. X_B has an exponential distribution; X_D has a PH distribution with $C_D^2 = 8$. The means of X_B and X_D are as labeled. Switching times are exponential with mean 0 or 1 as labeled, where $K \equiv K_{sw} = K_{ba}$.*

as the donor job. We can conclude that cycle stealing is most effective when the switching time is small relative to the size of the *donor* jobs.

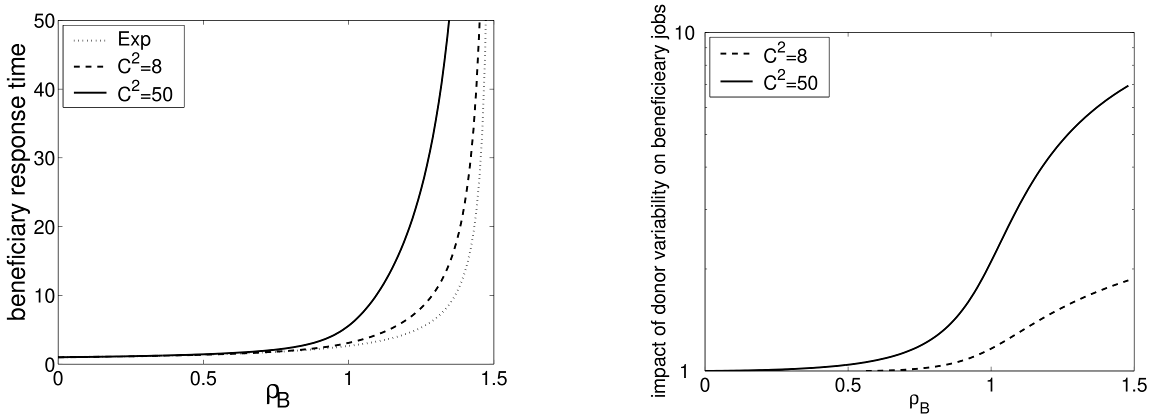
Focusing on columns 2 and 4 of Figure 6.5, which depict the effect on overall mean response time, we see that, for all rows, when the switching time is zero, cycle stealing always overwhelms the dedicated policy. When switching time is non-zero, cycle stealing is a good idea provided either ρ_B is high, or the switching time is short compared to X_D. These trends continue for longer switching times.

Effect of donor job size variability

> *For $.5 < \rho_B < 1$, variability of donor job sizes has very little effect on beneficiary mean response time.*

For $.5 < \rho_B < 1$, we find variability of donor job sizes has very little effect on beneficiary mean response time. This finding surprised us; we expected the beneficiary to gain far less from the bursty help of a donor with irregular (highly variable) job sizes.

It seems intuitive that when donor job sizes are made more variable, two things should happen. (i) The donor pain should drop. This is because the donor mean response times will be higher overall, and so the relative pain will appear diminished. (ii) The beneficiary gain should drop. This is because high variability in the donor job sizes implies high variability in the length of the donor busy periods, which implies that the donor's visits to the beneficiary queue will be more irregular. Sporadic help should be inferior to regular help for the beneficiary. Figure 6.6 shows that hypothesis (i) is in fact true, while hypothesis (ii) is surprisingly false, at least for $\rho_B < 1$. Comparing Figure 6.5 row 1 (X_D has low variability: $C_D^2 = 1$) with Figure 6.6 (X_D has high variability: $C_D^2 = 8$), we see that there is no discernible difference in beneficiary performance.



(a) (b)

Figure 6.7: *(a) The mean response time of the beneficiary jobs under different donor job size variability; (b) the "impact" of donor job size variability on the mean response time of the beneficiary jobs (the vertical axis is shown in the log scale). We fix ρ_D at 0.5, and ρ_B varies from 0 to the stability condition of 1.5. Switching time is zero; X_D and X_B have mean 1.*

To study this effect more closely, we next increase the variability in donor job sizes further. Figure 6.7(a) shows the mean response time of the beneficiary jobs when C_D^2 is 1, 8, or 50. Figure 6.7(b) shows the "impact" of C_D^2 on the mean response time of the beneficiary jobs, where the "impact" is defined to be the mean response time of the beneficiary jobs when $C_D^2 > 1$ (specifically, two cases where $C_D^2 = 8, 50$ are shown) relative to the mean response time of the beneficiary jobs when $C_D^2 = 1$. In Figure 6.7, switching times are set zero, ρ_D is fixed at 0.5, and ρ_B is varied from 0 to $\rho_B^{\max}$. As observed in Figure 6.6, the effect of the variability of X_D on the mean response time of X_B is small at $\rho_B < 1$, and negligible at $\rho_B < 0.75$, when $C_D^2 = 8$, although this effect increases slightly when C_D^2. When $\rho_B > 1$ the effect of variability in donor sizes can be significant. A critical factor seems to be whether the beneficiary queue is stable in isolation; when this is not the case, high variability in donor visits leads to prolonged intervals of instability, which inflates the mean response time.

This is the same phenomenon seen in other related models studied by Borst, Boxma and van Uitert [25] and by Borst, Boxma and Jelenkovic [26] as well as in Chapter 5. Borst, Boxma and van Uitert study the coupled processor model, where two processors each serve their own class of jobs, and if either is idle it may help the other, increasing the rate of the other processor. This help incurs no switching time and has a benefit even if only a single job is present (i.e. two processors can work on the single job). They find that if a processor has a load less than one, it is "insulated" from the heavy-tail of the other, as long periods without cooperation will not lead to large backlogs. This is not the case if the load is greater than one, as the queue now must rely on help to be stable. Borst, Boxma and Jelenkovic study the generalized processor sharing, where n classes of jobs can share a processor with arbitrary weights. Using probabilistic bounds, they show that different service rates can either insulate the performance of different classes from the others or not, again depending on whether the non-cooperative load is larger than one.

Effect of thresholds

The thresholds N_B^{th} and N_D^{th} have very different effects.

In this section, we study the effect of threshold settings on performance. We will see that increasing N_B^{th} helps alleviate donor pain given nonzero switching time, without appreciably diminishing beneficiary gain. Thus, the optimal value of N_B^{th} tends to be well above 1. By contrast, increasing N_D^{th} increases beneficiary gain substantially (by increasing their stability region), but also increases donor pain. Overall, the impact of changes in N_D^{th} is more dramatic than the impact of changes in N_B^{th}.

Figure 6.8 shows the mean response time for beneficiary jobs (rows 1 and 3) and the mean response time for donor jobs (rows 2 and 4) as a function of ρ_B for different threshold values. In the top half of the figure we study the effect of changing N_B^{th} from 1 to 10 as we hold N_D^{th} fixed at 1. In the bottom half of the figure we study the effect of changing N_D^{th} from 1 to 10 as we hold N_B^{th} fixed at 1. Throughout, X_B and X_D are exponential with mean 1, K_{sw} and K_{ba} are exponential with mean 0 or 1, and we fix $\rho_D = 0.5$.

As N_B^{th} is increased from 1 to 10, Figure 6.8 shows only slightly higher response times for the beneficiary jobs. Recall that increasing N_B^{th} does not change the beneficiary stability region, although the beneficiary queue is helped less frequently. In fact, under longer switching times, the effect of raising N_B^{th} on beneficiary mean response time is even more negligible, since the decreased frequency of helping beneficiary jobs is counteracted by the positive benefit of wasting less time on switching. We also see that increasing N_B^{th} creates less penalty for the donor, as the donor does not have to visit the beneficiary queue as frequently. Observe that when the switching times are nonzero, the donor mean response time is always bounded above by the mean response time for a corresponding M/GI/1 queue with setup time K_{ba}, and this bound is tight for all N_B^{th} values as ρ_B reaches its maximum, since the beneficiary queue always exceeds N_B^{th} in this case. We conclude that N_B^{th} has somewhat small impact; however higher values of N_B^{th} are more desirable for the system as a whole under longer switching time.

By contrast increasing N_D^{th} from 1 to 10 has dramatic effects. In general (assuming non-zero switching time) increasing N_D^{th} can drastically improve beneficiary response time. This result is not obvious, since increasing N_D^{th} allows the donor to spend more time at the beneficiary queue before leaving, but also means that when the donor leaves the beneficiary queue, the donor will be absent for a longer time (since more time is needed to empty the donor queue). Another positive effect of increasing N_D^{th} is less switching overall. In the end, it is the enlargement of the stability region due to higher N_D^{th} which substantially improves the beneficiary response time when the switching times are large and beneficiary load is high. When switching times are very short, increasing N_D^{th} only slightly worsens the mean response time for beneficiary jobs, as beneficiary jobs experience longer intervals between help. In all cases evaluated, increasing N_D^{th} results in much higher mean response times for donor jobs, since, for $N_D^{th} > 1$, the donor job arriving at an empty queue must wait for another $N_D^{th} - 1$ jobs to arrive before being served. We conclude that increasing N_D^{th} can have large impact, positive for the beneficiary jobs, but negative for the donor jobs. Thus setting N_D^{th} is much trickier than N_B^{th}.

Finally we seek to determine good values for the thresholds, N_B^{th} and N_D^{th}, as a function of the system parameters. Above, we have already observed some characteristics of N_B^{th}. (i) Increasing N_B^{th} leads to lower gain for the beneficiary jobs and lower pain for the donor jobs. (ii) Perhaps less obvious, the relative drop in gain for the beneficiary jobs is slight compared to the drop in pain for

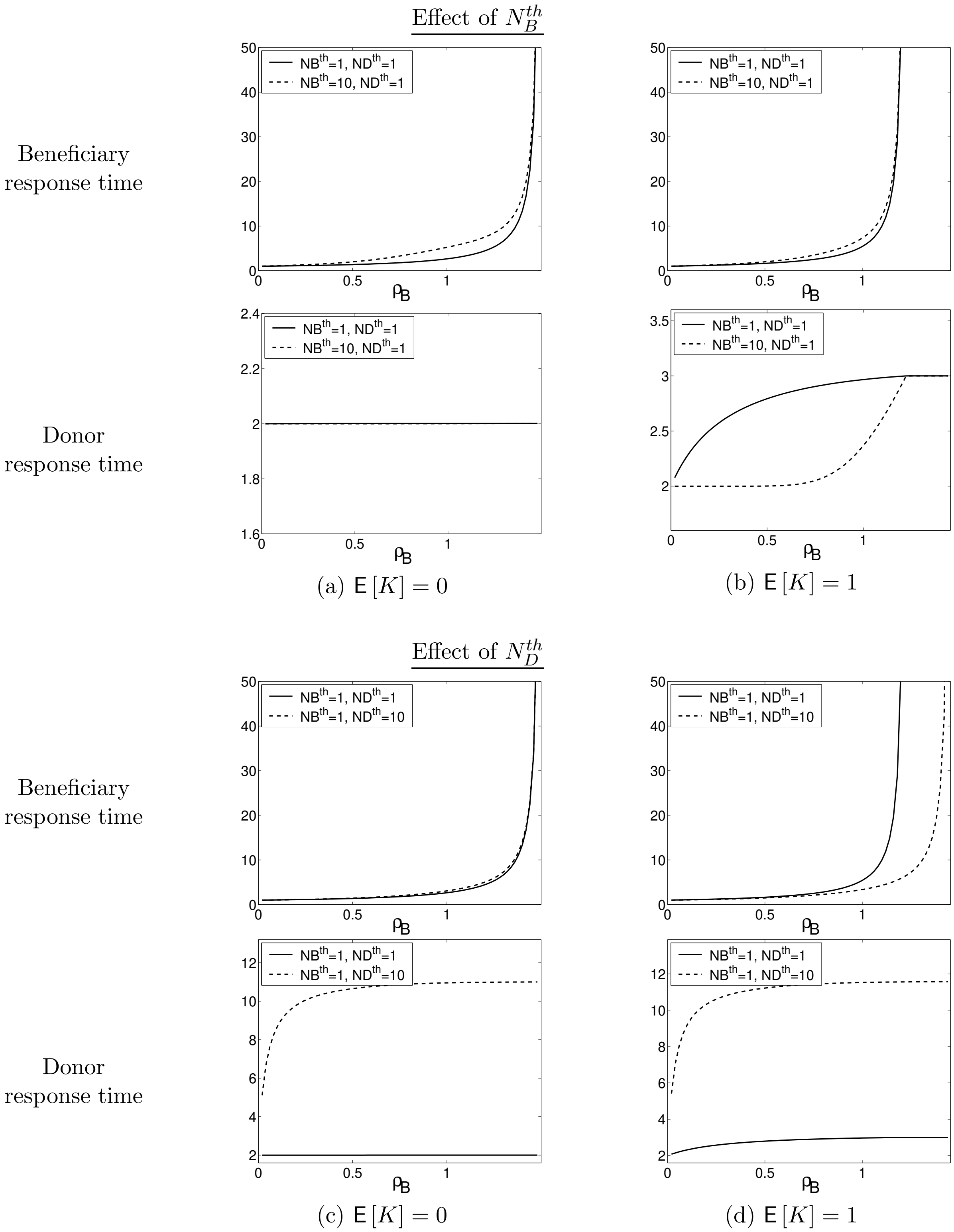
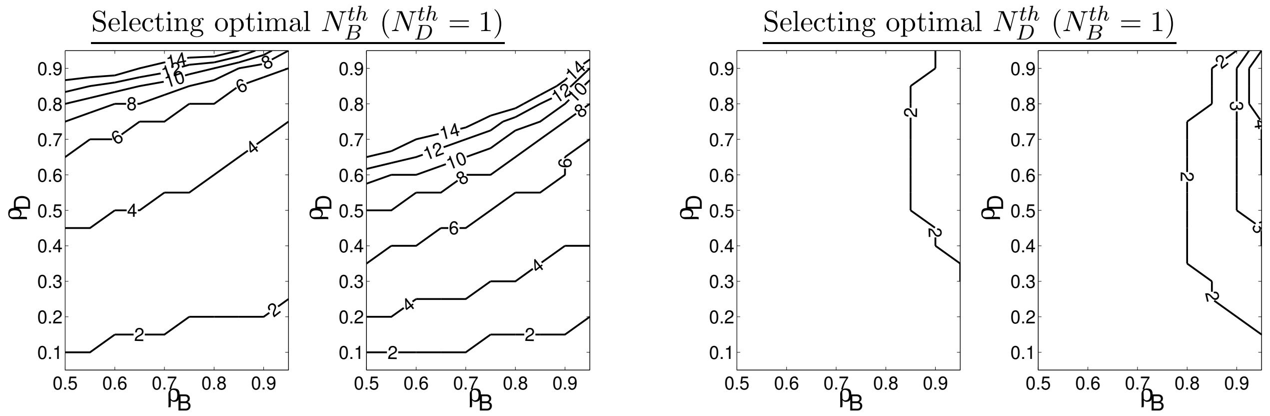


Figure 6.8: *The mean response time for beneficiary jobs and donor jobs as a function of ρ_B. Graphs show the case of (i) $N_B^{th} = N_D^{th} = 1$, (ii) $N_B^{th} = 10$ and $N_D^{th} = 1$, and (iii) $N_B^{th} = 1$ and $N_D^{th} = 10$. X_B and X_D are exponential with mean 1. Switching times are exponential with mean 0 or 1 as labeled, where $K \equiv K_{sw} = K_{ba}$. $\rho_D = 0.5$.*



(a) $\mathsf{E}[K] = 1$ (b) $\mathsf{E}[K] = 2$ (c) $\mathsf{E}[K] = 1$ (d) $\mathsf{E}[K]\,2$

Figure 6.9: *Optimal values of N_B^{th} and N_D^{th} with respect to overall mean response time, where X_B and X_D have an exponential distribution with mean 1.*

the donor jobs. This points towards choosing a higher value of N_B^{th}. Thus, if the switching time is zero, the optimal N_B^{th} is 1 (or 0), since there is never any pain for the donor jobs. Figure 6.9 (a) and (b) show optimal values of N_B^{th} for minimizing overall mean response time (over all jobs) as a function of ρ_B and ρ_D under various switching times when $N_D^{th} = 1$. The numbers on the contour curves show the optimal N_B^{th} at each load. For clarity we only show lines up to $N_B^{th} = 14$. The following additional characteristics of N_B^{th} are implied by the figure: (iii) the optimal N_B^{th} is an increasing function of ρ_D and a decreasing function of ρ_B; (iv) increasing the switching time increases the optimal N_B^{th}.

Figure 6.9 (c) and (d) show optimal values of N_D^{th} for minimizing overall mean response time when $N_B^{th} = 1$. First observe that (i) under low ρ_D or low ρ_B the optimal N_D^{th} is 1. When ρ_D is low and $N_D^{th} > 1$, the pain for donor jobs is so huge that the optimal N_D^{th} is always 1. When ρ_B is low, the beneficiary gains little from increasing N_D^{th}, while the donor can have nonnegligible pain, which increases with N_D^{th}; hence the optimal N_D^{th} is always 1. The following characteristics of N_D^{th} are also implied by the figure: (ii) the optimal N_D^{th} is not a monotonic function of ρ_D, but is an increasing function of ρ_B; (iii) increasing the switching time increases the optimal N_D^{th}. Note that although the range of the optimal values of N_D^{th} is smaller than N_B^{th} in Figure 6.9, Figure 6.8 tells us that the performance effect of changing N_D^{th} on the mean response time of both beneficiary jobs and donor jobs is more significant than changing N_B^{th}.

6.5 Concluding remarks

In this chapter, the performance of a threshold-based policy for reducing delay in cycle stealing is studied via dimensionality reduction (DR), as its analysis involves multidimensional Markov chains. DR allows us to evaluate the mean response time under a wide range of parameter settings, and this leads to many insights into the characteristics and performance of cycle stealing.

Our analysis shows that cycle stealing can provide unbounded benefit even under high switching costs. The unbounded benefit stems from the increased stability region under cycle staling, and we have provided the necessary and sufficient condition for the stability under the threshold-based policy for reducing delay in cycle stealing. Interestingly, we find that the beneficiary side threshold, N_B^{th}, does not affect the stability region, but increasing the donor side threshold, N_D^{th}, increases the stability region. We also find that the impact of changes in N_D^{th} on *mean response time* is more dramatic than the impact of changes in N_B^{th}

Switching times (both K_{sw} and K_{ba}) also affect the stability region (of the beneficiary jobs), and they also have a significant impact on the mean response time when the beneficiary is overloaded without cycle stealing ($\rho_B \geq 1$). However, when $\rho_B < 1$, we find that whereas the mean response time of the donor jobs is still sensitive to the switching times, the mean response time of the beneficiary jobs is far less sensitive.

An advantage of the analysis via DR is that the job sizes can be modeled as a PH distribution. This allows us to study the impact of job size variability on performance. In particular, we study the impact of the variability of the donor job sizes on the mean response time of the beneficiary jobs. We find that the impact is surprisingly small when $\rho_B < 1$. This is in parallel to our findings in Chapter 5, where we find that the variability of the long job sizes has a surprisingly small impact on the mean response time of the short jobs under `SBCS-ID` (size-based task assignment with cycle stealing under immediate dispatching).

In this chapter, we limit our discussion to a particular multiserver system, but the analysis via DR can be extended to other multiserver systems in various ways. For example, we do not need to require that the donor switches back immediately when N_D^{th} is reached; we can allow the donor to first complete the beneficiary job in progress. Completing the beneficiary job obviates the need for checkpointing the job; however it sometimes reduces system performance, particularly when beneficiary jobs have higher variability than donor jobs. We found that this change has almost no effect on performance, even under long switching times (see [150]). It is also easy to generalize our analysis to the case where the beneficiary requires a "switching time" before it can resume a job that the donor started, and to the case where the donor must complete the beneficiary job in progress before it switches back.

Another interesting case is where servers function as both beneficiaries and donors (two way cycle stealing), as in [180]. Since the two way cycle stealing does not appear to be modeled as a GFB process, DR does not allow us to analyze this model. In [180], the state of each processor is assumed to be stochastically independent and identical to the state of the other processors; i.e. a multidimensional Markov chain is decomposed into 1D Markov chains. In theory, the analysis of the two way cycle stealing can be reduced to a boundary value problem (see Section 3.3), but this leads to a complex expression whose evaluation often experiences numerical instability. It would be interesting to pursue how computational approaches such as DR can be applied to an analysis of the two way cycle stealing.

Chapter 7

Designing robust resource allocation policies

How to design robust resource allocation policies?

In this chapter, we study robustness of resource allocation policies. DR allows us to evaluate the mean response time of a broad class of threshold-based policies under a wide range of load conditions, and this leads to lessons useful in designing robust resource allocation policies.

7.1 Introduction

A common problem in multiserver systems is deciding how to allocate resources (e.g. operators, CPU time, and bandwidth) among jobs so as to maximize system performance, e.g. with respect to mean response time or throughput. Since good parameter settings typically depend on environmental conditions such as system loads, an allocation policy that is optimal in one environment may provide poor performance when the environment changes, or when the estimation of the environment is wrong. In other words, the policy may not be *robust*. In this chapter, we design several multiserver allocation policies, quantifying their performance with respect to mean response time and robustness, and providing insights into which types of policies perform well in different operating environments.

7.1.1 Static and dynamic robustness

We consider a multiserver model that consists of two servers and two queues (Beneficiary-Donor model), as shown in Figure 7.1. Jobs arrive at queue 1 and queue 2 according to Poisson processes (or Markov modulated Poisson processes, as defined in Section 3.2) with average arrival rates λ_1 and λ_2, respectively. Jobs have exponentially distributed service demands; however, the running time of a job may also depend on the affinity between the particular server and the particular job/queue. Hence, we assume that server 1 (beneficiary server) processes jobs in queue 1 (type 1 jobs) with rate μ_1 (jobs/sec), while server 2 (donor server) can process type 1 jobs with rate μ_{12}, and can process jobs in queue 2 (type 2 jobs) with rate μ_2. We define $\rho_1 = \lambda_1/\mu_1$, $\rho_2 = \lambda_2/\mu_2$, and $\hat{\rho}_1 = \lambda_1/(\mu_1 + \mu_{12}(1-\rho_2))$. Note that $\rho_2 < 1$ and $\hat{\rho}_1 < 1$ are necessary for the queues to be stable under any allocation policy, since the maximum rate at which type 1 jobs can be processed is μ_1, from server 1, plus $\mu_{12}(1-\rho_2)$, from server 2.

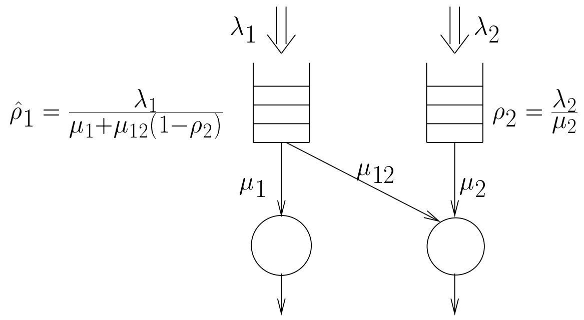


Figure 7.1: *The Beneficiary-Donor model.*

The Beneficiary-Donor model has a wide range of applications in service industries such as call centers and repair facilities. For example, in call centers, the donor server may be a bilingual operator, and the beneficiary server may be a monolingual operator [58, 105, 176, 185, 186], or the donor server may be a cross-trained or experienced operator who can handle all types of calls, and the beneficiary server may be a specialized operator who is only trained to handle a specific type of calls [58, 176]. (See also Chapter 8) In a repair facility, the donor server may be a technician who can handle jobs of any difficulty, and the beneficiary server may be a technician with limited expertise [61].

We design and evaluate allocation policies for the Beneficiary-Donor model with respect to three objectives. First, as is standard in the literature, we seek to minimize the overall weighted mean response time, $c_1 p_1 \mathsf{E}[R_1] + c_2 p_2 \mathsf{E}[R_2]$, where c_i is the weight (importance) of type i jobs, $p_i = \lambda_i/(\lambda_1 + \lambda_2)$ is the fraction of type i jobs, and $\mathsf{E}[R_i]$ is the mean response time of type i jobs, for $i = 1, 2$. Here, response time refers to the total time a job spends in the system. Below, we refer to overall weighted mean response time simply as *mean response time*.

In addition to mean response time, we consider an additional metric, *robustness*, introducing two types of robustness: *static robustness* and *dynamic robustness*. Static robustness measures robustness against *misestimation* of load; to evaluate static robustness, we analyze the mean response time of allocation polices for a range of loads to see how a policy tuned for *one* load behaves under different loads. Dynamic robustness measures the robustness against *fluctuations* in load; to evaluate dynamic robustness, we analyze the mean response time of allocation policies under Markov modulated Poisson processes, where arrivals follow a Poisson process at each moment, but the arrival rate changes over time.

7.1.2 State of the art in resource allocation policies for the Beneficiary-Donor model

There has been a large amount of prior work on the Beneficiary-Donor model, the majority of which has focused on proving the optimality of allocation policies in limiting or special cases. With respect to calculating mean response times, only coarse approximations exist for most of the allocation policies in our model. By contrast, dimensionality reduction (DR) developed in Chapter 3 allows us to analyze these and other allocation policies with respect to mean response time, as well as static and dynamic robustness.

One common allocation policy is the $c\mu$ rule [45], which biases in favor of jobs with high c (high importance) and high μ (small expected size). Applying the cμ rule to our setting, server 2 serves type 1 jobs (rather than type 2 jobs) if $c_1\mu_{12} > c_2\mu_2$, or queue 2 is empty. The $c\mu$ rule is provably optimal when server 1 does not exist [45] or in the fluid limit [125, 182]. However, Harrison as well as Squillante et. al. have shown that $c\mu$ rule may lead to instability even if $\hat{\rho}_1 < 1$ and $\rho_2 < 1$ [71, 181] More recently, Mandelbaum and Stolyar as well as van Mieghem have introduced and analyzed the *generalized $c\mu$ rule*. However, in our model, the generalized $c\mu$ rule reduces to the $c\mu$ rule and hence has the same stability issues [121, 126].

In light of this instability, Squillante et. al. and Williams have independently proposed a threshold-based policy that, under the right choice of threshold value, improves upon the $c\mu$ rule with respect to mean response time, guaranteeing stability whenever $\hat{\rho}_1 < 1$ and $\rho_2 < 1$ [181, 205]. We refer to this threshold-based policy as the T1 policy, since it places a threshold value, T_1, on queue 1, so that server 2 processes type 1 jobs only when there are at least T_1 jobs of type 1, or if queue 2 is empty. The rest of the time, server 2 works on type 2 jobs. This "reserves" a certain amount of work for server 1, preventing server 1 from being under-utilized and server 2 from becoming overloaded, as can happen under the $c\mu$ rule. Bell an Williams prove the optimality of the T1 policy for a model closely related to ours in the heavy traffic limit [17].

However, studies by Ahn et. al. and Meyn suggest that the T1 policy is *not* optimal in general [4, 125]. Ahn et. al. characterize the optimal policy with respect to minimizing the total holding cost until all the jobs in the system at time zero leave the system, assuming that there are no arrivals after time zero. They find that the optimal policy is in general a "flexible" T1 policy that allows a continuum of T1 thresholds, $\{T_1^{(i)}\}$, where threshold $T_1^{(i)}$ is used when the length of queue 2 is i. Meyn obtains, via a numerical approach, the optimal allocation policy in the Beneficiary-Donor model when both queues have finite buffers. Although not proven, the optimal policy appears to be a "flexible" T1 policy in the Beneficiary-Donor model.

All of the work above investigates a class of allocation policies that are optimal in limiting or special cases. In contrast, there has been little work on the *analysis and evaluation* of the mean response time of general allocation policies in our model, and no work evaluating robustness. Complicating this problem is the fact that the analysis involves 2D Markov chains; i.e., the Markov chains need to track both the number of type 1 jobs and the number of type 2 jobs. Hence, only approximate analyses exist for most of allocation policies in our model. For example, Squillante et. al. derive a coarse approximation for the mean response time of the T1 policy under Poisson arrivals based on vacation models [181]. The mean response time of other simple allocation policies (in more general models) such as (idle) cycle stealing, where server 2 works on type 1 jobs when queue 2 is empty, have also been analyzed (with approximation) either by matrix analytic methods with state space truncation [61, 185, 186] or by approximate solutions of a 2D Markov chain via state space decomposition [176].

7.1.3 Summary of results

A wide range of threshold-based allocation policies for the Beneficiary-Donor model can be analyzed via DR by modeling the system as GFB processes, as discussed in Section 3.4. This allows us to analytically study static and dynamic robustness as well as mean response time of these allocation policies. In particular, DR allows us to evaluate the mean response time under the T1 policy, proposed by Squillante et. al. and Williams [181, 205], for which only coarse approximations exist. Our analysis will lead to the following conclusions.

- We first consider single-threshold policies, and study which queue should control the single threshold. We find that it is better to determine when help is provided based on the beneficiary queue length (the T1 policy) rather than the donor queue length for all the cases that we study (not just in the heavy traffic limit, as proved in [17]).

- Surprisingly, we find that the use of multiple thresholds improves upon the T1 policy only by a small amount with respect to mean response time, although the optimal allocation policy appears to be a "flexible" T1 policy, which uses an infinite number of thresholds [4, 125].
- We introduce two types of robustness: static robustness and dynamic robustness, and analytically study a wide range of threshold-based allocation policies with respect to both types of robustness via DR. Surprisingly, we will see that policies that excel in static robustness do not necessarily excel in dynamic robustness.
- Specifically, we find that an allocation policy with multiple thresholds can experience significant benefit over allocation policies with a single threshold with respect to *static* robustness. Illustrating this, we introduce the adaptive dual threshold (ADT) policy, which places two thresholds on queue 1, and show this has significant advantage over single-threshold allocation policies with respect to static robustness. The ADT policy operates like a T1 policy, but the threshold value is self-adapted to the load.
- In contrast to this, we find that multiple thresholds surprisingly offer only small advantage over a single threshold with respect to *dynamic* robustness.

The rest of this chapter is organized as follows. Section 7.2 discusses single-threshold allocation policies, including the T1 policy. Section 7.3 discusses multiple threshold allocation policies, including the ADT policy. In Sections 7.2-7.3, we evaluate the policies with respect to mean response time and static robustness. In Section 7.4, we shift our interest to dynamic robustness.

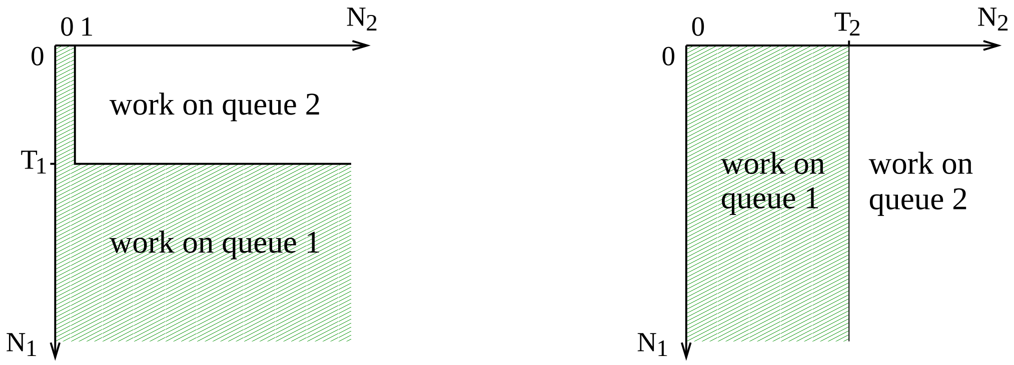
7.2 Static robustness and mean response time of single-threshold allocation policies

In this section, we analytically study the mean response time and static robustness of two single threshold allocation policies: the T1 policy and the T2 policy. The T1 policy places a threshold, T_1, on queue 1, whereby server 2 serves type 1 jobs whenever the length of queue 1 is at least T_1. Thus, under T1, the *beneficiary* queue (queue 1) has control. The T2 policy places a threshold, T_2, on queue 2, whereby server 2 serves type 1 jobs whenever the length of queue 2 is *below* T_2. In the T2 policy, the donor queue (queue 2) has control. Our analysis will show that the T1 policy is superior to the T2 policy with respect to minimizing mean response time, but that the T2 policy is superior with respect to static robustness.

7.2.1 Single-threshold allocation policies: T1 and T2

The T1 and T2 policies are formally defined as follows:

Definition 16 *Let N_1 (respectively, N_2) denote the number of jobs at queue 1 (respectively, queue 2). The T1 policy with parameter T_1 is characterized by the following set of rules, all of which are enforced preemptively (preemptive-resume):*



(a) T1 policy (b) T2 policy

Figure 7.2: *Figure shows whether server 2 works on jobs from queue 1 or queue 2 as a function of N_1 and N_2 under (a) the T1 policy with parameter T_1 and (b) the T2 policy with parameter T_2.*

- *Server 1 serves only its own jobs.*
- *Server 2 serves jobs from queue 1 if either (i) $N_1 \geq T_1$ or (ii) $N_2 = 0$ & $N_1 \geq 2$. Otherwise, server 2 serves jobs from queue 2.*

To achieve maximal efficiency, we assume the following exceptions. When $N_1 = 1$ and $N_2 = 0$, the job is processed by server 2 if and only if $\mu_1 < \mu_{12}$. Also, when $T_1 = 1$ and $N_1 = 1$, the job in queue 1 is processed by server 2 if and only if $\mu_1 < \mu_{12}$ regardless of the number of type 2 jobs.

Definition 17 *The T2 policy with parameter T_2 is characterized by the following set of rules, all of which are enforced preemptively (preemptive-resume):*

- *Server 1 serves only its own jobs.*
- *Server 2 serves jobs from queue 1 if $N_2 < T_2$. Otherwise server 2 serves jobs from queue 2.*

When $N_1 = 1$ and $N_2 = 0$, we allow the same exception as in the T1 policy.

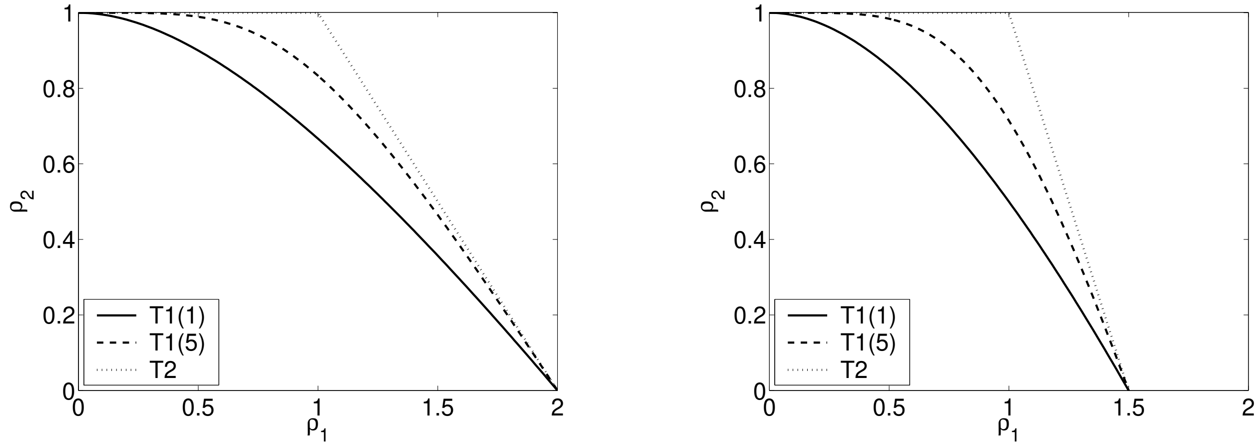
Below, the T1 policy with parameter t_1 is also denoted by the T1(t_1) policy, and the T2 policy with parameter t_2 is also denoted by the T2(t_2) policy.

Figure 7.2 shows the jobs processed by server 2 as a function of N_1 and N_2 (a) under the T1 policy and (b) under the T2 policy. Note that the T1(1) policy is the cμ rule when $c_1\mu_{12} > c_2\mu_2$, and the T1(∞) policy is the cμ rule when $c_1\mu_{12} < c_2\mu_2$; thus the $c\mu$ rule falls within the broader class of T1 policies.

7.2.2 Stability under single-threshold allocation policies

We first consider the stability conditions for the T1 and T2 policies. In the T1 policy, higher T_1 values yield the larger stability regions, and in the limit as $T_1 \to \infty$, the queues under the T1 policy are stable as long as $\hat{\rho}_1 < 1$ and $\rho_2 < 1$. In contrast, the T2 policy guarantees stability whenever $\hat{\rho}_1 < 1$ and $\rho_2 < 1$ for *any* finite T_2. The T2 policy clearly dominates the T1 policy in this respect. The stability conditions under the T1 and T2 policies are illustrated in Figure 7.3.

More formally, the necessary and sufficient conditions for the stability under the T1 and T2 policies are given by the following theorems.



(a) $\mu_{12} = \mu_1$ (b) $\mu_{12} = \frac{\mu_1}{2}$

Figure 7.3: *Stability conditions for the T1 policies, T1(1) and T1(5), and for the T2 policy. Upper bounds on ρ_2 are plotted as functions of ρ_1.*

Theorem 14 *Under the T1 policy with parameter $T_1 < \infty$, queue 1 is stable if and only if $\lambda_1 < \mu_1 + \mu_{12}$. Stability of queue 2 is given by the following conditions:*

- *For $1 < T_1 < \infty$, queue 2 is stable if and only if*

$$\rho_2 \quad < \quad \begin{cases} \frac{1-\rho_1^{T_1}}{1-\rho_1^{T_1}+\frac{(1-\rho_1)\rho_1^{T_1}}{1-\rho_1+\mu_{12}/\mu_1}} & \text{if } \rho_1 \neq 1, \\ \\ \frac{T_1}{T_1+\lambda_1/\mu_{12}} & \text{if } \rho_1 = 1. \end{cases} \tag{7.1}$$

- *For $T_1 = 1$, if $\mu_1 \geq \mu_{12}$, queue 2 is stable if and only if equation (7.1) holds with $T_1 = 2$.*

- *For $T_1 = 1$, if $\mu_1 < \mu_{12}$, queue 2 is stable if and only if*

$$\rho_2 < \frac{1}{1 + \frac{\rho_1+\lambda_1/\mu_{12}}{1-\rho_1+\mu_{12}/\mu_1}}.$$

Proof: We prove only the case when $T_1 > 1$ and $\rho_1 \neq 1$. The case when $T_1 = 1$ or $\rho_1 = 1$ can be proved in a similar way. Let $N = (N_1, N_2)$ be the joint process of the number of jobs in queue 1 and queue 2, respectively. The expected length of a "busy period," during which $N_1 \geq T_1$, is finite if and only if $\lambda_1 < \mu_1 + \mu_{12}$. This proves the stability condition for queue 1.

Based on the strong law of large numbers, the necessary and sufficient condition for stability of queue 2 is $\rho_2 < F$, where F denotes the time average fraction of time that server 2 processes jobs from queue 2 given $N_2 > 0$. Below, we derive F. Let $\widetilde{N} = (\widetilde{N}_1, \widetilde{N}_2)$ be a process in which $\widetilde{N}$ behaves the same as N except that it has no transition from $\widetilde{N}_2 = 1$ to $\widetilde{N}_2 = 0$. Consider a semi-Markov process of $\widetilde{N}_1$, where the state space is (0,1,2,...,$T_1 - 1$,T_1^+). The state n denotes there are n jobs in queue 1 for $n = 0, 1, ..., T_1 - 1$, and the state T_1^+ denotes there are *at least* T_1 jobs in queue 1. The expected sojourn time is $1/\lambda_1$ for state 0, $1/(\lambda_1 + \mu_1)$ for states $n = 1, ..., T_1 - 1$, and $b = \frac{1/(\mu_1+\mu_{12})}{1-\lambda_1/(\mu_1+\mu_{12})}$ for state T_1^+, where b is the mean duration of the busy period in an M/M/1 queue with arrival rate λ_1 and service rate $\mu_1 + \mu_{12}$. The limiting probabilities for the corresponding *embedded* discrete time Markov chain are $\pi_n = (1 + \rho_1)\rho_1^{n-1}\pi_0$ for $n = 1, ..., T_1 - 1$ and $\pi_{T_1^+} = \rho_1^{T_1}\pi_0$, where

$$\pi_0 = \frac{1 - \rho_1}{(1 + \rho_1^{T_1-1})(1 - \rho_1) + (1 + \rho_1)(1 - \rho_1^{T_1-1})}.$$

As server 2 can work on queue 2 if and only if $\widetilde{N}_1 < T_1$, the fraction of time that server 2 can work on queue 2 is

$$F = \frac{\pi_0/\lambda_1 + (1 - \pi_0 - \pi_{T_1^+})/(\lambda_1 + \mu_1)}{\pi_0/\lambda_1 + (1 - \pi_0 - \pi_{T_1^+})/(\lambda_1 + \mu_1) + b\pi_{T_1^+}} = \frac{1 - \rho_1^{T_1}}{1 - \rho_1^{T_1} + \frac{(1-\rho_1)\rho_1^{T_1}}{1-\rho_1+\mu_{12}/\mu_1}}.$$

■

The following corollary is an immediate consequence of Theorem 14.

Corollary 4 *Under the T1 policy, the stability region increases with T_1 (i.e., the right hand side of equation (7.1) is an increasing function of T_1).*

Theorem 15 *Under the T2 policy with $T_2 < \infty$, queue 1 is stable if and only if $\hat{\rho}_1 < 1$, and queue 2 is stable if and only if $\rho_2 < 1$.*

Theorem 15 can be proved in a similar way as Theorem 14.

7.2.3 Mean response time of single-threshold allocation policies

In this section, we characterize the performance of the T1 and T2 policies. In particular, we will find that

- The characteristics of the T1 policy are quite different depending on whether queue 2 prefers type 1 or type 2 ($c_1\mu_{12} \le c_2\mu_2$ or $c_1\mu_{12} > c_2\mu_2$). Specifically, the optimal T_1 threshold is typically finite when $c_1\mu_{12} > c_2\mu_2$, and typically infinite when $c_1\mu_{12} \le c_2\mu_2$, where the optimality is with respect to minimizing mean response time.
- The optimal T_2 threshold for the T2 policy is typically small.
- The mean response time under the T1 policy is at least as low as that under the T2 policy for all the cases that we study.

Case 1: $c_1\mu_{12} \le c_2\mu_2$

We first consider the case where $c_1\mu_{12} \le c_2\mu_2$, where server 2 "prefers" to run its own jobs in a $c\mu$ sense. In this case, we will see that the T1(∞) policy typically minimizes mean response time in the class of T1 policies, and the T2(1) policy typically minimizes mean response time in the class of T2 policies. Note that the T1(∞) policy and the T2(1) policy are identical, and they also coincide with the policy of following the $c\mu$ rule when $c_1\mu_{12} \le c_2\mu_2$. Note also that if $T_1 = \infty$ minimizes the mean response time of the T1 policy, $T_1 = \infty$ is the optimal choice with respect to both mean response time and stability region, since $T_1 = \infty$ maximizes the stability region of the T1 policy (see Corollary 4). Likewise, if $T_2 = 1$ minimizes the mean response time of the T2 policy, $T_2 = 1$ is the optimal choice with respect to both mean response time and stability region, since the stability region of the T2 policy is independent of its parameter T_2 (see Theorem 15).

In fact, when $c_1 = c_2$ (and $c_1\mu_{12} \le c_2\mu_2$), the following theorems hold (the theorem *may* extend to the case of general c_1 and c_2 with $c_1\mu_{12} \le c_2\mu_2$, but the general case is not proved).

Theorem 16 *If $c_1 = c_2$ and $c_1\mu_{12} \le c_2\mu_2$, the mean response time of the T1 policy is minimized at $T_1 = \infty$.*

Proof: Due to Little's law, it is sufficient to prove that the number of jobs completed under the T1(∞) policy is stochastically larger than those completed under the T1 policy with $T_1 < \infty$ at any moment. Let $N^{\mathrm{inf}}(t) = (N_1^{\mathrm{inf}}(t), N_2^{\mathrm{inf}}(t))$ be the joint process of the number of jobs in queue 1 and queue 2, respectively, at time t when $T_1 = \infty$. Let $N^{\mathrm{fin}}(t) = (N_1^{\mathrm{fin}}(t), N_2^{\mathrm{fin}}(t))$ be defined analogously for $T_1 < \infty$. With $T_1 = \infty$, server 2 processes type 2 jobs as long as there are type 2 jobs, and thus $N_1^{\mathrm{inf}}(t)$ is stochastically larger than $N_1^{\mathrm{fin}}(t)$ for all t. Consequently, the number of jobs completed by sever 1 is stochastically smaller when $T_1 < \infty$ than when $T_1 = \infty$ at any moment, since server 1 is work-conserving.

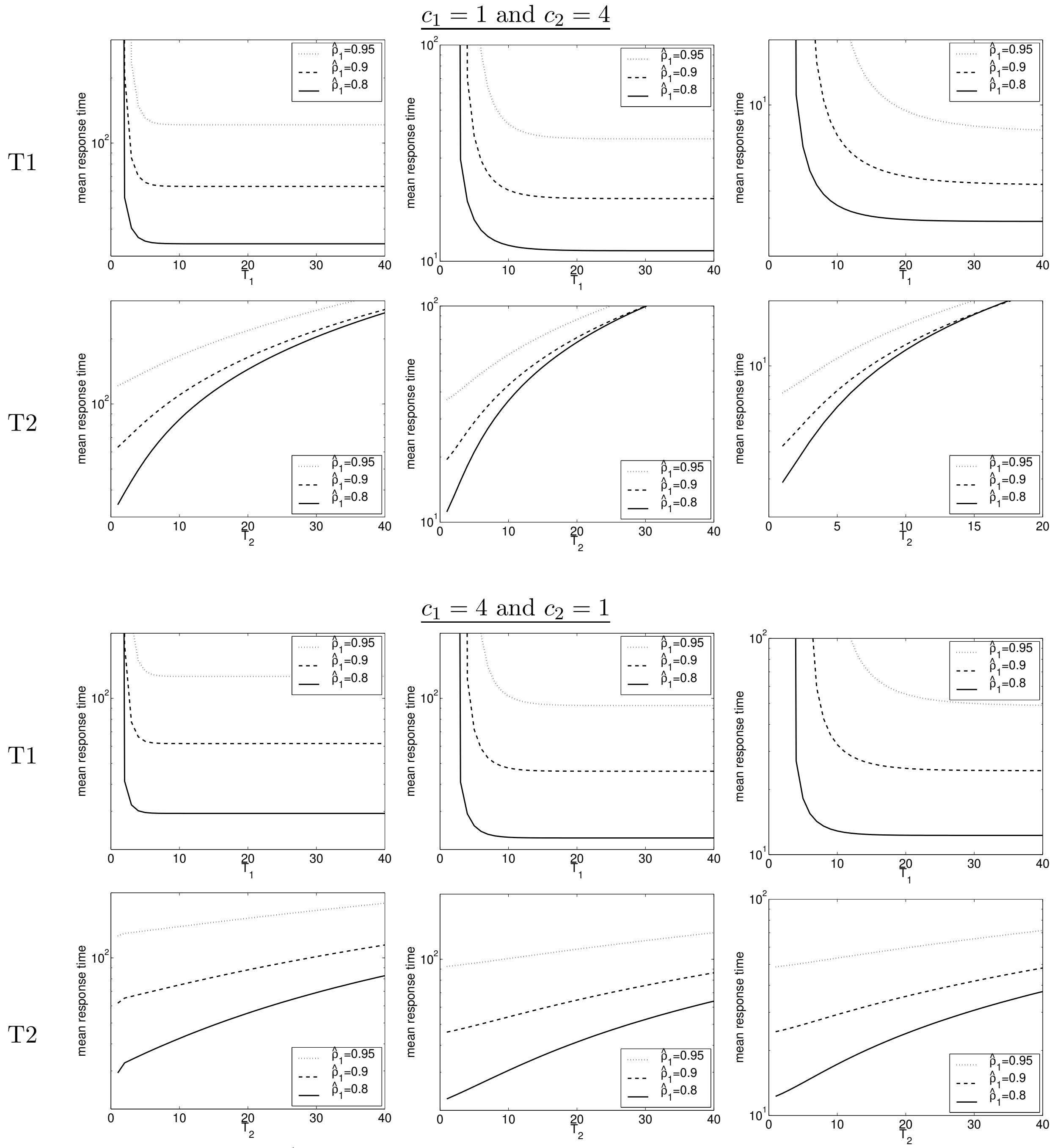
As long as server 2 is busy, the number of jobs completed by sever 2 is stochastically smaller when $T_1 < \infty$ than when $T_1 = \infty$, since $\mu_{12} \le \mu_2$. Also, using coupling argument, we can show that the system with $T_1 = \infty$ has server 2 go idle (both queues empty) earlier (stochastically) than the $T_1 < \infty$ system. Thus, when server 2 is idle the number of jobs completed by the $T_1 = \infty$ system is stochastically larger. Thus, the number of jobs completed (either by server 1 or by server 2) under the T1(∞) policy is stochastically larger than that completed under the T1 policy with $T_1 < \infty$. ■

Theorem 17 *If $c_1 = c_2$ and $c_1\mu_{12} \le c_2\mu_2$, the mean response time of the T2 policy is minimized at $T_2 = 1$.*

Theorem 17 can be proved in the same way as Theorem 16: in the proof of Theorem 17, the T1 policy with parameter $T_1 = \infty$ is replaced by the T2 policy with parameter $T_2 = 1$, and the T1 policy with parameter $T_1 < \infty$ is replaced by the T2 policy with parameter $T_2 > 1$.

Further, even when $c_1 \ne c_2$ (and $c_1\mu_{12} \le c_2\mu_2$), our analysis of the T1 and T2 policies via DR shows that the T1(∞) policy minimizes mean response time in the class of T1 policies, and the T2(1) policy minimizes mean response time in the class of T2 policies, for all the cases we have considered. Figure 7.4 shows a subset of such numerical analyses, where $c_1\mu_{12}$ and $c_2\mu_2$ are chosen to be the critical case: $c_1\mu_{12} = c_2\mu_2$. In the top half, $c_1 = 1$ and $c_2 = 4$, and in the bottom half, $c_1 = 4$ and $c_2 = 1$. Different columns correspond to different μ_1's. In each plot, mean response time is evaluated at three loads, $\hat{\rho}_1 = 0.8, 0.9, 0.95$, by changing λ_1. Although figures are not shown, when $c_1\mu_{12} < c_2\mu_2$, the mean response time under non-optimized T1 and T2 policies becomes much higher than the optimized T1 and T2 policies (T1(∞) and T2(1)), as compared to the case of $c_1\mu_{12} = c_2\mu_2$. This makes intuitive sense, since non-optimized policies have a bias toward jobs with smaller $c\mu$ values (less important and/or larger jobs).

Mean response time of T1 and T2: $c_1\mu_{12} = c_2\mu_2 = 1$ and $c_1 \neq c_2$



(a) $c_1\mu_1 = \frac{1}{4}$ (b) $c_1\mu_1 = 1$ (c) $c_1\mu_1 = 4$

Figure 7.4: *The mean response time under the T1 policy as a function of T_1 (rows 1 and 3) and the mean response time under the T2 policy as a function of T_2 (rows 2 and 4) when $c_1\mu_{12} = c_2\mu_2 = 1$ and $c_1 \neq c_2$. Here, $\rho_2 = 0.6$ are fixed.*

Case 2: $c_1\mu_{12} > c_2\mu_2$

In the rest of this chapter, we limit our attention to the case of $c_1\mu_{12} > c_2\mu_2$, where server 2 "prefers" to run type 1 jobs in a $c\mu$ sense. Note that condition $c_1\mu_{12} > c_2\mu_2$ is achieved when type 1 jobs are smaller than type 2 jobs, when type 1 jobs are more important than type 2 jobs, and/or in the pathological case when type 1 jobs have good affinity with server 2. (These, in addition, may motivate user of the Beneficiary-Donor model, giving smaller or more important jobs better service.) We will see that, for the T1 policy, the optimal T_1 threshold is typically finite when $c_1\mu_{12} > c_2\mu_2$, in contrast to the $c\mu$ rule. For the T2 policy, we will see that the optimal T_2 threshold is usually small, but not necessarily $T_2 = 1$.

Figure 7.5 shows the mean response time under the T1 policy as a function of T_1 (rows 1 and 3), and the mean response time under the T2 policy as a function of T_2 (rows 2 and 4) when $c_1\mu_{12} > c_2\mu_2$ (and $c_1 = c_2$). Throughout, $c_1\mu_{12} = 1$ is fixed, and we set $c_2\mu_2 = \frac{1}{16}$ in the top half, and $c_2\mu_{12} = \frac{1}{4}$ in the bottom half. Again, different columns correspond to different μ_1's. In each plot, mean response time is evaluated at three loads, $\hat{\rho}_1 = 0.8, 0.9, 0.95$, by changing λ_1

In Figure 7.5 (rows 1 and 3), we see that optimal T_1 is finite and depends on environmental conditions such as load ($\hat{\rho}_1$) and job sizes (μ_1). By Theorem 14, a larger value of T_1 leads to a larger stability region, and hence there is a tradeoff between good performance at the estimated load, $(\hat{\rho}_1, \rho_2)$, which is achieved at smaller T_1, and stability at higher $\hat{\rho}_1$ and/or ρ_2, which is achieved at larger T_1. Note also that the curves have sharper "V shapes" in general at higher $\hat{\rho}_1$ and/or smaller $c_2\mu_2$, which complicates the choice of T_1, since mean response time quickly diverges to infinity as T_1 becomes smaller. Also, when ρ_2 is lower (and thus ρ_1 is higher for a fixed $\hat{\rho}_1$), the optimal T_1 tends to become smaller, and hence the tradeoff between low mean response time at the estimated load and stability at higher loads is more significant. This makes intuitive sense, since at lower ρ_2, server 2 can help more.

In Figure 7.5 (rows 2 and 4), we see that the mean response time of the T2 policy is minimized at a small T_2, as in the case of $c_1\mu_{12} \le c_2\mu_2$. However, in contrast to the case of $c_1\mu_{12} \le c_2\mu_2$, the optimal T_2 is not necessarily 1 when $c_1\mu_{12} > c_2\mu_2$. Also, observe that the mean response time under the optimized T2 policy can be much higher than that under the optimized T1 policy. This difference becomes larger when μ_1 is smaller (or μ_{12} is larger), since the type 1 jobs are better served by server 2 and the optimized T1 policy can give more bias toward the type 1 jobs than T2 policies.

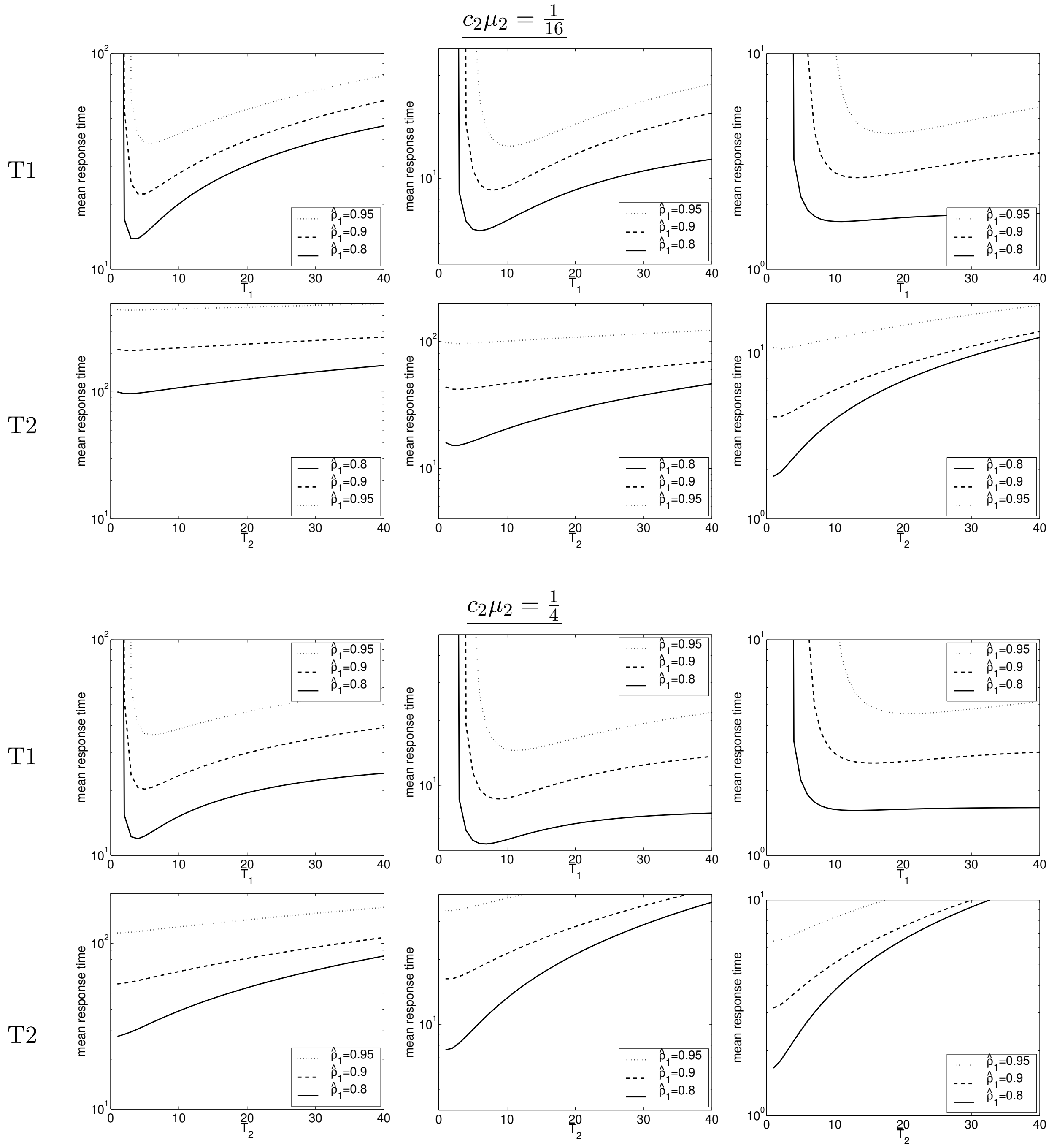
Although figures are not shown, we find that the above findings also hold when $c_1 \neq c_2$ (and $c_1\mu_{12} > c_2\mu_2$). That is, the value of the $c\mu$ product primarily determines the qualitative behavior of the T1 policy, and individual values of c and μ have smaller effect.

7.2.4 Static robustness of single-threshold allocation policies

We next study static robustness of the T1 and T2 policies by evaluating the mean response time of these policies under a range of loads. In particular, we examine how a policy (T1 or T2) tuned for a certain load behaves at other loads.

Figure 7.6 (top row) highlights static robustness of the T1 policy, plotting the mean response time of two T1 policies as a function of ρ_1 (only λ_1 is changed) in column (a) and as a function of ρ_2 (only λ_2 is changed) in column (b). For example, in column (b), $T_1 = 3$ is the optimal threshold when $\rho_2 = 0.4$. However, if it turns out that $\rho_2 = 0.8$ is the actual load, then the T1(3) policy leads to instability (infinite mean response time), while the T1(19) policy minimizes mean response time. Thus, choosing a higher T_1 (=19) guarantees stability against misestimation of ρ_2, but results in worse

Mean response time of T1 and T2: $c_1\mu_{12} > c_2\mu_2$



(a) $c_1\mu_1 = \frac{1}{4}$ (b) $c_1\mu_1 = 1$ (c) $c_1\mu_1 = 4$

Figure 7.5: *The mean response time under the T1 policy as a function of T_1 (rows 1 and 3), and the mean response time under the T2 policy as a function of T_2 (rows 2 and 4), when $c_1\mu_{12} > c_2\mu_2$. Here, $c_1 = c_2 = 1$, $c_1\mu_{12} = 1$, and $\rho_2 = 0.6$ are fixed.*

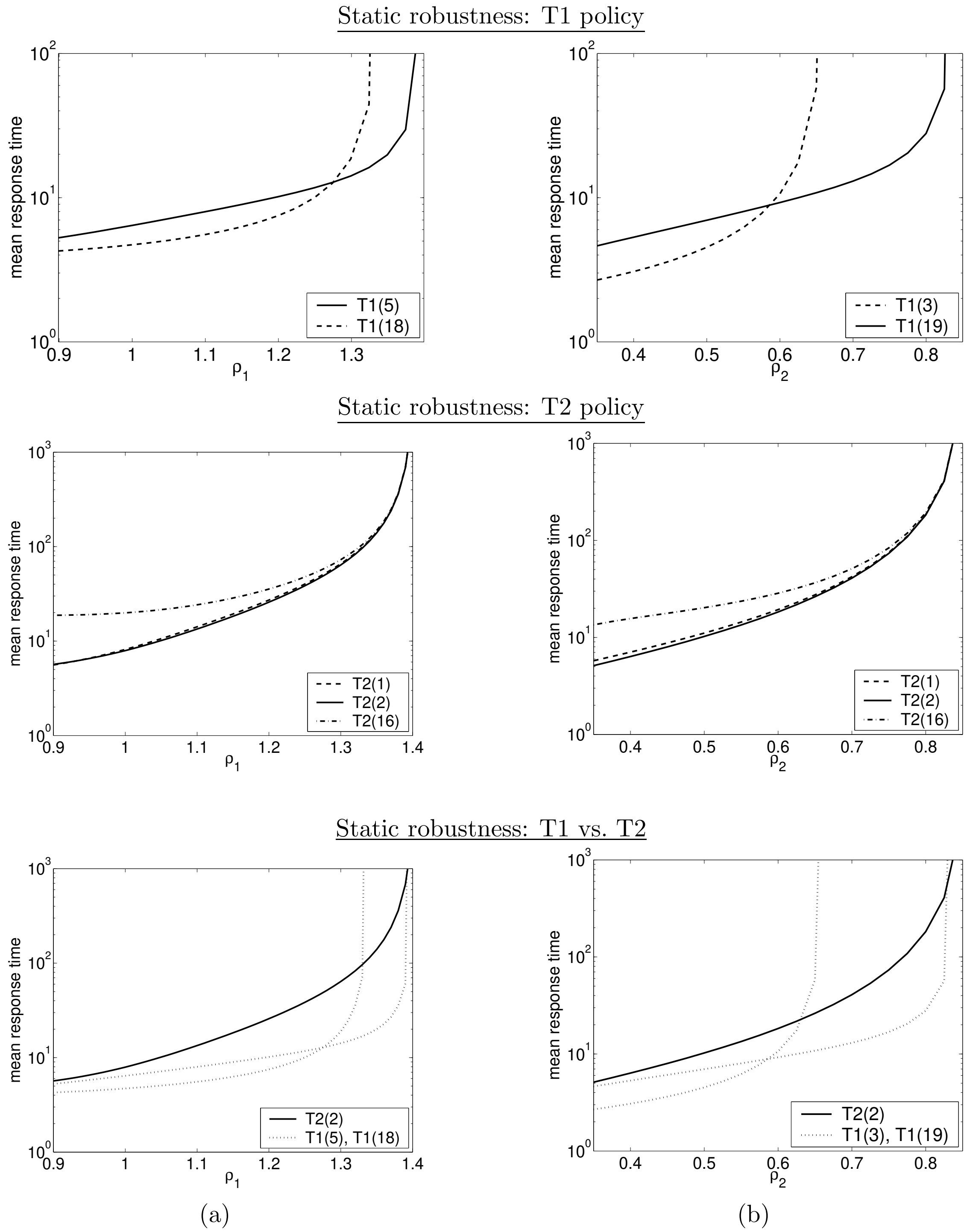


Figure 7.6: *Static robustness of single-threshold allocation policies is illustrated by plotting their mean response time as functions of (a)* ρ_1 *(*$\rho_2 = 0.6$ *is fixed) and (b)* ρ_2 *(*$\rho_1 = 1.15$ *is fixed). Here,* $c_1 = c_2 = 1$, $c_1\mu_1 = c_1\mu_{12} = 1$, *and* $c_2\mu_2 = \frac{1}{16}$ *are fixed. Bottom row compares the mean response time under the "optimized" T2 policy and two T1 policies.*

performance at the estimated load. Similar observation holds for stability against misestimation of ρ_1 (see column (a)), although the T1 policy appears to have more stability against misestimation of ρ_1 than against misestimation of ρ_2. Overall, we conclude that the T1 policy optimized at a lower load is poor with respect to static robustness. While it is possible to tune the T1 policy for higher loads (by choosing a larger T_1), this would degrade mean response time at lower loads.

Figure 7.6 (middle row) illustrates the static robustness of the T2 policy, plotting the mean response time of three T2 policies as functions of ρ_1 and ρ_2, as in the top row. In both columns (a) and (b), the T2(2) policy minimizes the mean response time in the class of T2 policies for a range of loads. That is, $T_2 = 2$ is the optimal choice for these parameter settings $(c_1, c_2, \mu_1, \mu_2, \mu_{12})$ regardless of the loads (λ_1, λ_2). Recall that the T2 policy is optimized at small T_2 values ($T_1 = 1$ or 2) for a wide range of parameter settings (see Figures 7.4-7.5). Since the T2 policy has the widest possible stability region regardless of its T_2 value, its mean response time is not very sensitive to the changes (or misprediction) in loads for a wider range of load, as compared to the T1 policy with parameter $T_1 < \infty$. In this sense, the T2 policy has more static robustness than the T1 policy.

However, Figure 7.6 (bottom) shows that the mean response time under the optimized T2 policy, T2(2), can be much higher than that under the T1 policy optimized at each load, which is in parallel to our previous findings (see Figure 7.5). For example, in column (b), the mean response time under T2(2) is twice as high as that under T1(3) when $\rho_2 = 0.4$, and the mean response time under T2(2) is eight times as high as that under T1(19) when $\rho_2 = 0.8$. Only at very high ρ_2, the mean response time of T2(2) becomes lower than that of T1(19). Similar observation holds for column (a) as well. We conclude that the T2 policy performs worse than the T1 policy with respect to mean response time.

7.3 Static robustness and mean response time of multi-threshold allocation policies

The tradeoff between the low mean response time of the T1 policy and good static robustness of the T2 policy motivates us to introduce multi-threshold allocation policies: T1T2 and ADT policies. The T1T2 policy places a threshold on each queue, while the ADT policy places two thresholds on queue 1 as well as a threshold on queue 2. We will study how the mean response time and static robustness of these multi-threshold allocation policies compare to that of the single-threshold allocation policies.

7.3.1 T1T2 policy

One might argue that the stability or static robustness issue of the T1 policy with small optimal T_1 is resolved simply by placing an additional threshold, T_2, on queue 2, so that whenever the length of queue 2, N_2, exceeds T_2, server 2 works on type 2 jobs regardless of the length of queue 1, thus preventing queue 2 from becoming unstable. We refer to this policy as the T1T2 policy. Surprisingly, we will see that the T1T2 policy improves upon static robustness of the T1 policy only slightly. Since the T1T2 policy generalizes the T1 policy (by setting $T_2 = \infty$, the T1T2 policy is reduced to the T1 policy), one might also expect that the mean response time of the optimized T1T2 policy could be strictly better than that of the optimized T1 policy as long as $T_2 < \infty$. This surprisingly turns out to be largely false; the mean response time of the optimized T1T2 policy does not appreciably improve upon that of the optimized T1 policy.

The T1T2 policy is formally defined as follows:

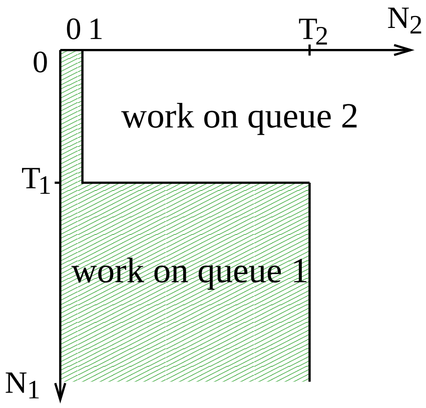


Figure 7.7: *Figure shows whether server 2 works on jobs from queue 1 or queue 2 as a function of N_1 and N_2 under the T1T2 policy with parameters (T_1, T_2).*

Definition 18 *The T1T2 policy with parameters (T_1, T_2) operates as the T1 policy with parameter T_1 if $N_2 < T_2$; otherwise, it operates as the T2 policy with parameter T_2.*

Figure 7.7 shows the jobs processed by server 2 a function of N_1 and N_2 under the T1T2 policy. Below, the T1T2 policy with parameters (t_1, t_2) is also denoted by the T1T2(t_1, t_2) policy.

The stability region of the T1T2 policy is the same as that of the T2 policy; i.e. the T1T2 policy has the widest possible stability region regardless of its parameters (T_1, T_2). This makes intuitive sense, since a T1T2 policy behaves like a T2 policy at high load. The following theorem can be proved in a similar way as Theorem 14.

Theorem 18 *Under the T1T2 policy with parameters (T_1, T_2) where $T_2 < \infty$, queue 1 is stable if and only if $\hat{\rho}_1 < 1$, and queue 2 is stable if and only if $\rho_2 < 1$.*

Figure 7.8 (top row) shows the mean response time under the T1T2 policy with various T_2 values (a) as a function of ρ_1 (b) as a function of ρ_2. In column (a), $T_1 = 3$ and T_2 ranges over 1, 40, and 100. In column (b), $T_1 = 5$ and T_2 ranges over 1, 10, and 100. Figure 7.8 (bottom row) compares the mean response time under the T1T2 policy with a "good" T_2 value ($T_2 = 40$ in column (a), and $T_2 = 10$ in column (b)) with that under two T1 policies studied in Figure 7.6. Note that the parameter settings are exactly the same as in Figure 7.6. The T1T2 policy with parameters (3,T_2) studied in column (b) is designed to provide a wider stability region with similar mean response time to the optimized T1 policy at $\rho_2 = 0.4$. Recall that the T1 policy achieves its lowest mean response time, given $\rho_2 = 0.4$, when $T_1 = 3$ (Figure 7.6). Likewise, the T1T2 policy with parameters (5,T_2) studied in column (a) is designed to provide a wider stability region with similar mean response time to the T1(5) policy.

Figure 7.8 shows that when T_2 is too large (respectively, too small), the T1T2 policy with parameters (T_1, T_2) behaves like the T1 policy with parameter T_1 (respectively, the T2 policy with parameter T_2). Specifically, by comparing the T1T2(3,100) policy and the T1(3) policy in column (b), we observe that the T1T2 policy with parameters (3,100) behaves like the T1(3) policy. Further, by comparing the T1T2(3,1) policy in column (b) and the T2(1) policy shown in the middle row of Figure 7.6(b), we observe that the T1T2 policy (3,1) behaves like the T2(1) policy. Similar observation holds for column (a) as well.

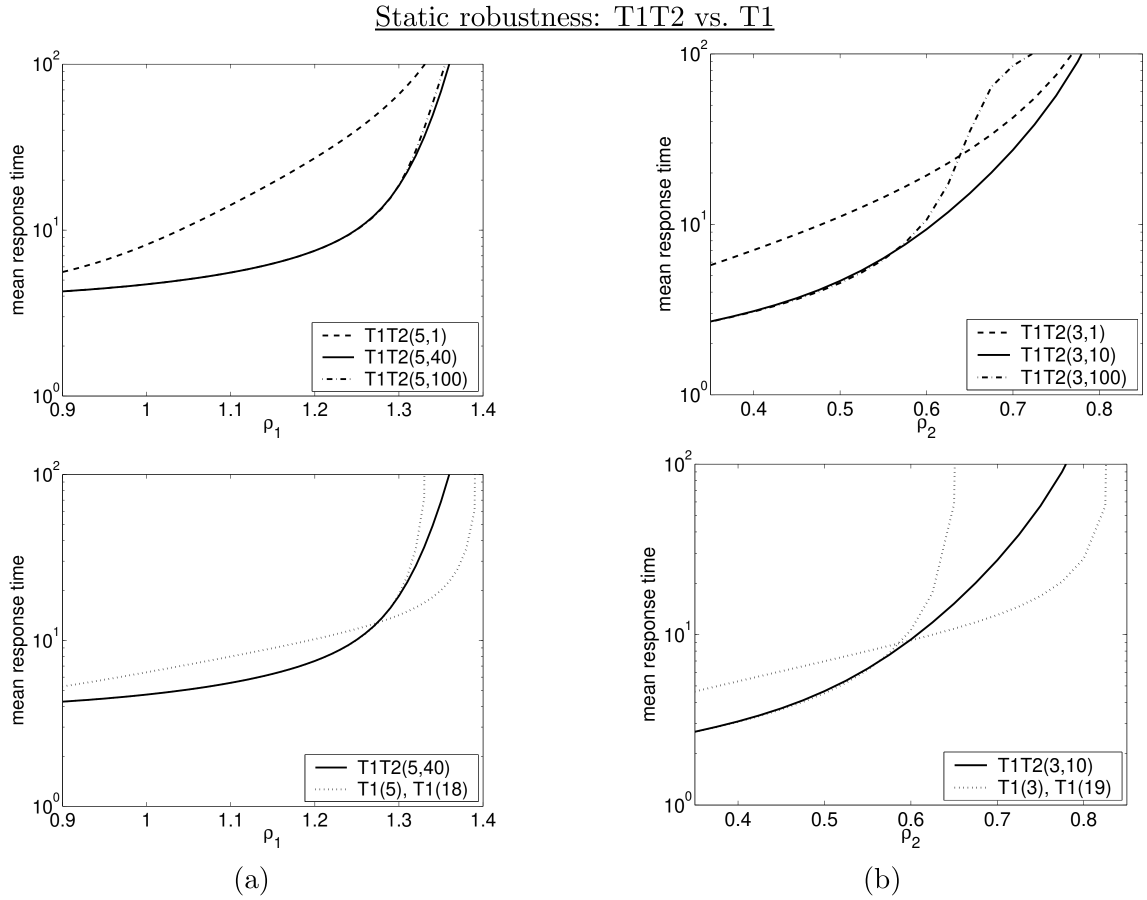


Figure 7.8: *Static robustness of the T1T2 policy is illustrated by plotting their mean response time as functions of (a) ρ_1 ($\rho_2 = 0.6$ is fixed) and (b) ρ_2 ($\rho_1 = 1.15$ is fixed). Here, $c_1 = c_2 = 1$, $c_1\mu_1 = c_1\mu_{12} = 1$, and $c_2\mu_2 = \frac{1}{16}$ are fixed. The top row shows the mean response time under the T1T2 policy with various (T_1, T_2) threshold values, and the bottom row compares the mean response time under the "optimized" T1T2 policy and two T1 policies.*

When T_2 is chosen appropriately, the T1T2 policy has a slight advantage over the T1 policy. The bottom row of Figure 7.8(b) shows that when $\rho_2 = 0.4$, the mean response time under the T1T2(3,10) policy is comparable to that under the T1(3) policy. For higher ρ_2 (specifically, $\rho_2 > 0.6$), the T1T2(3,10) policy provides lower mean response time than the T1(3) policy, hence being more robust. However, the range of load for which the T1T2(3,10) policy provides low response time is limited. For example, when $\rho_2 = 0.8$, the mean response time under the T1T2 policy with parameters (3,T_2) with any value of T_2 (the top row of Figure 7.8(b)) is significantly higher than the T1(19) policy. Again, similar observation holds for column (a) as well.

The inadequacy of the T1T2 policy is primarily due to the fact that the T1T2 policy operates like the T2 policy at higher load, but the mean response time of the T2 policy is typically poor at any load as compared to the optimized T1 policy. This motivates us to introduce a new policy, the ADT policy.

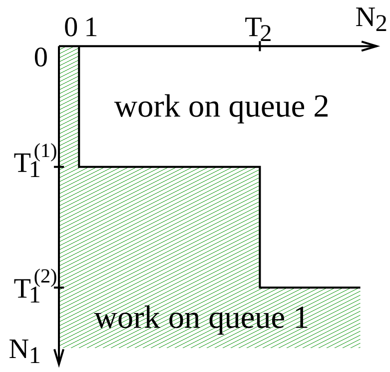


Figure 7.9: *Figure shows whether server 2 works on jobs from queue 1 or queue 2 as a function of N_1 and N_2 under the ADT policy with parameters $(T_1^{(1)}, T_1^{(2)}, T_2)$.*

7.3.2 The adaptive dual threshold (ADT) policy

The key idea in the design of the ADT policy is that, in contrast to the T1T2 policy, the ADT policy is *always* operating as a T1 policy, but unlike the standard T1 policy, the value of T_1 adapts, depending on the length of queue 2, to provide static robustness. Specifically, the ADT policy behaves like the T1 policy with parameter $T_1^{(1)}$ if the length of queue 2 is less than T_2 and otherwise like the T1 policy with parameter $T_1^{(2)}$, where $T_1^{(2)} > T_1^{(1)}$.

We will see that the ADT policy, unlike the T1T2 policy, is far superior to the T1 policy with respect to static robustness, due to the dual thresholds on queue 1. In addition, one might also expect that the mean response time of the optimized ADT policy will significantly improve upon that of the optimized T1 policy, since the ADT policy generalizes the T1 policy (the ADT policy is reduced to the T1 policy by setting $T_1^{(1)} = T_1^{(2)}$). However, this turns out to be largely false, as we see below. Formally, the ADT policy is characterized by the following rule.

Definition 19 *The ADT policy with parameters $(T_1^{(1)}, T_1^{(2)}, T_2)$ operates as the T1 policy with parameter $T_1^{(1)}$ if $N_2 \le T_2$; otherwise, it operates as the T1 policy with parameter $T_1^{(2)}$.*

Below, the ADT policy with parameters $(t_1^{(1)}, t_1^{(2)}, t_2)$ is also denoted by the ADT$(t_1^{(1)}, t_1^{(2)}, t_2)$ policy.

Figure 7.9 shows the jobs processed by server 2 under the ADT policy as a function of N_1 and N_2. At high enough $\hat{\rho}_1$ and ρ_2, N_2 usually exceeds T_2, and the policy behaves similar to the T1 policy with parameter $T_1^{(2)}$. Thus, the stability condition for the ADT policy is the same as that for the T1 policy with parameter $T_1^{(2)}$. The following theorem can be proved in a similar way as Theorem 14.

Theorem 19 *The stability condition for the ADT policy with parameters $(T_1^{(1)}, T_1^{(2)}, T_2)$ is given by the stability condition for the T1 policy with parameter $T_1^{(2)}$ (Theorem 14).*

Static robustness of the ADT policy

Figure 7.10 illustrates static robustness of the ADT policy, showing the mean response time under the ADT policy (a) as a function of ρ_1 and (b) as a function of ρ_2. For comparison, the mean response time under two T1 policies are plotted with dotted lines. The parameter settings in the middle row are exactly the same as those in Figure 7.6. In the top and bottom rows, only the value of $c_1\mu_1$ is changed, as labeled.

Figure 7.10 shows that the ADT policy with parameters $(T_1^{(1)}, T_1^{(2)}, T_2)$ achieves at least as low mean response time as the *better* of the two T1 policies, one with parameter $T_1^{(1)}$ and the other with parameter $T_1^{(2)}$, throughout the range of ρ_1 and ρ_2, if T_2 is chosen appropriately. In plotting Figure 7.10, we found "good" T_2 values manually by trying a few different values so that the ADT policy provides low mean response time throughout the range of load, which took only a few minutes. Observer that if T_2 is set too low, the ADT policy behaves like the T1 policy with parameter $T_1^{(2)}$, degrading the mean response time at lower loads, since $T_1^{(2)}$ is larger than the optimal T_1 in the T1 policy at lower loads. If T_2 is set too high, the ADT policy behaves like the T1 policy with parameter $T_1^{(1)}$. This worsens the mean response time at higher loads.

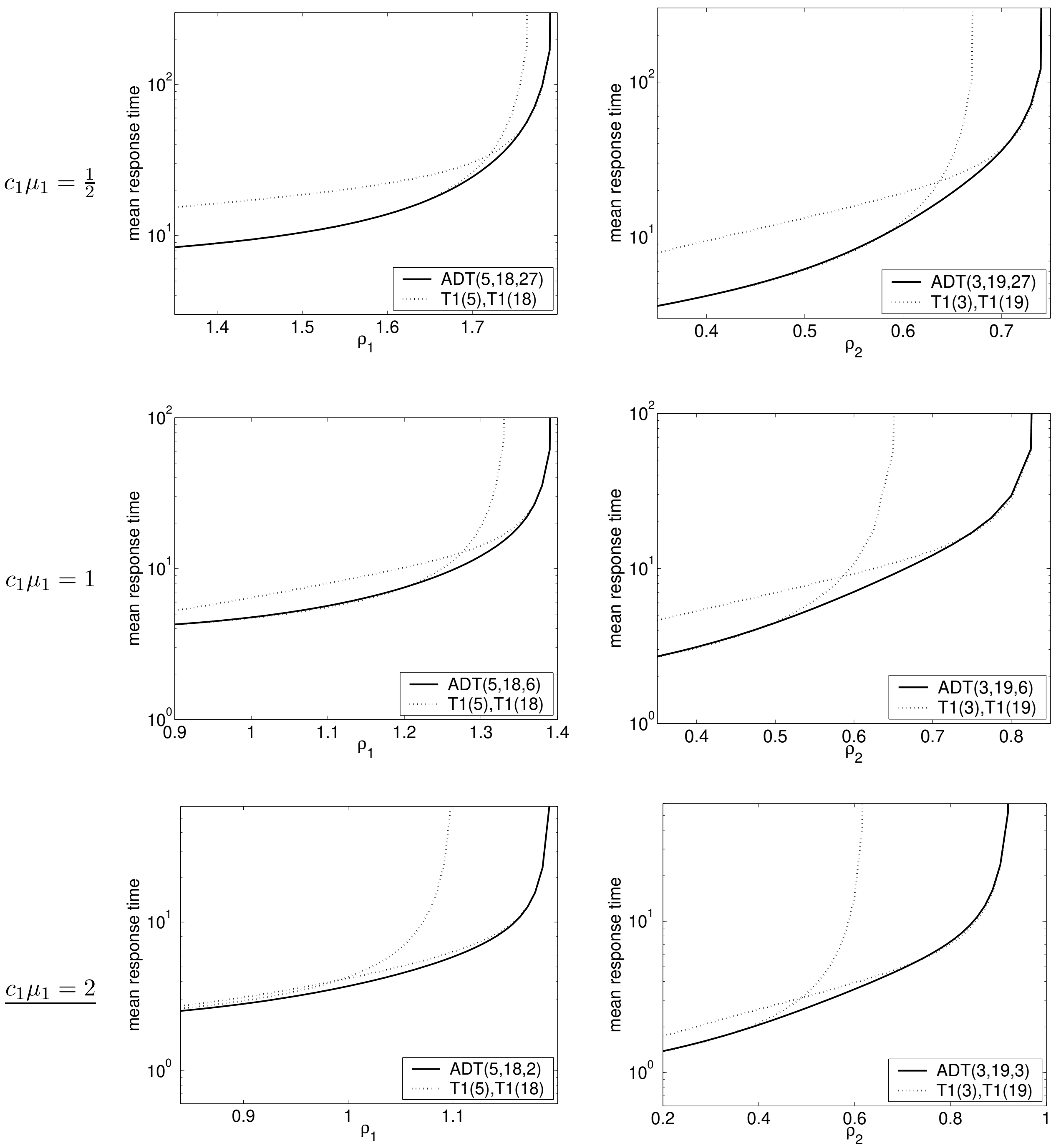
Static robustness of the ADT policy can be attributed to the following. The dual thresholds on queue 1 make the ADT policy adaptive to misestimation of load, in that the ADT policy with parameters $(T_1^{(1)}, T_1^{(2)}, T_2)$ operates like the T1 policy with parameter $T_1^{(1)}$ at the estimated load and like the T1 policy with parameter $T_1^{(2)}$ at a higher load, where $T_1^{(2)} > T_1^{(1)}$. Thus, server 2 can help queue 1 less when there are more type 2 jobs, preventing server 2 from becoming overloaded. This leads to the increased stability region and improved performance.

Figure 7.10 makes an additional point about the effect of $c_1\mu_1$. When $c_1\mu_1$ is smaller (top row), the difference in the stability regions of the two T1 policies becomes smaller, and the difference in the mean response times of the two T1 policies at low loads becomes larger. Thus, the benefit of the ADT policy over the T1 policy becomes smaller at smaller $c_1\mu_1$, since the T1 policy with smaller T_1 can provide low mean response time with only a slightly reduced stability region. This makes intuitive sense, since in the limit as $c_1\mu_1 \to 0$, the Beneficiary-Donor model reduces to a single (donor) server with two queues, where the policy following the $c\mu$ rule is provably optimal [45]. Also, when $c_1\mu_1$ is larger (bottom row), the difference in the stability region of the two T1 policies becomes larger, and the difference in the mean response times of the two T1 policies at low loads becomes smaller. Again, the benefit of the ADT policy over the T1 policy becomes smaller at larger $c_1\mu_1$, since the T1 policy with larger T_1 can provide a wide stability region with only a slightly high mean response time at lower loads. This makes intuitive sense, since the jobs in queue 1 are much better served by server 1 and the help from server 2 becomes negligible when $c_1\mu_1$ is much larger than $c_1\mu_{12}$. Thus, the ADT policy is most effective when $c_1\mu_1$ and $c_1\mu_{12}$ are comparable and $c_1\mu_{12} > c_2\mu_2$, where the T1 policy with $1 < T_1 < \infty$ provides lower mean response time than T1(1) and T1(∞) but suffers from static robustness.

Mean response time of the ADT policy

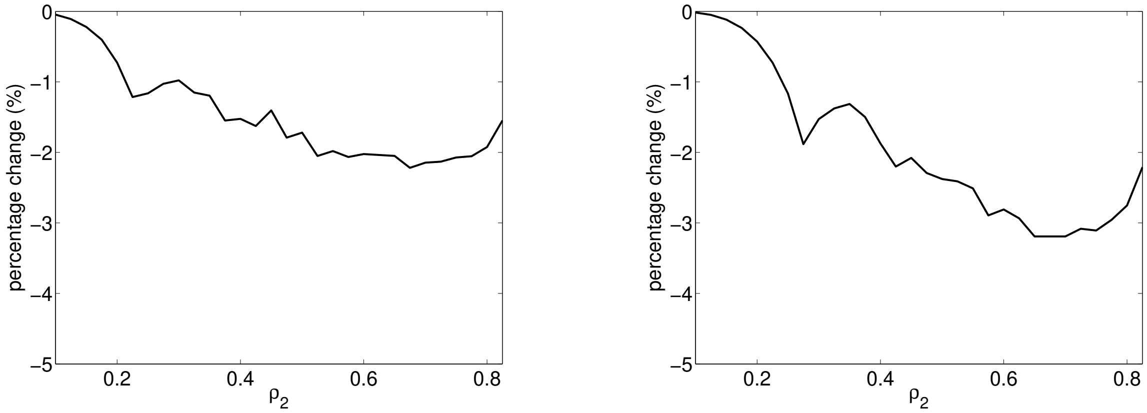
We have already seen the benefits of the ADT policy when the load is not exactly known (static robustness). One might also expect that, even when the load is known exactly, the ADT policy might significantly improve upon the T1 policy with respect to mean response time. Earlier work of Ahn et al. and Meyn provides some support for this expectation [4, 125]. For example, Meyn shows via numerical examples that, in the case of finite buffers for both queues, the policy that minimizes mean response time is a "flexible" T1 policy which allows a continuum of T1 thresholds, $\{T_1^{(i)}\}$, where threshold $T_1^{(i)}$ is used when the length of queue 2 is i [125]. The ADT policy can be seen as

Static robustness: ADT vs. T1



(a) (b)

Figure 7.10: *Static robustness of the ADT policy is illustrated by plotting the mean response time as functions of (a)* ρ_1 *(*$\rho_2 = 0.6$ *is fixed) and (b)* ρ_2 *(*$\rho_1 = 1.15$ *is fixed). Here,* $c_1 = c_2 = 1$, $c_1\mu_1 = c_1\mu_{12} = 1$, *and* $c_2\mu_2 = \frac{1}{16}$ *are fixed. For comparison, the mean response time under two T1 policies are also shown.*



(a) $c_2\mu_2 = \frac{1}{4}$ (b) $c_2\mu_2 = \frac{1}{16}$

Figure 7.11: *Comparison of the ADT policy with the T1 policy with respect to the mean response time. The percentage change (%) in the mean response time of the (locally) optimized ADT policy over the optimized T1 policy at each given load is shown as a function of ρ_2 (0% corresponds to the T1 policy optimized at each load). Here, $c_1 = c_2 = 1$, $c_1\mu_1 = c_1\mu_{12} = 1$, and $\rho_1 = 1.15$ are fixed.*

an approximation of a "flexible" T1 policy, using only two T_1 thresholds.

To evaluate the benefit of the ADT policy, we compare it over a range of ρ_2 against the T1 policy optimized for the given ρ_2. Since the search space of the threshold values for the ADT policy is large, we find *locally* optimal threshold values, which are found to be optimal within a search space of ± 5 for each threshold. We measure the percentage change in the mean response time of ADT versus T1:

$$\frac{\mathsf{E}[R_{ADT}] - \mathsf{E}[R_{T1}]}{\mathsf{E}[R_{T1}]} \times 100 \quad (\%), \tag{7.2}$$

where $\mathsf{E}[R_X]$ denotes the mean response time in policy $X \in \{\text{T1,ADT}\}$.

Figure 7.11 shows the percentage reduction in the mean response time of the locally optimized ADT policy over the T1 policy optimized at each ρ_2, as a function of ρ_2. The parameter settings are exactly the same as those in Figure 7.6(b). Figure 7.11 shows that, surprisingly, the benefit of the ADT policy is quite small with respect to mean response time under fixed Poisson arrivals; the improvement of the ADT policy is larger at moderately high ρ_2 and at smaller $c_2\mu_2$ value, but overall the improvement is typically within 3%. We conjecture that adding more thresholds (approaching the flexible T1 policy) will not improve mean response time appreciably, given the small improvement from one to two thresholds. Thus, whereas the ADT policy has significant benefits over the simpler T1 policy with respect to static robustness, the two policies are comparable with respect to mean response time.

7.4 Dynamic robustness of threshold based policies

We have seen, in Section 7.3.2, that the mean response time of the optimized ADT policy is similar to that of the optimized T1 policy, although the ADT policy has greater static robustness. Note that this observation is based on the assumption of Poisson arrivals. Consider, to start, an alternate scenario, in which the load at queue 2 fluctuates. For example, a long high load period (e.g., $\rho_1 = 1.15$

and $\rho_2 = 0.8$) is followed by a long low load period (e.g., $\rho_1 = 1.15$ and $\rho_2 = 0.4$), and the high and low load periods alternate. The T1 policy with a fixed threshold value must have a high mean response time either during the high load period or during the low load period (recall Figure 7.6). On the other hand, the ADT policy may provide low mean response time during both high and low load periods, since the T_1 threshold value is self-adapted to the load (recall Figure 7.10). In this section, we study the mean response time of the ADT policy when the load fluctuates, or the *dynamic robustness* of the ADT policy.

We use a Markov modulated Poisson process of order two (MMPP(2)), as defined in Section 3.2, as an arrival process. An MMPP(2) has two types of periods, which we denote as the high load period and the low load period. The duration of each type of period has an exponential distribution, which can differ in each type of period. During the high (respectively, low) load period, the arrival process follows a Poisson process with rate λ_H (respectively, λ_L), where $\lambda_H > \lambda_L$.

Figure 7.12 shows the percentage change in the mean response time of the (locally) optimized ADT policy over the optimized T1 policy, when arrivals at queue 1 follow a Poisson process and arrivals at queue 2 follow an MMPP(2). The arrival rates in the MMPP(2) are chosen such that the load during the high load period is $\rho_2 = 0.8$ and the load during the low load period is $\rho_2 = 0.2$, 0.4, or 0.6, while $\rho_1 = 1.15$ is fixed throughout. Other parameter settings are the same as in Figure 7.11. We choose the horizontal axis to be the expected number of type 2 arrivals during a high load period, and the three lines (solid, dashed, and dotted) to be different expected number of type 2 arrivals during a low load period. Note that since there are less frequent arrivals during a low load period, having the same number of arrivals during the high and low load periods implies that the low load period is longer. Thus, the number of arrivals during each period is an indicator of how frequently the load changes, which we find to be an important parameter in studying dynamic robustness. The threshold values of the optimized T1 policy and the (locally) optimized ADT policy are chosen such that the overall weighted mean response time is minimized for each given arrival process.

The first thing to notice in Figure 7.12 is that the improvement of the ADT policy over the T1 policy is smaller when the duration of the high and low load periods is shorter, or equivalently when there are less arrivals in each period. This makes intuitive sense, since the MMPP(2) reduces to a Poisson process when the high and low load periods alternate infinitely quickly, and under the Poisson process, the optimized T1 policy and the optimized ADT policy provide similar mean response time (Section 7.3.2).

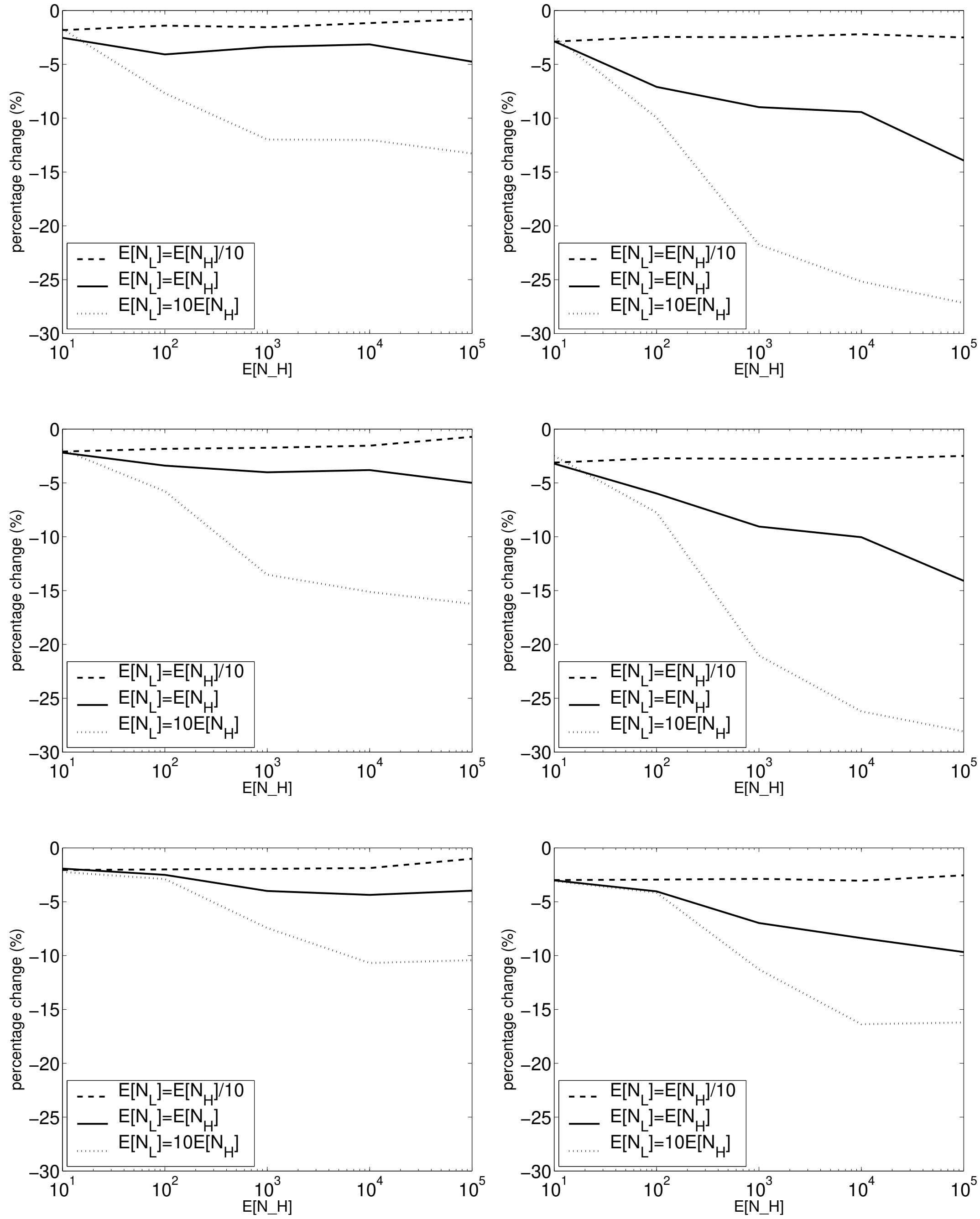
However, even when the durations are longer, the performance improvement of the ADT policy over the T1 policy is comparatively small (3 to 25%). This is mainly because the mean response time of the jobs arriving during the high load period tends to dominate the *overall* mean response time for two reasons: (i) the response time of jobs arriving during the high load period is much higher than that of jobs arriving during the low load period, partially due to the fact that any reasonable allocation policy (such as the optimized T1 policy and the optimized ADT policy) can provide low mean response time at low load, and (ii) assuming that a low load period and a high load period have the same *duration*, there are more arrivals during a high load period. Since the T1 policy with a fixed T_1 threshold can provide low mean response time for the jobs arriving during the high load period, it can provide low *overall* mean response time. (Of course, if there were *many* more arrivals during the low load period than the high load period, the T_1 threshold would be adjusted.)

It is only when the jobs arriving during the high load period and the jobs arriving during the

$\rho_2 = [0.2, 0.8]$

$\rho_2 = [0.4, 0.8]$

$\rho_2 = [0.6, 0.8]$



(a) $c_2\mu_2 = \frac{1}{4}$ (b) $c_2\mu_2 = \frac{1}{16}$

Figure 7.12: *Dynamic robustness of the ADT policy and the T1 policy. The percentage change (%) in the mean response time of the (locally) optimized ADT policy over the optimized T1 policy for each given MMPP(2), shown as a function of the expected number of arrivals during a high load period,* $\mathsf{E}[N_H]$*, and during a low load period,* $\mathsf{E}[N_L]$*. A negative percentage indicates the* improvement *of ADT over T1. Here,* $c_1 = c_2 = 1$*,* $c_1\mu_1 = c_1\mu_{12} = 1$*, and* $\rho_1 = 1.15$ *are fixed.*

low load period have roughly equal contribution to the *overall* mean response time [1] that the ADT policy can have appreciable improvement over the T1 policy. For example, Figure 7.12 suggests that the ADT policy can provide a mean response time that is 20-30% lower than that of the T1 policy, when the number of arrivals during a low load period is ($\sim$ 10 times) larger than that during a high load period.

In addition (not shown), we find that when arrivals at queue 1 follow an MMPP(2) or when arrivals at both queues follow MMPP(2)'s, the improvement of the ADT policy over the T1 policy tends to be smaller than when only arrivals at queue 2 follow an MMPP(2). Overall, we conclude that the ADT policy has appreciable improvement over the T1 policy only when there are more arrivals during the low load period than during the high load period, giving them comparable importance.

7.5 Concluding remarks

In this chapter, the performance of a wide range of threshold-based (resource) allocation policies for the Beneficiary-Donor model is studied via dimensionality reduction (DR), as their analyses involves multidimensional Markov chains. DR allows us to evaluate the mean response time under a broad class of threshold-based policies under a wide range of load conditions, for the first time, and we find surprising conclusions.

We first consider single threshold policies, T1 and T2, and find that the T1 policy is superior with respect to (overall weighted) mean response time. That is, the threshold for resource allocation is better determined by the beneficiary queue length (queue 1) than by the donor queue length (queue 2), in all cases studied.

We then compare single threshold policies to a multiple threshold policy, the adaptive dual threshold (ADT) policy, with respect to mean response time, assuming that the load is fixed and known. We find that when the threshold value is chosen appropriately, the mean response time of the T1 policy is at worst very close to the best mean response time achieved by the ADT policy. This is surprising, since the optimal policy appears to have infinitely many thresholds, but evidently the improvement these thresholds generate is marginal.

We next study static robustness, where the load is constant, but may have been misestimated. We find that the ADT policy not only provides low mean response time but also excels in static robustness, whereas the T1 policy does not. The increased flexibility of the ADT policy enables it to provide low mean response time under a range of loads. Hence, when the load is not exactly known, the ADT policy is a much better choice than the T1 policy.

Finally, and surprisingly, our analysis shows that this improvement in static robustness does not necessarily carry over to dynamic robustness, i.e. robustness against load fluctuation. We observe that when the load is fluctuating, the T1 policy, which lacks static robustness, can often provide low mean response time, comparable to ADT. This can occur for example because the mean response time of jobs arriving during the high load period may dominate the overall mean response time.

The results in this chapter have implications to the multiserver systems studied in previous chapters. For example, in Chapter 5, we have proposed size-based task assignment policies with cycle stealing, `SBCS-ID` and `SBCS-CQ`, but these task assignment policies can be improved upon by

[1] That is, when $\sum_{i=1}^{2} c_i p_i^L \mathsf{E}\left[R_i^L\right] \sim \sum_{i=1}^{2} c_i p_i^H \mathsf{E}\left[R_i^H\right]$, where p_i^L (respectively, p_i^H) is the fraction of jobs that are type i and arriving during the low (respectively, high) load period, and $\mathsf{E}\left[R_i^L\right]$ (respectively, $\mathsf{E}\left[R_i^H\right]$) is the mean response time of type i jobs arriving during the low (respectively, high) load period, for $i = 1, 2$.

a threshold-based policy, whereby the server for long jobs processes short jobs when the number of long jobs is below a threshold value, as in the T1 policy, or by a more complex threshold-based policy like the ADT policy. In designing such threshold-based *task assignment* policies, lessons learned in this chapter would be useful.

Also, robustness is more or less a common issue in resource allocation policies that have parameters to be tuned. For example, in Chapter 6, we have studied a threshold-based policy for reducing switching costs in cycle stealing, but the optimal threshold values depend on loads. When these resource allocation policies are used where the load is not known, it would be important to study the robustness of these allocation policies and to design robust allocation policies based on the lessons learned in this chapter.

Part III

Further discussion and conclusion

Chapter 8

Applications in contact center design

In this chapter, we discuss how the analytical tools developed in Chapters 2-3 (dimensionality reduction (DR) and moment matching algorithms) and the results of our analysis or lessons learned in Chapters 4-7 may be used in designing contact centers (or call centers) today. Although we discuss the specific example of contact center design, the analytical tools developed and lessons learned in this thesis may be applied similarly to designing other real-world systems as well.

8.1 Motivation

In today's competitive business, even high-tech products can be quickly reduced to commodities that are difficult to differentiate through features, functions, or price. As a result, it becomes increasingly important to provide *accessibility* to the company, which has a direct impact on the customers' perception of quality, in order to acquire, keep, and grow customers [8, 58, 123]. A contact center (or call center) provides such accessibility for customers who buy or reserve products or services and for customers who have questions, complaints, or need for technical supports and other customer services. Since contact centers are the critical element of customer relationship management (CRM), they are ubiquitous in PC companies, banks, airline companies, rent-a-car companies, hotels, and credit-card companies, etc.

Although a contact center is a significant revenue generator, its operating expenses are also huge, and human resource costs are estimated to account for 50-75% of the total operating expenses [58, 123]. As a result, a significant amount of research has addressed analytical tools and simulation techniques that support capacity management at contact centers [119].

However, operations at contact centers have become increasingly complex, and the increased complexity in the operations makes it difficult to apply existing analytical tools to support capacity management. For example, new technology for communication has become available, and a mere "call" center is transformed to a contact center that support various communication channels, including telephones, e-mails, and chat. Also, advancement of information technology such as data mining and information retrieval enables one to identify customers of high value and to give priority to these customers. Further, more and more specialized products and services are created and sold, and particular customers need to be served by particular agents.

Analytical tools developed and lessons learned in this thesis may be useful in supporting capacity management at contact centers with these complex operations, and for other purposes in contact

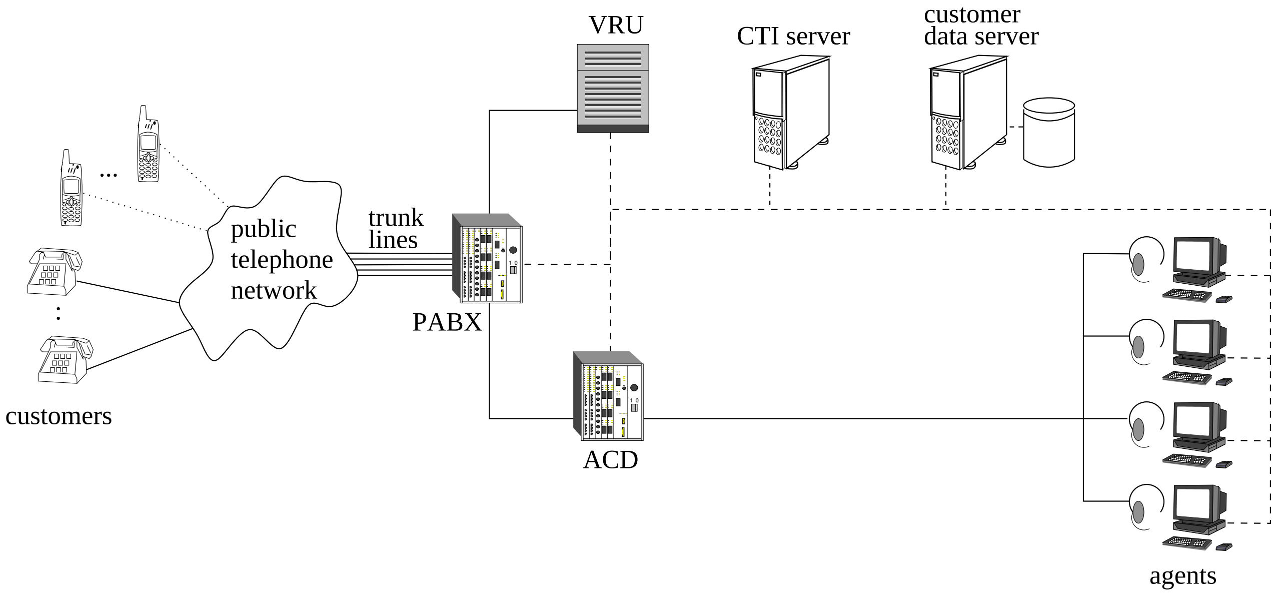


Figure 8.1: *Contact center architecture [58].*

center operations. In this chapter, we will provide an overview of contact center operations (Section 8.2), and discuss how the results in this thesis may be applied in designing contact centers today (Section 8.3).

8.2 Overview of contact center operation

In this section, we provide a brief overview of operations at contact centers. For a more comprehensive tutorial and survey of the literature, see [58, 119].

A contact center is a complex integration of computers, human operators (agents), telephone and packet networks, and their equipment. Contact centers can be classified into three types, depending of whether they handle inbound traffic, outbound traffic, or both inbound and outbound traffic, and their scale can vary from a few agents to thousands of agents. Figure 8.1 illustrates an example of a large scale contact center ("call" center, in particular) that handles inbound calls [58].

Consider a customer calling a contact center via a toll-free number. The telephone company that provides the toll-free service connects the call via the public telephone network to a private automatic branch exchange (PABX) of the contact center. Here, the PABX is connected to the public telephone network through multiple telephone lines, called trunk lines. If there is a free trunk line, the call is connected to the PABX; otherwise, the caller receives a busy signal.

The calls connected to the PABX are routed either to an (interactive) voice response unit (VRU) or to an automatic call distributor (ACD), possibly depending on the dialed number and/or the caller's ID (number). If a call is routed to the VRU, the caller hears an automatic response and performs self-service by pushing buttons (dialing) or by speaking to the VRU. Some customers complete service at the VRU, and others who request connection to an agent are routed to the ACD.

An automatic call distributor (ACD) is a specialized router that is designed to route calls to individual agents at the contact center. At the ACD, various information is available both on the caller and on the agent, and the information is used to route a caller to an agent. The caller's information such as the caller ID and the record of interaction at the VRU may be used to identify the customer, and this allows the ACD to retrieve more information, such as their language, priority, and the type of requested service, from a customer data server. Agents log into the ACD at their terminals when they start working, and their login ID can be used to retrieve full information on types and levels of their skills. Computer-telephone integration (CTI) is a middleware that closely ties computer systems, including the customer data server and the terminals at the agents, and telephone networks. CTI enables sophisticated routing at the ACD. In addition, CTI can be used to automatically display a caller's information on an agent's workstation screen.

Based on the caller information as well as the information and availability of agents, the call may be connected to an agent, or may be held on in a queue. A waiting customer may abandon the call before being served, or may eventually be connected to an agent.

8.3 Towards efficient contact center operation

An objective of contact centers is to provide high customer satisfaction at low operational cost. Customer satisfaction is achieved by providing high accessibility or short waiting time, as well as by providing a high quality interaction of agents with customers [58]. Operational cost is dominated (50-75%) by human resources [58, 123]; thus reducing the number of agents is quite effective in reducing the operational cost. However, since fewer agents implies longer waiting time, which also causes a higher abandon rate, contact centers trade off high customer satisfaction and low operational cost. Thus, a major challenge in contact centers is determining the number of agents to be assigned (capacity planning) such that the right level of customer satisfaction is achieved at the right operational cost. Designing good routing policies at the automatic call distributor (ACD) is also important, since good routing policies can increase the accessibility or shorten the waiting time without increasing the number of agents.

In this section, we discuss how the results (analytical tools developed and lessons learned) in this thesis may be used in capacity planning and routing policy design at contact centers. In Section 8.3.1, we briefly review management of contact centers (see also Figure 8.2), including capacity planning and routing policy design. In Section 8.3.2, we discuss how the results in this thesis may be used in capacity planning. In Section 8.3.3, we discuss how the results in this thesis may be used in designing routing policies.

8.3.1 Overview of contact center management

An enormous amount of operational data can in theory be collected (*measurement*) at the (interactive) voice response unit (VRU) and the automatic call distributor (ACD). In practice, only summary statistics of such data are often stored, for example at the granularity of 30 minutes, in order to reduce the necessary storage space. The summary statistics usually include the number of arrivals and abandoned jobs, average service time, agents' utilization, and the distribution (percentiles) of waiting time in the queue [58].

The collected data are used to *estimate* the arrival process, service time distribution, and distribution of the time a customer is willing to wait, for each "type" of calls [29]. The history of these estimates are then used to *forecast* future arrivals, service times, and the time willing to wait [30, 118].

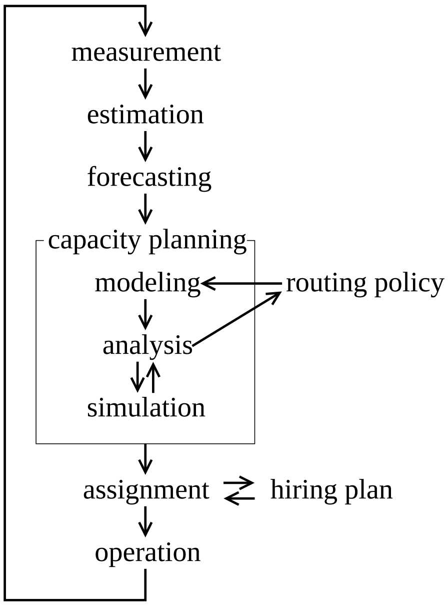


Figure 8.2: *A flow of routing policy design and capacity management/planning at a contact center.*

Forecasting allows capacity planning via a queueing analysis. Assuming that a routing policy at the ACD is given, a contact center can be *modeled* as a (multiserver) queueing system, where the ACD has queues of calls that are to be served by servers (agents). An *analysis* of the queueing system allows us to determine the number of agents (of each type and level of skill) to be assigned for each time slot (e.g. half an hour), and it also provides lessons and guidelines that are useful in designing a routing policy at the ACD. *Simulation* may/should be used for fine tuning the number of agents to be assigned. In Sections 8.3.2-8.3.3, we will elaborate on capacity planning and routing policy design.

Once the number of agents to be assigned for each time slot is given, an *assignment* of individual agents to the time slots can be determined. In determining the assignment, one usually needs to solve an integer program (IP), so that various constraints are satisfied. We study a similar problem in the context of nurse scheduling in [154]. The solution to the IP provides the working schedule of each agent, and it can also be used for hiring and planning training. For example, if there are no feasible solutions (satisfying all the constraints), new agents need to be hired, or some existing agents need to be trained for new skills. The contact center operates following the assignment schedule, and the operational data will be measured, which in turn provides feedback to capacity planning and agent assignment.

8.3.2 Capacity planning

The goal of capacity planning is to determine the number of agents to be assigned at each time slot so that a desired customer satisfaction (e.g. short waiting time and low abandon rate) is achieved at low operational cost (fewer number of agents), given the forecasted future arrivals, service times, time a customer is willing to wait, as well as the routing policy at the automatic call distributor (ACD). As we have seen in Section 8.3.1, capacity planning consists of (iterations of) modeling, analysis, and simulation.

When a contact center is operated in a simple manner, for example when calls are served by homogeneous agents in the FCFS order, the contact center can be modeled in such way that it can be analyzed efficiently and accurately, and such analytical methods are found to be quite useful in capacity planning at contact centers [58]. Analytical methods not only quickly provide the number of agents to be assigned, but also allow us to study the performance under a wide range of parameter settings, which is useful in designing routing policies as we will see in Section 8.3.3.

Recently, however, it has become popular to prioritize calls (value-based routing) and to route particular calls to particular agents (skill-based routing). Under value-based or skill-based routing, a contact center can be modeled as a multidimensional Markov chain, for which only coarse approximations exist in the literature. Below, we discuss how the analytical tools developed in this thesis can support modeling and analysis in capacity planning at contact centers today.

Modeling

A first step in capacity planning at a contact center is to model the contact center as a queueing model. In modeling a contact center, it is desirable that (i) the queueing model well captures the behavior of the contact center, (ii) the queueing model can be analyzed efficiently, and (iii) the modeling can be done easily and quickly. Property (i) is important, since a more precise queueing model provides a more accurate number of agents to be assigned. This can significantly reduce the necessary iterations of simulation. Properties (ii)-(iii) are important so that modeling and analysis does not become a bottleneck in capacity planning. An efficient analysis also helps in designing good routing policies, as we will see in Section 8.3.3.

A popular approach in modeling a contact center is to map the forecasted interarrival time, service time, and time a customer is willing to wait to phase type (PH) distributions (as defined in Section 2.2), so that the resulting queueing model can be analyzed as a Markov chain. The moment matching algorithms developed in Chapter 2 can be used exactly for this purpose. Recall the four desired properties that our moment matching algorithm has. Each of the four properties is desirable specifically in contact center modeling. Specifically, our moment matching algorithm can match the first *three* moments of the input distribution (and is defined for a *broadest* possible class of input distributions). This allows the queueing model to better capture the behavior of the contact center, as compared to existing moment matching algorithms that match only two moments. Also, the matching PH distribution provided by our moment matching algorithm has at most $\mathrm{OPT}(G)+1$ phases. This makes the state space of the queueing model (and the corresponding Markov chain) small, which in turn allows an efficient evaluation of the Markov chain. Further, our moment matching algorithm has short running time (closed form solutions are provided), which allows easy and quick modeling.

Analysis

An analysis of a queueing model of a contact center can often be reduced to an analysis of a Markov chain. However, in the contact centers with value-based or skill-based routing, the analysis of the queueing model often involves a *multidimensional* Markov chain to which existing analytical tools do not apply. Dimensionality reduction (DR) developed in Chapter 3 allows us to analyze (efficiently with high accuracy) some of the multidimensional Markov chains that model the contact centers with a value-based or skill-based routing.

For example, in Chapter 4, we have studied a multiserver system with multiple priority classes. This multiserver system with multiple priority classes can model a contact center with a value-based routing, where customers with higher priority are allowed to jump ahead of customers with lower priority. For example, big spenders or big investors may have higher priority at contact centers [8, 29]. In Chapter 4, we have considered "how many servers are best?" given a fixed total capacity. However, in capacity planning at contact centers, they would be interested in questions such as "what is the minimum number of agents so that the mean waiting time is less than a minutes?" DR can be used to answer such questions as well.

In Chapter 7, we have studied the Beneficiary-Donor model (see Figure 7.1), which can model a contact center with a skill-based or value-based routing. For example, the donor server may be a bilingual agent, and the beneficiary server may be a monolingual agent [58, 105, 176, 185, 186]. Instead, the donor server may be a cross-trained or experienced agent who can handle all types of questions, and the beneficiary server may be a specialized agent who are trained to handle a specific type of questions [8, 58, 176]. Alternatively, customers served by the beneficiary server may have higher priority, and the donor server contributes to maintaining an adequate service level of the high priority customers; conversely, customers served by the donor server may have higher priority, and the excess capacity of the donor server may be used to help the (otherwise overloaded) beneficiary server [58]. In Chapter 7, we fixed the number of donor and beneficiary servers, and studied the mean response time and robustness of a wide range of threshold-based policies. However, DR applies to the Beneficiary-Donor model with an arbitrary number of donor and beneficiary servers (the running time increases with the number of servers), and allows us to determine the number of beneficiary and donor agents to be assigned at a contact center with a skill-based or value-based routing.

8.3.3 Routing policy design

A goal of a routing policy is to provide high accessibility or short waiting times with a fixed number of agents. Thus, in designing and choosing a routing policy, it is important to understand the performance (e.g. mean waiting time) under various routing policies under various parameter settings. An efficient analysis of queueing models is useful in evaluating the performance of various routing policies by changing various parameters such loads and service time variability, as the running time of simulation is too long.

For example, in Chapters 5-7, we have repeatedly seen the effectiveness of resource sharing such as cycle stealing. Specifically, cycle stealing can significantly improve upon the `Dedicated` policy (without cycle stealing) with respect to mean response time. In contact centers, particular customers are often served by particular agents, e.g. due to the skill level of the agents, as discussed above. However, the effectiveness of cycle stealing suggests that training *some* of the agents and allowing them to serve multiple types of customers, can significantly improve mean response time.

In Chapter 7, we have studied a more sophisticated type of resource sharing towards minimizing mean response time. Specifically, we have studied a broad class of threshold-based policies for the Beneficiary-Donor model, and found that a single-threshold policy (the T1 policy) can improves upon cycle stealing, but the improvement of a multi-threshold policy over the T1 policy is only marginal.

In Chapter 7, we have also studied robustness of the threshold-based policies for the Beneficiary-Donor model. Specifically, we find that the T1 policy excels in static robustness (robustness against misestimation of load) but lacks dynamic robustness (robustness against fluctuations in load), while a multi-threshold policy (the ADT policy) excels in both static and dynamic robustness. Such lessons are useful in choosing an appropriate routing policy for a contact center. Specifically, if future loads can be forecasted with high accuracy, the T1 policy is sufficient, as it can provide low mean response time and excels in static robustness; otherwise, the ADT policy is a better choice.

Also, in Chapter 4, we have studied the impact of the variability in service demand and the impact of prioritization in multiserver systems. Specifically, we find that the impact of service demand variability on mean response time can be much higher when the number of servers is small (i.e. at small contact centers). Also, we find that the impact of prioritization on the mean response time of lower priority jobs can be much higher at small contact centers. These lessons suggest that smaller contact centers may not want to prioritize high priority customers if their service demand has very high variability, since it can significantly worsen the mean response time of low priority customers. Also, if the variability of the service demand is high, a smaller contact center may instead want to prioritize those customers whose service times are more likely to be short.

In Chapter 5, we have proposed a size-based task assignment policy with cycle stealing under central queue (`SBCS-CQ`), which combats the impact of service demand variability. Under `SBCS-CQ`, customers whose service times are expected to be short ("short" customers) are routed to the "short" agent, and other customers ("long" customers) are routed to the "long" agent. This prevents the short customers waiting behind a long customer. Further, under `SBCS-CQ`, when there are no long customers to be served, the long agent serves short customers. This increases the utilization of the long agent. In contact centers, service demand may be estimated based on the record of interaction at the (interactive) voice response unit (VRU) and the customer information available at the customer data server, as well as the history of measured service demands.

Chapter 9

Conclusion

In this thesis, new analytical tools for performance evaluation of multiserver systems are developed. The new analytical tools allow us to analyze the performance of various resource allocation policies for multiserver systems for the first time. Our analysis leads to many insights and lessons that are useful, for system designers, in capacity planning and designing resource allocation policies for multiserver systems. In particular, our analysis shows a tremendous benefit of resource allocation policies with resource sharing or cycle stealing, when these resource allocation policies are designed carefully. Further, principles and guidelines are provided for designing these resource allocation policies.

9.1 Analytical tools developed

We have first proposed moment matching algorithms, which can be used to model a multiserver system as a (multidimensional) Markov chain for analyzing the performance of the multiserver system. Also, to prove the minimality of the number of phases used in our moment matching algorithms, the set, $\mathcal{S}^{(n)}$, of distributions that are well-represented by an n-phase acyclic phase type (PH) distribution is characterized.

In developing moment matching algorithms, we have introduced various theoretical concepts such as the normalized moments, r-value, sets $\mathcal{S}^{(n)}$ and $\mathcal{T}^{(n)}$, function ϕ, and the EC distribution, and have proved their properties. These new concepts and their properties are found to be quite useful in developing moment matching algorithms, and they have recently stimulated the research in this area. For example, Bobbio, et al. [22] have recently used the normalized moments to provide *exact* characterizations of sets $\mathcal{S}^{(n)}$ and $\mathcal{S}^{(n)^*}$, where $\mathcal{S}^{(n)}$ is the set of distributions that can be well-represented by an n-phase acyclic PH distribution and $\mathcal{S}^{(n)^*}$ is the set of distributions that can be well-represented by an n-phase acyclic PH distribution *with no mass probability at zero*. Bobbio, et al. use their exact characterization of $\mathcal{S}^{(n)}$ to construct a *minimal* acyclic PH distribution that well-represents any distribution in $\mathcal{PH}_3$.

We have also proposed a new analytical tool, dimensionality reduction (DR), for analyzing multidimensional Markov chains (Markov chains on a multidimensionally infinite state space). The key idea in DR is in approximating an infinite portion of a Markov chain, which often corresponds to the "busy period" of a system, by a collection of PH distributions, which match the first three moments of the duration of the busy period conditioned on how the Markov chain enters and exits from the busy period. As DR involves approximations, we have validated its accuracy against simulation. Overall, we find that the error in DR is quite small: specifically, the error in DR is within 3% (often within 1%) for the first order metric (such as mean delay and mean queue length) and within 7% for the second order metric (such as the second moment of queue length) for a range of parameters. To reduce the computational complexity of DR, we have introduced two approximations of DR: DR-PI and DR-CI. Although these approximations slightly worsen the accuracy, they can significantly reduce the running time.

DR allows us to analyze the performance of a multiserver system whose behavior is modeled as a multidimensional Markov chain. Since it is a nontrivial task to determine whether DR can be applied to a particular multiserver system and how, we formally define a class of multidimensional Markov chains called RFB/GFB processes (recursive foreground-background processes or generalized foreground-background processes) that can be analyzed via DR. The definition of RFB/GFB processes makes it easier to determine whether a given multiserver system can be analyzed via DR, and our analysis of RFB/GFB processes enables one to analyze the multiserver system by simply modeling it as an RFB/GFB process. In fact, many multiserver systems with resource sharing or job prioritization can be modeled as RFB/GFB processes:

- multiserver systems with multiple priority classes (Chapter 4),
- size-based task assignment with cycle stealing under immediate dispatching, `SBCS-ID`, for server farms (Chapter 5),
- size-based task assignment with cycle stealing under central queue, `SBCS-CQ`, for server farms (Chapter 5),
- threshold-based policies for reducing switching costs in cycle stealing (Chapter 6), and
- various threshold-based policies for the Beneficiary-Donor model (Chapter 7).

DR allows us to analyze these multiserver systems with high accuracy for the first time.

Beyond an analysis of multiserver systems, our moment matching algorithms and DR as well as the ideas used in these tools have broad applicability in computer science, engineering, operations research, etc. For example, many optimization problems in stochastic environment can be formulated as Markov decision problems [159] by mapping general (input) probability distributions into PH distributions via our moment matching algorithms. The ideas in DR may then be used to reduce the size of the state space in the Markov decision processes from infinite to finite or from large to small, so that the computational complexity in solving the Markov decision problems is reduced.

In response to many requests, our moment matching algorithms and DR as well as its approximations, DR-PI and DR-CI, as described in this thesis are largely implemented, and the latest implementation is made available at an online code repository:

http://www.cs.cmu.edu/~osogami/code/.

9.2 Lessons learned in the analysis of multiserver systems

DR is then applied to the performance analysis of various multiserver systems, whose analysis involve multidimensional Markov chains. In this thesis, we primarily study fundamental natures of multiserver systems via DR, but DR also has a broad applicability in capacity planning of multiserver systems, as discussed in Chapter 8. Our analysis illuminates principles of the performance of multiserver systems, and provides lessons and guidelines that are useful in designing resource allocation policies in multiserver systems. Note that an advantage of DR is that it allows us to study the performance of multiserver systems under a wide range of environmental conditions. (For example, an arrival process can be modeled as a Markovian arrival process (MAP), service demand can be modeled as a PH distribution, and the performance can be analyzed with high accuracy under a wide range of loads, not just heavy traffic limits.) This allows us to study, for example, the impact of service demand variability on the performance, and robustness of resource allocation policies.

Impact of service demand variability and prioritization in multiserver systems

In the study of "how many servers are best?" in multiserver systems with multiple priority classes, we find that the performance of a multiserver system can be quite different from its single server counterpart. We find that the variability of service demand and prioritization of jobs can have a much larger impact on mean response time in single server (or a-few-server) systems than in multiserver (or many-server) systems. Specifically, we find that the mean response time in single server and multiserver systems can be characterized primarily by three rules of thumb.

(i) A single server system has an advantage over multiserver systems with respect to utilization, and this relative advantage becomes greater at higher load.

(ii) A multiserver system has an advantage over single server systems with respect to reducing the impact of *job size variability* on the mean response time, and this advantage becomes greater at higher load and at larger job size variability.

When jobs are served in FCFS order, the mean response time of single server and multiserver systems can be characterized primarily by the above two rules. However, the above two rules are not sufficient when there are priorities among jobs, since

(iii) A multiserver system has an advantage over a single server system with respect to reducing the impact of *prioritization* on the mean response time of low priority jobs, and this advantage becomes greater when the mean and/or variability of the higher priority job size are larger and/or when the load of the higher priority job is higher.

We find that these three rules well characterize how the optimal number of servers is affected by job size variability and prioritization.

These rules have important implications in designing resource allocation or scheduling policies in multiserver systems. Namely, combating the variability of service demand (e.g. by prioritizing small jobs) is important in designing resource allocation policies for single server (or a-few-server) systems, but prioritizing small jobs is not as effective in improving mean response time in multiserver (or many-server) systems.

We have also studied the impact of the variability in the donor job ("long" job in `SBCS-ID` and `SBCS-CQ`) sizes on the mean response time of the beneficiary jobs ("short" jobs in `SBCS-ID` and `SBCS-CQ`) under various systems with cycle stealing, where the donor server processes the beneficiary jobs when there are no donor jobs. Note that variable donor job sizes imply irregular help from the donor server. We find that the impact of donor job size variability is surprisingly small, but this impact can increase when the beneficiary jobs rely more on the help from the donor server.

Benefit and penalty of cycle stealing

In this thesis, various resource allocation policies with cycle stealing (resource sharing in general) are introduced, and their performance is analyzed via DR. These policies include the threshold-based policy for reducing switching costs in cycle staling (Chapter 6), the task assignment with cycle stealing under immediate dispatching (`SBCS-ID`; Chapter 5), the task assignment with cycle stealing under central queue (`SBCS-CQ`; Chapter 5), and the threshold-based policies for the Beneficiary-Donor model (Chapter 7).

Although cycle stealing provides obvious benefits to the beneficiary jobs ("short" jobs in `SBCS-ID` and `SBCS-CQ`), these benefits come at some cost to the donor jobs ("long" jobs in `SBCS-ID` and `SBCS-CQ`). For example, the donor jobs may suffer from switching costs or may suffer from the nonpreemptive beneficiary jobs stealing cycles of the donor server. DR allows us to quantify the benefit and penalty of cycle stealing under a wide range of parameter settings, and this leads to insights into good resource allocation policies.

Our analysis shows that cycle stealing can provide unbounded benefit overall (to the beneficiary jobs in particular) over the dedicated policy (without cycle stealing). The unbounded benefit stems from the increased stability region under cycle staling, and we have provided the necessary and sufficient conditions for stability under all the resource allocation policies with cycle stealing studied in this thesis. These stability conditions are often nontrivial. For example, under the threshold-based policy for reducing switching costs in cycle staling, we find that the beneficiary side threshold, N_B^{th}, does not affect the stability region, but increasing the donor side threshold, N_D^{th}, increases the stability region.

The mean response time of donor jobs is sensitive to the switching times, and large switching times can result in a significant penalty to the donor jobs. Switching times also diminish the benefit of cycle stealing to the beneficiary jobs. We find that switching times have a significant impact on the mean response time when the beneficiary is overloaded without cycle stealing ($\rho_B \geq 1$), but the mean response time of the beneficiary jobs is far less sensitive when $\rho_B < 1$.

Our analysis shows that long jobs experience only a small penalty under both `SBCS-ID` and `SBCS-CQ`. We also consider the case where the "short" jobs are indistinguishable from the "long" jobs, and the pathological case where the "short" jobs are longer than the "long" jobs. Our analysis shows that `SBCS-ID` and `SBCS-CQ` can still provide an unbounded benefit over `Dedicated` when the load of short jobs is high, but the benefit diminishes as the load becomes lower. In fact, when the load of short jobs is low, cycle stealing (especially, `SBCS-ID`) can worsen the overall mean response time.

Exploring robustness of resource allocation policies

The applicability of DR under a wide range of load conditions allows us to evaluate resource allocation policies not only with respect to mean response time but also with respect to *robustness*. We have introduced two types of robustness measures: static robustness (robustness against misestimation of load) and dynamic robustness (robustness against fluctuations in load). The study of resource allocation policies for the Beneficiary-Donor model with respect to both mean response time and robustness provides us with lessons that are useful in designing good resource allocation policies.

The study of single-threshold allocation policies, T1 and T2, reveals the tradeoff between low mean response time at an estimated load versus static robustness. Specifically, the T2 policy provides good static robustness, but its mean response time is high. On the other hand, the T1 policy optimized for an estimated load provides low mean response time at the estimated load, but it can lead to instability at higher loads (poor static robustness). While it is possible to tune the T1 policy for the higher load, this would degrade the mean response time at the estimated load.

The tradeoff between low mean response time and static robustness in single-threshold allocation policies motivates us to propose the adaptive dual threshold (ADT) policy. We find that the ADT policy not only provides low mean response time but also excels in static robustness. The ADT policy operates as a T1 policy, but its threshold value is self-adapted to the load. This flexibility allows the ADT policy to provide low mean response time under a range of loads. Hence, when the load is not exactly known, the ADT policy is a better choice than the T1 policy. However, we find that the advantage of the ADT policy over the T1 policy with respect to mean response time (under a given load) is quite small (the improvement is usually less than 5%). This may be surprising, since the optimal policy appears to have infinitely many thresholds.

Surprisingly, our analysis shows that poor static robustness does not necessarily imply poor dynamic robustness. For example, we observe that even when the load is fluctuating, the T1 policy, which lacks static robustness, can often provide low mean response time.

9.3 Future directions

The moment matching algorithms and DR proposed in this thesis by no means provide perfect solutions for the performance evaluation of multiserver systems. Below, we discuss interesting future directions for research along the lines of this thesis.

Interesting future directions in moment matching algorithms include the extension to matching more moments and extension to approximating a correlated sequence of random variables. Specifically, our moment matching algorithms match the first three moments of an input distribution, but it may be desirable to match more moments. Also, a PH distribution can be used to model a sequence of independent random variables; however, in some applications, it is desirable to capture the correlation in the sequence of random variables. The Markovian arrival process (MAP) can represent a sequence of correlated PH distributions. However, how we should map a sequence of correlated random variables into a MAP is not well understood (see e.g. [72] for this direction of work).

In designing moment matching algorithms that match more moments, it would be helpful to characterize the set of distributions whose first k-moments can be matched by an n-phase PH distribution for $k > 3$. This will extend our characterization of set $\mathcal{S}^{(n)}$, the set of distributions that are well-represented by an n-phase acyclic PH distribution. It would also be useful to extend this characterization to the *general* PH distribution, including cyclic PH distributions. Such a characterization could be used to prove the minimality of the number of phases used in our closed form solutions in a stronger sense. Although our experience via numerical experiments suggests that allowing cycles in the PH distribution does not help to reduce the number of phases used to match the three moments, a rigorous characterization is not known to date.

Interesting future directions in DR include establishing a theoretical guarantee on the accuracy of DR and extending DR. Although we validate the accuracy of DR against simulation, no theoretical guarantee is proved. It is desirable to establish theoretical bounds on the error in DR. Also, although we apply DR in the analysis of the RFB/GFB processes in this thesis, the idea in DR has a broad applicability beyond the RFB/GFB processes, and DR can be extended in various ways. Below, we list possible extensions of DR. It would be interesting to pursue these possible extensions and apply them to the analysis of more complex multiserver systems.

For example, throughout this thesis, we assume that service demands have PH distributions, but in some cases, they can be extended to general distributions. For example, in the analysis of the threshold-based policy for reducing switching costs in cycle stealing, the donor's service demand and switching back time can be extended to general distributions, as we show in [151, 152]. This extension relies on a (well known) analysis of the busy period in an M/GI/1 queue with a setup time.

In fact, the idea in DR can be applied to the RFB/GFB processes with a quite broad class of background processes. The analysis of the RFB/GFB processes relies on the analysis of the "busy period" in a QBD process via Neuts' algorithm; however, Neuts' algorithm applies to a much broader class of Markov chains, namely the M/G/1 type semi-Markov process. Fully utilizing Neuts' algorithm, DR can be applied to the RFB/GFB process with much less restriction on the background process.

Restrictions on the foreground process can also be relaxed. Our restrictions on the foreground process guarantee that the 1D Markov chains reduced from the RFB/GFB processes are QBD processes. However, the QBD process is not the only Markov chain that allows analytical tractability. For example, the M/G/1 and G/M/1 type processes can also be analyzed efficiently via matrix analytic methods [111]. Thus, the restrictions on the foreground process can be relaxed accordingly.

Further, the GFB process is limited to 2D Markov chains, but this can be extended to higher dimensions, as we extend the FB process to the RFB process. Also, we remark that the multidimensional Markov chain need not have multidimensionally *infinite* state space. DR can reduce a Markov chain that has a *large* number of states in multiple dimensions into a Markov chain that has a smaller number of states in each dimension, since matrix analytic methods and Neuts' algorithm apply to QBD processes with a finite number of levels as well.

Finally, an essential restriction in DR is that the background process must be independent of the foreground process while the background process is in levels $\geq \kappa$, although the foreground process can depend on the background process all the time. Unfortunately, it is not at all clear how DR can be extended to the symmetric case where the foreground and background processes depend on each other all the time. For example, two way cycle stealing, where two servers function as both beneficiaries and donors, does not appear to be modeled as a GFB process, since the foreground and background processes depend on each other all the time. In theory, the analysis of two way cycle stealing can be reduced to a boundary value problem (see Section 3.3), but this leads to a complex expression whose evaluation often experiences numerical instability. It would be interesting to pursue how computational approaches such as DR can be applied to an analysis of two way cycle stealing.

References

[1] I. J. B. F. Adan. *A compensation approach for queueing problems*. PhD thesis, Eindhoven University of Technology, 1991.

[2] I. J. B. F. Adan and J. Wessels. Analysis of the symmetric shortest queue problem. *Communications in Statistics - Stochastic Models*, 6(4):691–713, 1990.

[3] I. J. B. F. Adan, J. Wessels, and W. H. M. Zijm. A compensation approach for two-dimensional Markov processes. *Advances in Applied Probability*, 25(4):783–817, 1993.

[4] H.-S. Ahn, I. Duenyas, and R. Q. Zhang. Optimal control of a flexible server. *Advances in Applied Probability*, 36(1):139–170, 2004.

[5] D. Aldous and L. Shepp. The least variable phase type distribution is Erlang. *Communications in Statistics - Stochastic Models*, 3(3):467–473, 1987.

[6] T. Altiok. On the phase-type approximations of general distributions. *IIE Transactions*, 17(2):110–116, 1985.

[7] A. T. Andersen, M. F. Neuts, and B. F. Nielsen. On the time reversal of Markovian arrival processes. *Communications in Statistics - Stochastic Models*, 20(2):237–260, 2004.

[8] J. Anton. The past, present and future of customer access centers. *International Journal of Service Industry Management*, 11(2):120–130, 2000.

[9] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, M. Kalantar, and S. Krishnakumar. Oceano - SLA based management of a computing utility. In *Proceedings of the IFIP/IEEE Symposium on Integrated Network Management*, pages 855–868, May 2001.

[10] M. Arlitt and T. Jin. A workload characterization study of the 1998 World Cup web site. *IEEE Network*, 14(3):30–37, 2000.

[11] Y. Artsy and R. Finkel. Designing a process migration facility: The Charlotte experience. *IEEE Computer*, 22(9):47–56, 1989.

[12] S. Asmussen and G. Koole. Marked point processes as limits of Markovian arrival streams. *Journal of Applied Probability*, 30(2):365–372, 1993.

[13] B. Awerbuch, Y. Azar, A. Fiat, and T. Leighton. Making commitments in the face of uncertainty: How to pick a winner almost every time. In *Proceedings of the ACM STOC*, pages 519–530, May 1996.

[14] M. Baker, R. Buyya, and D. Laforenza. Grids and grid technologies for wide-area distributed computing. *Journal of Software: Practice and Experience*, 32(15):1437–1466, 2002.

[15] J. E. Baldeschwieler, R. D. Blumofe, and E. A. Brewer. ATLAS: An infrastructure for global computing. In *Proceedings of the ACM SIGOPS European workshop: Systems support for worldwide applications*, pages 165–172, September 1996.

[16] F. Baskett, K. M. Chandy, R. Muntz, and F. Palacios-Gomez. Open, closed and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22(2):248–260, 1975.

[17] S. L. Bell and R. J. Williams. Dynamic scheduling of a system with two parallel servers in heavy traffic with complete resource pooling: Asymptotic optimality of a continuous review threshold policy. *Annals of Applied Probability*, 11(3):608–649, 2001.

[18] F. Berman and R. Wolski. The AppLeS project: A status report. In *Proceedings of the NEC Research Symposium*, May 1997.

[19] D. Bertsimas and D. Nakazato. The distributional Little's law and its applications. *Operations Research*, 43(2):298–310, 1995.

[20] S. N. Bhatt, F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg. On optimal strategies for cycle-stealing in networks of workstations. *IEEE Transactions on Computers*, 46(5):545–557, 1997.

[21] J. P. C. Blanc. Performance analysis and optimization with the power-series algorithm. In *Performance Evaluation of Computer and Communication Systems (Lecture Notes in Computer Science, Vol. 729)*, pages 53–80, 1993.

[22] A. Bobbio, A. Horváth, and M. Telek. Matching three moments with minimal acyclic phase type distributions. *Communications in Statistics - Stochastic Models*, to appear.

[23] A. B. Bondi and J. P. Buzen. The response times of priority classes under preemptive resume in M/G/m queues. In *Proceedings of the ACM SIGMETRICS*, pages 195–201, August 1984.

[24] F. Bonomi and A. Kumar. Adaptive optimal load balancing in a nonhomogeneous multiserver system with a central job scheduler. *IEEE Transactions on Computers*, 39(10):1232–1250, October 1990.

[25] S. Borst, O. Boxma, and M. van Uitert. The asymptotic workload behavior of two coupled queues. *Queueing Systems: Theory and Applications*, 43(1-2):81–102, 2003.

[26] S. Borst, O. J. Boxma, and P. Jelenkovic. Reduced-load equivalence and induced burstiness in GPS queues with long-tailed traffic flows. *Queueing Systems: Theory and Applications*, 43(4):273–306, 2003.

[27] O. J. Boxma, J. W. Cohen, and H. Huffels. Approximations of the mean waiting time in an M/G/s queueing system. *Operations Research*, 27(6):1115–1127, 1979.

[28] L. W. Bright and P. G. Taylor. Calculating the equilibrium distribution in level dependent quasi-birth-and-death processes. *Communications in Statistics - Stochastic Models*, 11(3):497–514, 1995.

[29] L. Brown, N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn, and L. Zhao. Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of the American Statistical Association*, 100(469):36–50, 2005.

[30] R. G. Brown. *Smoothing, Forecasting and Prediction of Discrete Time Series*. Prentice-Hall, 1963.

[31] S. L. Brumelle. Some inequalities for parallel-server queues. *Operations Research*, 19:402–413, 1971.

[32] S. L. Brumelle. A generalization of $L = \lambda W$ to moments of queue length and waiting times. *Operations Research*, 20:1127–1136, 1972.

[33] J. P. Buzen and A. B. Bondi. The response times of priority classes under preemptive resume in M/M/m queues. *Operations Research*, 31(3):456–465, 1983.

[34] J. M. Calabrese. Optimal workload allocation in open queueing networks in multiserver queues. *Management Science*, 38(12):1792–1802, 1992.

[35] M. Calzarossa and G. Serazzi. A characterization of the variation in time of workload arrival patterns. *IEEE Transactions on Computers*, c-34(2):156–162, 1985.

[36] N. Carriero, E. Freeman, D. Gelernter, and D. Kaminsky. Adaptive parallelism and Piranha. *IEEE Computer*, 28(1):40–49, 1995.

[37] X. Chao and C. Scott. Several results on the design of queueing systems. *Operations Research*, 48(6):965–970, 2000.

[38] J. Chase, L. Grit, D. Irwin, J. Moore, and S. Sprenkle. Dynamic virtual clusters in a grid site manager. In *Proceedings of the International Symposium on High Performance Distributed Computing*, pages 90–103, June 2003.

[39] J. S. Chase, D. C. Anderson, P. N. Thankar, and A. M. Vahdat. Managing energy and server resources in hosting centers. In *Proceedings of the ACM Symposium on Operating Systems Principles*, pages 103–116, October 2001.

[40] M. Chetty and R. Buyya. Weaving computational grids: How analogous are they with electrical grids? *IEEE Computing in Science and Engineering*, 4(4):61–71, 2002.

[41] B. Christiansen, P. Cappello, M. F. Ionescu, M. O. Neary, K. Schauser, and D. Wu. Javelin: Internet-based parallel computing using Java. *Concurrency: Practice and Experience*, 9(11):1139–1160, 1997.

[42] J. W. Cohen. Boundary value problems in queueing theory. *Queueing Systems: Theory and Applications*, 3(2):97–128, 1988.

[43] J. W. Cohen. Two-dimensional nearest-neighbor queueing models, a review and an example. In F. Baccelli, A. Jean-Marie, and I. Mitrani, editors, *Quantitative Methods in Parallel Systems*, pages 141–152. Springer-Verlag, 1995.

[44] J. W. Cohen and O. J. Boxma. *Boundary Value Problems in Queueing System Analysis*. North-Holland Publ. Cy., 1983.

[45] D. R. Cox and W. L. Smith. *Queues*. Kluwer Academic Publishers, 1971.

[46] 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, December 1997.

[47] M. E. Crovella, M. S. Taqqu, and A. Bestavros. Heavy-tailed probability distributions in the World Wide Web. In *A Practical Guide To Heavy Tails*, chapter 1, pages 1–23. Chapman & Hall, New York, 1998.

[48] F. Douglis and J. Ousterhout. Transparent process migration: Design alternatives and the Sprite implementation. *Software: Practice and Experience*, 21(8):757–785, 1991.

[49] A. Ephremides, P. Varaiya, and J. Walrand. A simple dynamic routing problem. *IEEE Transactions on Automatic Control*, AC-25(4):690–693, 1980.

[50] A. K. Erlang. Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges. *Elektrotkeknikeren*, 13:5–13, 1917.

[51] G. Fayolle and R. Iasnogorodski. Two coupled processors: The reduction to a Riemann-Hilbert problem. *Zeitschrift fur Wahrscheinlichkeitstheorie und vervandte Gebiete*, 47(3):325–351, 1979.

[52] D. Feitelson, L. Rudolph, U. Schwiegelshohn, K. Sevcik, and P. Wong. Theory and practice in parallel job scheduling. In *Proceedings of IPPS/SPDP '97 Workshop. Lecture Notes in Computer Science, vol. 1291*, pages 1–34, April 1997.

[53] D. G. Feitelson and B. Nitzberg. Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860. In *Proceedings of IPPS '95 Workshop on Job Scheduling Strategies for Parallel Processing*, pages 215–227, April 1995.

[54] R. E. Felderman, E. M. Schooler, and L. Kleinrock. The Benevolent Bandit Laboratory: A testbed for distributed algorithms. *IEEE Journal on Selected Areas in Communications*, 7(2):303–311, 1989.

[55] A. Feldmann and W. Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. *Performance Evaluation*, 32(4):245–279, 1998.

[56] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.

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

[58] N. Gans, G. Koole, and A. Mandelbaum. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing and Service Operations Management*, 5(2):79–141, 2003.

[59] J. Gehring and A. Streit. Robust resource management for metacomputers. In *Proceedings of the International Symposium on High-Performance Distributed Computing*, pages 105–111, August 2000.

[60] D. Green. Lag correlations of approximating departure process for MAP/PH/1 queues. In *Proceedings of the Third International Conference on Matrix Analytic Methods in Stochastic Models*, pages 135–151, 2000.

[61] L. Green. A queueing system with general use and limited use servers. *Operations Research*, 33(1):168–182, 1985.

[62] A. S. Grimshaw and W. A. Wulf. The legion vision of a worldwide virtual computer. *Communications of the ACM*, 40(1):39–45, 1997.

[63] R. Haji and G. Newell. A relation between stationary queue and waiting time distributions. *Journal of Applied Probability*, 8(3):617–620, 1971.

[64] M. Harchol-Balter. Task assignment with unknown duration. *Journal of the ACM*, 49(2):260–288, 2002.

[65] M. Harchol-Balter, M. Crovella, and C. Murta. On choosing a task assignment policy for a distributed server system. *IEEE Journal of Parallel and Distributed Computing*, 59(2):204–228, 1999.

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

[67] M. Harchol-Balter, C. Li, T. Osogami, A. Scheller-Wolf, and M. S. Squillante. Task assignment with cycle stealing under central queue. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, pages 628–637, May 2003.

[68] M. Harchol-Balter, C. Li, T. Osogami, A. Scheller-Wolf, and M. S. Squillante. Task assignment with cycle stealing under immediate dispatch. In *Proceedings of the Fifteenth ACM Symposium on Parallelism in Algorithms and Architectures*, pages 274–285, June 2003.

[69] M. Harchol-Balter, T. Osogami, and A. Scheller-Wolf. Robustness of threshold policies for beneficiary-donor model. *ACM SIGMETRICS Performance Evaluation Review*, 2005 (to appear).

[70] M. Harchol-Balter, T. Osogami, A. Scheller-Wolf, and A. Wierman. Multi-server queueing systems with multiple priority classes. *Queueing Systems: Theory and Applications*, (accepted for publication).

[71] J. M. Harrison. Heavy traffic analysis of a system with parallel servers: Asymptotic optimality of discrete review policies. *Annals of Applied Probability*, 8(3):822–848, 1998.

[72] A. Heindl, K. Mitchell, and A. van de Liefvoort. The correlation region of second-order MAPs with application to queueing network decomposition. In *Proceedings of the Performance TOOLS (Lecture Notes in Computer Science, Vol. 2794)*, pages 237–254, September 2003.

[73] A. Heindl and M. Telek. Output models of MAP/PH/1($/K$) queues for an efficient network decomposition. *Performance Evaluation*, 49(1-4):321–339, 2002.

[74] A. Heindl, Q. Zhang, and E. Smirni. ETAQA truncation models for the MAP/MAP/1 departure process. In *Proceedings of the First International Conference on the Quantitative Evaluation of Systems*, pages 100–109, September 2004.

[75] G. Hooghiemstra, M. Keane, and S. van de Ree. Power series for stationary distributions of coupled processor models. *SIAM Journal on Applied Mathematics*, 48(5):1159–1166, 1988.

[76] A. Horváth and M. Telek. Approximating heavy tailed behavior with phase type distributions. In *Advances in Matrix-Analytic Methods for Stochastic Models*, pages 191–214. Notable Publications, July 2000.

[77] A. Horváth and M. Telek. PhFit: A general phase-type fitting tool. In *Proceedings of Performance TOOLS 2002*, pages 82–91, April 2002.

[78] S. Hotovy, D. Schneider, and T. J. O'Donnell. Analysis of the early workload on the Cornell Theory Center IBM SP2. In *Proceedings of the ACM SIGMETRICS*, pages 272–273, May 1996.

[79] D. L. Iglehart and W. Whitt. Multiple channel queues in heavy traffic, I. *Advances in Applied Probability*, 2:150–177, 1970.

[80] A. Iyengar, E. MacNair, M. S. Squillante, and L. Zhang. A general methodology for characterizing access patterns and analyzing web server performance. In *Proceedings of the IEEE/ACM MASCOTS*, pages 167–174, July 1998.

[81] A. Iyengar, M. S. Squillante, and L. Zhang. Analysis and characterization of large-scale web server access patterns and performance. *World Wide Web*, 2(1-2):85–100, 1999.

[82] V. S. Iyengar, L. H. Trevillyan, and P. Bose. Representative traces for processor models with infinite cache. In *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)*, pages 62–72, February 1996.

[83] J. R. Jackson. Networks of waiting lines. *Operations Research*, 5(4):518–521, 1957.

[84] J. R. Jackson. Jobshop-like queueing systems. *Management Science*, 10(1):131–142, 1963.

[85] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Mesurement, Simulation, and Modeling*. John Wiley & Sons, 1991.

[86] M. A. Johnson. Selecting parameters of phase distributions: Combining nonlinear programming, heuristics, and Erlang distributions. *ORSA Journal on Computing*, 5(1):69–83, 1993.

[87] M. A. Johnson and M. F. Taaffe. An investigation of phase-distribution moment-matching algorithms for use in queueing models. *Queueing Systems: Theory and Applications*, 8(2):129–147, 1991.

[88] M. A. Johnson and M. R. Taaffe. Matching moments to phase distributions: Mixtures of Erlang distributions of common order. *Communications in Statistics - Stochastic Models*, 5(4):711–743, 1989.

[89] M. A. Johnson and M. R. Taaffe. Matching moments to phase distributions: Density function shapes. *Communications in Statistics - Stochastic Models*, 6(2):283–306, 1990.

[90] M. A. Johnson and M. R. Taaffe. Matching moments to phase distributions: Nonlinear programming approaches. *Communications in Statistics - Stochastic Models*, 6(2):259–281, 1990.

[91] E. Jul, H. Levy, N. Hutchinson, and A. Black. Fine-grained mobility in the Emerald system. *ACM Transactions on Computer Systems*, 6(1):109–133, 1988.

[92] S. H. Kang, Y. H. Kim, D. K. Sung, and B. D. Choi. An application of Markovian arrival process (MAP) to modeling superposed ATM cell streams. *IEEE Transactions on Communications*, 50(4):633–642, 2002.

[93] E. P. C. Kao and K. S. Narayanan. Computing steady-state probabilities of a nonpreemptive priority multiserver queue. *Journal on Computing*, 2(3):211–218, 1990.

[94] E. P. C. Kao and K. S. Narayanan. Modelling a multiprocessor system with preemptive priorities. *Management Science*, 37(2):185–197, 1991.

[95] E. P. C. Kao and S. D. Wilson. Analysis of nonpreemptive priority queues with multiple servers and two priority classes. *European Journal of Operational Research*, 118(1):181–193, 1999.

[96] S. Karlin and W. J. Studden. *Tchebycheff Systems: With Applications in Analysis and Statistics*. John Wiley and Sons, 1966.

[97] F. P. Kelly. *Reversibility and Stochastic Networks*. Wiley, 1979.

[98] R. E. A. Khayari, R. Sadre, and B. Haverkort. Fitting World-Wide Web request traces with the EM-algorithm. *Performance Evaluation*, 52(3):175–191, 2003.

[99] J. F. C. Kingman. The single server queue in heavy traffic. *Proceedings of the Cambridge Philosophical Society*, 57:902–904, 1961.

[100] J. F. C. Kingman. Two similar queues in parallel. *Annals of Mathematical Statistics*, 32:1314–1323, 1961.

[101] L. Kleinrock. *Queueing Systems, Volume II: Computer Applications*. A Wiley-Interscience Publication, 1976.

[102] L. Kleinrock. Creating a mathematical theory of computer networks. *Operations Research*, 50(1):125–131, 2002.

[103] A. Konheim, I. Meilijson, and A. Melkman. Processor-sharing of two parallel lines. *Journal of Applied Probability*, 18(4):952–956, 1981.

[104] G. Koole. On the power series algorithm. In O. J. Boxma and G. M. Koole, editors, *Performance Evaluation of Computer and Communication Systems — Solution Methods, CWI Tract 105*, pages 139–155, 1998.

[105] G. Koole and A. Mandelbaum. Queueing models of call centers: An introduction. *Annals of Operations Research*, 113:41–59, 2002.

[106] G. Koole, P. D. Sparaggis, and D. Towsley. Minimizing response times and queue lengths in systems of parallel queues. *Journal of Applied Probability*, 36(4):1185–1193, 1999.

[107] B. Krishnamurthy, C. Wills, and Y. Zhang. On the use and performance of content distribution network. In *Proceedings of the ACM SIGCOMM*, pages 169–182, November 2001.

[108] D. P. Kroese, W. R. W. Scheinhardt, and P. G. Taylor. Spectral properties of the tandem Jackson network, seen as a quasi-birth-and-death process. *Annals of Applied Probability*, 14(4):2057–2089, 2004.

[109] P. Krueger and R. Chawla. The stealth distributed scheduler. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, pages 336–343, May 1991.

[110] M. Krunz and S. K. Tripathi. On the characteristics of VBR MPEG steams. In *Proceedings of the ACM SIGMETRICS*, pages 192–202, June 1997.

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

[112] H. Leemans. *The Two-Class Two-Server Queue with Nonpreemptive Heterogeneous Priority Structures*. PhD thesis, K.U.Leuven, 1998.

[113] W. E. Leland and T. J. Ott. Load-balancing heuristics and process behavior. In *Proceedings of the Performance and ACM SIGMETRICS*, pages 54–69, 1986.

[114] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[115] M. Litzkow, M. Livny, and M. W. Mutka. Condor — A hunter of idle workstations. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, pages 104–111, June 1988.

[116] M. Litzkow and M. Solomon. Supporting checkpointing and process migration outside the UNIX kernel. In *Proceedings of the USENIX Winter Conference*, pages 283–290, January 1992.

[117] D. M. Lucantoni, K. S. Meier-Hellstern, and M. F. Neuts. A single-server queue with server vacations and a class of non-renewal arrival processes. *Advances in Applied Probability*, 22(3):676–705, 1990.

[118] S. Makridakis, S. C. Wheelwright, and R. J. Hyndman. *Forecasting: Methods and Applications*. John Wiley & Sons, 1998.

[119] A. Mandelbaum. Call centers (centres): Research bibliography with abstracts, version 6, December 2004.

[120] A. Mandelbaum and M. I. Reiman. On pooling in queueing networks. *Management Science*, 44(7):971–981, 1998.

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

[122] R. Marie. Calculating equilibrium probabilities for $\lambda(n)/c_k/1/n$ queues. In *Proceedings of the Performance*, pages 117–125, 1980.

[123] V. Mehrotra. Ringing up big business. *OR/MS Today*, 24(4):18–24, 1997.

[124] R. Menich and R. F. Serfozo. Optimality of routing and servicing in dependent parallel processing systems. *Queueing Systems: Theory and Applications*, 9(4):403–418, 1991.

[125] S. Meyn. Sequencing and routing in multiclass queueing networks part I: Feedback regulation. *SIAM Journal on Control Optimization*, 40(3):741–776, 2001.

[126] J. A. V. Mieghem. Dynamic scheduling with convex delay costs: The generalized $c\mu$ rule. *Annals of Applied Probability*, 5(3):809–833, 1995.

[127] D. R. Miller. Steady-state algorithmic analysis of M/M/c two-priority queues with heterogeneous servers. In R. L. Disney and T. J. Ott, editors, *Applied probability - Computer science, The Interface, volume II*, pages 207–222. Birkhauser, 1992.

[128] D. S. Milojicic, W. Zint, A. Dangel, and P. Giese. Task migration on the top of the Mach migration. In *Proceedings of the USENIX Mach Symposium*, pages 273–290, April 1992.

[129] I. Mitrani and P. J. B. King. Multiprocessor systems with preemptive priorities. *Performance Evaluation*, 1(2):118–125, 1981.

[130] P. Molinero-Fernandez, K. Psounis, and B. Prabhakar. Systems with multiple servers under heavy-tailed workloads. Technical Report CENG-2004-02, University of Southern California, March 2004.

[131] P. Morse. *Queues, Inventories, and Maintenance.* John Wiley and Sons, 1958.

[132] R. Nelson. *Probability, Stochastic Processes, and Queueing Theory.* Springer-Verlag, 1995.

[133] M. F. Neuts. Probability distributions of phase type. In *Liber Amicorum Professor Emeritus H. Florin, University of Louvain, Belgium*, pages 173–206, 1975.

[134] M. F. Neuts. Moment formulas for the Markov renewal branching process. *Advances in Applied Probability*, 8:690–711, 1978.

[135] M. F. Neuts. A versatile Markovian point process. *Journal of Applied Probability*, 16(4):764–779, 1979.

[136] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach.* The Johns Hopkins University Press, 1981.

[137] M. F. Neuts. *Structured Stochastic Matrices of M/G/1 Type and Their Applications.* Marcel Dekker, 1989.

[138] B. Ngo and H. Lee. Analysis of a pre-emptive priority M/M/c model with two types of customers and restriction. *Electronics Letters*, 26(15):1190–1192, 1990.

[139] D. Nichols. Using idle workstations in a shared computing environment. In *Proceedings of the ACM Symposium on Operating Systems Principles*, pages 5–12, November 1987.

[140] T. Nishida. Approximate analysis for heterogeneous multiprocessor systems with priority jobs. *Performance Evaluation*, 15(2):77–88, 1992.

[141] S. Nozaki and R. Ross. Approximations in finite capacity multiserver queues with Poisson arrivals. *Journal of Applied Probability*, 15(4):826–834, 1978.

[142] C. A. O'Cinneide. Phase-type distributions and majorization. *Annals of Applied Probability*, 1(3):219 – 227, 1991.

[143] T. Osogami. Analysis of a QBD process that depends on background QBD processes. Technical Report CMU-CS-04-163, School of Computer Science, Carnegie Mellon University, 2004.

[144] T. Osogami and M. Harchol-Balter. Necessary and sufficient conditions for representing general distributions by Coxians. Technical Report CMU-CS-02-178, School of Computer Science, Carnegie Mellon University, 2002.

[145] T. Osogami and M. Harchol-Balter. A closed-form solution for mapping general distributions to minimal PH distributions. In *Proceedings of the Performance TOOLS (Lecture Notes in Computer Science, Vol. 2794)*, pages 200–217, September 2003.

[146] T. Osogami and M. Harchol-Balter. A closed-form solution for mapping general distributions to minimal PH distributions. Technical Report CMU-CS-03-114, School of Computer Science, Carnegie Mellon University, 2003.

[147] T. Osogami and M. Harchol-Balter. Necessary and sufficient conditions for representing general distributions by Coxians. In *Proceedings of the Performance TOOLS (Lecture Notes in Computer Science, Vol. 2794)*, pages 182–199, September 2003.

[148] T. Osogami and M. Harchol-Balter. Closed form solutions for mapping general distributions to quasi-minimal PH distributions. *Performance Evaluation*, 2005 (to appear).

[149] T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. Robustness and performance of threshold-based resource allocation policies. *Working paper*.

[150] T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. Analysis of cycle stealing with switching cost. Technical Report CMU-CS-02-192, School of Computer Science, Carnegie Mellon University, October 2002.

[151] T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. Analysis of cycle stealing with switching cost. In *Proceedings of the ACM SIGMETRICS*, pages 184–195, June 2003.

[152] T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. Analysis of cycle stealing with switching times and thresholds. *Performance Evaluation*, 2005 (to appear).

[153] T. Osogami, M. Harchol-Balter, A. Scheller-Wolf, and L. Zhang. Exploring threshold-based policies for load sharing. In *Proceedings of the 42nd Annual Allerton Conference on Communication, Control, and Computing*, September 2004.

[154] T. Osogami and H. Imai. Classification of various neighborhood operations for the nurse scheduling problem (extended abstract). *Lecture Notes in Computer Science (Proceedings of the 11th Annual International Symposium on Algorithms And Computation (ISAAC 2000))*, 1969:72–83, December 2000.

[155] T. Osogami, A. Wierman, M. Harchol-Balter, and A. Scheller-Wolf. How many servers are best in a dual-priority FCFS system? Technical Report CMU-CS-03-213, School of Computer Science, Carnegie Mellon University, 2004.

[156] T. Osogami, A. Wierman, M. Harchol-Balter, and A. Scheller-Wolf. A recursive analysis technique for multi-dimensionally infinite Markov chains. *ACM SIGMETRICS Performance Evaluation Review*, 32(2):3–5, 2004.

[157] E. W. Parsons and K. C. Sevcik. Implementing multiprocessor scheduling disciplines. In *Proceedings of IPPS/SPDP '97 Workshop. Lecture Notes in Computer Science, vol. 1459*, pages 166–182, April 1997.

[158] V. Paxson and S. Floyd. Wide-area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[159] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics, 1994.

[160] B. M. Rao and M. J. M. Posner. Parallel exponential queues with dependent service rates. *Computers and Operations Research*, 13(6):681–692, 1986.

[161] M. I. Reiman and B. Simon. Open queueing systems in light traffic. *Mathematics of Operations Research*, 14(1):26–59, 1989.

[162] A. Riska, V. Diev, and E. Smirni. An EM-based technique for approximating long-tailed data sets with PH distributions. *Performance Evaluation*, 55(1-2):147–164, 2004.

[163] A. L. Rosenberg. Guidelines for data-parallel cycle-stealing in network of workstations, I. On maximizing expected output. *Journal of Parallel and Distributed Computing*, 59(1):31–53, 1999.

[164] A. L. Rosenberg. Guidelines for data-parallel cycle-stealing in network of workstations, II: On maximizing guranteed output. *International Journal of Foundations of Computer Science*, 11(1):183–204, 2000.

[165] A. L. Rosenberg. Optimal schedules for cycle-stealing in a network of workstations with a bag-of-tasks workload. *IEEE Transactions on Parallel and Distributed Systems*, 13(2):179–191, 2002.

[166] K. W. Ross and D. D. Yao. Optimal load balancing and scheduling in a distributed computer system. *Journal of the ACM*, 38(3):676–690, July 1991.

[167] S. M. Ross. *Stochastic Processes*. John Wiley & Sons, Inc., second edition, 1996.

[168] S. M. Ross. *Simulation*. Academic Press, 1997.

[169] S. M. Ross. *Introduction to Probability Models*. Academic Press, seventh edition, 2000.

[170] K. D. Ryu, J. K. Hollingsworth, and P. J. Keleher. Mechanisms and policies for supporting fine-grained cycle stealing. In *Proceedings of the ACM/IEEE Conference on Supercomputing*, pages CD–ROM, May 1999.

[171] C. H. Sauer and K. M. Chandy. Approximate analysis of central server models. *IBM Journal of Research and Development*, 19(3):301–313, 1975.

[172] A. Scheller-Wolf. Necessary and sufficient conditions for delay moments in FIFO multiserver queues with an application comparing s slow servers with one fast one. *Operations Research*, 51(5):748–758, 2003.

[173] L. Schmickler. MEDA: Mixed Erlang distributions as phase-type representations of empirical distribution functions. *Communications in Statistics - Stochastic Models*, 8(1):131–156, 1992.

[174] B. Schroeder and M. Harchol-Balter. Evaluation of task assignment policies for supercomputing servers: The case for load unbalancing and fairness. In *Proceedings of the Ninth IEEE International Symposium on High Performance Distributed Computing*, pages 211–219, 2000.

[175] G. H. Shahkar and H. R. Tareghian. Designing a production line through optimization of M/G/c using simulation. *Mathematical Engineering in Industry*, 8(2):123–126, 2001.

[176] R. Shumsky. Approximation and analysis of a call center with flexible and specialized servers. *OR Spectrum*, 26(3):307–330, 2004.

[177] A. Sleptchenko. Multi-class, multi-server queues with non-preemptive priorities. Technical Report 2003-016, EURANDOM, Eindhoven University of Technology, 2003.

[178] A. Sleptchenko, A. van Harten, and M. van der Heijden. An exact solution for the state probabilities of the multi-class, multi-server queue with preemptive priorities, 2003 – Manuscript.

[179] P. Smith and N. C. Hutchinson. Heterogeneous process migration: The Tui system. *Software: Practice and Experience*, 28(6):611–638, 1998.

[180] M. S. Squillante and R. D. Nelson. Analysis of task migration in shared-memory multiprocessors. In *Proceedings of the ACM SIGMETRICS*, pages 143–155, May 1991.

[181] M. S. Squillante, C. H. Xia, D. D. Yao, and L. Zhang. Threshold-based priority policies for parallel-server systems with affinity scheduling. In *Proceedings of the IEEE American Control Conference*, pages 2992–2999, June 2001.

[182] M. S. Squillante, C. H. Xia, and L. Zhang. Optimal scheduling in queuing network models of high-volume commercial web sites. *Performance Evaluation*, 47(4):223–242, 2002.

[183] M. S. Squillante, D. D. Yao, and L. Zhang. Analysis of job arrival patterns and parallel scheduling performance. *Performance Evaluation*, 36-37(1-4):137–163, 1999.

[184] M. S. Squillante, D. D. Yao, and L. Zhang. Web traffic modeling and web server performance analysis. In *Proceedings of the IEEE Conference on Decision and Control*, December 1999.

[185] D. A. Stanford and W. K. Grassmann. The bilingual server system: A queueing model featuring fully and partially qualified servers. *INFOR*, 31(4):261–277, 1993.

[186] D. A. Stanford and W. K. Grassmann. Bilingual server call centers. In D. McDonald and S. Turner, editors, *Analysis of Communication Networks: Call Centers, Traffic and Performance*. American Mathematical Society, 2000.

[187] D. Starobinski and M. Sidi. Modeling and analysis of power-tail distributions via classical teletraffic methods. *Queueing Systems: Theory and Applications*, 36(1-3):243–267, 2000.

[188] S. Stidham. On the optimality of single-server queueing systems. *Operations Research*, 18(4):708–732, 1970.

[189] S. Stidham. Analysis, design, and control of queueing systems. *Operations Research*, 50(1):197–216, 2002.

[190] M. Telek and A. Heindl. Matching moments for acyclic discrete and continuous phase-type distributions of second order. *International Journal of Simulation*, 3(3-4):47–57, 2003.

[191] G. Thiel. Locus operating systems, a transparent system. *Computer Communications*, 16(6):336–346, 1991.

[192] D. Towsley, P. D. Sparaggis, and C. G. Cassandras. Optimal routing and buffer allocation for a class of finite capacity queueing systems. *IEEE Transactions on Automatic Control*, 37(9):1447–1451, 1992.

[193] K. S. Trivedi. *Probability & Statistics with Reliability, Queueing, and Computer Science Applications*. Prentice Hall, 1982.

[194] M. van der Heijden, A. van Harten, and A. Sleptchenko. Approximations for Markovian multi-class queues with preemptive priorities. *Operations Research Letters*, 32(3):273–282, 2004.

[195] N. M. van Dijk. *Queueing Networks and Product Forms: A Systems Approach*. Wiley, 1993.

[196] G. J. van Houtum. *New approaches for multi-dimensional queueing systems*. PhD thesis, Eindhoven University of Technology, 1995.

[197] G. J. van Houtum and W. H. M. Zijm. Computational procedures for stochastic multi-echelon production systems. *International Journal of Production Economics*, 23:223–237, 1991.

[198] G. J. van Houtum and W. H. M. Zijm. Incomplete convolutions in production and inventory models. *OR Spektrum*, 19(2):97–107, 1997.

[199] L. A. Wald and S. Schwarz. The 1999 southern California seismic network bulletin. *Seismological Research Letters*, 71(4), 2000.

[200] W. Whitt. Approximating a point process by a renewal process: Two basic methods. *Operations Research*, 30(1):125–147, 1982.

[201] W. Whitt. Deciding which queue to join: Some counterexamples. *Operations Research*, 34(1):226–244, 1986.

[202] W. Whitt. A review of $L = \lambda W$ and extensions. *Queueing Systems: Theory and Applications*, 9(3):235–268, 1991.

[203] W. Whitt. The impact of a heavy-tailed service-time distribution upon the M/GI/s waiting-time distribution. *Queueing Systems: Theory and Applications*, 36(1-3):71–87, 2000.

[204] A. Wierman, T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. Analyzing the effect of prioritized background tasks in multiserver systems. Technical Report CMU-CS-03-213, School of Computer Science, Carnegie Mellon University, 2004.

[205] R. J. Williams. On dynamic scheduling of a parallel server system with complete resource pooling. In D. McDonald and S. Turner, editors, *Analysis of Communication Networks: Call Centers, Traffic and Performance*. American Mathematical Society, 2000.

[206] W. Willinger, M. S. Taqqu, W. E. Leland, and D. V. Wilson. Self-similarity in high-speed packet traffic: Analysis and modeling of Ethernet traffic measurements. *Statistical Science*, 10(1):67–85, 1995.

[207] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, 1997.

[208] K. Windisch, V. Lo, B. Nitzberg, D. Feitelson, and R. Moore. A comparison of workload traces from two production parallel machines. In *Proceedings of the Symposium on the Frontiers of Massively Parallel Computation*, pages 319–326, May 1996.

[209] W. Winston. Optimality of the shortest line discipline. *Journal of Applied Probability*, 14(1):181–189, 1977.

[210] R. W. Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice Hall, 1989.

[211] E. Zayas. Attacking the process migration bottleneck. In *Proceedings of the ACM Symposium on Operating Systems Principles*, pages 13–24, November 1987.

Appendix A

Proofs

Proof of Lemma 3

We prove the lemma by induction on n.

Base case ($n = 1$): Any one-phase PH distribution is a mixture of O and an exponential distribution, and the r-value is always $\frac{3}{2}$.

Inductive case: Suppose that the lemma holds for $n \leq k$. We show that the lemma holds for $n = k + 1$ as well.

Consider any $(k + 1)$-phase acyclic PH distribution, G, which is not an Erlang distribution. We first show that there exists a PH distribution, F_1, with $r^{F_1} \leq r^G$ such that F_1 is the convolution of an exponential distribution, X, and a k-phase PH distribution, Y. The key idea is to see any PH distribution as a mixture of PH distributions whose initial probability vectors, $\vec{\tau}$, are base vectors. For example, the three-phase PH distribution, G, in Figure 2.1, can be seen as a mixture of O and the three 3-phase PH distribution, G_i ($i = 1, 2, 3$), whose parameters are $\vec{\tau}^{G_1} = (1, 0, 0)$, $\vec{\tau}^{G_2} = (0, 1, 0)$, $\vec{\tau}^{G_3} = (0, 0, 1)$, and $\mathbf{T^{G_1}} = \mathbf{T^{G_2}} = \mathbf{T^{G_3}} = \mathbf{T^{G}}$. Proposition 5 and Lemma 2 imply that there exists $i \in \{1, 2, 3\}$ such that $r^{G_i} \leq r^G$. Without loss of generality, let $r^{G_1} \leq r^G$ and let $F_1 = G_1$; thus, $r^{F_1} \leq r^G$. Note that F_1 is the convolution of an exponential distribution, X, and a k-phase PH distribution, Y.

Next we show that if F_1 is not an Erlang distribution, then there exists a PH distribution, F_2, with no greater r-value (i.e. $r^{F_2} \leq r^{F_1}$). Let Z be a mixture of O and an Erlang-k distribution, E_k, (i.e. $Z(\cdot) = pO(\cdot) + (1-p)E_k(\cdot)$), where p is chosen such that $\mu_1^Z = \mu_1^Y$ and $m_2^Z = m_2^Y$. There always exists such a Z, since the Erlang-k distribution has the least m_2 among all the PH distributions (in particular $m_2^{E_k} \leq m_2^Y$) and m_2 is an increasing function of p ($m_2^Z = m_2^{E_k}/(1-p)$). Also, observe that, by Proposition 5 and the inductive hypothesis, $r^Z \leq r^Y$. Let F_2 be the convolution of X and Z, i.e. $F_2(\cdot) = X(\cdot) * Z(\cdot)$. We prove that $r^{F_2} \leq r^{F_1}$. Let $y = \mu_1^Y/\mu_1^X$. Then,

$$\begin{aligned} r^{F_1} &= \frac{\left(r^X (m_2^X)^2 + 3m_2^X y + 3m_2^Y y^2 + r^Y (m_2^Y)^2 y^3\right)(1+y)}{\left(m_2^X + 2y + m_2^Y y^2\right)^2} \\ &\geq \frac{\left(r^X (m_2^X)^2 + 3m_2^X y + 3m_2^Z y^2 + r^Z (m_2^Z)^2 y^3\right)(1+y)}{\left(m_2^X + 2y + m_2^Z y^2\right)^2} = r^{F_2}, \end{aligned}$$

where the inequality follows from $\mu_1^Z = \mu_1^Y$, $m_2^Z = m_2^Y$, and $r^Z \leq r^Y$.

Finally, we show that an Erlang distribution has the least r-value. F_2 is the convolution of X and Z, and it can also be seen as a mixture of X and a distribution, F_3, where $F_3(\cdot) = X(\cdot) * E_k(\cdot)$. Thus, by Lemma 2, at least one of $r^X \le r^{F_2}$ and $r^{F_3} \le r^{F_2}$ holds. When $r^X \le r^{F_2}$, the r-value of the Erlang-$(k+1)$ distribution, $r^{E_{k+1}}$, is smaller than r^{F_2}, since $r^{E_{k+1}} < r^X \le r^{F_2}$. When $r^X > r^{F_2}$ (and hence $r^{F_3} \le r^{F_2}$), $r^{E_{k+1}} \le r^{F_3} \le r^{F_2}$ can be proved by showing that r^{F_3} is minimized when $\mu_1^X = \mu_1^{E_k}/k$. Let $y = \mu_1^X/\mu_1^{E_k}$. Then,

$$r^{F_3} = \frac{\left(r^{E_k}(m_2^{E_k})^2 + 3m_2^{E_k}y + 6y^2 + 6y^3\right)(1+y)}{\left(m_2^{E_k} + 2y + 2y^2\right)},$$

where $r^{E_k} = (k+2)/(k+1)$ and $m_2^{E_k} = (k+1)/k$. Therefore,

$$\frac{\partial r^{F_3}}{\partial y} = \frac{2k(k+1)(6ky^2 + 6ky + k - 1)}{\left(\frac{k+1}{k} + 2y + 2y^2\right)}\left(y - \frac{1}{k}\right).$$

Since $k > 1$, r^{F_3} is minimized at $y = 1/k$. ■

Proof of Lemma 5

It is easy to check, by substitution, that conditions (2.10)-(2.12) are satisfied. It is easy to see $0 < p < 1$, since $m_3^G < 2m_2^G - 1$. Also, $m_2^X \ge \frac{N+2}{N+1}$ implies $w > 0$. Thus, it suffices to prove that condition (2.9) is satisfied.

We first consider the first inequality of condition (2.9). The assumption on r^G in the lemma gives

$$\frac{3}{2} - r^G \le \frac{3}{2} - \frac{(N+1)m_2^G + (N+4)}{2(N+2)} = \frac{N+1}{N+2} \cdot \frac{2 - m_2^G}{2}.$$

Therefore, since $\frac{3}{2} > r^G$, it follows that

$$m_2^X = 2w = \frac{2 - m_2^G}{2} \cdot \frac{1}{\frac{3}{2} - r^G} \ge \frac{N+2}{N+1}.$$

We next consider the second inequality of condition (2.9). We begin by bounding the range of m_2^G for G considered in the lemma. Condition $G \in \widehat{\mathcal{L}}_N$ implies $m_2^G \ge \frac{N+2}{N+1}$. Also, if $m_2^G > 2$, then by the assumption on r^G in lemma,

$$r^G \ge \frac{(N+1)m_2^G + (N+4)}{2(N+2)} > \frac{2(N+1) + (N+4)}{2(N+2)} = \frac{3}{2}.$$

This contradicts $r^G < \frac{3}{2}$. Thus, $m_2^G \le 2$. So far, we derived the range of m_2^G as $\frac{N+2}{N+1} < m_2^G \le 2$.

We prove $m_2^X < \frac{N+1}{N}$ in two cases: (i) $\frac{N+1}{N} \le m_2^G \le 2$ and (ii) $\frac{N+2}{N+1} \le m_2^G < \frac{N+1}{N}$. (i) When $\frac{N+1}{N} \le m_2^G \le 2$,

$$m_2^X = \frac{2 - m_2^G}{2\left(\frac{3}{2} - r^G\right)}. < \frac{2 - \frac{N+1}{N}}{2\left(\frac{3}{2} - \frac{N+2}{N+1}\right)} = \frac{N+1}{N}.$$

The inequality follows from $m_2^G < \frac{N+1}{N}$ and $r^G \le \frac{N+2}{N+1}$. (ii) When $\frac{N+2}{N+1} \le m_2^G < \frac{N+1}{N}$,

$$m_2^X = \frac{2 - m_2^G}{2\left(\frac{3}{2} - r^G\right)}. < \frac{2 - m_2^G}{2\left(\frac{3}{2} - \frac{2m_2^G - 1}{m_2^G}\right)} = m_2^G < \frac{N+1}{N}.$$

The inequality follows from

$$r^G = \frac{m_3^G}{m_2^G} < \frac{2m_2^G - 1}{m_2^G},$$

which follows from $G \in \widehat{\mathcal{L}}_N$. ■

Proof of Lemma 6

For each case, it is easy to check, by substitution, that conditions (2.14)-(2.16) are satisfied. Below, we prove condition (2.13) and $z > 0$.

We begin with the first case, where $m_2^G = 2$. It is easy to see (2.13) is true if $z > 0$, since

$$m_2^X = 2(1+z) > 2 > \frac{N+2}{N+1}.$$

Further, $z > 0$ if $2\frac{N+3}{N+2} < m_3^G < 3$, which is true by $G \in \mathcal{L}_N$, $r^G < \frac{3}{2}$, and $m_2^G = 2$.

Below, we consider the second case, where $m_2^G \neq 2$. We first prove $z > 0$ by showing that z is the larger solution of the two solutions of a quadratic equation that has a unique positive solution. Observe that

$$\begin{aligned}
& m_3^G(m_2^X + 2z + 2z^2)(1+z) = m_2^X m_3^X + 3m_2^X z + 6z^2 + 6z^3 \quad \text{(by (2.16))} \\
\iff\quad & m_3^G m_2^G (1+z)^3 = \frac{N+3}{N+2}(m_2^X)^2 + 3z(m_2^X + 2z + 2z^2) \quad \text{(by (2.14) and (2.15))} \\
\iff\quad & m_3^G m_2^G (1+z)^3 = \frac{N+3}{N+2}(1+z)^2\left(m_2^G(1+z) - 2z\right)^2 + 3z(1+z)^2 m_2^G \quad \text{(by (2.15))} \\
\iff\quad & m_3^G m_2^G (1+z) = \frac{N+3}{N+2}\left(m_2^G(1+z) - 2z\right)^2 + 3z m_2^G
\end{aligned}$$

Thus, z is a solution of the following quadratic equation: $f(z) = 0$, where

$$f(z) \equiv \frac{N+3}{N+2}(m_2^G - 2)^2 z^2 - m_2^G\left((m_3^G - 3) - 2\frac{N+3}{N+2}(m_2^G - 2)\right)z - (m_2^G)^2\left(r^G - \frac{N+3}{N+2}\right).$$

Since the coefficient of the leading term, $\frac{N+3}{N+2}(m_2^G - 2)^2$, is positive and $f(0) < 0$, there exists a unique positive solution of $f(z) = 0$.

Second, we show $m_2^X \ge \frac{N+2}{N+1}$. We consider two cases: (i) $m_2^G \ge 2$ and (ii) $m_2^G < 2$. Case (i) is easy to show. Suppose $m_2^G \ge 2$. Observe that by (2.15),

$$m_2^X = z\left((m_2^G - 2)z + 2(m_2^G - 1)\right) + m_2^G.$$

Thus, if $m_2^G \ge 2$, then $m_2^X \ge m_2^G \ge \frac{N+2}{N+1}$. Below, we consider case (ii).

Suppose $m_2^G < 2$. Observe that

$$m_2^X = -(2 - m_2^G)z^2 + 2(m_2^G - 1)z + m_2^G,$$

again by (2.15). Thus, $m_2^X \geq \frac{N+2}{N+1}$ iff $0 < z \leq z^*$, where z^* is a larger solution, x, of the following quadratic equation: $\chi(x) = 0$, where

$$\chi(x) = -(2 - m_2^G)x^2 + 2(m_2^G - 1)x + m_2^G - \frac{N+2}{N+1}.$$

That is,

$$z^* \equiv \frac{m_2^G - 1 + \sqrt{\frac{N+2}{N+1}m_2^G - \frac{N+3}{N+1}}}{2 - m_2^G}. \tag{A.1}$$

Thus, it suffices to show $f(z^*) \geq 0$. Since $\chi(z^*) = 0$, we obtain $(z^*)^2$ as a linear function of z^*:

$$(z^*)^2 = \frac{2(m_2^G - 1)z^* + m_2^G - \frac{N+2}{N+1}}{2 - m_2^G}.$$

By substituting this $(z^*)^2$ into the expression for $f(z^*)$, we obtain

$$\begin{aligned}
f(z^*) &= \frac{N+3}{N+2}(2 - m_2^G)^2 \left(\frac{2(m_2^G - 1)z^* + m_2^G - \frac{N+2}{N+1}}{2 - m_2^G}\right) \\
&\quad - m_2^G \left(2\frac{N+3}{N+2}(2 - m_2^G) - (3 - m_3^G)\right) z^* - (m_2^G)^2 \left(r^G - \frac{N+3}{N+2}\right) \\
&= \left(3m_2^G - 2\frac{N+3}{N+2}(2 - m_2^G) - m_2^G m_3^G\right) z^* + 2\frac{N+3}{N+2}m_2^G - \frac{N+3}{N+1}(2 - m_2^G) - m_2^G m_3^G \\
&> \left(3m_2^G - 2\frac{N+3}{N+2}(2 - m_2^G) - (m_2^G)^2 \left(\frac{(N+1)m_2^G + (N+4)}{2(N+2)}\right)\right) z^* \\
&\quad + 2\frac{N+3}{N+2}m_2^G - \frac{N+3}{N+1}(2 - m_2^G) - (m_2^G)^2 \left(\frac{(N+1)m_2^G + (N+4)}{2(N+2)}\right) \\
&= \frac{2 - m_2^G}{2(N+2)} \Big((N+1)(m_2^G)^2 + 3(N+2)m_2^G - 4(N+3)\Big) z^* \\
&\quad - \frac{(N+1)m_2^G - (N+2)}{2(N+1)(N+2)} \Big((N+1)(m_2^G)^2 + (2N+6)m_2^G - 4(N+3)\Big)
\end{aligned}$$

where the inequality follows from the assumption on r^G in the lemma. By substituting (A.1) into the last expression, we obtain

$$f(z^*) = g(m_2^G) + h(m_2^G)\sqrt{\frac{(N+2)m_2^G - (N+3)}{N+1}},$$

where

$$\begin{aligned}
g(m_2^G) &\equiv \frac{(N+1)^2(m_2^G)^2 - (N-3)(N+2)m_2^G - 4(N+3)}{2(N+1)(N+2)} \\
h(m_2^G) &\equiv \frac{(N+1)(m_2^G)^2 + 3(N+2)m_2^G - 4(N+3)}{2(N+2)}.
\end{aligned}$$

Since

$$g'(m_2^G) = 2(N+1)^2 \left(m_2^G - \frac{N+2}{N+1} \right) + (N+2)(N+5) > 0$$

for $\frac{N+2}{N+1} \le m_2^G < 2$, $g(m_2^G)$ and $h(m_2^G)$ are increasing functions of m_2^G in the range of $\frac{N+2}{N+1} \le m_2^G < 2$. Since

$$g(\frac{N+2}{N+1}) = \frac{2}{(N+1)^2(N+2)} > 0 \quad \text{and} \quad h(\frac{N+2}{N+1}) = \frac{2}{N^2+3N+2} > 0,$$

we have $g(m_2^G) \ge 0$ and $h(m_2^G) \ge 0$ for $\frac{N+2}{N+1} \le m_2^G < 2$. This implies $f(z^*) \ge 0$. ■

Proof of Theorem 8

We will prove that if the distribution of K is in $\mathcal{T}^{(n)}$, the distribution of T is in $\mathcal{T}^{(n)}$ for $n \ge 2$. When $\rho = 0$, $B = K$; thus, the theorem follows immediately.

Below, we assume $0 < \rho < 1$. The first three moments of T are

$$\begin{aligned}
\mathsf{E}[T] &= \frac{\mathsf{E}[K]}{1-\rho} \\
\mathsf{E}\left[T^2\right] &= \frac{\mathsf{E}[K^2]}{(1-\rho)^2} + \frac{\rho}{(1-\rho)^3}\frac{\mathsf{E}[K]\,\mathsf{E}[G^2]}{\mathsf{E}[G]} \\
\mathsf{E}\left[T^3\right] &= \frac{\mathsf{E}[K^3]}{(1-\rho)^3} + \frac{3\rho}{(1-\rho)^4}\frac{\mathsf{E}[K^2]\,\mathsf{E}[G^2]}{\mathsf{E}[G]} + \frac{\rho}{(1-\rho)^4}\frac{\mathsf{E}[K]\,\mathsf{E}[G^3]}{\mathsf{E}[G]} + \frac{3\rho^2}{(1-\rho)^5}\frac{\mathsf{E}[K]\,(\mathsf{E}[G^2])^2}{(\mathsf{E}[G])^2},
\end{aligned}$$

where ρ is the load of the M/G/1 queue. Let $t_2 \equiv \frac{\mathsf{E}[T^2]}{(\mathsf{E}[T])^2}$ and $t_3 \equiv \frac{\mathsf{E}[T^3]}{(\mathsf{E}[T])^3}$. Observe that if $t_2 > \frac{n+1}{n}$ and $t_3 - \frac{n+2}{n+1}t_2^2 \ge 0$, then the distribution of T is in $\mathcal{T}^{(n)}$.

Thus, it suffices to prove that $t_2 > \frac{n+1}{n}$ and $t_3 - \frac{n+2}{n+1}t_2^2 \ge 0$, given that the distribution of K is in $\mathcal{T}^{(n)}$. First, $t_2 > \frac{n+1}{n}$ is easy to prove:

$$t_2 = k_2 + \frac{\rho}{1-\rho} g_2 \frac{\mathsf{E}[G]}{\mathsf{E}[K]} > k_2 > \frac{n+1}{n},$$

where $k_2 = \frac{\mathsf{E}[K^2]}{(\mathsf{E}[K])^2}$ and $g_2 = \frac{\mathsf{E}[G^2]}{(\mathsf{E}[G])^2}$. Below, we prove that $t_3 - \frac{n+2}{n+1}t_2^2 \ge 0$. Note that

$$t_3 = k_3 + \frac{3\rho}{1-\rho} g_2 k_2 \frac{\mathsf{E}[G]}{\mathsf{E}[K]} + \frac{\rho}{1-\rho} g_3 \left(\frac{\mathsf{E}[G]}{\mathsf{E}[K]}\right)^2 + \frac{3\rho^2}{(1-\rho)^2} g_2^2 \left(\frac{\mathsf{E}[G]}{\mathsf{E}[K]}\right)^2,$$

where $k_3 = \frac{\mathsf{E}[K^3]}{(\mathsf{E}[K])^3}$ and $g_3 = \frac{\mathsf{E}[G^3]}{(\mathsf{E}[G])^3}$. Thus,

$$\begin{aligned}
& t_3 - \frac{n+2}{n+1}t_2^2 \\
&= k_3 - \frac{n+2}{n+1}k_2^2 + \frac{n-1}{n+1}\frac{\rho}{1-\rho} g_2 k_2 \frac{\mathsf{E}[G]}{\mathsf{E}[K]} + \frac{\rho}{1-\rho} g_3 \left(\frac{\mathsf{E}[G]}{\mathsf{E}[K]}\right)^2 + \frac{2n+1}{n+1}\frac{\rho^2}{(1-\rho)^2} g_2^2 \left(\frac{\mathsf{E}[G]}{\mathsf{E}[K]}\right)^2 \\
&> k_3 - \frac{n+2}{n+1}k_2^2 \ge 0.
\end{aligned}$$

■

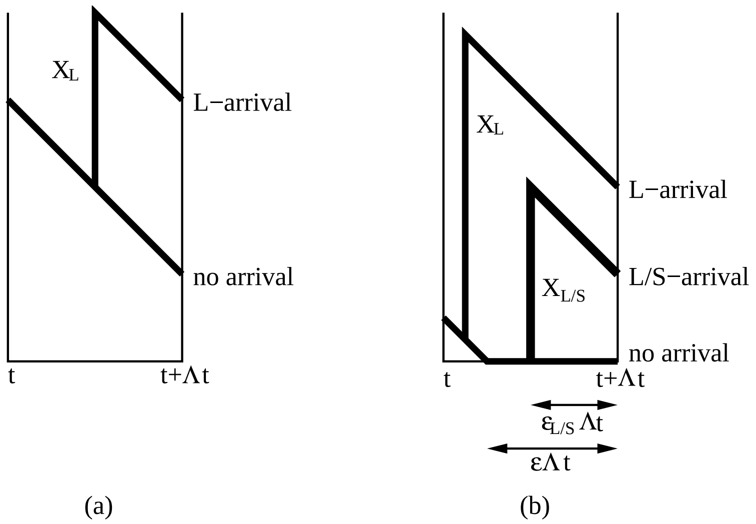


Figure A.1: *Virtual waiting time analysis: relationship between* $W(t)$ *and* $W(t+\Delta t)$*, when (a)* $W(t) \geq \Delta t$ *and (b)* $0 \leq W(t) < \Delta t$.

Proof of Theorem 9

Let $W(t)$ be the virtual waiting time for the queue of long jobs at time t. That is, a long job arriving at time t would wait $W(t)$ before it starts being processed. By the PASTA (Poisson arrival sees the time average) principle, the virtual waiting time W is equal in distribution to the waiting time. Therefore, it suffices to analyze the virtual waiting time W for the derivation of the response time of long jobs.

The following steps allow us to obtain the moments of $W = \lim_{t\to\infty} W(t)$:

(i) Set up a differential equation for $\widetilde{W}(t,s)$, the Laplace transform of $W(t)$.

(ii) Let $t \to \infty$; then, $\frac{d\widetilde{W}(t,s)}{dt} \to 0$, because the queue reaches the stationary state. Now, $\widetilde{W}(s)$ is obtained as a function of π_0.

(iii) Evaluate $\widetilde{W}(s=0)$ to obtain π_0.

(iv) Differentiate $\widetilde{W}(s)$ to obtain moments of W.

(i) We first set up a differential equation for $\widetilde{W}(t,s)$. For this purpose, we carefully examine the relationship between $W(t)$ and $W(t+\Delta t)$. First, suppose $W(t) \geq \Delta t$ (see Figure A.1(a)). Since the long server is always busy between t and $t+\Delta t$, only long jobs could arrive at the queue. Since the arrival process is Poisson with rate λ_L, the probability of having a job arrival in time Δt is $\lambda_L \Delta t + o(\Delta t)$. Any such arrival will have service time X_L. Therefore,

$$W(t+\Delta t) = \begin{cases} W(t) - \Delta t & \text{w/ prob. } 1 - \lambda_L \Delta t + o(\Delta t), \\ W(t) + X_L - \Delta t & \text{w/ prob. } \lambda_L \Delta t + o(\Delta t), \\ \text{something else} & \text{w/ prob. } o(\Delta). \end{cases}$$

Next, suppose $0 \le W(t) < \Delta t$ (see Figure A.1). Let a random variable ϵ be the fraction of time that the long server was idle during $(t, t+\Delta t)$ given that there were no arrivals during this interval. Let a random variable ϵ_L be the fraction of time that the long server was busy during this interval given that there was a long job arrival. Let a random variable ϵ_S be the fraction of time that the long server was busy during the interval giving that there was a short job arrival. Then,

$$W(t+\Delta t) \quad = \quad \begin{cases} 0 & \text{w/ prob. } 1-(\lambda_L+\lambda_S\epsilon)\Delta t + o(\Delta t), \\ W(t)+X_L-\epsilon_L\Delta t & \text{w/ prob. } \lambda_L\Delta t + o(\Delta t), \\ X_S-\epsilon_S\Delta t & \text{w/ prob. } \lambda_S\epsilon\Delta t + o(\Delta t), \\ \text{something else} & \text{w/ prob. } o(\Delta). \end{cases}$$

Note that $0 \le \epsilon, \epsilon_L, \epsilon_S \le 1$.

Based on the above observation, the Laplace transform $\widetilde{W}(t+\Delta t, s)$ of $W(t+\Delta t)$ is obtained as follows:

$$\begin{aligned} \widetilde{W}(t+\Delta t, s) \quad &\equiv \quad \mathsf{E}\left[e^{-sW(t+\Delta t)}\right] \\ &= \quad \int_{x=0}^{\infty} \mathsf{E}\left[e^{-sW(t+\Delta t)} | W(t)=x\right] d\Pr(W(t) \le x) \\ &= \quad \left(1+(s-\lambda_L+\lambda_L\widetilde{X}_L(s))\Delta t\right)\widetilde{W}(t,s) + \int_{x=0^+}^{\Delta t} O(\Delta t) d\Pr(W(t) \le x) \\ &\quad + \left(-\lambda_S+\widetilde{X}_S(s)\lambda_S - s\right)\Delta t \Pr(W(t)=0) + o(\Delta t). \end{aligned}$$

Thus, we obtain the next formula.

$$\begin{aligned} \frac{\widetilde{W}(t+\Delta t, s)-\widetilde{W}(t,s)}{\Delta t} \quad &= \quad \left(s-\lambda_L+\lambda_L\widetilde{X}_L(s)\right)\widetilde{W}(t,s) + \int_{x=0^+}^{\Delta t} O(1) d\Pr(W(t) \le x) \\ &\quad + \left(-\lambda_S+\widetilde{X}_S(s)\lambda_S - s\right)\Pr(W(t)=0) + \frac{o(\Delta t)}{\Delta t}. \end{aligned}$$

Letting $\Delta t \to 0$ in the above formula, we obtain a differential equation for $\widetilde{W}(t,s)$.

$$\frac{d\widetilde{W}(t,s)}{dt} \quad = \quad \left(s-\lambda_L+\lambda_L\widetilde{X}_L(s)\right)\widetilde{W}(t,s) + \left(-\lambda_S+\widetilde{X}_S(s)\lambda_S - s\right)\Pr(W(t)=0).$$

(ii) Let $t \to \infty$. Then, $\frac{d\widetilde{W}(t,s)}{dt} \to 0$ because the queue reaches the stationary state. Let $\widetilde{W}(s) \equiv \lim_{t\to\infty}\widetilde{W}(t,s)$. Then, $\widetilde{W}(s)$ is obtained as a function of $\pi_0 = \Pr(W(t)=0)$:

$$\widetilde{W}(s) \quad = \quad \frac{s+\lambda_S-\widetilde{X}_S(s)\lambda_S}{s-\lambda_L+\widetilde{X}_L(s)\lambda_L}\pi_0.$$

(iii) Next, we will obtain π_0 by evaluating $\widetilde{W}(s)$ at $s=0$. Note that the Laplace transform $\widetilde{Z}(s)$ of a probability distribution Z always has the property $\widetilde{Z}(0)=1$.

$$1 = \widetilde{W}(0) = \frac{1+\mathsf{E}[X_S]\lambda_S}{1-\mathsf{E}[X_L]\lambda_L}\pi_0.$$

The second equality follows from the L'Hopital's rule. Therefore,

$$\pi_0 \quad = \quad \frac{1-\lambda_L \mathsf{E}[X_L]}{1+\lambda_S \mathsf{E}[X_S]}.$$

(iv) The moments of waiting time, and subsequently response time, are easily obtained by differentiating $\widetilde{W}(s)$ and evaluating at $s=0$. In particular, the n-th moment of W is

$$\mathsf{E}[W^n] = \widetilde{W}^{(n)}(0).$$

■

Proof of Theorem 11

The donor queue can be seen as an M/GI/1 queue with generalized vacations, where a vacation starts when the number of donor jobs becomes zero and ends when the donor server starts working on donor jobs. In an M/GI/1 queue with generalized vacations, the mean response time has the following expression [57]:

$$\mathsf{E}[T] = \mathsf{E}[X_D] + \frac{\lambda_D \mathsf{E}[X_D^2]}{2(1-\rho_D)} + \frac{\mathsf{E}[A(A-1)]}{2\lambda_D \mathsf{E}[A]}, \tag{A.2}$$

where A denotes the number of jobs that arrive during a vacation period. Observe that the first two terms constitute the mean response time in a corresponding M/GI/1 queue (without vacation). Therefore, it suffices to analyze $\mathsf{E}[A]$ and $\mathsf{E}[A(A-1)]$.

There are two types of vacations, depending on how the vacation ends. The first type of vacation ends when a donor job arrives at an empty donor queue while the donor server is staying at the donor queue. In this case, $A=1$. The second type of vacation ends when a donor job arrives at a donor queue with $N_D^{th}-1$ jobs while the donor server is staying at the beneficiary queue or in the process of switching to the beneficiary queue. In this case, $A = N_D^{th} + B$, where B is the number of donor job arrivals during K_{ba}. Let p be the probability that a vacation is of the second type. Then,

$$\begin{aligned} \mathsf{E}[A] \quad &= \quad (N_D^{th} + \lambda_D \mathsf{E}[K_{ba}])p + (1-p) && \text{(A.3)} \\ \mathsf{E}[A(A-1)] \quad &= \quad \left(N_D^{th}(N_D^{th}-1) + 2N_D^{th}\lambda_D \mathsf{E}[K_{ba}] + \lambda_D^2 \mathsf{E}\left[K_{ba}^2\right]\right)p. && \text{(A.4)} \end{aligned}$$

All that remains is to derive p. Observe that the first type of vacation starts when a donor job arrives while the donor server is at the donor queue *and* the number of donor jobs is zero. Also, observe that the second type of vacation starts when a donor job arrives while the donor server is either at the beneficiary queue or in the process of switching to the beneficiary queue *and* the number of donor jobs is $N_D^{th}-1$. Since the arrival process is Poisson, p is given by the expression (6.1). The theorem now follows from (A.2)-(A.4). ■

Appendix B

Moment matching algorithm by Bobbio, Horváth, and Telek

In this section, we summarize the recent results by Bobbio et al. [22] on characterization of PH distributions and moment matching algorithms, which build upon our results in Chapter 2. Recall that $\mathcal{S}^{(n)}$ denotes the set of distributions that are well represented by an n-phase acyclic PH distribution (Definition 5). Bobbio et al. provide exact conditions for a distribution G to be in set $\mathcal{S}^{(n)}$ (see Theorem 21) as well as exact conditions for $G \in \mathcal{S}^{(n)^*}$ (see Theorem 20), where $\mathcal{S}^{(n)^*}$ is defined as follows:

Definition 20 *Let $\mathcal{S}^{(n)^*}$ denote the set of distributions that are well-represented by an n-phase acyclic PH distribution with no mass probability at zero for positive integer n.*

Further, Bobbio et al. provide a closed form solution for mapping any $G \in \mathcal{PH}_3$ to a minimal-phase acyclic PH distribution without mass probability at zero[1] (see Theorem 22). As we define the EC distribution and map an input distribution to an EC distribution in Chapter 2, Bobbio et al. also define a subset of PH distributions, and map an input distribution to a PH distribution in the subset. Specifically, Bobbio et al. map an input distribution to an Erlang-Exp distribution (see Figure B.1) or an Exp-Erlang distribution (see Figure B.2).

Theorem 20 *[22] A distribution G is in set $\mathcal{S}^{(n)^*}$ iff its normalized moments m_2^G and m_3^G satisfy the following conditions:*

$$m_2^G \geq \frac{n+1}{n}$$

and

$$m_3^G \textit{ lower bound:} \quad \begin{cases} l_n \leq m_3^G & \textit{if } \frac{n+1}{n} \leq m_2^G \leq \frac{n+4}{n+1} \\ \frac{n+1}{n} m_2^G < m_3^G & \textit{if } \frac{n+4}{n+1} \leq m_2^G \end{cases}$$

$$m_3^G \textit{ upper bound:} \quad \begin{cases} m_3^G \leq u_n & \textit{if } \frac{n+1}{n} \leq m_2^G \leq \frac{n}{n-1} \\ m_3^G < \infty & \textit{if } \frac{n}{n-1} < m_2^G, \end{cases}$$

[1] In [22], Bobbio et al. also provide a closed form solution for mapping any $G \in \mathcal{PH}_3$ to a minimal-phase general acyclic PH distribution, which can have mass probability at zero (i.e. the number of phases used is $\mathrm{OPT}(G)$).

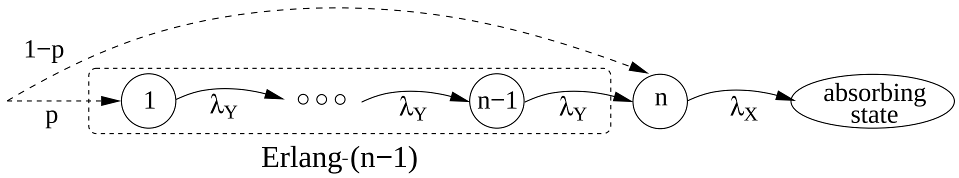


Figure B.1: *The Markov chain whose absorption time defines an Erlang-Exp distribution.*

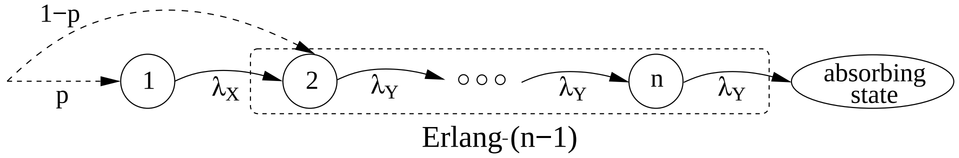


Figure B.2: *The Markov chain whose absorption time defines an Exp-Erlang distribution.*

where l_n and u_n are defined as follows:

$$\begin{aligned} l_n &= \frac{(3+a_n)(n-1)+2a_n}{(n-1)(1+a_n p_n)} - \frac{2a_n(n+1)}{2(n-1)+a_n p_n(na_n+2n-2)} \\ u_n &= \frac{1}{n^2 m_2^G}\left(2(n-2)(nm_2^G-n-1)\sqrt{1+\frac{n(m_2^G-2)}{n-1}} + (n+2)(3nm_2^G-2n-2)\right) \end{aligned}$$

where

$$\begin{aligned} p_n &= \frac{(n+1)(m_2^G-2)}{2m_2^G(n-1)}\left(\frac{-2\sqrt{n+1}}{\sqrt{4(n+1)-3nm_2^G}} - 1\right) \\ a_n &= \frac{m_2^G-2}{p_n(1-m_2^G)+\sqrt{p_n^2+\frac{p_n n(m_2^G-2)}{n-1}}} \end{aligned}$$

Theorem 21 *[22] A distribution G is in set $\mathcal{S}^{(n)}$ iff its normalized moments m_2^G and m_3^G satisfy the following conditions:*

$$m_2^G \geq \frac{n+1}{n}$$

and

$$\begin{aligned} &m_3^G \textit{ lower bound:} && \frac{n+2}{n+1}m_2^G \leq m_3^G \\ &m_3^G \textit{ upper bound:} && \begin{cases} m_3^G \leq u_n & \textit{if } \frac{n+1}{n} \leq m_2^G \leq \frac{n}{n-1} \\ m_3^G < \infty & \textit{if } \frac{n}{n-1} < m_2^G, \end{cases} \end{aligned}$$

where u_n is the same as in Theorem 20.

Theorem 22 *[22] Let G be a distribution in $\mathcal{S}^{(n)^*} \setminus \mathcal{S}^{(n-1)^*}$. If $m_2^G \leq \frac{n}{n-1}$ or $m_3^G \leq 2m_2^G - 1$, then G is well-represented by the Erlang-Exp distribution with the following parameters:*

$$
\begin{aligned}
\lambda_X &= \frac{ap+1}{\mu_1^G} \\
\lambda_Y &= \frac{\lambda_X(n-1)}{a} \\
a &= \frac{(bm_2^G-2)(n-1)b}{(b-1)n} \\
p &= \frac{b-1}{a} \\
b &= \frac{2(4-n(3m_2^G-4))}{m_2^G(4+n-nm_3^G)+\sqrt{nm_2^G c}} \\
c &= 12(m_2^G)^2(n+1)+16m_3^G(n+1)+m_2^G(n(m_3^G-15)(m_3^G+1)-8(m_3^G+3))
\end{aligned}
$$

If $m_2^G > \frac{n}{n-1}$ and $m_3^G > u_{n-1}$, then G is well-represented by the Exp-Erlang distribution with the following parameters:

$$
\begin{aligned}
\lambda_X &= \frac{a+p}{\mu_1^G} \\
\lambda_Y &= \frac{\lambda_X(n-1)}{a} \\
a &= \frac{2(f-1)(n-1)}{(n-1)(m_2^G f^2-2f+2)-n} \\
p &= (f-1)a,
\end{aligned}
$$

and f is given by the following steps[2]:

$$
\begin{aligned}
K_1 &= n-1 \\
K_2 &= n-2 \\
K_3 &= 3m_2^G-2m_3^G \\
K_4 &= m_3^G-3 \\
K_5 &= n-m_2^G \\
K_6 &= 1+m_2^G-m_3^G \\
K_7 &= n+m_2^G-nm_2^G
\end{aligned}
$$

[2] f is a solution for the fourth order equation $c_4f^4+c_3f^3+c_2f^2+c_1f+c_0=0$, where

$$
\begin{aligned}
c_4 &= m_2^G(3m_2^G-2m_3^G)(n-1)^2 \\
c_3 &= 2m_2^G(m_3^G-3)(n-1)^2 \\
c_2 &= 6(n-1)(n-m_2^G) \\
c_1 &= 4n(2-n) \\
c_0 &= n(n-2).
\end{aligned}
$$

$$
\begin{aligned}
K_8 &= 3+3(m_2^G)^2+m_3^G-3m_2^G m_3^G \\
K_9 &= 108K_1^2\left(4K_2^2K_3n^2m_2^G+K_1^2K_2K_4^2n(m_2^G)^2+4K_1K_5(K_5^2-3K_2K_6nm_2^G)\right. \\
&\quad \left.+\sqrt{-16K_1^2K_7^6+(4K_1K_5^3+K_1^2K_2K_4^2n(m_2^G)^2+4K_2nm_2^G(K_4n^2-3K_6m_2^G+K_8n)^2}\right) \\
K_{10} &= \frac{K_4^2}{4K_3^2}-\frac{K_5}{K_1K_3m_2^G} \\
K_{11} &= \frac{2^{\frac{1}{3}}\left(3K_5^2+K_2(K_3+2K_4)nm_2^G\right)}{K_3K_9^{\frac{1}{3}}m_2^G} \\
K_{12} &= \frac{K_9^{\frac{1}{3}}}{2^{\frac{7}{3}}3K_1^2K_3m_2^G} \\
K_{13} &= \sqrt{K_{10}+K_{11}+K_{12}} \\
K_{14} &= \frac{6K_1K_3K_4K_5+4K_2K_3^2n-K_1^2K_4^3m_2^G}{4K_1^2K_3^3K_{13}m_2^G} \\
K_{15} &= -\frac{K_4}{2K_3} \\
K_{16} &= \sqrt{2K_{10}-K_{11}-K_{12}-K_{14}} \\
K_{17} &= \sqrt{2K_{10}-K_{11}-K_{12}+K_{14}} \\
K_{18} &= 36K_5^3+36K_2K_4K_5nm_2^G+9K_1K_2K_4^2n(m_2^G)^2 \\
&\quad -\sqrt{81(4K_5^3+4K_2K_4K_5nm_2^G+K_1K_2K_4^2n(m_2^G)^2)^2-48(3K_5^2+2K_2K_4nm_2^G)^3} \\
K_{19} &= -\frac{K_5}{K_1K_4m_2^G}-\frac{2^{\frac{2}{3}}(3K_5^2+2K_2K_4nm_2^G)}{3^{\frac{1}{3}}K_1K_4m_2^GK_{18}^{\frac{1}{3}}}-\frac{K_{18}^{\frac{1}{3}}}{6^{\frac{2}{3}}K_1K_4m_2^G} \\
K_{20} &= 6K_1K_3K_4K_5+4K_2K_3^2n-K_1^2K_4^3m_2^G \\
K_{21} &= K_{11}+K_{12}+\frac{K_5}{2nK_1K_3} \\
K_{22} &= \sqrt{\frac{3K_4^2}{4K_3^2}-\frac{3K_5}{K_1K_3m_2^G}+\sqrt{4K_{21}^2-\frac{nK_2}{m_2^GK_1^2K_3}}} \\
f &= \begin{cases}
K_{13}+K_{15}-K_{17} & \textit{if } u_{n-1}<m_3^G<\frac{3m_2^G}{2} \\
K_{19} & \textit{if } m_3^G=\frac{3m_2^G}{2} \\
-K_{13}+K_{15}+K_{16} & \textit{if } m_3^G>\frac{3m_2^G}{2} \textit{ and } K_{20}>0 \\
K_{15}+K_{22} & \textit{if } K_{20}=0 \\
K_{13}+K_{15}+K_{17} & \textit{if } K_{20}<0.
\end{cases}
\end{aligned}
$$

Appendix C

Properties of Markovian arrival processes

Below, we summarize some of the basic properties of the MAP. First, the set of MAPs is quite broad and, in theory, any stationary point process can be approximated arbitrarily closely by a MAP.

Proposition 6 *[12] The set of MAPs is dense in the set of all the stationary point processes.*

Second, the set of MAPs is closed under some operations. In particular, a superposition of two independent MAPs is a MAP.

Proposition 7 *[117] A superposition of two independent MAPs, MAP($\mathbf{C_0}, \mathbf{C_1}$) and MAP($\mathbf{D_0}, \mathbf{D_1}$) is a MAP, MAP($\mathbf{E_0}, \mathbf{E_1}$), where*

$$\mathbf{E_0} = \mathbf{C_0} \otimes \mathbf{D_0} \quad \textit{and} \quad \mathbf{E_1} = \mathbf{C_1} \otimes \mathbf{D_1}.$$

Here, $\otimes$ denotes the Kronecker product.

Third, the average rate of events and the marginal distribution of the inter-event time of a MAP have simple mathematical expressions.

Proposition 8 *[117, 92] The average rate of events (the number of events in a unit time) in a MAP is called the fundamental rate of the MAP. The fundamental rate of MAP($\mathbf{D_0}, \mathbf{D_1}$) is given by*

$$\lambda = \overrightarrow{\theta}\mathbf{D_1}\overrightarrow{1},$$

where $\overrightarrow{\theta}$ is the stationary probability vector in the Markov chain with infinitesimal generator $\mathbf{D} = \mathbf{D_0} + \mathbf{D_1}$ (i.e., $\overrightarrow{\theta}\mathbf{D} = \mathbf{0}$ and $\overrightarrow{\theta}\overrightarrow{1} = 1$).

The marginal distribution of the inter-event time of the above MAP is a PH($\overrightarrow{\phi}, \mathbf{D_0}$) distribution, where

$$\overrightarrow{\phi} = \frac{\overrightarrow{\theta}\mathbf{D_1}}{\lambda},$$

which is the stationary probability vector immediately after the event. Note the difference between PH($\overrightarrow{\phi}, \mathbf{D_0}$) and PH($\overrightarrow{\theta}, \mathbf{D_0}$). PH($\overrightarrow{\theta}, \mathbf{D_0}$) is the distribution of the time from an arbitrary epoch, i.e., the excess of the inter-event time.

Since the marginal distribution of the inter-event time in a MAP is a PH distribution, its moments, density function, and distribution function can be calculated via Proposition 4.

The variability of a point process can be characterized by the covariance of the inter-event times or the index of dispersion for intervals. The index of dispersion for intervals, IDI($\cdot$), of a MAP is defined by

$$\mathrm{IDI}(i) = \frac{Var(S_i)}{i/\lambda^2},$$

where S_i is the sum of the first i inter-event times of the MAP. Observe that IDI(i) is the ratio of the variance of S_i to the corresponding variance of a Poisson process with the same rate. The MAP have a convenient mathematical expressions for the covariance and the index of dispersion for intervals.

Proposition 9 *[7, 92] Consider MAP($\mathbf{D_0}, \mathbf{D_1}$) with the fundamental rate λ. Recall the vectors, $\overrightarrow{\theta}$ and $\overrightarrow{\phi}$, of the MAP defined in Proposition 8. The covariance between the two inter-event times separated by $i-1$ events of the MAP is given by*

$$\lambda^{-1}\overrightarrow{\theta}\left((-\mathbf{D_0}^{-1}\mathbf{D_1})^i - \overrightarrow{1}\overrightarrow{\phi}\right)(-\mathbf{D_0})^{-1}\overrightarrow{1}.$$

The IDI($\cdot$) of the above MAP is given by

$$\begin{aligned} IDI(i) &= 2\lambda\overrightarrow{\theta}\left(\mathbf{I} + \mathbf{D_0}^{-1}\mathbf{D_1} + \overrightarrow{1}\overrightarrow{\phi}\right)^{-1}(-\mathbf{D_0})^{-1}\overrightarrow{1} - 1 \\ &\quad -\frac{2}{i}\lambda\overrightarrow{\theta}\left(\mathbf{I} - (-\mathbf{D_0}^{-1}\mathbf{D_1})^i\right)\left(\mathbf{I} + \mathbf{D_0}^{-1}\mathbf{D_1} + \overrightarrow{1}\overrightarrow{\phi}\right)^{-1}(-\mathbf{D_0}^{-1}\mathbf{D_1})(-\mathbf{D_0})^{-1}\overrightarrow{1}. \end{aligned}$$

Finally, a MAP can be translated into a counting process, i.e., the number of events by time t, N_t. The MAP has a simple z-transform of N_t, and this leads to a convenient mathematical expression for the index of dispersion for counts, which characterizes the variability of a MAP via the variability of N_t. The index of dispersion for counts, IDC($\cdot$), of the MAP is defined by

$$\mathrm{IDC}(t) = \frac{Var(N_t)}{\mathsf{E}[N_t]}.$$

Observe that IDC(t) is the ratio of the variance of N_t to the corresponding variance of a Poisson process with the same rate.

Proposition 10 *[7] Let N_t be the number of events by time t in MAP($\mathbf{D_0}, \mathbf{D_1}$). Recall vectors, $\overrightarrow{\theta}$ and $\overrightarrow{\phi}$, and the fundamental rate λ of the MAP defined in Proposition 8. Let $P(i,t)$ be the probability that $N_t = i$. The z-transform of $P(\cdot,t)$, $\hat{P}(z,t) \equiv \sum_{i=0}^{\infty} z^i P(i,t)$, is given by*

$$\hat{P}(z,t) = \exp((\mathbf{D_0} + z\mathbf{D_1})t).$$

for $|z| \le 1$, $t \ge 0$.

The IDC($\cdot$) of the above MAP is given by

$$\mathrm{IDC}(t) = 1 - 2\lambda + \frac{2}{\lambda}\overrightarrow{\theta}\mathbf{D_1}\overrightarrow{d} - \frac{2}{\lambda t}\overrightarrow{c}\left(\mathbf{I} - e^{\mathbf{D}t}\right)\overrightarrow{d},$$

where

$$\overrightarrow{c} = \overrightarrow{\theta}\mathbf{D_1}(\overrightarrow{1}\overrightarrow{\theta} - \mathbf{D})^{-1} \quad and \quad \overrightarrow{d} = (\overrightarrow{1}\overrightarrow{\theta} - \mathbf{D})^{-1}\mathbf{D_1}\overrightarrow{1}.$$